أسس
OSOS

توثيق أسس

التكامل والتوسعة

قسم من مركز التوثيق والمساعدة في أسس

النسخة العربية

إصدار المنصّة 1.10

تاريخ النشر: 23 نوفمبر 2026

الأدلة · الإرشادات

المحتويات

التكامل والتوسعة – ابدأ من هنا 5

الأدلة

التحزيم والتوزيع

التحزيم والتوزيع 6
صيغة الحزمة 23
مخطّطات موارد الحزمة 31
مخطط التبعيات 39
توليد حزمة 45
واجهة الاستيراد 50
إصدارات المخطط 61
نقل الوحدة بصيغة JSON 65
الحزم والمتجر والمتجر داخل التطبيق 69
الوظائف: شغّل رمازك الخاص 90
الملفات: قواعد التخزين 98
صيغ المعايير ومسارات الحقول 102
استدعاء واجهة HTTP 107
المصادقة OAuth 120
الزمن الحقيقي: الأحداث الحيّة والمكالمات 127
الويب هوك والتكاملات 131
لوحات معلومات Grafana 135
نموذج البيانات والأحداث 141
تشغيل المنصّة من المصدر (التطوير المحلي) 158

أسس كواجهة خلفية

أسس كواجهة خلفية 170
البدء السريع مع الواجهة الخلفية 174
اختيار بيانات الاعتماد 179
دورة حياة الرموز 182
القاطنون والهوية 185
قراءة البيانات 189
كتابة البيانات 193
التعامل مع الملفات 200
الاشتراك في الأحداث 205
العمل طويل الأمد 209
الأخطاء وإعادة المحاولة 214
توليد عميل 239
البيئات والترقية 243
الحدّ الأمني 246
نموذج سير العمل ومرجع واجهته البرمجية 249
مرجع واجهة معارض البيانات البرمجية 273
واجهة الأذونات البرمجية 292

مرجع نقاط النهاية

مرجع نقاط النهاية 303

Comments Service

Comment – comments-service 306
Reaction – comments-service 309

Data Service

Access – data-service 311
BackupAdmin – data-service 318

321 CounterWidget – data-service
323 Counters – data-service
324 Dashboard – data-service
326 DataCollections – data-service
334 DataEntities – data-service
341 DataEntityAggregate – data-service
344 DataEntityBlocks – data-service
347 DataEntityCategories – data-service
350 DataEntityFields – data-service
353 DataEntries – data-service
373 DataEntryTemplateExport – data-service
374 DataModule – data-service
377 DataQuery – data-service
381 DataRelation – data-service
389 DataViews – data-service
403 Delegations – data-service
404 EntityEntriesExport – data-service
405 Expression – data-service
406 GenericView – data-service
410 Groups – data-service
411 Icons – data-service
412 ImportExport – data-service
414 Job – data-service
415 Locks – data-service
417 OrgConfig – data-service
418 QuickFilterGroup – data-service
420 ReportEntry – data-service
425 Request – data-service
426 SensitiveFieldGrants – data-service
429 SideBar – data-service
430 Spaces – data-service
433 StatisticQuery – data-service
437 Stores – data-service
439 SystemButton – data-service
443 SystemLink – data-service
444 SystemObject – data-service
445 SystemSettings – data-service
446 TagGroup – data-service
448 Tags – data-service
449 Templates – data-service
450 UiSettings – data-service
454 User – data-service
457 UserPreferences – data-service
458 Version – data-service
459 ViewApiDocs – data-service
460 ViewEntriesExport – data-service
461 Workflow – data-service

Functions Service

462 Functions – functions-service

Notification Api

463 Notification – notification-api
465 NotificationTemplate – notification-api
466 PushNotificationsTokens – notification-api

Reaction Service

468 Announcement – reaction-service

DataReaction – reaction-service 471
ReactionLog – reaction-service 476
SystemReaction – reaction-service 478
WhatsAppTemplates – reaction-service 480

Timeline Service

Timeline – timeline-service 481

Workflow Service

Workflow – workflow-service 484
الطلبات والخدمات (الواجهة البرمجية والبنية الداخلية) 489
أمان الحساب والتحقق الإضافي (التفاصيل الداخلية) 497
مرجع نموذج الكيان وواجهته البرمجية 502
مرجع واجهة برمجة الصفحات والشريط الجانبي وروابط المشاركة 530
نموذج التفاعل ومرجع واجهته البرمجية 542
مرجع واجهة الأزرار البرمجية 580
تسجيل الدخول إلى البوابة والمساحات (التفاصيل الداخلية) 590
نموذج الاستعلام ومرجع واجهته البرمجية 596
العلاقات والمراجع وحقول الربط: النموذج والواجهة البرمجية 604
واجهة المستخدمين البرمجية وحسابات الهوية 614
مرجع المحصّلات والتجميع 621
مرجع واجهة برمجة قوالب التصدير 630
المجموعات والوحدات التنظيمية والانتماء 639
إطار البوابة: الصفحة الرئيسية والشريط الجانبي وروابط المشاركة (الواجهة البرمجية) 647
الكيانات الوهمية: النموذج والواجهة البرمجية والبنية الداخلية 651
واجهة التفويض البرمجية وآلية فرضه 666
مرجع واجهة برمجة السجل التاريخي والشريط الزمني 673
الإشعارات وتسليمها (الواجهة البرمجية والتفاصيل الداخلية) 685
نموذج التقرير ومرجع واجهته البرمجية 691
التسلسلات ذاتية الزيادة وقطع العدّادات: البنية الداخلية والواجهة البرمجية 700
التفاصيل الداخلية لوضع المعاينة 707
مخازن البيانات: النموذج والتوجيه والواجهة البرمجية 713
مرجع الواجهة البرمجية للخلاصات ولوحات المعلومات 720
مرجع المشاركة في سير العمل 735
مرجع التعليقات والإشارات 742
الحقول الحسّاسة والمشفَّرة: النموذج والواجهة البرمجية والإنفاذ 747
مرجع نموذج الوحدات وواجهتها البرمجية 777
مرجع إعدادات الحقول 788
أنواع الحقول الداخلية 805

خادوم MCP لوكلاء الذكاء الاصطناعي

خادوم MCP لوكلاء الذكاء الاصطناعي 808
البداية السريعة مع MCP 810
مرجع أدوات MCP 812
المُدخلات والقوائم: مرجع السلوك 814

الإرشادات

استخدم رمز API 820
أسئلة التكامل الشائعة 823

التكامل والتوسعة – ابدأ من هنا

هذا القسم موجَّه إلى **مطوّري التكامل والمطوّرين المحترفين**: من يربطون أنظمة أخرى بأسس أو يوسّعونها بالرماز. والتفاصيل التقنية هنا هي المقصودة: نقاط النهاية، والحمولات، وبيانات الاعتماد، ورموز الأخطاء.

أما إن كنت تبني تطبيقًا في المصمِّم، فمكانك بناء التطبيقات [بناء التطبيقات › بناء التطبيقات – ابدأ من هنا]، ولا يتطلب ذلك أي رماز.

ساعتك الأولى

1. **احصل على بيانات اعتماد.** حدّد النوع الذي يحتاجه تكاملك في اختيار بيانات الاعتماد، ثم أنشئه (ينشئ مسؤول القاطن مستخدم التطبيق ورمز الوصول الخاص به – راجع رموز الوصول إلى API).
2. **نفّذ أول استدعاء.** اتبع البدء السريع: اقرأ مُدخلات، ثم أنشئ مُدخلًا.
3. **تعرّف على العقد.** تغطي صفحات قراءة البيانات وكتابة البيانات والأخطاء وإعادة المحاولة ما يحتاجه أي تكامل.

اعثر على ما تحتاجه

أريد أن…	ابدأ من
أستدعي المنصة من نظام آخر	أسس كواجهة خلفية · واجهة HTTP
أصادق المستخدمين أو الخدمات	المصادقة OAuth · دورة حياة الرمز
أتفاعل مع التغييرات في نظام آخر	خطافات الويب والتكاملات · الاشتراك في الأحداث · الوقت الحقيقي
أشغّل رمازي الخاص داخل إجراء	الوظائف (FaaS)
أرفع الملفات وأنزّلها	الملفات · تخزين الملفات والمرفقات
أنقل التطبيقات بين البيئات	البيئات والترقية · الحزم والتوزيع
أتيح لوكيل ذكاء اصطناعي العمل مع المنصة	خادم MCP لوكلاء الذكاء الاصطناعي
أعرض بيانات المنصة في Grafana	مصدر بيانات Grafana
أبحث عن نقطة نهاية	مرجع نقاط النهاية
أشغّل المنصة من الرماز المصدري	التطوير المحلي

المرافِقات التقنية لأدلة البناء والإدارة

تصف صفحات البناء والإدارة الشاشات والنتائج، أما الواجهة البرمجية وسلوك المحرّك وراءها فمكانهما هنا: الطلبات والخدمات · واجهة الأذونات البرمجية · نموذج سير العمل وواجهته البرمجية · نموذج البيانات والأحداث · لهجات المعايير ومسارات الحقول.

التحزيم والتوزيع

هذه الصفحة موجّهة إلى **مطوّري التكامل والمهندسين** الذين يديرون التحزيم عبر الواجهة البرمجية، أو يبنون أدوات للتعامل مع الحزم، أو يحتاجون إلى معرفة ما يفعله محرّك التصدير والاستيراد بدقّة. وتشرح كيف تتكامل أجزاء المحرّك، والقواعد التي يفرضها، وكيف يحلّ التبعيات، وتحيل إلى الصفحات المرجعية لصيغة الأرشيف، وعقد الاستيراد على مستوى الاتصال، والمتجر.

تعمل من واجهة المستخدم؟

تجد توثيقًا للتصدير والاستيراد وإدارة الإصدارات والنشر والتثبيت من شاشات **الحزم**، موجّهًا إلى بُناة التطبيقات، في تصدير التطبيقات واستيرادها وتثبيتها [بناء التطبيقات].

ينقل **التحزيم** مجموعةً من موارد المنصّة المبنية – الوحدات، والكيانات، والمعارض، والاستعلامات، والعلاقات، والتفاعلات، والتقارير، والصفحات (المعارض العامة)، وأزرار النظام، والمجموعات، والاستعلامات الإحصائية، وقطع العدّادات، والوظائف، والإعلانات – من بيئة (أو مساحة) إلى أخرى، مع جميع تبعياتها. والحزمة ملفّ **ZIP** واحد يجمع الموارد المحدّدة وما تحتاج إليه من بيانات على مستوى المُدخلات، وقواعد مخازن البيانات المشتركة، وملفّات القوالب والملفّات الثنائية. والتحزيم هو الآلية التي يقوم عليها مسار الترقية المعتاد **التطوير › التجهيز › الإنتاج**.

التوفّر

منذ 1.8. أُطلق محرّك التصدير والاستيراد القائم على ZIP ضمن المعلَم 1.8 (وظهرت أولى إيداعاته على وسوم ما قبل إصدار 1.7، وملفّات تعريف التخطيط على وسوم ما قبل إصدار 1.8). **ومنذ 1.10** تعتمد عليه الحزم المسمّاة ذات الإصدارات، وصندوق المراجعة، والاتصال بالمتجر، والتثبيتات، والمتجر داخل التطبيق – انظر الحزم والمتجر والمتجر داخل التطبيق. أمّا تاريخ توفّر كل ميزة على حدة فمبيّن في قسمها.

أين تجد ما تحتاج إليه

ما تحتاج إليه	الصفحة
المحرّك، وقواعد الوصول إليه، وكيف يحلّ التبعيات	هذه الصفحة
الملفّات التي يحتويها ملف ZIP للحزمة بدقّة، وكل حقل في البيان، وحدود وقت الاستيراد والمُحقِّق	صيغة الحزمة
بنية JSON داخل كل ملف مورد	مخطّطات موارد الحزمة
الحواف في `dependencies.json` وكيف توجّه إعادة التخطيط والترتيب	مخطط التبعيات
كل نقطة نهاية لجلسة الاستيراد، وآلة الحالات، ومستند الجلسة، وقواعد التزامن	واجهة الاستيراد
قيم `schemaVersion` الموجودة والبيئات التي تقبلها	إصدارات المخطط
إنتاج حزمة أو تحويلها برمجيًا	توليد حزمة
الحزم، والإصدارات، وطلبات النشر، والمتجر، والتثبيت والتحديث	الحزم والمتجر والمتجر داخل التطبيق
آلية نقل الوحدات والكيانات الأقدم بملف JSON واحد	نقل الوحدة بصيغة JSON

كيف تتكامل أجزاء المحرّك

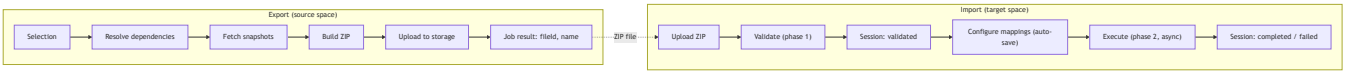
تتوزّع مهام التحزيم على خدمتين، كلتاهما مبنيّة على LoopBack 4 ومحميّة بـ Auth2:

الخدمة	المسار الأساسي	الدور
`packaging-service`	`/api/packaging`	مسار تنفيذ التصدير، والتحقّق من الاستيراد وتنفيذه، والجلسات، والتجاوزات، وملفّات تعريف التخطيط
`dependency-service`	`/api/dependency`	مخطط تبعيات موجَّه؛ يوفّر الحواف، وشجرة المعتمِدين، والترتيب الطوبولوجي

تُنتَج الحزمة عبر **مهمّة تصدير** وتُستهلَك عبر **جلسة استيراد**. وتُنفَّذ هاتان العمليتان الطويلتان بصورة غير متزامنة بواسطة عمّال **BullMQ** (المدعومة بـ Redis)؛ إذ تُدرج نقاط نهاية HTTP العمل في الصفّ وتُعيد معرّف مهمّة أو جلسة يمكن استخدامه لاستقصاء الحالة.

كل متحكّم في التحزيم موسوم بـ `@authenticate` باستراتيجية Auth2، وتتطلّب كل نقطة نهاية ترويسة الطلب `x-space-id`؛ وعند غيابها، يُعاد الخطأ `400 x-space-id header is required`.

دورة حياة من مرحلتين



تصدير حزمة

يبدأ التصدير بمجموعة الموارد الجذرية المحدّدة، ويحلّ تبعياتها، ثم يكتب ملف ZIP. ويجري ذلك في تطبيق الويب من صفحة **تصدير الموارد** ضمن **الحزم › البناء** (انظر دليل بُناة التطبيقات [بناء التطبيقات › تصدير التطبيقات واستيرادها وتثبيتها])؛ أمّا عبر الواجهة البرمجية، فتتولّاه نقاط النهاية أدناه.

> **تلميح:** شغّل **تحليل** التصدير أولًا دائمًا، وعالج تحذيرات التبعيات الخارجية قبل شحن الحزمة؛ فالمورد الذي تفتقد الحزمةُ إحدى تبعياته الصلبة لن يُستورَد استيرادًا سليمًا.

ما تحتويه الحزمة

تحتوي كل حزمة على **بيانًا** يصف المصدر (إصدار المخطط، وبيئة المصدر، ومساحة المصدر، وإصدار المصدر)، وعلى مجموعة موارد **مرتّبة طوبولوجيًا** – بحيث يسبق الآباء الأبناء. ويتتبّع المُصدِّر حواف التبعية **الصلبة** للموارد المحدّدة لتكون الحزمة مكتفية بذاتها؛ أمّا المراجع **الليّنة** (الاختيارية) فتُسجَّل دون تتبّعها. وتُجمَع معها المراجع على مستوى المُدخلات، وقواعد مخازن البيانات المشتركة، وملفّات القوالب والقطع والصفحات الثنائية التي تحتاج إليها مواردك. أمّا مواصفات الأرشيف نفسه، فترد في صيغة الحزمة.

واجهة التصدير البرمجية

تقع جميع نقاط نهاية التصدير تحت `/api/packaging/v1/export`، وتشترك في جسم الطلب `ExportRequest` وحارس `x-space-id`.

الأسلوب	المسار	الغرض	يُعيد
`POST`	`/v1/export/preview`	أعداد دقيقة للموارد بعد تطبيق خطّ الجلب والترشيح الفعلي	`ExportPreviewResponse`
`POST`	`/v1/export/analysis`	المعاينة + تحليل التبعيات الخارجية في طلب واحد	`AnalyzeExportResponse`
`POST`	`/v1/export`	إدراج مهمّة تصدير في الصفّ، و `Since 1.10` كتابة سجلّ لها	`{ jobId, exportId }`
`GET`	`/v1/export/{jobId}/status`	استقصاء حالة المهمّة ونتيجتها	`ExportStatusResponse`

يشترط `GET /status` أن تكون مساحة المستدعي مالكةً للمهمّة (وإلا أُعيد `403 Access denied to this export job`)، و`Since 1.10` يشترط أن يكون المستدعي هو المستخدم الذي بدأها أو مسؤول نظام؛ ويتلقّى أي شخص آخر `404` كما لو كانت

المهمّة غير موجودة. وعند النجاح، تتضمّن نتيجة الحالة معرّف التخزين `fileId` والاسم المولَّد `name`؛ ويُخزَّن ملف ZIP القابل للتنزيل في **storage-service** بوصفه ملفًّا مؤقّتًا. ويُشتقّ اسم الملف من الاسم الإنجليزي للتصدير، أو من اسمه العربي عند غياب الإنجليزي، بعد تنقيته ليقتصر على `[a-zA-Z0-9_-]`؛ وإذا لم يبقَ منه شيء (كأن يكون الاسم عربيًا خالصًا)، يُستخدم الاسم البديل `<NAMESPACE>-package-<ISO-timestamp>.zip`.

تُعيد المعاينة أعداد الموارد بحسب النوع والكيان؛ ويضيف التحليل **تحليل التبعيات الخارجية**، أي كل ما تشير إليه الموارد المحدّدة ويقع *خارج* نطاقها، مجمّعًا بحسب الوحدة، مع إبراز التبعيات الخارجية غير المحلولة بوصفها تحذيرات.

حقول ExportRequest

الحقل	النوع	ملاحظات
`resources`	`ResourceRef[]`	**مطلوب** (≥1، وإلا `400 At least one resource is required`). كل مرجع بصيغة `{type, id, withChildren?}`.
`excludes`	`ResourceRef[]`	أزواج `(type,id)` محدّدة تُستبعَد من المجموعة الناتجة عن حلّ التبعيات.
`excludeTypes`	`ResourceType[]`	أنواع موارد كاملة تُستبعَد من المجموعة الناتجة عن حلّ التبعيات.
`description`	`string`	نصّ حرّ يُخزَّن في `manifest.description`.
`name`	`ILocalizedValue` (`{en?, ar?}`)	`Since 1.10` **مطلوب في `POST /v1/export`**: يُعاد `400 EXPORT_NAME_REQUIRED` إذا لم تتضمّن أيٌّ من اللغتين نصًّا (وتتجاهله المعاينة والتحليل). ويُستخدم لتسمية مصدر التصدير وملف ZIP (انظر أعلاه). وكان في 1.8 نصًّا عاديًا اختياريًا.
`maxClosureRounds`	`number`	عمق الإغلاق المتعدّي لـ **مراجع المُدخلات**. القيمة الافتراضية `5`؛ وتعطّله القيمة `0` (الحواف المباشرة فقط). وكلّما زاد العمق، شمل الجمع مُدخلات متداخلة على مستويات أعمق، مقابل زيادة عدد استدعاءات الواجهة البرمجية وحجم الحزمة.

يُفعِّل `ResourceRef.withChildren` «الحلّ الذكي»: ففي `DataModule` و `DataEntity`، توسّع الخدمة شجرة الموارد المعتمِدة ذات النوع `MEMBERSHIP` المأخوذة من dependency-service، وتجلب الأبناء (باستثناء الأبناء المقيّدين بمستخدم والأبناء الخارجيين). وفي تطبيق الويب، يُرسَل تحديد عقدة وحدة أو كيان مصحوبًا بـ `withChildren`.

لا يحمل التصدير العادي ملف README `Since 1.10`. فلا يتضمّن `POST /v1/export` حقل `readme`، إذ يرتبط ملف README بالحزمة وإصداراتها (ويُحرَّر في صفحة الحزمة ويُكتب في ملف ZIP لكل إصدار). ولا تزال آلية الاستيراد تقرأ أي ملف README تجده في الأرشيف وتحتفظ به. ويبلغ حدّ حجم جسم الطلب في `POST /v1/export` **30 MiB** (ويُعاد `413 PAYLOAD_TOO_LARGE` عند تجاوزه).

سجلّات التصدير `Since 1.10`

يكتب كل طلب `POST /v1/export` **سجلّ تصدير** قبل إدراج المهمّة في الصفّ، مما يتيح لتبويب **التصديرات** في صفحة البناء سرد التصديرات السابقة، وإعادة فتح أحدها، وبدء التصدير نفسه من جديد (تُفتح صفحة التصدير مع `?from=<exportId>`). وتكون الاستجابة `{jobId, exportId}`، ويكون معرّف المهمّة `export-<exportId>`؛ ويمكن استقصاء `GET /v1/export/{jobId}/status` كالمعتاد.

الأسلوب	المسار	الغرض
`GET`	`/v1/exports`	سجلّات تصدير المساحة، الأحدث أولًا، مع `total`. تُقسَّم النتائج إلى صفحات باستخدام `skip` (الافتراضي `0`) و `limit` (الافتراضي `50`، وبحدّ أقصى `100`).
`GET`	`/v1/exports/{id}`	سجلّ واحد **مع تحديده** (`resources` و `excludes` و `excludeTypes`)، لبدء التصدير نفسه من جديد. وإذا كان المعرّف يخصّ مساحةً أخرى، يُعاد `404 EXPORT_NOT_FOUND`.

الأسلوب	المسار	الغرض
`DELETE`	`/v1/exports/{id}`	يحذف **السجلّ** وحده.

وتُتاح نقاط النهاية الثلاث لكل من يستطيع بدء تصدير (من يملك مستوى Developer في موضع ما؛ انظر صلاحية التصدير). ويتضمّن السجلّ `id` و`name` و`description?` و`resourceCount` (التحديد كما أُرسل) و`packagedCount?` (ما ضمّته الحزمة بعد حلّ التبعيات) و`status` (`running` أو `completed` أو `failed`) و`createdBy` و`createdAt` و`completedAt?` و`fileId?` و`size?` و`failedReason?`. ويُحدَّث السجلّ ذو الحالة `running` عند قراءته إذا كانت مهمّته قد انتهت بالفعل؛ أمّا السجلّ الذي يتعذّر العثور على مهمّته بعد 60 ثانية، فتُغيَّر حالته إلى `failed` مع `EXPORT_JOB_LOST`.

حذف السجلّ لا يحذف ملف ZIP. يبقى الملف في التخزين، وتكتمل المهمّة إذا كانت لا تزال قيد التشغيل. وفي المقابل، يُعدّ ملف ZIP للتصدير العادي ملفًّا مؤقّتًا، لذا قد يشير `fileId` في السجلّ إلى ملف انتهت صلاحيته بالفعل.

صلاحية التصدير `Since 1.10`

التصدير إجراء **للمطوّر** لا للمسؤول. ويُمنَح إذن استدعاء `POST /v1/export` و`/v1/export/preview` و`/v1/export/analysis` بناءً على الجذور الواردة في `resources`:

- **منحة على مستوى النظام بمستوى Developer أو أعلى** تجيز تصدير أي جذور؛
- تؤدّي منحة على مستوى النظام بمستوى **None** إلى الرفض، أيًّا كانت المنح على مستوى الموارد؛
- وفيما عدا ذلك، يحتاج **كل** جذر إلى مستوى Developer أو أعلى على الوحدة أو الكيان الذي يملكه؛ فتُستخدَم للوحدة منحة الوحدة، وللكيان منحة الكيان إن وُجدت، وإلّا فمنحة وحدته، وللمجموعة منحة كيانها (فقط حين تكون المجموعة والكيان في الوحدة نفسها)، وللصفحة (المعرض العام) منحة الوحدة التي تنتمي إليها. ويتطلّب أي نوع جذر آخر، أو قائمة فارغة، منحةً على مستوى النظام.

وتعكس صفحة التصدير وشجرة مواردها هذه القواعد؛ إذ تُتاح الصفحة لمطوّري الوحدات، ولا تعرض إلّا الجذور التي يستطيع المستخدم تطويرها.

ويرفض التصدير نفسه الموارد التي تنتمي إلى **حزمة مثبّتة** بالرمز `403 EXPORT_INSTALLED_RESOURCE` (`details.resourceIds` و`packageId`)، إذ يتولّى ناشر القالب المثبّت شحنه. انظر الموارد المثبّتة.

ما الذي يُصدَّر (وما الذي لا يُصدَّر)

`EXPORTABLE_TYPES` — هذه الأنواع وحدها هي التي تُكتَب إلى `resources/` داخل ملف ZIP:

`DataEntity`، `DataView`، `DataQuery`، `DataRelation`، `DataCollection`، `DataReaction`، `GenericView`، `DataModule`، `SystemButton`، `StatisticQuery`، `Report`، `Function`، `CounterWidget`، `Announcement`، `DataEntityAggregate` (`Since 1.10`)، و`QuickFilterGroup` و`TagGroup` (كلاهما `Since 1.10`).

يُصدَّر نوعا المجموعات بوصفهما أبناءً للكيان. وعند الاستيراد، لا يُكتب `id` ولا `dataEntity` أبدًا (فالكيان يُحدَّد من مسار الهدف)، ويُستخدَم `normalizedName` بوصفه الهوية التي تُطابَق بها مجموعة موجودة. ولا يغيّر التحديث إلّا `name` و`queries`؛ كما لا يتغيّر `color` لمجموعة الوسوم عند التحديث.

يستبعد `isNonExportableResource` ما يلي، حتى لو حُدِّد:

- الكيانات **المؤرشَفة** (`state === Archived`)، مع شجرتها الفرعية المتداخلة بأكملها.
- الموارد **المقيّدة بمستخدم**.

- الكيان `DataEntity` **الضمني** (ما لم يكن `implicitOptions.allowEdit`) وأي `DataReaction` **ضمني**.

كيانات الهوية الأساسية – `USER` و`GROUP` و`ORGUNIT` و`WORKING_HOURS` (`CORE_ENTITY_IDS`) – لا تُكتَب أبدًا بوصفها موارد؛ فهي موجودة بصورة مستقلة في كل بيئة، وتُعامَل بوصفها مراجع (انظر مراجع القيم).

التفاصيل الداخلية لخطّ التصدير

ينفّذ `PackageExportService.executeExport` الخطوات التالية بالترتيب:

1. **حلّ التحديدات** – توسيع `withChildren`، وتطبيق `excludes` / `excludeTypes`، وإزالة التكرار بحسب `(type,id)`.
2. **الجلب والترشيح** – جلب لقطات JSON الكاملة لكل نوع على دفعات من data-service (الإسقاط `value,dataEntity,state` للمُدخلات)، وفهرسة كلٍّ منها بقيمة `normalizedName` فريدة (يُلحَق بها المعرّف عند التصادم)، ثم تطبيق `isNonExportableResource` واستبعاد الشجرة الفرعية المؤرشَفة تسلسليًا.
3. **تفريغ عناصر مجموعات الكيانات** – تُصدَّر لقطات `DataCollection` غير التابعة لـ ADT مع `items: []` (لأن عناصرها تشير إلى معرّفات `DataEntry` غير موجودة في الحزمة)؛ أمّا مجموعات ADT فتحتفظ بعناصرها العددية المضمّنة.
4. `Since dev` **نزع أسرار الوظائف** من كل `Function` في الحزمة؛ انظر أدناه.
5. **جمع الحمولات العابرة** – مراجع المُدخلات (المراجع على مستوى القيمة إلى معرّفات UUID للنواة أو المُدخلات، مع الإغلاق المتعدّي)، والمخازن المشتركة (`storeRules` حيث `type === 2`)، ومراجع ملفّات القوالب والملفّات الثنائية.
6. **تجريد** الحقول التي يديرها الخادم (`SERVER_MANAGED_KEYS`، مثل `creationInfo` / `updateInfo`) و`executionPlan` من المعارض، إذ يعيد الاستيراد اشتقاقها.
7. **بناء البيان والفهرس**، وترتيب الموارد طوبولوجيًا (خوارزمية Kahn على الحواف الصلبة)، و**بثّ** ملف ZIP عبر `archiver` (مستوى zlib السادس)، و**رفعه** إلى storage-service. ويحافظ البثّ على استهلاك محدود لذاكرة العامل (حدّ علوي ~16 MB)، كي لا تنفد الذاكرة في عمليات التصدير التي تبلغ مئات الميغابايت.

أسرار الوظيفة تبقى في بيئتها `Since dev`

في خطّ التطوير، يمكن أن تحمل الوظيفة **أسرارًا** مشفّرة، وأن تكون لها **هوية** خاصة بها (مستخدم تطبيق)؛ انظر أسرار الوظيفة وهويتها الخاصة. ولا ينقل التحزيم أبدًا قيمة أي سرّ أو هوية أي وظيفة بين البيئات، لأن النص المشفّر مرتبط بمفاتيح بيئة المصدر.

الخطوة	ما يحدث لـ `Function`
التصدير	يحتفظ كل صفّ سرّ بـ `key` (وبأي حقل آخر)، لكن قيمته `value` تصبح `null`؛ ويُزال صفّ `OSOS_API_TOKEN` الذي يديره النظام ومرجع `identity`. ويجري ذلك قبل جمع مراجع المُدخلات، كي لا تؤدّي الهوية إلى تضمين سجلّ مستخدم الوظيفة في الحزمة. وتُنزَع أسرار الوظيفة التي يقتصر وجودها على أن موردًا مُصدَّرًا *يشير إليها* بالطريقة نفسها.
تجزئة المحتوى	تشمل التجزئة **أسماء الأسرار بترتيبها**، ولا تشمل أيًّا من القيم التي تضبطها البيئة الهدف لنفسها؛ وبذلك تبقى تجزئة الوظيفة المثبّتة مساوية لتجزئة حزمتها، أيًّا كانت القيم التي تُملأ بها أسرارها لاحقًا.
التثبيت (الإنشاء)	تُنزَع القيمة المحزَّمة مرة أخرى، كي لا تتمكّن حزمة صُدّرت قبل تطبيق هذه القاعدة – وقد تحمل النص المشفّر للمصدر وصفّ الرمز والهوية – من إدخالها إلى الهدف.
تحديث وظيفة مثبّتة	تُنقل حالة الهدف نفسه: هويته، وصفّ `OSOS_API_TOKEN` الخاص به، والقيمة المخزّنة لكل سرّ لا تزال الحزمة تعلن مفتاحه. وتُسقَط الأسرار التي لم تعد الحزمة تعلنها، بينما تصل الأسماء المعلنة حديثًا بقيم فارغة. ويقرأ التحديث الهدف أولًا دائمًا، ويفشل إذا تعذّرت تلك القراءة، تجنّبًا للكتابة دون معرفة حالته.

ولا يشترط التحزيم ضبط أي سرّ: فالوظيفة المثبّتة حديثًا تحمل الأسماء المعلنة بقيم فارغة إلى أن يملأها مسؤول في البيئة الهدف.

صيغة الحزمة (ZIP)

ترد مواصفات بنية الأرشيف، وكل حقل في البيان، و `index.json` ، و `dependencies.json` ، والملفّ الجانبي لحقول الربط، وملفّات الإصدار المنشور، وحدود الأمان وقت الاستيراد، والمُحقِّق الصارم في صيغة الحزمة. أمّا قيم `schemaVersion` المتاحة، والبيئات التي تقبلها، وكيفية تعامل التحقّق مع `sourceVersion` و `specificationVersion` لكل مورد، فترد في إصدارات المخطط. وإصدارات الصيغة مستقلة عن إصدار المنصّة، وتبقى الحزم الأقدم قابلة للاستيراد بحكم التصميم.

استيراد حزمة

الاستيراد عملية من ثلاث مراحل تعتمد على الجلسات. وينشئ كل استيراد **جلسة استيراد** تتعقّب التقدّم ويمكن إعادة محاولتها أو إلغاؤها. وتحدّد واجهة الاستيراد عقد الاتصال لكل مرحلة – أجسام الطلبات، والاستجابات، وبوّابات الحالة، ورمز التزامن – ويشرح هذا القسم ما يفعله المحرّك في كل مرحلة. وفي تطبيق الويب تمرّ العملية عبر **الحزم ← المثبتة ← التثبيت من ملف** (انظر دليل بُناة التطبيقات [بناء التطبيقات ، تصدير التطبيقات واستيرادها وتثبيتها]).

ملف ZIP الذي يتضمّن بيانه كتلة `package` هو إصدار حزمة منشور، ويُستورَد بوصفه **تثبيتًا** – انظر التثبيت.

المرحلة 1 – التحقّق

يأخذ `POST /v1/import/validate` القيمة `{fileId, fileName?, importStrategy?}` ، وينزّل ملف ZIP من storage-service، ويشغّل `ValidationService.validate` الذي:

1. يستخرج ملف ZIP ويتحقّق منه ويفحص `schemaVersion` .
2. يصنّف كل مورد قابل للتصدير بمقارنته بالمساحة الهدف، محدِّدًا **النطاق** الصحيح (نطاق الكيان، أو نطاق الوحدة، أو النطاق العام – انظر `resolveResourceScope`) ومطابِقًا حسب `normalizedName` (على دفعات لكل نوع، مع استعلامات متكرّرة مرشَّحة بحسب الكيان أو الوحدة، والرجوع إلى المعرّف عند الحاجة). وتكون النتيجة لكل مورد: `NEW` أو `EXISTING` (`ValidationStatus`).
3. يولّد تخطيطات الموارد الافتراضية: `NEW → CREATE` ؛ و `EXISTING → SKIP` (أو `OVERWRITE` حين تكون الاستراتيجية `OVERWRITE_ALL`).
4. يُثري التخطيطات ببيانات وصفية للتبعيات (`dependsOn` ، و `requiredBy` ، وملكية الوحدة، و `cascadeSkipCount`). ويكتشف **التبعيات غير المخطَّطة** (المراجع الخارجية الصلبة غير الموجودة في الحزمة، و `Since dev` مراجع قطع الصفحات إلى عناصر خارجها)، ويكتشف **مراجع القيم**.
5. يحفظ `ImportSession` (بالحالة `validated` ، أو بالحالة `failed` مع تعذّر المتابعة حين توجد أخطاء حاجبة) ويُعيد `{sessionId, validationReport, updateIdentifier}` .

ويتيح لك `ValidationReport` (`resources` و `errors` و `warnings` و `canProceed` و `unmappedDependencies?` و `availableTargets?` ، و `Since 1.10`) مراجعة كل شيء قبل أي كتابة. فالتحقّق للقراءة فقط: لا يُكتب شيء في الهدف في هذه المرحلة. وتُبرَز هنا تعارضات الهوية والتبعيات غير المخطَّطة لتراجعها.

بعد نجاح التحقّق، يعيد تطبيق الويب تحميل الجلسة الجديدة ليعرض مراجع القيم التي حلّها الخادم. وإذا أخفقت إعادة التحميل، فإنه يفتح خطوة المراجعة بالتقرير المتاح لديه، ويوضّح أن مراجع القيم تعذّر تحميلها. `Since 1.8`

المرحلة 2 – الإعداد

يحدّد الإعداد كيفية كتابة كل مورد في الهدف. وهو **يُحفظ تلقائيًا** مع قفل تفاؤلي، فلا يستطيع شخصان يعدّلان الجلسة نفسها أن يكتب أحدهما فوق تعديلات الآخر دون تنبيه. ويتعمّد `PATCH /…/config` إتاحة المرونة – فكل قسم اختياري، وتُؤجَّل

القيود الصارمة إلى التنفيذ. وهو يدمج تخطيطات الموارد، وخيارات مراجع القيم، وأهداف التبعيات غير المخطَّطة، ويحذف مراجع القيم المتقادمة، ويعيد تشغيل حارس المراجع المتداخلة في `AUTO_CREATE`، ويُعيد `dependencyImpact` / `mappingSummary`. و`Since 1.10` يتضمّن طلب الحفظ نفسه `mappedFieldDecisions` ويُعيد `mappedFields` و`canProceed` بعد إعادة حسابهما (انظر حقول الربط).

الإجراء لكل مورد (`MappingAction`). القيم المستخدمة في الاتصال هي `CREATE` و`OVERWRITE` و`SKIP`؛ ويعرضها تطبيق الويب على هيئة أربع نتائج:

النتيجة	على مستوى الاتصال	المعنى
إنشاء	`CREATE`	إنشاء مورد جديد في الهدف، بمعرّف جديد.
تعيين (إعادة استخدام)	`SKIP` مع `targetId` على مورد `EXISTING`	إبقاء المورد الموجود في الهدف دون تغيير؛ ويُسجَّل تخطيط المعرّف مع ذلك، فتُحَلّ إليه المراجع الصادرة من الموارد المستورَدة الأخرى.
استبدال	`OVERWRITE`	استبدال المورد الموجود في الهدف بالنسخة المحزَّمة.
تخطي	`SKIP`	عدم استيراد هذا المورد. وإذا كان المورد `NEW`، تُتخطّى الموارد التي تعتمد عليه بتبعيات صلبة تسلسليًا.

ويمكنك أيضًا:

- تطبيق **استراتيجية جماعية** على كل التخطيطات عبر `PATH /…/strategy` – **يدوي**، أو **دمج (الجديد فقط)** (`MERGE_NEW_ONLY`)، أو **استبدال الكل** (`OVERWRITE_ALL`) – مع الحفاظ على خيارات المستخدم لإعادة تخطيط الوحدات.
- **قبول اقتراحات المطابقة التامة** (`PATH /…/suggestions/accept-exact`) لتحويل كل اقتراح مطابقة تامّة لصاحب صلاحية لم يُتّخذ بشأنه إجراء، دفعةً واحدة، إلى `REMAP` نحو هدفه المقترح.
- **إعادة تخطيط أصحاب الصلاحيات** – تخطيط المستخدمين والمجموعات والوحدات التنظيمية من المصدر إلى الهدف. وتحفظ ملفّات تعريف التخطيط هذه التخطيطات بين مصدر وهدف معيّنين، فيطبّقها الاستيراد التالي تلقائيًا.
- **تقرير ما ينسخ منه كل حقل ربط** حين يتعذّر على الاستيراد تحديده (انظر حقول الربط).
- اختياريًا، **تحرير حمولة مورد يدويًا** (تجاوز الحمولة) قبل الاستيراد.

المرحلة 3 – التنفيذ

يتحقّق `POST /…/execute` أولًا من حسم كل تبعية غير مخطَّطة وكل حقل ربط وكل مرجع قيمة يحتاج إليه التشغيل، و`Since 1.10` من أن لدى القاطن سعةً كافيةً لكل مُدخل سينشئه التشغيل؛ أمّا تسلسل بوّابات التحقّق كاملًا ورموز أخطائه ففي واجهة الاستيراد. ثم ينتقل ذرّيًا من `validated → importing` (بحارس يمنع التنفيذ المزدوج)، ويُدرج مهمّة استيراد BullMQ في الصفّ، ويحفظ `importJobId`.

يستورد المحرّك الموارد بترتيب التبعيات، منفّذًا **إعادة تخطيط المعرّفات** بحيث يُحوَّل كل معرّف مصدر إلى معرّف الهدف الصحيح عند إنشاء الموارد، ويعالج الحالات المعقّدة تلقائيًا – التخطّي التسلسلي، وكسر الدورات، وتصفير التبعيات الليّنة، ومعالجة مراجع القيم، وتمريرة إعادة المحاولة (انظر حلّ التبعيات وإعادة التخطيط).

- **إعادة المحاولة** (`POST /…/retry`) تُعيد الإدراج في الصفّ، وتُعيد الصفوف الفاشلة والمتخطّاة تسلسليًا إلى حالة الانتظار، ولا تعاود إلّا ما لم يُكتب. ويُحال تنفيذ جلسة حالتها أصلًا `completed` / `failed` إلى إعادة المحاولة. ويُفضَّل استخدام إعادة المحاولة على بدء الاستيراد من الصفر.
- **الإلغاء** (`POST /…/cancel`) يوقف استيرادًا في الصفّ أو قيد التشغيل؛ وتبقى الموارد المكتوبة سلفًا. و`Since 1.10` أصبح الإلغاء **نهائيًا** – فابدأ استيرادًا جديدًا.

تجاوزات حمولة كل مورد

بين التحقّق والتنفيذ، يمكن للمستدعي استبدال JSON لمورد واحد في التنفيذ التالي؛ ويبقى الأرشيف دون تغيير. وفي تطبيق الويب يجري ذلك عبر **تحرير JSON** في صفّ المورد، وهو خيار يفتح محرّر رماز مضمّنًا مع تلوين الصياغة. وترد نقاط النهاية وقواعد التحقّق وبوّابات الحالة في واجهة الاستيراد؛ أمّا متى تُستخدم هذه التجاوزات بدل تحويل ملف ZIP، فيوضّحه توليد حزمة.

حواجز الوصول والتنفيذ في الاستيراد `Since 1.10`

- **الاستيراد لمسؤول المساحة وحده.** التحقّق والإعداد والتنفيذ وإعادة المحاولة والإلغاء وتجاوزات الحمولة وملفّات تعريف التخطيط كلها تتطلّب منحة مسؤول على مستوى النظام (أو استدعاء خدمة). فعمليات الكتابة التي يجريها الاستيراد في data-service تتجاوز المنح على مستوى الموارد، ولذلك يُطبَّق حاجز الوصول عند نقطة الدخول. أمّا التصدير، فيتطلّب Developer على الجذور المصدَّرة.
- **مراجع القيم تحتاج إلى إجراء صريح.** يُعيد التنفيذ `400 VALUE_REFERENCE_ACTION_UNSET` ما دام هناك مرجع قيمة سيكتبه الاستيراد دون إجراء محدّد أو قيمة افتراضية آمنة، بدل كتابة معرّف مصدر في الهدف.
- **لا إنشاء صامت لأصحاب الصلاحيات.** يُرفض **إنشاء تلقائي** لمرجعٍ تطابق هويته – البريد الإلكتروني للمستخدم أو اسم المستخدم مثلًا – مُدخلًا موجودًا في الهدف، ويقتصر الرفض على هذا المرجع (`VALUE_REFERENCE_IDENTITY_COLLISION`)؛ ولا يُنشأ تلقائيًا أبدًا صاحب صلاحية لم تختره.
- **الأهداف المؤرشفة ليست «موجودة».** تُعدّ مطابقة مورد في الحزمة لهدف مؤرشف خطأً حاجبًا بالرمز `TARGET_ARCHIVED`، بدل الكتابة فوق سجلّ مؤرشف؛ ويُجرى الفحص مجددًا عند إعادة المحاولة.
- **لا كتابة فوق الموارد المثبّتة.** يُرفض الاستيراد العادي الذي سيكتب موردًا يملكه تثبيت حزمة بالرمز `409 TARGET_INSTALLED`؛ فاشتقّ التثبيت أولًا (الاشتقاق والإزالة).
- **تُستبدل فهارس الكيان، ويُسقط المتعذّر منها.** يتضمّن تحديث الكيان فهارسه دائمًا، ولذلك **يستبدل** الاستيراد الفهارس المخصّصة للكيان الهدف بالفهارس المحزَّمة (ولا تُصدَّر حالة الفهرس التي يديرها الخادم ولا تُكتب أبدًا). ويُسقَط الفهرس المخصّص الذي لا يستطيع الهدف فهرسة مسار حقله – لأن المسار مفقود، أو له بنية لا يستطيع data-service فهرستها، أو يحدّد لغة غير متاحة في الهدف – ويُبلَّغ عنه في `importWarnings` ضمن النتيجة (`Dropped index "<name>": field path <path> …`) بدل إفشال استيراد الكيان كله.
- **حقول الربط تحتاج إلى قرار.** يُعدّ حقل الربط الذي لا يجد الاستيراد مصدره خطأ تحقّق حاجبًا إلى أن يختار المثبِّت ما ينسخ منه، أو يختار عدم النسخ (انظر حقول الربط).
- **يُفحص حدّ المُدخلات قبل التشغيل.** قبل بدء تثبيت أو إعادة محاولة، تتحقّق آلية التحزيم من أن لدى القاطن سعةً كافيةً **لجميع** المُدخلات التي سينشئها التشغيل، وترفض العملية بالرمز `403 ENTRY_CAP_EXCEEDED` ("Archive or delete entries, or upgrade the plan, then install again") إن لم تتوافر هذه السعة. ولا يُكتب شيء. وإذا أخفق الفحص نفسه لسبب آخر، يُسجَّل ذلك ويستمر التشغيل، لأن كل عملية إنشاء منفردة تظلّ خاضعة للفحص.
- **الملفّات التي تعذّر نقلها تُسقَط ولا تُفشل الاستيراد.** يُزال الملف من المورد الذي لا يزال يحمل معرّفه من القاطن المصدر – لأن التصدير لم يستطع تنزيل الملف، أو لم يستطع الاستيراد رفعه – (بإفراغه، أو إسقاطه من مصفوفة، أو إزالة الصورة المصغّرة)، ويُسجَّل **تحذير استيراد**. أمّا معرّف الملف الذي يمكن حلّه بالفعل في تخزين الهدف، كما في الاستيراد داخل القاطن نفسه، فيُحتفظ به.
- **الصفحة التي تشير قطعتها إلى خارج الحزمة تُثبَّت مع ذلك.** تُترك قطعة الاستعلام الإحصائي في صفحةٍ لم يُخطَّط مُدخل بياناتها **غير مُعدّة** – وهي الحالة نفسها التي ينتجها محرّر القطعة – بدل إفشال الصفحة بالرمز `UNRESOLVED_PAYLOAD_REFERENCES`. أمّا عُقد التنقّل ولوحات المعلومات فلا يزال استيرادها يفشل صراحةً.

المراجع المصفَّرة لأصحاب الصلاحيات تغادر المصفوفات وتنحاز إلى المنع `Since 1.4` . حين يصفّر الاستيراد مرجع قيمة يقع داخل مصفوفة مراجع – جمهور خدمة من المستخدمين المصادَق عليهم مثلًا – يُحذف العنصر من المصفوفة بدل إبقائه خانة `null` . وإذا أدّى ذلك إلى إفراغ قائمتي المستخدمين والمجموعات في قاعدة إعادة توجيه لمعرض، تصبح القاعدة `Restricted` فلا تطابق أحدًا، بدل أن تشمل كل مستخدم مصادَق عليه. وتسري القاعدة على كل خط إصدارات مدعوم (1.4 و1.6 و1.8 وما بعدها)، وتُصلح عملية ترحيل في data-service القواعد التي كانت تتضمّن `null` من قبل.

حلّ التبعيات وإعادة التخطيط

أصعب ما في الاستيراد هو تحويل معرّفات UUID في بيئة المصدر إلى مراجع صالحة في بيئة الهدف. ويحسب المحرّك مسبقًا معرّفًا في الهدف لكل مورد سينشئه، مما يتيح إعادة كتابة المراجع حتى قبل إنشاء المورد المشار إليه فعليًا، وإجراء الاستيراد في تمريرة واحدة مرتّبة حسب التبعيات. وللتبعيات درجتان: التبعيات **الصلبة** يجب أن توجد كي يكون المورد صالحًا (فالمعرض يعتمد على كيانه)؛ وهي تُتتبَّع عند التصدير وتحكم الاستيراد. أمّا التبعيات **الليّنة** فاختيارية، تُسجَّل ولا تُتتبَّع عند التصدير، وتُصفَّر عند الاستيراد إن تعذّر حلّها. وصيغة الحافة التي يقرؤها المحرّك موثَّقة في مخطط التبعيات.

مراجع القيم `Since 1.8`

مرجع القيمة معرّف UUID على مستوى المُدخل، مضمَّن في حقل مورد ويشير إلى مُدخل في كيان أساسي أو كيان آخر (مثل تفاعل يشير إلى مُدخل USER أو ORGUNIT بعينه، أو إلى عنصر DataCollection). ويسجّل `ValueReferenceItemModel` قيمتَي `entryId` و `entityType` ، والمواضع (كل مسار يظهر فيه)، وما إذا كان `existsInTarget` ، وأي مرشّح لمطابقة الهوية (`existing`)، وإجراء `ValueReferenceAction` المختار:

الإجراء	الأثر عند الاستيراد
`KEEP`	إعادة استخدام المُدخل نفسه (صالح حين يوجد في الهدف).
`REMAP`	توجيه المرجع إلى مُدخل مختار في الهدف (`remapTargetId`).
`NULL`	مسح المرجع.
`AUTO_CREATE`	إعادة إنشاء المُدخل في الهدف من `entryData` المحزَّمة (الأب أولًا في المُدخلات المتداخلة).

لا تُنشأ مُدخلات `AUTO_CREATE` إلّا حين *يُكتب فعليًا مورد واحد على الأقل يشير إليها* (CREATE/OVERWRITE)؛ أمّا المراجع التي لا تستخدمها إلّا موارد متخطّاة أو معيَّنة فتُوسم بالحالة `skipped` ("not needed"). وتُنشأ مُدخلات GROUP في كيانها المصدر الفعلي (`sourceEntityType` ، مثل `ROLE` و `ORGUNIT` و `USERS_GROUP`).

ويُعيد `GET /v1/import/{sessionId}` حمولة `entryData` المحزَّمة لمرجع القيمة (وأي `sourceEntryData`) كاملةً، ولا تُحذف هذه الحمولة من الجلسة أبدًا. وهي ما يتيح خيار **إنشاء تلقائي** أصلًا: إذ يستند إليها المعالج لعرض الإجراء، وإظهار محرّر حمولة المُدخل المضمّن، وحساب شارات المراجع المتداخلة. أمّا المرجع الذي يصل دون حمولة محزَّمة فلا يمكن إلّا إبقاؤه، أو إعادة تعيينه، أو مسحه.

يجب أن تكون الأهداف نشطة. يحتفظ فهرس المُدخلات عمدًا بالمُدخلات المؤرشَفة، ولذا لا يكفي الوجود وحده:

- يجب أن يكون هدف `REMAP` **نشطًا**: يُرفض بالرمز `400` حفظُ إعداد يوجّه مرجعًا إلى مُدخل ليس `Active` ، بما يتوافق مع ما يعرضه منتقي المُدخلات في المعالج نفسه. ويكشف هذا الفحص الهدف الذي أُرشف *بعد* أن أدرجه المنتقي، أو بعد آخر تأكيد له من ملفّ تعريف تخطيط مخزَّن. أمّا المُدخل الذي لا تُذكر حالة دورة حياته إطلاقًا فيبقى مسموحًا.
- ويخضع `KEEP` التلقائي (الناتج عن `existsInTarget`) للمعيار نفسه: فالعثور على المُدخل مؤرشفًا في الفهرس يُعامَل بوصفه غيابًا، فينتقل المرجع إلى مرحلتي ملفّ التعريف والمطابقة بالمفتاح الطبيعي بدل إبقائه على هدف مؤرشف يتعذّر حلّه.

حقول الربط: المثبِّت يقرّر `Since 1.10`

حقل الربط يعكس قيمةً من كيان أب، كبند طلب يعرض عميل الطلب مثلًا. وهو يسجّل معرّف الحقل الذي ينسخ منه على الكيان الأب، لكن المعرّفات لا تبقى صالحة بعد الانتقال بين البيئات: فالكيان الأب يُطابَق عادةً مع كيان موجود أصلًا في الهدف، فيحتفظ الأب بمعرّفات الهدف نفسها، ولا يشير المعرّف المحزَّم إلى شيء هناك. ويعالج الاستيراد ذلك أولًا بالمطابقة على اسم الحقل بدلًا من معرّفه، إذ يبقى الاسم ثابتًا عبر البيئات. ويُنقل الاسم في الملفّ الجانبي `field-mapping-hints.json` الذي أضافه المخطط 1.2؛ ومن دونه لا يستطيع الاستيراد قراءة الاسم إلّا من نسخة الكيان الأب نفسها داخل الحزمة.

وحين تفشل مطابقة الاسم، **يقرّر المثبِّت** مصدر النسخ للحقل: *النسخ من* حقل على الأب الهدف، أو *ألغِ النسخ*، ولا يبدأ التشغيل حتى يُحسم كل حقل من هذا النوع. ولا تُعرض إلّا الحقول من النوع نفسه، ذات أهداف الارتباط المتوافقة، وذات الإلزامية المطابقة في الربط ثنائي الاتجاه، وقد تكون القائمة فارغة. ويخصّ القرار الأب الهدف الذي اتُّخذ له: فإذا غيّرت الهدف الذي يُخطَّط إليه الأب، أُسقط القرار الذي لم يعد مناسبًا. ولا يستطيع التثبيت من المتجر أو التحديث الذي يجري دون المعالج اتخاذ القرار نيابةً عنك: فحقل الربط غير المحسوم يُعاد ضمن الخيارات التي عليك حسمها في صفحة الاستيراد.

أما قيم `state` لكل حقل، وقواعد المرشّحين، وحمولة القرار، وحالات رفض التنفيذ (`MAPPED_FIELD_SOURCE_UNDECIDED`، `MAPPED_FIELD_SOURCE_UNAVAILABLE`) فموضَّفة في واجهة الاستيراد.

التبعيات الخارجية، مشروحة `Since 1.10`

الحزمة التي تشير إلى مورد لا تتضمّنه، عادةً كيان تملكه وحدة أخرى، تسرده بوصفه **تبعية خارجية**، وتذكر أيضًا **السبب**. فكل عنصر من هذا النوع (في بيان الإصدار المنشور، وفي التبعيات غير المخطَّطة التي يراها المثبِّت) يحمل `usedBy`، أي ما يستخدمه في الحزمة: حقل **يشير** إليه، أو حقل ربط **ينسخ** أحد حقوله (مع تسمية حقل الأب المنسوخ)، أو استخدام **آخر** بلا حقل، كمعرض يشير إلى الكيان، ويحمل أيضًا `requiredFields`، أي الحقول التي يجب أن يوفّرها الكيان الهدف كي تعمل عمليات النسخ تلك. وتعرض هذه المعلومات شاشة النشر ومعالج التثبيت («يستخدمه …»، «الحقل *field* ينسخ منه قيمة الحقل *source*»)، فيستطيع المثبِّت اختيار الكيان الصحيح لتخطيط التبعية إليه قبل حسم الحقول المنسوخة أعلاه.

وحين يرفض النشر معرّفًا متروكًا داخل قيمة مكتوبة (`PUBLISH_DEPENDENCY_REFERENCE_UNSUPPORTED`)، تحدّد رسالة المشكلة نوع المورد ومعرّفه واسمه، وتحدّد الحقل أيضًا حين يتعلّق الأمر بتعيين قيمة لحقل، كما في التفاعل.

قطع الصفحات التي تشير إلى خارج الحزمة `Since dev`

مرجع قطعة الصفحة (مصدر قطعة عرض البيانات أو هدفها، وكل خيار آخر للقطعة يقرؤه مخطط التبعيات) حافة **ليّنة**. وفي خط إصدارات 1.10، لا يعلن النشر إلّا الأهداف الخارجية الصلبة، ويصفّر الاستيراد هذا المرجع الليّن دون طلب قرار، فتُثبَّت القطعة معطوبة. أمّا في خط التطوير، فأصبح بالإمكان ربط هذه المراجع:

- **النشر** يعلن أيضًا هدفًا خارجيًا ليّنًا ضمن التبعيات الخارجية للإصدار حين يكون أحد استخداماته **مرجعًا من قطعة صفحة**: يكون المورد المستخدِم صفحة (`GenericView`)، ويقع المرجع تحت إحدى قطعها (`components[n]` أو `groups[n]`، لا تحت إعدادات ظهور القطعة)، ولا يكون الهدف وحدة. ويُعلَن **دون أن يصير تبعية حزمة**.
- **تحقّق التثبيت** يعرض ربطًا لكل مرجع خارجي صلب **أو** من هذا النوع. ويحمل صفّه في `unmappedDependencies` القيمة `hard: false` حين تكون كل استخداماته مراجع من قطع الصفحات (وغياب الحقل `hard` يعني أنه صلب). وتبقى تبعية الصفحة نفسها عليه ليّنة، فلا تتغيّر أعداد التخطّي التسلسلي والأثر. ولا يُقترح بالاسم هدفٌ ليّن محصور في نطاق كيان (كالمعرض)، لأن المرجع لا يذكر الكيان الذي ينتمي إليه، والأسماء من هذا النوع فريدة داخل الكيان فقط؛ أمّا الهدف الموجود بالمعرّف نفسه في هذه البيئة فيبقى مقترحًا.
- **إذا رُبط** أشارت القطعة إلى المورد المختار. **وإذا تُرك دون ربط** صُفّر مرجع القطعة. وتَسِم صفحة الاستيراد الصفّ بأنه *اختياري* (وتلميحه: «تستخدمه عناصر الصفحات فقط. إذا تركته دون ربط، تُفرَّغ هذه العناصر ويستمر الاستيراد.») ولا

تحجب **التنفيذ** بسببه. ولا يُعرض أبدًا ربط لمرجع قطعة إلى وحدة، ولا تتغيّر مراجع الصفحة الليّنة الأخرى: المشاركة، والنطاق، وأصحاب الصلاحيات في إعدادات الظهور.

> ⚠ **فحص التنفيذ في الخادم غير متوافق بعد**
>
> في أحدث نسخة من خط التطوير التي استندت إليها هذه المراجعة (الواجهة الخلفية `1.11.0-alpha.4`)، لا يقرأ فحص التنفيذ في الخادم ولا قائمةُ الخيارات غير المحسومة في المتجر الحقلَ `hard` بعد: فالصفّ الاختياري غير المربوط الذي سيكتب الاستيراد صفحته لا يزال يُرفض عند التنفيذ (`400`، «All unmapped dependencies must be resolved before execution»)، ولا يزال يُعاد في تثبيت المتجر بوصفه خيارًا غير محسوم. وإلى أن يتحقّق هذا التوافق، اربط الصفّ للمتابعة.

مالكو خطوات سير العمل المطلوبون `Since 1.8`

يجب أن يكون لخطوة سير العمل التي يكون مالكها **ثابتًا** مالك رئيسي؛ إذ يرفض data-service الكيان الذي يكون فيه `workflows[n].steps[n].owner.source.value` فارغًا. ويفرض التحزيم القاعدة نفسها عند الدخول، فيُحجب أي مرجع قيمة يؤدّي إجراؤه إلى **مسح** ذلك المسار (`NULL`) على مورد سيكتبه الاستيراد فعلًا (`CREATE` أو `OVERWRITE`). أمّا `fallbackOwner[n].source.value` للخطوة فمستثنى عمدًا، لأن المالك الاحتياطي اختياري.

ويُطبَّق هذا الفحص عند ثلاث نقاط، وتُقيَّم عند كلٍّ منها الحالة **الفعلية** بعد التغيير الذي تجريه، لا التغيير بمعزل عنها:

الموضع	ما تراه
التحقّق (المرحلة 1)	خطأ حاجب `REQUIRED_OWNER_REFERENCE_CLEARED` في تقرير التحقّق، مع `canProceed: false`.
حفظ الإعداد	يُرفض الحفظ التلقائي بالرمز `400`. ويشمل الفحص كذلك الحفظ المقتصر على التخطيطات حين يعيد تفعيل مورد سبق أن مُسح مالكه.
تطبيق استراتيجية جماعية	يُرفض بالطريقة نفسها التحويلُ الجماعي من **تخطي** إلى **إنشاء/استبدال** الذي يجعل المالك الممسوح مؤثّرًا من جديد.

ولرفع الحجب، حدّد للمالك هدفًا حقيقيًا: **إعادة تعيين** المرجع إلى صاحب صلاحية موجود في المساحة الهدف، أو **إنشاء تلقائي** له من الحمولة المحزَّمة. ولا يبقى **إبقاء** و**مسح** خيارين بعد تفعيل هذا الفحص. كما أن إعادة المورد المالك إلى **تخطي** ترفع الحجب أيضًا، إذ لا يشمل الفحص إلّا الموارد التي سيكتبها الاستيراد، ولذا يُسقَط الخطأ الذي لم يعد قائمًا من التقرير ويُعاد حساب `canProceed`.

إعادة توجيه مرجع قيمة `Since 1.8`

تحكم قاعدتان ما يرسله معالج الاستيراد حين يتغيّر هدف مرجع، وتظهر أهميتهما خصوصًا حين يكون ملفّ تعريف تخطيط مخزَّن قد ملأ الصفّ مسبقًا:

- **الهدف الذي تختاره يعلو على افتراضي الصفّ.** فاختيار هدف محدّد يُرسَل بوصفه إعادة توجيه صريحة حتى لو كان `REMAP` هو الإجراء الافتراضي للصفّ أصلًا. والمرجع المستعاد من ملفّ تعريف مخزَّن يُعاد تحميله وقد ضُبط إجراؤه *وافتراضيه* كلاهما على `REMAP`، ولذا يجب مع ذلك إرسال إعادة توجيه الهدف وحده، وإلّا بقي هدف ملفّ التعريف القديم.
- **أهداف كل موضع تبقى بعد الحفظ.** فحين يظهر صاحب صلاحية واحد في مسارات عدّة (المالك الاحتياطي نفسه على عدّة خطوات سير عمل مثلًا)، يمكن تجاوز الهدف لموضع واحد بعينه. واختيار هدف لموضع واحد يسجّل أيضًا إجراء الصفّ على ذلك الموضع، فيكتمل التجاوز بما يكفي لإرساله؛ أمّا إعادة الهدف إلى افتراضي الصفّ فتُسقط التجاوز، ويعود الموضع إلى وراثة قرار الصفّ. وإعادة توجيه موضع واحد تُرقّي كذلك المورد المالك من **تخطي** إلى **استبدال**، لأن تغيير المرجع لا يسري إلّا إذا كُتب المورد فعلًا.

ويُوسَم المرجعُ الذي جاء قراره من ملفّ تعريف تخطيط مخزَّن في المعالج بما يدلّ على ذلك: شارة **ملف التعيين** بدل شارة المطابقة بالمفتاح الطبيعي، ويحمل شارة تحذير *التعيين المحفوظ غير متاح* حين يتعذّر حلّ الهدف المحفوظ.

إعادة تخطيط صاحب صلاحية مشترك لكل مورد على حدة `Since 1.8`

حين يُشار إلى صاحب الصلاحية **نفسه**، سواء أكان مستخدمًا واحدًا أم مجموعةً أم وحدةً تنظيميةً، من أكثر من **مورد** في الاستيراد ذاته (كمالك خطوة سير عمل على كيان، والمستخدم الذي يقتصر عليه معرض)، يمكن إعادة توجيه نسخة كل مورد من ذلك المرجع على حدة. فالمعالج يرصد ظهور صاحب الصلاحية تحت أكثر من صفّ مورد، ويجعل مجموعة كل مورد تقرأ مواضعها وتكتبها وحدها. أمّا صاحب الصلاحية المحصور في مورد واحد فيظلّ يستخدم قرار الصفّ، وهو أيضًا ما يحفظه ملفّ تعريف التخطيط المخزَّن.

يُحتفَظ بافتراضي الصفّ حين تنحرف عنه إحدى المجموعات. فالاختيار على مستوى الموضع ليس إلّا تجاوزًا *جزئيًا* يُطبَّق فوق قرار صاحب الصلاحية على مستوى الصفّ، والخادم يرفض إعادة توجيه على مستوى العنصر لا تحمل هدفًا خاصًّا بها. ولذلك فإن أول هدف يُختار لصاحب صلاحية مشترك **يُرسي كذلك افتراضي الصفّ**: ثم تختار مجموعة لاحقة لصاحب الصلاحية نفسه هدفًا يختلف *عن* ذلك الافتراضي بدل أن تحلّ محلّه. وهذا ما يحافظ على الاختيار الذي تتّخذه في **الإعداد** حتى **التنفيذ**، بما في ذلك لصاحب صلاحية لم يكن له بعدُ هدف على مستوى الصفّ.

أمّا **إنشاء تلقائي** لصاحب صلاحية مشترك بين عدّة موارد فيُطبَّق دائمًا على مستوى الصفّ، لا بوصفه تجاوزًا لكل موضع، لأن مفتاح المُدخلات المُنشأة تلقائيًا هو صاحب الصلاحية نفسه، لا موضع ظهوره.

التخطّي التسلسلي والدورات وتصفير التبعيات الليّنة

- **التخطّي التسلسلي** – إذا تُخطّيت تبعية صلبة أو فشلت، فإن كل الموارد التي تعتمد عليها اعتمادًا صلبًا تُتخطَّى تسلسليًا (`cascade_skipped`) قبل أي كتابة.
- **كسر الدورات** – تُكتشف دورات التبعيات الصلبة بين كيانات `DataEntity` (SCC) وتُكسر: يُنشأ أحد الأعضاء في صورة هيكل مبدئي عند CREATE، ثم يُحدَّث عبر PATCH بالمراجع الحقيقية العابرة للدورة متى وُجد الأعضاء جميعًا.
- **تصفير التبعيات الليّنة** – يُضبط مسار التبعية الليّنة التي يتعذّر حلّها على `null` بدل ترك معرّف UUID معطوب من المصدر فيه (مع استثناءات خاصة بالنوع، مثل ترك `dataSource.query` كما هو كي يُظهر data-service خطأً أوضح). و `Since dev` يمكن بدلًا من ذلك ربط مرجع قطعة الصفحة بمورد خارج الحزمة.
- **صفّ إعادة المحاولة** – يُعاد تنفيذ الموارد التي تفشل برمز «هدف تبعية مفقود» (`DATA_QUERY_NOT_EXIST` ، `DATA_ENTITY_NOT_EXIST` ، `DATA_VIEW_NOT_EXIST` ، `DATA_COLLECTION_NOT_EXIST` ، `RELATION_NOT_EXIST`) مرة واحدة بعد الحلقة الرئيسية، بعد أن يكون اكتشاف الأبناء الضمنيين قد ملأ جدول المعرّفات.

الترتيب الطوبولوجي

يعيد كلٌّ من التصدير والاستيراد ترتيب الموارد باستخدام `topologicalSort` (خوارزمية Kahn على الحواف الصلبة) كي تُنشأ الكيانات الآباء قبل الأبناء (مثل `DataEntity` قبل `DataView` التابع له). ويعيد الاستيراد الترتيب حتى لو كانت الحزمة أقدم أو غير مرتّبة.

مخطط dependency-service

تتولّى `dependency-service` (`/api/dependency`) إدارة مخطط التبعيات، ويستعلم عنه التحزيم أثناء التصدير (الإغلاق الصادر، وشجرة المعتمِدين لـ `withChildren`) ويستخدمه عند الاستيراد عبر `dependencies.json` . وتحمل الحواف المخزَّنة `fromType/fromId` ، و `toType/toId` ، و `kind` (`MEMBERSHIP` أو `REFERENCE` ، وهما قيمتا `DependencyKind` الوحيدتان)، و `hard` (يحجب حذف الهدف)، و `path` ، و `resolvedVia` (`DIRECT_ID` / `NORMALIZED_NAME` / `REFERENCE_PROPERTY` /

`PIPELINE_STAGE`)، و`resourceVersion` (بصمة SHA-256 للقطة، لضمان أن تعطي إعادة البناء النتيجة نفسها مهما تكرّرت). أمّا صيغة الحافة عند نقلها عبر الواجهة فموثَّقة في مخطط التبعيات. وفيما يلي نقاط نهاية القراءة:

الأسلوب	المسار	الغرض
`GET`	`/v1/dependencies/{type}/{id}`	الحواف الصادرة (على ماذا يعتمد هذا؟)
`GET`	`/v1/dependents/{type}/{id}`	الحواف الواردة (ما الذي يعتمد على هذا؟)
`GET`	`/v1/validate-delete/{type}/{id}`	المعتمِدون الصلبون مقابل الليّنين (الحذف الآمن)
`GET`	`/v1/impact/{type}/{id}`	تحليل الأثر المتعدّي
`GET`	`/v1/external-dependents/{type}/{id}`	`Since 1.10` . ما الذي يقع *خارج* وحدة أو كيان ويشير إليه أو إلى أي شيء يملكه – القائمة التي تعرضها مربّعات حوار حذف الوحدة وحذف الكيان والإزالة

تُستخرج الحواف عبر لغة تصريحية مخصّصة لتعريف المخططات، ويُعاد بناؤها استجابةً للأحداث (أحداث دورة الحياة في RabbitMQ ← عمّال BullMQ).

ملفّات تعريف التخطيط المخزَّنة `Since 1.8`

بعد نجاح الاستيراد، تُحفظ تخطيطات **أصحاب الصلاحيات** المؤكَّدة (USER/GROUP/ORGUNIT) في ملفّ تعريف من نوع `MappingProfile`، مفتاحه `(sourceEnvironmentId, sourceSpaceId, targetSpaceId)` . ويطبّق أيّ استيراد لاحق من المصدر نفسه تلك القرارات مسبقًا، قبل المطابقة بالمفتاح الطبيعي، فتُبنى عمليات الترقية المتكرّرة على الخيارات السابقة.

القرارات التي تُحفظ:

- **قرار واحد على الأكثر لكل صاحب صلاحية في المصدر.** لا يُحفظ صاحب الصلاحية الذي أُعيد تخطيطه لكل مورد على حدة واختلفت مواضعه بوصفه `REMAP` (وإلّا لأُعيد تطبيق الاختيار المخالف على كل موضع في المرة التالية). ولا يتأثّر `AUTO_CREATE` الناجح ولا المسح بقيمة `NULL` على مستوى العنصر، إذ لا يمثّل أيّ منهما قيمة مُرقّاة من موضع.
- **يُتذكَّر `REMAP` المؤكَّد** `Since 1.10` . فالتجاوز الفارغ على مستوى الموضع يُعَدّ غيابًا للتجاوز، فلا يُظهر صاحب الصلاحية بوصفه مخالفًا؛ ويُطبَّق مسبقًا التخطيط الذي أكّدته لصاحب الصلاحية في الاستيراد التالي من المصدر نفسه. (وفي 1.8، كان حفظ الإعداد يخزّن كل تجاوز غير مضبوط بوصفه قيمة فارغة صريحة، فلم يكن يصل إلى ملفّ التعريف عمليًا إلّا أصحاب الصلاحيات المُنشؤون تلقائيًا.)

واجهة الإدارة البرمجية (`mapping-profile.controller.ts`، يقتصر استخدامها على مسؤول المساحة، ويحدّد نطاقها `x-space-id`):

الأسلوب	المسار	الغرض
`GET`	`/v1/admin/mapping-profiles/count`	عدّ ملفّات التعريف في المساحة
`GET`	`/v1/admin/mapping-profiles`	سرد ملخّصات ملفّات التعريف
`DELETE`	`/v1/admin/mapping-profiles/{profileId}`	حذف ملفّ تعريف كامل
`DELETE`	`/v1/admin/mapping-profiles/{profileId}/entries/{sourcePrincipalId}`	حذف عنصر محفوظ واحد

`MappingProfileEntry`: `sourcePrincipalId`، و`action` (`remap` افتراضيًا / `clear`)، و`targetPrincipalId?`، و`entityType`، و`matchedBy?`، و`confidence?`، و`updatedAt`. ويسري الحذف في جلسة الاستيراد التالية. وإذا كان معرّف ملفّ التعريف يخصّ مساحةً أخرى أو كان الملفّ غير موجود، تُعاد `404` (فلا يُكشف أبدًا عن وجوده في مساحة أخرى).

وقبل إعادة تطبيق `targetPrincipalId` محفوظ، تتحقّق أسس من أن صاحب الصلاحية ما يزال `Active` – أي من كونه نشطًا، لا من مجرّد وجوده في الفهرس. فالعنصر المحفوظ الذي **أُرشف** هدفه لاحقًا يُعامَل بوصفه متقادمًا، تمامًا كالعنصر الذي حُذف هدفه: فلا يُطبَّق بوصفه إعادة توجيه نشطة، وينتقل المرجع إلى المطابقة بالمفتاح الطبيعي، ويُعلَّم الصفّ (`profileTargetMissing`) مع تحذير `PROFILE_TARGET_MISSING` كي يتمكّن المعالج من وسمه بوضوح. وخلافًا للهدف المحذوف، لا **يُحذف** الهدف المؤرشف من ملفّ التعريف المخزَّن – فالأرشفة قابلة للتراجع، ولذا يبقى القرار المحفوظ ويسري من جديد فور استعادة صاحب الصلاحية. ولا تُسقَط إلّا العناصر التي اختفى هدفها من الفهرس تمامًا.

المجموعات في التحزيم

`DataCollection` نوع قابل للتصدير، لكنه يحتاج إلى معالجة خاصة:

- تُصدَّر **المجموعات غير ADT (المدعومة بكيان)** مع `items: []` – فعناصرها معرّفات UUID لمُدخلات `DataEntry` لا تتضمّنها الحزمة، والإبقاء عليها يُنشئ مراجع معلّقة. أمّا **مجموعات ADT** (التي يكون `dataEntity` فيها ثابتًا أوليًا من ADT مثل `ADTString`) فتحتفظ بعناصرها العددية المضمّنة.
- المجموعات **مقيّدة بنطاق**: نطاق الكيان حين يكون `dataEntity` معرّف UUID لكيان حقيقي، و**النطاق العام** حين يكون ثابت ADT. ويحدّد النطاق كيفية مطابقة المورد الموجود عند التحقّق، ومفتاح البحث داخل ZIP.
- ليس للمجموعات العامة/ADT **حافة عضوية في وحدة** يمكن تتبّعها صعودًا، ولذا تُعرض أثناء تحليل تبعيات التصدير ضمن مجموعة اصطناعية باسم «Global» بدلًا من إسقاطها.
- تُعامَل **عناصر** المجموعة التي تشير إليها موارد أخرى بوصفها مراجع قيم مع `entityResourceType = DataCollection`؛ وتستخدم فحوص الوجود و`AUTO_CREATE` واجهات DataCollection البرمجية بدلًا من واجهات فهرس المُدخلات.

آلة حالات الاستيراد

تمرّ جلسة الاستيراد بالحالات **validating › validated › importing › completed** (أو **failed**). ولا يمكن تنفيذها إلّا مرة واحدة – إذ تُحسم محاولات التنفيذ أو إعادة المحاولة أو الإلغاء المتزامنة ذرّيًا. وتُعاد الجلسة العالقة في **importing** بعد انهيار أحد العمّال إلى **failed** كي تتمكّن إعادة المحاولة من تحرير حجزها. ولا تعيد المحاولة معالجة إلّا الموارد الفاشلة والمتخطّاة تسلسليًا. أمّا جدول الانتقالات الكامل ونافذة الاستعادة ففي واجهة الاستيراد.

شاشات التحزيم `Since 1.10`

تضمّ قائمة **الحزم** أربع شاشات: **المتجر** و**المثبتة** و**البناء** و **المراجعات**. وتُوثَّق كيفية استخدام باني التطبيقات لها في تصدير التطبيقات واستيرادها وتثبيتها [بناء التطبيقات]؛ أمّا مساراتها وقواعد ظهورها ففي صفحة المتجر. وتهمّك السلوكيات التالية عند البناء على هذه الشاشات، أو اختبارها، أو دعم مستخدميها.

صفحة البناء وصفحة الحزمة

- لصفحة **البناء** (`/admin/packages/build`) تبويبان: *الحزم* (قابلة للفرز بحسب أيّ عمود؛ ويبقى زرّ **حذف** الحزمة ظاهرًا لكنه معطّل، مع بيان السبب، ما دامت للحزمة إصدارات) و*التصديرات* (`/admin/packages/build/exports`، سجلّات التصدير). وتُعيد العناوين القديمة `/admin/packages/mine/...` التوجيه إلى `/build/...`.
- تتيح **صفحة الحزمة** (`/admin/packages/build/{packageId}`) تحرير الاسم والوصف في موضعهما (Enter أو Ctrl/Cmd+Enter للحفظ، Escg للإلغاء). ويصدّر زرّ **تنزيل**، الموجود في الترويسة بجانب *عرض في المتجر*، وحدات الحزمة

إلى ملفّ، ويظهر للمطوّرين ما دامت الحزمة تضمّ وحدةً لا تزال موجودةً. أمّا **إلغاء النشر** و**إعادة النشر**، و**حذف** لحزمة بلا إصدارات، فتوجد في قائمة ⋯.

- **المحتويات** (التبويب الافتراضي) قائمة واحدة بما سيحويه الإصدار التالي – *الوحدات* و*سجلات مُشار إليها* و*الصفحات* و*يعتمد على* – ويبيّن كل صفّ ما تغيّر منذ أحدث إصدار، مع سهم يفتح العناصر المتغيّرة. ويبيّن صفّ الوحدة أو الصفحة المحذوفة الإجراء المطلوب. أمّا *سجلات مُشار إليها* فهي السجلات التي تشير إليها إعدادات الحزمة (المجموعة المالكة لخطوة في سير عمل مثلًا)، وتحمل الحزمة نسخةً منها.
- يأتي **README** في نهاية قائمة المحتويات ويُعلَّم حين يتغيّر. ويُحوَّل Markdown الملصوق في محرّر README بوصفه نصًّا عاديًا – عناوين، وأسوار رماز، واقتباسات، وجداول، وخط عريض، وروابط، ورماز مضمّن، أو سطران فأكثر من قائمة – إلى نص منسَّق في خطوة تراجع واحدة. و`Since 1.10` (من 1.10.2)، يوضّح القسم، قبل إنشاء أيّ إصدار، أن الإصدار التالي سيحتفظ بنسخة من ملف README هذا – أو ينبّه إلى ضرورة إضافته أولًا، عند التأكّد من غيابه. و`Since dev` تُنسخ صورة الويب الملصقة إلى الحزمة، وترد تفاصيل السلوكين في صفحة المتجر.
- **التعديلات المتزامنة** `Since dev`. ترسل الصفحة رمز تحديث الحزمة مع كل حفظ للمسوّدة أو لملف README، فيُرفض الحفظ الصادر من تبويب أقدم، أو الذي يتعارض مع حفظ أحدث أجراه شخص آخر («تم تغيير هذا في تبويب آخر أو من مستخدم آخر منذ فتحته…»)، بدلًا من الكتابة فوق الحفظ الأحدث – انظر رفض الحفظ المتقادم.
- **الإصدارات.** يوجد زرّ *إنشاء إصدار* في هذا التبويب؛ ويُغلق الحوار عند إنشاء الإصدار، لأن إرساله للمراجعة قرار منفصل. ويمكن **حذف** الإصدار الذي لم يتلقَّه المتجر قط؛ أمّا الإصدار الذي أُرسل للمراجعة أو ثُبّت في مكان ما فيُرفض حذفه (`VERSION_SENT`، `VERSION_INSTALLED`). ويزيل حذف الإصدار **سجلّه** – ويبقى ملف ZIP في التخزين – ولا يُعاد استخدام رقمه أبدًا. وتُظهر *مقارنة الإصدارات* أيضًا متى أُضيف ملف README أو أُزيل أو حُدِّث. ولا يوجد إجراء *إيقاف* (إذ كان يقتصر على تعليم الإصدار في هذه البيئة)؛ ويبقى *إلغاء الإيقاف* متاحًا للإصدارات الموقوفة سابقًا.
- **الشريط الجانبي للسجل.** يسجّل الإنشاء، والتعديل (الاسم، والوصف، والوحدات والصفحات المضافة أو المزالة، ونطاقات التبعيات)، وتعديل README، والإصدارات المُنشأة والموقوفة، والمُلغى إيقافها، والمحذوفة، إضافةً إلى أحداث المراجعة – الإرسال، والموافقة، والإعادة، وإدراج صفحة المتجر أو رفضها.
- **إعادة التحميل الحيّة.** يُعاد تحميل الصفحة حين ترسل المنصّة حدث `packages/changed` للحزمة المفتوحة (قرار مراجعة مثلًا)، ومرةً أخرى حين يعود التركيز إلى تبويب المتصفّح.
- **الصفحات العامة.** يسرد تبويب *الصفحات* الصفحات العامة للحزمة – أي الصفحات التي لا تنتمي إلى وحدة ولا كيان. ويجوز أن تضمّ الحزمة صفحات فقط. ولا يمكن نشر الصفحة التي تنتقل لاحقًا إلى وحدة أو كيان (`PUBLISH_PAGE_NOT_GLOBAL`)، وكذلك الصفحة التي ثُبّتت أو اشتُقّت من حزمة أخرى.

قائمة المثبتة وصفحة التثبيت

- **قائمة المثبتة** (`/admin/packages/installed`): تتضمّن قائمة *الحالة* (المثبتة، والفاشلة، والمُزالة) وقائمة *المصدر* بجانب البحث والترتيب؛ وتبدأ القائمتان فارغتين، فتظهر كل الحالات والمصادر. وتشمل *المثبتة* التثبيتات الفاشلة والجارية، فيظهر التثبيت الفاشل تحت *المثبتة* و*الفاشلة* معًا. ويفتح الصفّ المرتبط بتثبيت قائم صفحة ذلك التثبيت؛ كما يعيد سجلّ الاستيراد المرتبط بتثبيت قائم التوجيه إليها.
- **صفحة التثبيت** (`/admin/packages/installed/installations/{installationId}`): يُستمدّ المصدر والأصل ووقت التصدير والنتيجة من أحدث سجلّ استيراد يستند إليه التثبيت، مع ملف README الخاص به. ويسرد *السجل* كل تشغيل للتثبيت (مع بيان التشغيل الفاشل)، إضافةً إلى الاشتقاق والإزالة.
- **التثبيتات الفاشلة** تتيح إعادة المحاولة من جلسة الاستيراد ما دامت موجودةً، أو رفع ملف الحزمة نفسه مرةً أخرى ليتولّى التثبيت. وحين يتوفّر إصدار أحدث متوافق، يُعرض *تحديث* بدلًا من ذلك: يكتب فوق ما أنشأه التشغيل الفاشل

ويثبّت الباقي. ويعرض التثبيت الفاشل من المتجر أو التحديث المتوقّف من المتجر، أيضًا **إبلاغ *الناشر*** `Since 1.10` (من 1.10.1) – انظر الإبلاغ عن تثبيت أو تحديث فاشل.

- **الاشتقاق** يتطلّب إدخال اسم الحزمة للتأكيد، ويُرفض ما دامت حزم أخرى تعتمد على التثبيت أو كان هناك تحديث قيد التشغيل. وتُظهر موارد التثبيت المشتقّ «تم التفرع، لذا لم تعد التغييرات على هذه الموارد تُفحص». وتستطيع الصفحة عرض وسم *محذوف* لمورد أُزيل بعد الاشتقاق، لكن الواجهة البرمجية تبلّغ عن هذا المورد بوصفه *مفقودًا*، لذا لا يظهر ذلك الوسم بعد.
- **الإزالة** متاحة للتثبيت النشط والمشتقّ (ويحذّر الحوار من أن تغييراتك على الموارد المشتقّة تُحذف معها). ويبقى الزرّ معطّلًا حتى تُزال جميع الروابط الواردة من **خارج** الحزمة؛ ويفرض الخادم القاعدة نفسها بالرمز `409` `UNINSTALL_BLOCKED_BY_OUTSIDE_LINKS`، فيُرفض أيضًا الاستدعاء الذي يتجاوز الحوار. وتتيح الإزالة المتوقّفة خيار *إعادة المحاولة*.

المعارض والصفحات المثبَّتة في محرّراتها

المعرض أو الصفحة الذي تثبّته حزمة **مقفل** بها. وتكون أقسام الإعدادات والمساعدة والمحتوى غير مفعّلة، ويُرفض الحفظ باختصاره أو بزرّ **تم** أو بزرّ الحفظ في الترويسة؛ ويبقى نشر المعرض أو إخفاؤه متاحًا من قائمة معارض الكيان. ويظلّ تغيير الاسم والوصف والأيقونة متاحًا (يعرض المحرّر قيمة الحزمة بجانب قيمتك، ويُحتفظ بتسميتك في كل تحديث، ويبقى اسم المعرض الضمني للقراءة فقط)؛ وترد قواعد الخادم في معارض البيانات [بناء التطبيقات › معارض البيانات وأوجهها] و الصفحات والقطع [بناء التطبيقات]. وتبقى تجاوزات العلاقات قابلة للاستخدام (فهي تُحفظ على المعرض الأب)، وتكون محرّرات النصوص البرمجية للقراءة فقط مع إمكان تحديد النص، فيبقى رمازها مقروءًا وقابلًا للنسخ.

الحدود التشغيلية والأمان

الحارس	القيمة	الموضع
الحدّ الأقصى للحزمة بعد فكّ الضغط	200 MB	`MAX_DECOMPRESSED_SIZE`
الحدّ الأقصى لمُدخلات ZIP	10 000	`MAX_ZIP_ENTRIES`
اجتياز المسارات / المُدخلات غير الآمنة	مرفوضة	`isSafeEntryName` ، `SAFE_PATH_COMPONENT`
رسالة خطأ الجلسة المحفوظة	بحدّ أقصى 2 000 حرف	`MAX_ERROR_MESSAGE_LENGTH`
مدّة بقاء الجلسة	7 أيام	`import-session.model.ts`
استعادة الاستيراد العالق	15 دقيقة في `importing`	`IMPORT_STUCK_THRESHOLD_MS`
تزامن التصدير والاستيراد	عمّال BullMQ (Redis)	`IMPORT_QUEUE_NAME` / `EXPORT_QUEUE_NAME`
وتيرة فحص الإلغاء	كل 5 موارد مُعالَجة	`CANCEL_CHECK_INTERVAL`

يمكن تجاوز إعدادات انتظار الأبناء الضمنيين واستقصائهم عبر متغيّرات البيئة (`PKG_IMPLICIT_SETTLE_MS` ، `PKG_IMPLICIT_POLL_ATTEMPTS` ، `PKG_IMPLICIT_POLL_DELAY_MS`) – وتتيح هذه الإعدادات لعملية الاستيراد انتظار data-service ريثما يولّد المعارض والاستعلامات الضمنية بصورة غير متزامنة بعد إنشاء كيان (CREATE).

القيود والمزالق

- **الحواف الصلبة وحدها.** يُضمن انتقال المراجع الصلبة وحدها مع الحزمة؛ أمّا المراجع اللّيّنة فقد تُصفَّر عند الاستيراد – فراجع تحذيرات تحليل التصدير، وأعد توجيهها بعد ذلك إن لزم. و `Since dev` ، يُعرض عند التثبيت ربطُ مرجع قطعة الصفحة بوجهة خارج الحزمة (قطع الصفحات التي تشير إلى خارج الحزمة).

- **أسرار الوظائف لا تنتقل** `Since dev`. تحمل الوظيفة المحزَّمة أسماء أسرارها فقط؛ فاضبط القيم في كل بيئة هدف (أسرار الوظيفة تبقى في بيئتها).
- **نادرًا ما يتطابق تخطيط أصحاب الصلاحيات تلقائيًا** بين البيئات – فاستخدم ملفّات تعريف التخطيط واقتراحات المطابقة التامة لتخطيط المستخدمين والمجموعات.
- **التخطّي التسلسلي عند الفشل.** يكفي فشل تبعية صلبة واحدة لتُتخطّى الموارد التي تعتمد عليها. فأصلح السبب الجذري و**أعد المحاولة**.
- **القوالب تحتاج إلى حقولها.** يفشل عرض القالب المُصدَّر دون كيانه (أو الذي لا تتوفّر في الهدف حقوله أو رموزه المشار إليها) بعد الاستيراد.
- **حزم أحدث إلى أهداف أقدم.** يتسامح المحرّك مع بعض الفوارق بين الإصدارات (فهو مثلًا يولّد موارد الأبناء الضمنية التي لا ينشئها الهدف الأقدم تلقائيًا)، لكن الحزمة المبنية على إصدار مخطط أحدث مما يقبله الهدف تُرفض بخطأ عدم توافق المخطط.
- `ValidationStatus` **لا يكون إلا** `NEW` **أو** `EXISTING`. تُذكر قيمة `CONFLICT` في تعليق توثيقي، وتوجد في تطبيق الويب تسمية *تعارض*، لكن مسار التحقّق الحالي لا يُنتجها أبدًا – إذ تُبرَز تصادمات الهوية بصورة منفصلة في تقرير التحقّق.
- **كائنات النقل على مستوى الحقول** معرَّفة في عقود `@osos/shared` وفي بُناة المواصفات في المتحكّمات؛ وتوثّق هذه الصفحات الحقول والأشكال التي جرى التحقّق منها، ولا تشمل كل خاصية اختيارية في كل كائن نقل. ولا تنشر packaging-service ولا dependency-service وصف OpenAPI، ولذا لا تظهران في مرجع نقاط النهاية المولَّد.

توزيع الإضافات الجاهزة

يتيح التحزيم نقل موارد *تطبيقك* بين البيئات. وتأتي أسس أيضًا مزوّدةً بـ **حزم إضافات جاهزة للواجهة الأمامية** (إضافات مخصّصة) توفّر مكوّنات عرض إضافية. وتُوزَّع هذه الإضافات بوصفها **سجلّ مكوّنات ديناميكيًا داخليًا ضمن التطبيق**؛ إذ تُعرِّف كل حزمة إضافة مكوّنات (قطعًا وقوالب) تُحمَّل عند الحاجة وتُسجَّل داخليًا، ليستدعيها التطبيق بأسمائها وقت التشغيل. **الإضافات داخلية فقط – ولا يوجد تحميل عن بُعد أو اتّحادي** – فهي تأتي مدمجةً في بناء التطبيق ولا تُجلب من مصدر خارجي وقت التشغيل، وتُضاف عبر منشئ المعارض العامة. وتختلف هذه الآلية عن آلية تحزيم ZIP المذكورة أعلاه، التي توزّع موارد البيئات لا الرماز المدمج.

> **ملاحظة:** هياكل الإضافات النموذجية الموجودة في مكتبة الإضافات **قوالب فارغة بلا أي وظيفة فعلية** – فهي نقاط انطلاق لبناء إضافتك الخاصة، وليست مكوّنات عاملة.

ذو صلة لكنه متمايز: نقل الوحدات والكيانات بصيغة JSON

يوفّر `data-service` أيضًا آليةً أقدم ومنفصلةً لتصدير واستيراد **الوحدات والكيانات ملفّ JSON واحدًا**. وتختلف هذه الآلية عن نظام تحزيم ZIP المذكور أعلاه، ولا تشترك معه في الرماز أو الصيغة أو نقاط النهاية؛ لذا استخدم التحزيم لنقل تطبيق كامل. أمّا تفاصيل صيغتها ونقاط نهايتها وحدودها، فترد في نقل الوحدة بصيغة JSON.

صيغة الحزمة

توثّق صفحة التحزيم والتوزيع **سير العمل** – مهامّ التصدير، وجلسات الاستيراد، والتخطيطات، والتنفيذ. أما هذه الصفحة فتوثّق **الناتج** الذي تنتجه إجراءات سير العمل تلك وتستهلكه: ما الملفات التي يحتوي عليها ملف ZIP للحزمة تحديدًا، وما معنى كل حقل في البيان، وما الذي تقبله آلية الاستيراد عند حدودها وما الذي ترفضه. وهي موجّهة إلى من يريد فحص حزمة أو تحويلها أو توليدها بدلًا من استخدام شاشة التصدير. وتفصّل صفحتان مرافقتان ما تحيل إليه هذه الصفحة عمدًا: تُحدّد مخطّطات موارد الحزمة شكل الحمولة داخل كل ملف مورد، وتوضّح صفحة توليد حزمة الطرق المدعومة لإنتاج حزمة، مع الموازنة بينها. وقد أُتيح محرّك التحزيم الذي يحدّده هذا العقد ضمن سلسلة إصدارات **1.8** (`Since 1.8`)؛ ولا تتوافر هذه المسارات وواجهة التصدير في عمليات النشر الأقدم.

وقبل الدخول في التفاصيل، ينبغي توضيح أمر يقود السؤال عنه القارئ عادةً إلى هنا: **حزمة أسس ليست ملف JSON واحدًا.** بل هي أرشيف ZIP يضمّ ملفات متعددة يجب أن تتوافق أجزاؤه فيما بينها – إذ يجب أن يسمّي الفهرس كل ملف مورد، وأن تشير حواف التبعيات إلى مسارات فعلية داخل الحمولات، وأن يصف البيان كليهما.

بنية الأرشيف

تتولّى أدوات ZIP في خدمة التحزيم بناء الأرشيف وتحليله. ويُسمح بظهور أربعة عشر صنفًا من المُدخلات في المستوى الأعلى – أربعة منها `Since 1.10` ؛ وأي مُدخل آخر يُفشل التحقّق (انظر المُحقِّق الصارم أدناه):

```
manifest.json              required
index.json                 required
dependencies.json          written by every export; required at import
config.json                echo of the export request
entry-references.json      optional
store-references.json      optional
file-references.json       optional
field-mapping-hints.json   optional, schema 1.2 and later
hashes.json                published versions only, schema 1.3 and later
package-references.json    published versions only, schema 1.3 and later
README.md                  optional, schema 1.4
readme/{image}             optional, schema 1.4
files/{fileId}.bin         one per file reference
resources/{Type}/{normalizedName}.json
```

المُدخل	الوجود	المحتوى
`manifest.json`	مطلوب	البيانات الوصفية للحزمة – انظر أدناه.
`index.json`	مطلوب	قائمة الموارد المرتّبة وخريطة تربط المعرّف باسم الملف – انظر أدناه.
`dependencies.json`	مطلوب عند الاستيراد	حواف التبعيات المُسلسَلة – انظر أدناه. يؤدي غيابه إلى فشل الاستخراج بالخطأ `Invalid package: missing dependencies.json`.
`config.json`	يكتبه كل تصدير	نسخة من طلب التصدير الأصلي: `resources` ، إضافةً إلى `excludes` و `excludeTypes` و `name` و `description` متى ضُبطت. للعلم فقط – لا يستند إليه الاستيراد في سلوكه.
`entry-references.json`	اختياري	`PackageEntryReference[]` – مراجع مُدخلات على مستوى القيمة (مستخدمون أو مجموعات أو عناصر مجموعات بعينها)، مع حمولة `entryData` المحزَّمة التي تتيح إنشاءها تلقائيًا في الهدف.
`store-references.json`	اختياري	`PackageStoreReference[]` – لقطات للمخازن المشتركة (العابرة للمساحات) التي تشير إليها الكيانات المصدَّرة.

المُدخل	الوجود	المحتوى
`file-references.json`	اختياري	`PackageFileReference[]` – بيانات وصفية `{fileId, fileName, mimeType, size}` لكل مرفق قالب/ملف ثنائي.
`field-mapping-hints.json` Since `1.10`	اختياري	الملفّ الجانبي لحقول الربط – وهو الإضافة البنيوية **الوحيدة** في المخطط `1.2` . لا يُكتب إلا حين يكون غير فارغ، ووجوده هو ما يجعل الحزمة تعلن `1.2` . انظر قسم الملفّ الجانبي.
`hashes.json` Since `1.10`	الإصدارات المنشورة	تجزئات المحتوى بصيغته المعيارية لكل مورد، و `rootHash` للحزمة. انظر ملفات الإصدار المنشور.
`package-references.json` Since `1.10`	الإصدارات المنشورة	كل مرجع من هذه الحزمة إلى حزمة أخرى. انظر ملفات الإصدار المنشور.
`README.md` و `readme/{image}` Since `1.10`	اختياري	ملف README للحزمة والصور التي يربط إليها – وهي إضافة المخطط `1.4` . انظر ملفات الإصدار المنشور.
`files/{fileId}.bin`	واحد لكل مرجع ملف	الحمولة الثنائية نفسها. يفشل الاستخراج بالخطأ `missing binary for file reference` إذا وُجد مرجع بلا مُدخل `.bin` مطابق.
`resources/{Type}/{normalizedName}.json`	الحمولة	لقطة JSON واحدة لكل مورد مصدَّر، تُجمَّع اللقطات في مجلد لكل نوع.

وثمة اسم آخر، `unmapped-dependencies.json` ، مدرج في قائمة السماح لدى المُحقِّق، لكنّ المصدِّر الحالي لا يكتبه – لذا عامله بوصفه اسمًا محجوزًا.

يُكتب كل ملف JSON في سطر واحد (`JSON.stringify` دون فراغات). ولا يشترط المستورِد إلا أن يكون JSON صالحًا، دون تنسيق بعينه؛ لذا يجوز إعادة تنسيق الحزمة المحوَّلة بحرّية.

manifest.json

البيان هو أول ملف يُقرأ عند الاستيراد. ويحدّد العقد المشترك `PackageManifest` تعريفه المرجعي، بينما يكتبه جانب التصدير عبر `buildPackageManifest` .

الحقل	مطلوب	المعنى
`schemaVersion`	نعم	إصدار **صيغة** الحزمة. يسجّل الكاتب أدنى إصدار يصف الحزمة (`pickSchemaVersion`): `1.4` حين تحمل الحزمة ملف README، وإلا `1.3` حين تحمل كتلة `package` ، وإلا `1.2` حين يوجد `field-mapping-hints.json` ، وإلا `1.1` – والقيم الثلاث الأحدث كلها `Since 1.10` . ويقبل المستورِد من الإصدار 1.10 فصاعدًا الإصدارات من `1.0` إلى `1.4` ؛ أما المستورِد الأقدم من 1.10 فيقبل `1.0` و `1.1` فقط (`SUPPORTED_SCHEMA_VERSIONS`). وتؤدي أي قيمة أخرى إلى خطأ التحقّق المانع `INCOMPATIBLE_SCHEMA` . ولقواعد التوافق صفحتها الخاصة.
`createdAt`	نعم	طابع زمني بصيغة ISO 8601 للتصدير. للعلم فقط.
`resourceCount`	نعم	عدد الموارد المحزَّمة فعلًا – أي طول `orderedResources` في `index.json` بعد تطبيق مرشّحات التصدير.
`resourceTypes`	نعم	الأنواع الفريدة التي لها ملف واحد على الأقل تحت `resources/` . لا تظهر هنا إلا الأنواع القابلة للتصدير.
`sourceEnvironment`	لا	اسم قديم للبيئة، بصيغة نصّ حرّ (قيمة `NAMESPACE` في النشر). حلّ محلّه `sourceEnvironmentId` ، لكنه ما يزال يُكتب.
`sourceEnvironmentId`	لا	معرّف ثابت لبيئة المصدر، مأخوذ من إعداد المنصة `ENVIRONMENT_ID` . يُحذف حين يكون الإعداد غير مضبوط.
`sourceSpaceId`	لا	المساحة التي صُدِّرت منها الحزمة.

الحقل	مطلوب	المعنى
`sourceVersion`	لا	إصدار التطبيق وقت التصدير. للعلم فقط – يُقارَن لإصدار تحذير فحسب (`NEWER_SOURCE_VERSION`)، ولا يمنع الاستيراد أبدًا.
`description`	لا	نص حرّ يُقدَّم عند التصدير.
`packageSince 1.10`	لا	كتلة هوية الإصدار المنشور – لا توجد إلا في إصدار حزمة منشور، وهي ما يحوّل الاستيراد إلى تثبيت. انظر ملفات الإصدار المنشور.
`origin Since 1.10`	لا	مصدر الملف: `exportId` و `name` و `environment {id, name}` و `space {id, name}` و `exportedAt` و `exportedBy {id, name}`. ولا تكون أعضاؤه فارغة إلا في الملفات المصدَّرة قبل إتاحة بيانات المصدر. ويسجّله المستورِد لكل مورد، ليتيح للهدف تحديد مصدره.
`readme Since 1.10`	لا	`{path: 'README.md', images[]}` – موضع ملف README وصوره المدرجة (المخطط `1.4`).
`content Since s 1.10`	لا	ملخّص مخصّص للعرض: `counts` لكل نوع، و `modules[]` بكياناتها وأعداد حقولها، وخانة `other` اختيارية، و `pages[]` (الصفحات العامة في الحزمة، `{id?, name}`)، و `truncated` عند بلوغ سقف 100 وحدة / 500 كيان / 100 صفحة. وتُسجَّل أيضًا لكل وحدة وكيان وصفحة قيمة `id` الخاصة بها في بيئة المصدر، مع `moduleId` الخاص بكل كيان، لتمكين المراجِع من ربطها بأصولها؛ أما الإصدارات المبنية قبل بدء تسجيل المعرّفات فتحمل الأسماء فقط.

وتتجاوز أهمية `sourceEnvironmentId` و `sourceSpaceId` ما قد يبدو للوهلة الأولى: فهما يشكّلان، مع المساحة الهدف، مفتاح **ملف تعريف التخطيط** المخزَّن. وتتيح هذه الآلية لعمليات الاستيراد المتكررة من المصدر نفسه أن تطبّق مسبقًا تخطيطات أصحاب الصلاحيات (المستخدمين والمجموعات والوحدات التنظيمية) التي أُكّدت سابقًا. ويظلّ البيان المُنشأ يدويًا الذي يُغفلهما قابلًا للاستيراد، لكنه لن يطابق أي ملف تعريف مخزَّن أبدًا – لذا تبدأ عملية التخطيط من الصفر في كل استيراد.

ولا يفحص المُحقِّق الصارم أنواع بيانات الحقول إلا للحقول الأربعة المطلوبة؛ أما الحقول الاختيارية فتُقرأ عند استعمالها.

index.json

يضمّ `PackageIndex` ثلاثة أعضاء:

العضو	النمط	المعنى
`orderedResources`	`ResourceRef[]`	جميع الموارد المحزَّمة بصيغة `{type, id, …}` ، **مرتّبةً طوبولوجيًا** – الآباء قبل الأبناء.
`nameMap`	`Record<id, filename>`	معرّف المورد ← الاسم الأساسي لملفه تحت `resources/{Type}/` . **وهذا هو مفتاح اسم الملف المرجعي.**
`displayNameMap`	`Record<id, localized name>`	معرّف المورد ← الاسم المعروض `{en, ar}` ، لاستخدامه في معالج الاستيراد. اختياري.

ولاعتماد `nameMap` مرجعًا سبب يجب على المنتِج مراعاته: حين يتحوّل موردان من النوع نفسه إلى الاسم النموذجي `normalizedName` ذاته، يُلحق المصدِّر باسم ملف المورد الثاني معرّفَه (`projects` ← `projects_<id>`). فحقل `normalizedName` في المورد نفسه ليس إلا الاسم *الخام* الذي تتشاركه جميع الموارد الشقيقة المتشابهة في الاسم. ويحدّد المستورِد (`resolvePackagedResource`) الملفات عبر `nameMap` أولًا، ويتحقّق باستخدام المعرّف من أي نتيجة يصل إليها عبر الاسم الخام، منعًا لأن يؤدي تصادم الأسماء إلى إرجاع مورد مختلف يحمل الاسم نفسه دون تنبيه. والمولِّد الذي يشتقّ أسماء الملفات دون الرجوع إلى `nameMap` الخاص به يعيد إنتاج الخلل الذي وُضع هذا المنطق لمنعه.

dependencies.json

مصفوفة `DependencyEdge[]` مُسلسَلة، تمثّل إغلاق التبعيات الناتج عن التحديد المصدَّر، والمستخلَص من مخطط التبعيات وقت التصدير. وبنيتها كما يلي:

الحقل	المعنى
`from` ، `to`	مرجعا `ResourceRef` – المورد المعتمِد وتبعيته.
`path`	مسار JSON داخل لقطة المورد `from` يحدّد موضع المرجع. وهو ما يبيّن للمستورِد **أين يعيد كتابة المعرّف**.
`resolvedVia`	كيفية العثور على المرجع: `DIRECT_ID` أو `NORMALIZED_NAME` أو `REFERENCE_PROPERTY` أو `PIPELINE_STAGE` .
`hard`	ما إذا كانت التبعية بنيوية. تتحكّم الحواف الصلبة في ترتيب الاستيراد والتخطي المتسلسل؛ أما الليّنة فقد تُصفَّر حين يتعذّر حلّها.
`kind`	`MEMBERSHIP` (ابن ينتمي إلى أب) أو `REFERENCE` (إشارة عرضية) – وهما قيمتا التعداد `DependencyKind` الوحيدتان.
`fromActive` ، `fromArchived` ، `metadata`	أعلام للحالة وسياق إضافي.

وتؤدّي الحواف مهمّتين معًا: فهي توجّه **إعادة تخطيط المعرّفات** (عبر `path`)، وتحدّد **الترتيب الطوبولوجي** الذي ينظّم الاستيراد. وإذا أضاف المنتِج موارد بلا حواف مطابقة، استُوردت بترتيب اعتباطي دون إعادة كتابة أي مراجع – فيُنشأ المورد، لكن جميع مراجعه تظلّ تحمل معرّفات بيئة المصدر. وتفصّل صفحة مخطط التبعيات دلالات الحواف وواجهة المخطط وكيفية استخلاص الحواف.

Since 1.10 field-mapping-hints.json

يمثّل هذا الملفّ الجانبي الفارق البنيوي **كلّه** بين المخطط `1.1` والمخطط `1.2` . فلم تُضَف أي خاصية إلى حمولة أي مورد، ولم يُضَف حقل إلى البيان أو يُحذف منه، ولم يتغيّر شكل أي مُدخل قائم: لا يضيف `1.2` سوى مُدخل اختياري واحد في المستوى الأعلى.

و**حقل الربط** (map field) حقلٌ من `DataEntityField` يحمل `mapping.enabled` – أي حقل يعكس قيمةً من كيان أب. وتحمل كتلة `mapping` فيه معرّفَي حقلين: `referenceFieldId` ، وهو حقل في *هذا* الكيان يشير إلى الأب، و `sourceFieldId` ، وهو حقل في الكيان *الأب* تُعكَس قيمته (إضافةً إلى `updateMode` : `Local` ، `0` ، `Never` ، `1` ، `ParentOnly` ، `2` ، `TwoWay` ، `3`).

ويظلّ `referenceFieldId` صالحًا بعد الاستيراد دون معالجة إضافية، لأنه داخل الكيان نفسه، ولأن data-service تطابق الحقول باستخدام المعرّف `id` الذي تحمله الحمولة. أما `sourceFieldId` فلا يظلّ صالحًا بالضرورة؛ إذ إن الكيان الأب **يُخطَّط** عادةً على كيان هدف موجود مسبقًا بدلًا من كتابته من جديد، فيحتفظ بمعرّفات حقوله في بيئة الهدف، ولا يطابق `sourceFieldId` المحزَّم أي حقل هناك. والمفتاح الذي يربط الحقل بين البيئتين فعلًا هو `fullNormalizedName` – أي `<normalizedName>` للحقل في المستوى الأعلى، و `<block>.<field>` للحقل الموجود في الكيان الداخلي لكتلة.

وقبل `1.2` ، لم تكن هناك آلية تنقل ذلك الاسم بين البيئتين، فكان أي ربط يغيب كيانه الأب عن الحزمة **يُسقَط بصمت بينما يبلّغ الاستيراد بالنجاح**. ويسدّ الملفّ الجانبي هذه الثغرة، إذ يربط كل مفتاح `` `${entityId}:${fieldId}` `` – بمعرّفات بيئة المصدر – بالبنية التالية:

```
interface FieldMappingHint {
  sourceEntityId: string;                // source-env id of the mapped field's PARENT entity
  sourceFieldFullNormalizedName: string;  // flattened name of the source field on that parent
}
```

وتحكم هذا المُدخل، بوصفه عضوًا في الأرشيف، قاعدتان. فهو **لا يُكتب إلّا حين يكون غير فارغ**، مما يُبقي الحزمة الخالية من حقول الربط على المخطط `1.1`، ومن ثمّ قابلةً للاستيراد في بيئة لم تُرقَّ. وهو **يُقرأ اختياريًا**: فغيابه ليس خطأً في أي إصدار مدعوم، لأن آلية الحلّ ترجع إلى لقطة الكيان الأب داخل الحزمة – وهو المسار الذي اتّبعته حزم `1.0` و `1.1` دائمًا.

وتُولَّد التلميحات عمدًا عند كل تصدير بدلًا من تثبيتها على الكيان، كي لا تصبح غير مطابقة عند إعادة تسمية حقل. وإذا تعذّر مع ذلك العثور على حقل المصدر في الهدف، يُطلب من المثبِّت تحديد الحقل الذي تُنسخ منه القيمة – وهو قرار يتوقّف عليه بدء التنفيذ – كما هو موثَّق في واجهة الاستيراد.

ملفات الإصدار المنشور `Since 1.10`

إصدار الحزمة **المنشور** من منطقة الحزم – انظر الحزم والمتجر والمتجر داخل التطبيق – هو ملف ZIP نفسه مع أربع إضافات. ولا يتضمّن التصدير العادي أيًّا منها.

`manifest.package` (`ManifestPackageInfo`):

العضو	المعنى
`id`	معرّف الحزمة الدائم، `pkg_` متبوعًا بمعرّف ULID من 26 حرفًا.
`version`	إصدار بصيغة semver الصارمة.
`displayName` `description?`	مترجمان.
`releaseNotes`	مترجمة ومطلوبة.
`minPlatformVersion`	أدنى إصدار للمنصة يُسمح له بتثبيت الحزمة.
`builtOn?`	إصدار المنصة في بيئة التأليف.
`license`	دائمًا `'free'` .
`rootHash`	التجزئة المعيارية للحزمة (من `hashes.json`).
`dependencies[]`	`{packageId, range, displayName?}` – حزم أخرى تحتاجها هذه الحزمة، بنطاقات إصدارات semver.
`externalDependencies[]`	`{type, sourceId, name, requiredBy[]}` – عناصر خارج أي حزمة تشير إليها هذه الحزمة. و `Since 1.10` يتضمّن كل عنصر أيضًا بيانات توضّحه: `displayName?` و `usedBy[]` (`resourceId, resourceType, resourceName?, fieldName?, fieldLabel?, kind,`) `{resourceId, resourceType, resourceName?, fieldName?, fieldLabel?, kind,}`. حيث تكون `kind` إحدى القيم `reference` أو `copiesField` أو `other`) و `requiredFields[]` (`{name, label?,}`). أي الحقول التي يجب أن يوفّرها الكيان الهدف كي تعمل نُسَخ الحزمة). و `Since dev` قد تتضمّن القائمة أيضًا هدفًا ليّنًا `{dataType?}` تعرضه قطعة في صفحة (معرضًا لكيان لا تتضمّنه الحزمة)؛ ويمكن ربط هذا العنصر عند التثبيت، لكنه ليس تبعية حزمة، ويظهر للمثبِّت في صفّ اختياري (`hard: false`) – انظر قطع الصفحات التي تشير إلى خارج الحزمة. `copiedField?`

عند الاستيراد، تحدّد كتلة `package` **وضع التثبيت**، الذي يشترط سلامة صيغة كل عضو باستثناء `description` و `builtOn` (وإلا ظهر الخطأ `INSTALL_MANIFEST_INVALID`). وتُرفض أي قيمة لـ `package.id` لا تمثّل معرّف `pkg_` صالحًا – أي ملف نُشر قبل إتاحة معرّفات الحزم – بالرمز `422 PACKAGE_FORMAT_OUTDATED` .

`hashes.json` – يتضمّن `algorithm` و `rootHash` ، وتجزئة `readme` اختيارية (تدخل في `rootHash` حين تُضبط)، ولكل مفتاح مورد: `type` و `name` و `hash` وتجزئات اختيارية لكل عنصر. و `Since 1.10` يتضمّن أيضًا قيمة `parentKey` اختيارية – وهي مفتاح المورد المحزَّم الذي ينتمي إليه المورد (كيان المعرض، ووحدة الكيان). وتُسجَّل هذه القيمة خارج التجزئة، لذا لم تؤدِّ إضافتها إلى تغيير أي `rootHash` ، ولا توجد في الإصدارات المنشورة قبل بدء تسجيل الآباء. وقيمة `algorithm` هي `sha256/osos-canonical-v1` أو `-v2` أو `-v3` : تُكتب الإصدارات الجديدة بالخوارزمية **v3**، ويجب على القارئ قبول الخوارزميات الثلاث. وتستبعد

v2 المفاتيح التي يجوز للعميل تغييرها في مورد مثبَّت (انظر مخططات موارد الحزمة)؛ أما v3 فتجزّئ، إضافةً إلى ذلك، السجلّ المُشار إليه الذي يطابقه الاستيراد بالهوية – مستخدمًا أو مجموعة أو نطاقًا أو مساحة – اعتمادًا على تلك الهوية وحدها، فلا يُعدّ تعديل ملف مستخدم مُشار إليه تغييرًا في الحزمة. و`Since dev` لا تغطّي تجزئة `Function` إلا **أسماء** أسرارها، ضمن إصدار الخوارزمية نفسه (انظر أسرار الوظيفة). ويُجزَّأ كل مورد بصيغة JSON معيارية، مع استبعاد المفاتيح التي يديرها الخادم والمفاتيح الخاصة بالبيئة، واستبدال المعرّفات بمفاتيح نسبية داخل الحزمة؛ وبذلك يعطي المحتوى نفسه التجزئة نفسها في أي بيئة. وهذا ما يتيح لعملية التثبيت الإبلاغ عمّا إذا كان المورد المثبّت قد تغيّر (`changed`).

`package-references.json` – يتضمّن كل مرجع من هذه الحزمة إلى حزمة أخرى: `resourceKey` و`path` و`packageId` و`targetResourceKey` . ويؤدي تعذّر حلّ مرجع في مساحة التثبيت إلى الخطأ `INSTALL_REFERENCE_UNRESOLVED` .

`README.md` و `readme/` – نص Markdown لملف README والصور التي قد يربط إليها، وتُدرج في `manifest.readme` . ولا يُكتب ملف README إلا في **إصدار منشور**: فالتصدير العادي لا يتضمّن أيًّا من هذه الملفات، وإن كان المستورِد يقرأ ما يجده منها ويحتفظ به. ويُتحقَّق منها عند الاستيراد وفق الحدود التالية:

الحد	القيمة
نص Markdown	256 KiB
كل صورة	2 MiB؛ بصيغة PNG أو JPEG أو WebP أو GIF
الصور	50 صورة على الأكثر، و20 MiB إجمالًا

ويُبلَّغ عن مشكلة في README إذا كان الملف مفقودًا، أو كان مساره أو نوعه غير صحيح، أو تجاوز أحد الحدود، أو لم يطابق تجزئته. والإضافات الأربع كلها مدرجة في قائمة السماح لدى المُحقِّق الصارم: أما الخدمة الأقدم من 1.10 فترفض الحزمة التي تعلن `1.3` أو `1.4` بالخطأ `INCOMPATIBLE_SCHEMA` قبل الوصول إلى هذه الإضافات أصلًا.

ملفات الموارد

يُمثَّل كل مورد محزَّم بملف JSON واحد في `resources/{Type}/{name}.json` ، حيث تؤخذ `{name}` من `nameMap` . وتتناول صفحة مخطّطات موارد الحزمة مخطّطات الحمولة لكل نوع؛ أما هذه الصفحة فتتناول القواعد التي تنطبق على الملفات بوصفها *أعضاء في الأرشيف*:

- **المفاتيح المُدارة خادميًا غائبة.** يحذف التصدير `_id` و`_resolved` و`createdAt` و`updatedAt` و`createdBy` و`updatedBy` و`creationInfo` و`updateInfo` و`stateHistory` و`updateIdentifier` و`spaceId` و`space` و`version` و`__v` من كل لقطة (`SERVER_MANAGED_KEYS`)، وتحذفها محلّلات الاستيراد مجددًا احترازيًا قبل الكتابة. لذا لا تُصدِّرها ولا تتوقّع وجودها.
- **`id` يبقى.** فهو معرّف UUID في بيئة المصدر، ويُحتفَظ به ليتحقّق المستورِد من نتائج البحث عن أسماء الملفات ويبني جدول إعادة التخطيط.
- **`executionPlan` يُجرَّد** من لقطات `DataView` و`GenericView` – فهو مشتق خادميًا ويُعاد توليده في الهدف.
- **لقطات `DataCollection` المستندة إلى كيان تحمل `items: []`** – فعناصرها معرّفات مُدخلات لا تُنقل في الحزمة. أما مجموعات ADT (ذات الأنماط الأولية) فتحتفظ بعناصرها العددية المضمّنة.
- **تحمل `Function` أسماء الأسرار لا قيمها** `Since dev` . تحتفظ صفوف `value.secrets[]` فيها بـ `key` وبأي حقل آخر، لكن قيمة كل `value` تصبح `null` ؛ ويُزال صفّ `OSOS_API_TOKEN` ومرجع `identity` . انظر أسرار الوظيفة تبقى في بيئتها.

سبعة عشر نوعًا قابلًا للتصدير – والنوع الثامن عشر

تمثّل `EXPORTABLE_TYPES` القائمة الكاملة للأنواع التي يُسمح بظهورها تحت `resources/` – وهي سبعة عشر نوعًا:

`DataEntity` ، `DataView` ، `DataQuery` ، `DataRelation` ، `DataCollection` ، `DataReaction` ، `GenericView` ، `DataModule` ، `SystemButton` ، `StatisticQuery` ، `Report` ، `Function` ، `CounterWidget` ، `Announcement` ، `DataEntityAggregate` ، `QuickFilterGroup` ، `TagGroup` .

والأنواع الثلاثة الأخيرة جديدة – `Since 1.10` . وتمثّل `QuickFilterGroup` و `TagGroup` المجموعات التي تُنظَّم فيها المرشّحات السريعة والوسوم في الكيان؛ وتُنقل بوصفها أبناءً للكيان، ولا تحدّد هذه الصفحة شكل حمولتيهما (صدّر موردًا من أيٍّ منهما واقرأه). أما `DataEntityAggregate` فهو المورد الذي تستند إليه قائمة **المحصّلات (Aggregators)** في الكيان، ويُنقل بوصفه ابنًا للكيان تحت `resources/DataEntityAggregate/` ، وشكل حمولته موثَّق في مخطّطات الموارد. ولا تتضمّن الحزمة المُنتَجة قبل إتاحته هذا المجلد.

أما التعداد `ResourceType` الذي تنتمي إليه هذه القيم فيضمّ **ثمانية عشر** عضوًا. والعضو الإضافي هو `Dashboard` ، وهذا الاختلاف مقصود – إذ تجيب القائمتان عن سؤالين مختلفين:

- `ResourceType` هو معجم عُقد **مخطط التبعيات**؛ وينصّ تعليقه التوثيقي على: «Resource types tracked by the dependency graph».
- `EXPORTABLE_TYPES` هي **المجموعة الجزئية الخاصة بالتحزيم** من ذلك المعجم.

فلوحة المعلومات مورد **قائم على مُدخل**: تمثّلها `dependency-service` في المخطط انطلاقًا من الكيان الأساسي `DASHBOARD` : أي إن لوحة المعلومات *مُدخل بيانات* في كيان نظامي، وليست وثيقة مورد مستقلة. وهي تشارك في المخطط – بل إن التحزيم يجتاز مسارات `Dashboard.widgets[*]` عند إعادة تخطيط مركّبات العلاقات – لكنها لا تُكتب أبدًا إلى `resources/` . وإذا نُقلت لوحة معلومات أصلًا، فإنما تُنقل بوصفها مرجع مُدخل.

وتُستبعد أنواع الهوية لسبب مختلف: فكيانات النواة `USER` و `GROUP` و `ORGUNIT` و `WORKING_HOURS` (`CORE_ENTITY_IDS`) موجودة بصورة مستقلة في كل بيئة؛ لذا تُدرج الإشارات إلى مُدخلاتها في `entry-references.json` ، ويُعاد تخطيطها عند الاستيراد بدلًا من تحزيمها بوصفها موارد.

أما تحديد أيٍّ من *أفراد* الأنواع السبعة عشر يُسقَط حتى عند اختياره – الكيانات المؤرشفة وأشجارها الفرعية، والموارد المقيّدة بمستخدم، والموارد الضمنية – فيخضع لسلوك خطّ التصدير، وهو موثَّق في سير عمل التصدير.

حدود الأمان عند الاستيراد

يفرض `extractPackageZip` حدودًا صارمة قبل قراءة أي محتوى، بالترتيب التالي:

الحارس	القيمة	عند المخالفة
الحد الأقصى لمُدخلات ZIP	10 000 (`MAX_ZIP_ENTRIES`)	يفشل الاستخراج.
اجتياز المسارات – وجود `..` في أي موضع، أو `/` في البداية، أو أي شرطة مائلة عكسية في اسم مُدخل	– (`isSafeEntryName`)	يفشل الاستخراج.
الحد الأقصى للحجم بعد فك الضغط، محسوبًا لجميع المُدخلات	200 MB (`MAX_DECOMPRESSED_SIZE`)	يفشل الاستخراج.
مكوّنات المسار تحت `resources/`	يجب أن تطابق `^[a-zA-Z0-9_.-]+$` (`SAFE_PATH_COMPONENT`)	يُتخطّى المُدخل **بصمت**، ولا يُرفض.
مُدخلات `resources/` التي لا تتكوّن مساراتها من ثلاثة مقاطع بالضبط	–	تُتخطّى بصمت.

ويستحقّ الصفّان الأخيران الانتباه، لأن سلوكهما يختلف عن بقية الصفوف. فملف المورد الذي يحتوي اسم مجلد نوعه أو اسمه على مسافة أو نقطتين رأسيتين أو حرف غير لاتيني لا يُفشل الرفع، بل يُتجاهل بصمت أثناء الاستخراج. ثم يظهر الفشل لاحقًا بصورة أقل وضوحًا: إذ لا يعثر الاستيراد على المورد، ويتخطّى بالتسلسل كل ما يعتمد عليه اعتمادًا صلبًا. وهذا أكثر أنماط الفشل شيوعًا لدى المولِّد الذي يشتقّ أسماء الملفات من الأسماء المعروضة للمستخدمين بدلًا من `nameMap`، الذي لا يحتوي إلا أسماء نموذجية آمنة.

المُحقّق الصارم

بعد اجتياز الحدود، يعمل `validatePackageStrict` عند حدود الاستخراج، لتتمكّن المراحل اللاحقة من افتراض سلامة بنية الأرشيف. وهو يفحص ثلاثة أمور – المُدخلات المطلوبة، وبنية البيان، وقائمة السماح للمستوى الأعلى – ويبلّغ عن الإخفاقات بخطأ منمّط `PackageFormatError`، تكون فئته `detail.category` واحدةً من أربع:

الفئة	تُرفع عندما
`MISSING_ENTRY`	يغيب `manifest.json` أو `index.json`.
`INVALID_JSON`	يوجد `manifest.json`، لكن يتعذّر تحليله بوصفه JSON.
`SCHEMA_VIOLATION`	يُحلَّل البيان، لكنه لا يجتاز فحص البنية: إما أنه ليس كائنًا، أو أن `schemaVersion` أو `createdAt` ليس سلسلة نصية، أو أن `resourceCount` ليس عددًا، أو أن `resourceTypes` ليست مصفوفة. ويُبلَّغ عن المسار الذي أخفق في التحقّق في `detail.schemaPath`.
`UNKNOWN_ENTRY`	يوجد أي مُدخل في المستوى الأعلى خارج قائمة السماح – ملفات JSON الأحد عشر المسمّاة، و `README.md`، والبادئات `resources/` و `files/` و `readme/`. ويُعامَل الملف غير المتوقَّع في جذر الأرشيف بوصفه عبئًا أو خروجًا عن الصيغة، ويؤدي إلى الفشل مبكرًا. وهذا أيضًا ما *كانت* ستبلّغ عنه خدمة أقدم من `1.2` عند وجود `field-mapping-hints.json`؛ ولهذا تحديدًا يجب أن تعلن الحزمة التي تحمل الملفّ الجانبي `1.2`، لتُرفض بالخطأ الأوضح `INCOMPATIBLE_SCHEMA` بدلًا منه.

ويحمل الخطأ `detail.entryName` / `detail.schemaPath` ليتمكّن المنتِج من معالجة الإخفاقات برمجيًا بدلًا من مطابقة أنماط الرسائل. وتوقَّع أن تصادف الفئات الأربع كلها أثناء تطوير مولِّدك بالتجربة والتكرار – فقد وُضعت لهذا الغرض.

ولاحظ هذا الاختلاف: `dependencies.json` *ليس* ضمن قائمة المُدخلات المطلوبة لدى المُحقِّق الصارم، لكنّ الاستخراج يفشل بدونه بخطأ عادي (غير منمّط). لذا، فإن ملفات JSON الجذرية الثلاثة كلها إلزامية عمليًا.

إنتاج حزمة أو تحويلها

توجد ثلاثة مسارات للإنتاج، تعرض صفحة توليد حزمة موازنة صريحة بينها، تشمل كلفة كل منها. والخلاصة: ابدأ من تصدير فعلي كلما أمكن، فهو المسار الوحيد الذي ينطلق من ناتج مضمون الاستيفاء لجميع متطلبات هذه الصفحة.

أما لتكييف الحزمة مع كل بيئة، فلن تحتاج غالبًا إلى تعديل ملف ZIP أصلًا. ففي جلسة الاستيراد، تتيح نقاط نهاية تجاوز الحمولة (`GET` / `PATCH` / `DELETE` على `/v1/import/{sessionId}/resource/{sourceType}/{sourceId}/payload`) تبديل حمولة مورد واحد وقت التنفيذ، بينما يبقى الأرشيف كما هو. وتشكّل بُنى النقل (`SaveResourcePayloadRequest`، `GetResourcePayloadResponse`) جزءًا من عقود التحزيم المشتركة. وسير عمل الجلسة الذي يحيط بهذه العمليات موثَّق في واجهة الاستيراد.

مخطّطات موارد الحزمة

Availability

Since 1.8.

تحدّد صفحة صيغة الحزمة الحاوية الأرشيفية. أما هذه الصفحة فتحدّد ما يقع **داخل** `resources/{Type}/{normalizedName}.json` – أي شكل الحمولة لكل نوع مورد، مأخوذًا من أصناف النماذج التي تعرّفه في رماز الخادم المصدري. وهي الجزء الذي يحتاجه القارئ ليحوّل حمولة حزمة أو يولّدها، والجزء الذي لا سبيل إلى اكتشافه بغير قراءة تصدير حقيقي. ومحرّك التحزيم الذي ينتمي إليه هذا العقد شُحن في خطّ **1.8**؛ وعلى نشر أقدم تغيب هذه المسارات وواجهة التصدير.

ما تغطيه هذه الصفحة – وما لا تغطيه

هذا مرجع **متدرّج**، وهو لا يغطي الأنواع السبعة عشر القابلة للتصدير كلها. فالأنواع الستة من الفئة الأولى أدناه هي ما تتكوّن منه حزمة وحدة نموذجية؛ وتوثيق السبعة عشر دفعةً واحدة يعني توثيق نموذج البيانات الوصفية للمنصة بأكمله، وتوثيق ستة بدقة أنفع من توثيق سبعة عشر بسطحية.

النوع	الحالة
`DataModule`	**موثّق أدناه**
`DataEntity`	**موثّق أدناه**
`DataView`	**موثّق أدناه**
`DataQuery`	**موثّق أدناه**
`DataReaction`	**موثّق أدناه** – الشكل الأعلى مستوى؛ وشجرة الإجراءات بعمق الملامح فقط
`DataEntityAggregate`	**موثّق أدناه** – جديد في هذه الدورة (`Since 1.10`)
`DataRelation` ، `DataCollection` ، `GenericView` ، `SystemButton` ، `StatisticQuery` ، `Report` ، `Function` ، `CounterWidget` ، `Announcement`	**غير موثّقة.** صدِّر حزمة تحوي أحدها واقرأ JSON الناتج – وهو الإجراء نفسه الذي تحقّقت به هذه الصفحة. وحقيقتان متحقَّق منهما: `Since dev` تحمل صفوف `value.secrets[]` في `Function` مفتاح السرّ `key` بقيمة `value` تساوي `null` ، ولا `identity` لها ولا صفّ `OSOS_API_TOKEN` (أسرار الوظيفة)؛ و `Since dev` قطعة `GenericView` (الصفحة) التي تشير إلى معرض خارج الحزمة مرجع ليّن قابل للربط (قطع الصفحات التي تشير إلى خارج الحزمة).
`QuickFilterGroup` ، `TagGroup` (كلاهما `Since 1.10`)	**غير موثّقة.** لم يُتحقَّق إلا من قواعد استيرادهما: لا يُكتب `id` ولا `dataEntity` أبداً، و `normalizedName` هو هوية المطابقة، ولا يغيّر التحديث إلا `name` و `queries` (ولا يغيّر أبداً `color` لمجموعة الوسوم). وحوافّهما في المخطط موجودة في مخطط التبعيات.

لا تستنتج شكل نوع من الفئة الثانية من الأنواع الستة الموثَّقة هنا؛ فالأنواع تتباين أكثر مما توحي أسماؤها.

الثوابت في كل مورد محزَّم

ستة أمور تصدق على كل الأنواع أدناه، ومعرفتها أولًا تختصر قراءة الجداول كثيرًا.

- `id` **حاضر وهو معرّف UUID في بيئة المصدر.** فكل نموذج يرث قاعدة `Identified` (`services/data-service/src/models/identified.model.ts`) التي تعطي `id` قيمة UUID v4 افتراضيًا. ويستعمله المستورِد للتحقّق

من نتائج البحث عن أسماء الملفات ولإقامة جدول إعادة تخطيط المعرّفات – والحمولة الخالية منه لا يمكن التثبّت من هويتها.

- `specificationVersion` **هو إصدار الصيغة الخاص بالمورد**، وهو غير `schemaVersion` الخاص بالحزمة. تعلنه نماذج الفئة الأولى كلها مطلوبًا بقيمة افتراضية `1` . والمورد الذي لا يدعم محلِّلٌ إصدارَه يُنتج تحذير `UNSUPPORTED_SPEC_VERSION` عند التحقّق، لا فشلًا صلبًا.
- `name` **كائن مترجَم، لا سلسلة نصية مجرّدة أبدًا** – فهو `LocalizedValue` بالشكل `{"en": …, "ar": …}` . و`description` ، حيث وُجد، هو `NullableLocalizedValue` . والمولِّد الذي يُصدر سلسلة نصية عارية هنا ينتج حمولة سترفضها data-service.
- `normalizedName` **هو المعرّف الآلي الثابت.** يُتحقّق منه على النمط `^[a-zA-Z_][a-zA-Z0-9_$]{2,127}$` (من 3 إلى 128 محرفًا، `NormalizedNameValidation` في `packages/osos-shared/src/utils/normalize-name.ts`)، ويحوّل مُنظِّم المنصة نفسه الأحرف إلى الصغيرة ويستبدل بكل محرف آخر `_` . وهو مفهرس فهرسة فريدة على `DataModule` و`DataEntity` ، لكنه سلسلة نصية مُتحقَّق منها فحسب على `DataView` و`DataQuery` و`DataReaction` . كما أنه **ليس** بالضرورة اسم ملف المورد في ZIP – فتلك مهمّة `nameMap` (انظر صيغة الحزمة).
- **التعدادات تُسلسَل أرقامًا لا أسماء.** فـ `type` و`state` و`visibility` وأشباهها تعدادات رقمية (`packages/osos-shared/src/models/enums.ts`): الكيان المحزَّم يحمل `"state": 1` ولا يحمل `"state": "Active"` أبدًا.
- **المفاتيح المُدارة خادميًا غائبة.** تُحذف `creationInfo` و`updateInfo` و`stateHistory` و`updateIdentifier` و`version` و`createdAt` و`updatedAt` و`createdBy` و`updatedBy` و`spaceId` و`space` و`_id` و`_resolved` و`__v` عند التصدير وتجرّدها محلّلات الاستيراد مجددًا (`SERVER_MANAGED_KEYS` ، في `services/packaging-service/src/resource-parsers/parser-utils.ts`). وما تزال الجداول أدناه تُدرج ما يعلنه كل نموذج منها – موسومًا *بُيجرَّد* – كي لا تفاجئ الفجوةُ قارئًا يقارن حمولةً بمصدر النموذج.

وقاعدة ترشيح واحدة من خط التصدير تستحق التكرار لأنها تحدّد ما يمكن أن يظهر في حزمة أصلًا: الكيانات المؤرشفة (مع أشجارها الفرعية المتداخلة)، والموارد المقيّدة بمستخدم (أيّ مورد يحمل `relatedTo.user` قيمةَ `dataEntry`)، وكيانات `DataEntity` الضمنية بلا `implicitOptions.allowEdit` ، وتفاعلات `DataReaction` الضمنية – كلها لا تُصدَّر أبدًا (`isNonExportableResource` ، في `services/packaging-service/src/libs/resource-utils.ts`).

DataModule

أصغر أشكال الفئة الأولى (`services/data-service/src/models/data-module/data-module.model.ts`) – فالوحدة حاوية مسمّاة ذات أيقونة، تُعبَّر عضويتها بإشارة أبنائها إليها.

الخاصية	النمط	ملاحظات
`specificationVersion`	`1`	مطلوبة؛ القيمة الحرفية `1` .
`normalizedName`	`string`	مطلوبة، مفهرسة فهرسة فريدة، مُتحقَّق من نمطها.
`name`	`LocalizedValue`	مطلوبة.
`description`	`NullableLocalizedValue`	اختيارية، تقبل null.
`icon`	`Icon`	مطلوبة.
`isImplicit`	`boolean`	مطلوبة.
`sharing`	`ApplicationSharing`	مطلوبة – مستوى المشاركة.
`creationInfo` ، `updateInfo`	`ActionInfo`	معلنتان على النموذج؛ *تُجرَّدان* من الحزم.

ولاحظ الغائب: لا توجد مصفوفة `entities` . فالعضوية تجري في الاتجاه المعاكس – كل `DataEntity` يحمل خاصية `module` تشير إلى وحدته. ولهذا يستشير تصديرُ وحدةٍ مع أبنائها (`withChildren`) حوافَّ العضوية في رسم الاعتماديات بدل قراءة أي شيء من وثيقة الوحدة نفسها.

DataEntity

أخطر أشكال الفئة الأولى شأنًا وأكبرها: **40 خاصية معلنة** (`services/data-service/src/models/data-entity/data-entity.model.ts`).

خاصيتان منها مكتوبتان خطأً **في المنتج نفسه**، والخطآن معتمَد عليهما: الخاصية `extentions` (يحمل النموذج بجوارها تعليق `TODO rename and fix typo`) والنمط `DataEntityFromSettings` – بحرفي `From` لا `Form` ، مع أن ملفه المصدري اسمه `data-entity-form-settings.model.ts` . أصدرهما كما يتهجّاهما المنتج بالضبط؛ فمفتاح `extensions` الصحيح إملائيًا يُتجاهل بكل بساطة.

الخاصية	النمط	ملاحظات
`specificationVersion`	`number`	مطلوبة؛ الافتراضي `1` .
`name`	`LocalizedValue`	مطلوبة.
`description`	`NullableLocalizedValue`	اختيارية.
`normalizedName`	`string`	مطلوبة، مفهرسة فهرسة فريدة، مُتحقَّق من نمطها.
`version`	`number`	مراجعة مخطّط الكيان؛ *تُجرَّد* من الحزم.
`isImplicit`	`boolean`	مطلوبة.
`virtual`	`DataEntityVirtualOptions`	اختيارية – تحضر على الكيانات الوهمية والمجمّعات.
`implicitOptions`	`DataEntityIsImplicitOptions`	اختيارية. تقرّر `allowEdit` هل الكيان الضمني قابل للتصدير أصلًا.
`type`	`DataEntityTypeEnum`	مطلوبة، **رقمية** (`Custom = 0` ، `ADT = 1` ، `Service = 2` ، …).
`typeOptions`	`TypeOptions | VirtualProxyEntityOptions | AggregatorVirtualEntityOptions`	اختيارية؛ مفتوحة النمط.
`icon`	`Icon`	مطلوبة.
`archiveInfo`	`ActionInfo`	اختيارية.
`state`	`DataEntityStateEnum`	رقمية: `Draft = 0` ، `Active = 1` ، `Archived = 2` . لا تُحزَّم الكيانات المؤرشفة أبدًا، فتوقَّع `0` أو `1` .
`stateHistory`	`StateHistory[]`	*تُجرَّد* من الحزم.
`creationInfo` ، `updateInfo`	`ActionInfo`	*تُجرَّدان* من الحزم.
`internalEntities`	`InternalEntity[]`	الافتراضي `[]` . تعريفات كيانات متداخلة، لكل منها `fields` خاصة بها.
`internalCollections`	`InternalCollection[]`	الافتراضي `[]` .
`fields`	`DataEntityField[]`	الافتراضي `[]` – وهي عمليًا جُلّ كل حمولة حقيقية. يحمل كل حقل مميِّز نمط وكائنَ خيارات خاصًا بالنمط؛ والفهرس موثَّق في أنماط الحقول [بناء التطبيقات › اختر نوع الحقل] ولا يُعاد هنا.
`validation`	`string | null`	تعبير تحقّق اختياري على مستوى الكيان.

الخاصية	النمط	ملاحظات
`tags`	`string[]`	الافتراضي `[]` .
`workflows`	`DataEntityWorkflow[]`	الافتراضي `[]` .
`meta`	`ServiceMeta | Record<string, any>`	اختيارية، مفتوحة.
`module`	`UUID`	مطلوبة – حافة العضوية نحو `DataModule` المالكة.
`displayOption`	`DataEntityDisplayOptionEnum`	تعداد رقمي اختياري؛ الافتراضي `KeyValue` .
`formSettings`	`DataEntityFromSettings`	اختيارية. النمط ذو الاسم الخاطئ إملائيًا.
`deletedFields`	`DataEntityDeletedFields[]`	اختيارية.
`breakingFields`	`DataEntityBreakingFields[]`	معلنة، لكنها مدرجة في `hiddenProperties` للنموذج – لا تُسلسَل أبدًا، فلا تُحزَّم أبدًا.
`extentions`	`DataEntityExtensions`	اختيارية. الخاصية ذات الاسم الخاطئ إملائيًا.
`entryNames`	`DataEntityEntryNames`	اختيارية.
`discussions`	`DataEntityDiscussions`	اختيارية.
`templates`	`DataEntityExportTemplate[]`	اختيارية – قوالب التصدير/الطباعة.
`treeStructure`	`DataEntityTreeStructureConfig`	اختيارية.
`indexes`	`DataEntityIndex[]`	الافتراضي `[]` .
`dataSources`	`DataEntityFieldSources[]`	اختيارية.
`storeRules`	`StoreRule[]`	اختيارية. تُلتقط للمخازن المشتركة (`type === 2`) لقطةٌ إضافية في `store-references.json` بالحزمة.
`sharing`	`ApplicationSharing`	اختيارية على هذا النموذج (مطلوبة على `DataModule` و `DataView`).
`updateIdentifier`	`string`	رمز التزامن التفاؤلي؛ *يُجرَّد* من الحزم.
`displayTemplate`	`EntityDisplayTemplate | null`	اختيارية.
`scriptAttributes`	`object`	اختيارية، تقبل null، مفتوحة.

`Since 1.10` – بات `typeOptions.sources[*].sourceViewId` لكيان `VirtualProxy` حافة تبعية صارمة (hard) جديدة إلى `DataView` من النوع `kind: Proxy` المرتبط به؛ وعلى جانب القراءة (لا الحمولة المحزَّمة) ينضم إثراء الاستجابة فقط `proxyCapabilities` إلى `forbiddenOnWrite` إلى جانب `moduleDetails` / `typeDetails` . انظر حوكمة معرض الوكيل [بناء التطبيقات ‹ الكيانات الوهمية] في الكيانات الوهمية.

DataView

46 خاصية معلنة – أعرض أشكال الفئة الأولى. الصنف اسمه `DataViews` (بصيغة الجمع) في `services/data-service/src/models/data-view/data-views.model.ts`، وهذا جدير بالمعرفة لأن البحث في الخادم عن `class DataView` لا يجد شيئًا. (وثمة خطأ إملائي قريب: يعلن `data-view-exceptions.ts` الصنفَ `DataViewExcpetion` – ليس جزءًا من الحمولة، لكنك ستصادفه عند قراءة مسارات رماز المعارض.)

مجمّعةً بحسب الدور:

الهوية والموضع – `specificationVersion` (مطلوبة، الافتراضي `1`)، و `name` (مطلوبة)، و `normalizedName` (مُتحقَّق منها، غير مفهرسة فهرسة فريدة)، و `description` ، و `icon` (اختيارية هنا، خلافًا للكيان والوحدة)، و `dataEntity` (مطلوبة – الكيان

الذي يقرؤه المعرض)، و `module` (اختيارية)، و `displayOrder` (الافتراضي `0`)، و `domain` (`NullableReferenceProperty`).

المصدر – `dataSource` (`DataEntrySource` ، مطلوبة)، و `sourceOptions` ، و `sourceEntitiesMapping` (الافتراضي `[]`)، و `relations` (`ViewRelation[]` ، الافتراضي `[]` – يعاد تخطيط مركّبات العلاقات داخلها عند الاستيراد)، و `fields` (`string[]` ، **مطلوبة بعنصر واحد على الأقل** – مجموعة الحقول المعروضة).

الوصول – `isPublic` (**مهجورة** في مخطّط النموذج نفسه)، و `anonymous` (`AnonymousViewOption | null`)، و `authenticated` (`AuthenticatedViewOption | null`)، و `isPublished` (الافتراضي `false`)، و `isPinned` (الافتراضي `false`)، و `sharing` (`ApplicationSharing` ، مطلوبة)، و `qrCode` (الافتراضي `{enabled: false}`).

الإعداد – الكتل الثلاث المطلوبة التي تحمل معظم أي حمولة حقيقية: `viewConfiguration` (`DisplayViewConfiguration`) و `editConfiguration` (`EditDataViewConfiguration`) و `entriesConfiguration` (`EntriesDataViewConfiguration`)؛ إضافةً إلى الاختيارية `dashboardConfiguration` و `sideBarConfig` و `displayConfig` و `actionButtonsConfig` و `systemButtons` (الافتراضي `[]`) و `systemButtonsOrder` (**مهجورة** لمصلحة الترتيب داخل `systemButtons`) و `screenOrder` و `defaultViewTemplate` و `redirectionRules` (الافتراضي `[]`) و `help` والمفتوحتين `uiAttributes` / `scriptAttributes` .

مفاتيح السلوك – `showArchivedEntries` (الافتراضي `false`)، و `showInvalidEntries` ، و `allowObsoleteFilter` (الافتراضي `false`)، و `allowInvalidRequirementsFilter` (الافتراضي `false`).

المحصّلات المفعَّلة على المعرض `Since 1.10` – `viewConfiguration.aggregates` مصفوفةُ `string[]` من معرّفات `DataEntityAggregate` : أي المحصّلات التي يعرضها هذا المعرض. ويرشّح الاستيراد المصفوفة إلى السلاسل غير الفارغة، والمعرّف الذي يتعذّر إعادة تخطيطه – لأن المحصّلة لم تُضمَّن في الحزمة أو حُذفت قبل التصدير – **يُسقَط ولا يُعامَل خطأً**. فينتهي المعرض عارضًا محصّلة أقلّ لا غير، وهي النتيجة نفسها التي يعطيها حذف المحصّلة مباشرةً. أما النوع الذي تشير إليه هذه المصفوفة فموثَّق أدناه.

المشتق والمُدار خادميًا – يُحذف `executionPlan` صراحةً عند التصدير ويعيد الهدفُ اشتقاقه: فالحزمة المولَّدة التي تتضمنه تُتجاهل في أحسن الأحوال. وتَسِم `isImplicit` (الافتراضي `false`) و `implicitType` (رقمية، الافتراضي `NONE`) المعارضَ التي تولّدها المنصة – ولأن data-service تُنشئ المعارض الضمنية *بشكل غير متزامن* بعد إنشاء الكيان، فإن لخط الاستيراد مُهلًا قابلة للضبط بمتغيرات البيئة (`PKG_IMPLICIT_SETTLE_MS` و `PKG_IMPLICIT_POLL_ATTEMPTS` و `PKG_IMPLICIT_POLL_DELAY_MS` في `services/packaging-service/src/services/package-import.service.ts`) لانتظارها. أما `creationInfo` و `updateInfo` و `updateIdentifier` فمعلنة لكنها *تُجرَّد* من الحزم.

المفاتيح التي يملكها العميل و `packageLabels` `Since 1.10`

المعرض المثبَّت مقفل بحزمته، باستثناء قائمة قصيرة من المفاتيح يجوز للعميل تغييرها ويحتفظ بها التحديث (`CLIENT_KEYS` في `services/packaging-service/src/libs/update-carry-over.ts`): `isPublished` و `displayOrder` و `authenticated` و `anonymous` و `permissions` (صفوف حقوق الوصول التي تضمّها قراءة المعرض). وتستبعدها تجزئة المحتوى، فلا يُعدّ تغيير العميل لأحدها تعديلًا للحزمة – و**يجب ألّا تعتمد الحزمة على شحنها**: فالتحديث يعيد قيم العميل فوق المورد الوارد.

وتسجّل `packageLabels` (`{name?, description?, icon?}`) ما شحنته الحزمة لتسمية أعاد العميل تسميتها على معرض أو صفحة مثبَّتين. وهي تُخزَّن على المستند وتُستبعد من التجزئة، لكنها لا تُقبل أبدًا من مستدعٍ عادي – إذ يُسقطها data-service من كل إنشاء، ومن كل تحديث ما لم يكن المستدعي معفى من حارس الموارد المثبَّتة – لذلك **يجب ألّا تُصدرها الحزمة المولَّدة**. أما كيف يقرأ العميل تلك التسميات ويغيّرها ففي المعارض [بناء التطبيقات › معارض البيانات وأوجهها].

DataQuery

`services/data-service/src/models/data-query/data-query.model.ts` :

الخاصية	النمط	ملاحظات
`specificationVersion`	`number`	مطلوبة؛ الافتراضي `1` .
`name`	`LocalizedValue`	مطلوبة.
`normalizedName`	`string`	مُتحقَّق من نمطها؛ غير مطلوبة وغير مفهرسة فهرسة فريدة.
`description`	`NullableLocalizedValue`	اختيارية.
`dataEntity`	`UUID`	مطلوبة – الكيان المُستعلَم.
`dataSource`	`DataEntrySource`	اختيارية.
`visibility`	`VisibilityEnum`	مطلوبة، رقمية: `Organization = 0` ، `Private = 1` ، `Public = 2` .
`criteria`	`FilterExpression`	مطلوبة – شجرة التعبير العودية. انظر أدناه.
`relatedTo`	`DataQueryRelatedTo`	اختيارية، تقبل null. الاستعلام الذي يحمل `relatedTo.user` قيمةَ `dataEntry` مقيّدٌ بمستخدم، ولذلك **لا يُصدَّر أبدًا**.
`renderedCriteria`	`string`	موسومة **مهجورة** في مخطّط النموذج. صياغة مشتقة من `criteria` ؛ إن عدّلت إحداهما فأبقِ الأخرى متسقة – أو احذفها.
`isImplicit`	`boolean`	الافتراضي `false` .
`useInQuickFilter`	`boolean`	الافتراضي `false` – هل يُعرض الاستعلام مرشِّحًا سريعًا.
`creationInfo` ، `updateInfo`	`ActionInfo`	*تُجرَّدان* من الحزم.
`tag`	`DataQueryTag | null`	اختيارية.
`updateIdentifier`	`string`	*يُجرَّد* من الحزم.
`parameters`	`DataQueryParameter[]`	اختيارية.
`isParameterized`	`boolean`	اختيارية.

وأنفعُ حقيقة هنا لمن يولّد الحزم: **`criteria` هي عين صنف `FilterExpression` الذي تقبله واجهة HTTP API عضوًا باسم `criteria` داخل `filterCriteria`** – فمخطّط مرشِّح المعايير في `data-service` يستورده من `models/data-query/filter-expression.model.ts` (`customJsonSchema/filter-criteria-json-schema.ts`). ومفردات المعاملات موثَّقة مع لهجة الترشيح v2/v3، ويمكن التحقّق من شجرة معايير عبر نقطة نهاية محلِّل المعايير الحية قبل كتابتها في حزمة – وهو التحقّق الوحيد المتاح قبل الاستيراد لأي جزء من حمولة مورد.

DataReaction

يقيم نموذج التفاعل في خدمة غير خدمة أشقائه الأربعة: `services/reaction-service/src/models/data-reaction/data-reaction.model.ts`.

الخاصية	النمط	ملاحظات
`specificationVersion`	`number`	مطلوبة؛ الافتراضي `1` .
`normalizedName`	`string`	اختيارية، مُتحقَّق من نمطها.

الخاصية	النمط	ملاحظات
`definingEntity`	`ReferenceProperty`	مطلوبة – الكيان المالك بالشكل `{dataEntity, dataEntry}`. تقود حافة العضوية الصلبة. يجرّدها محلِّل الاستيراد من جسم الكتابة (فالأب يسافر في مسار URL)، لكن أبقِها في الحمولة: فالتحقّق ورسم الاعتماديات يقرآنها.
`eventType`	`EventTypeEnum`	مطلوبة، رقمية.
`executorType`	`CollectionValueModel<EventExecutorTypeEnum>`	اختيارية.
`query`	`ReferenceProperty | null`	اختيارية – حافة ليّنة نحو `DataQuery`.
`criteria`	`FilterExpression`	اختيارية – نموذج التعبير نفسه الذي في `DataQuery.criteria`.
`eventConfig`	`DataReactionEventConfig`	اختيارية، مفتوحة النمط. بعمق الملامح – انظر أدناه.
`actions`	`DataReactionAction[]`	مطلوبة. بعمق الملامح – انظر أدناه.
`isImplicit`	`boolean`	الافتراضي `false`. التفاعلات الضمنية **لا تُصدَّر أبدًا**.
`disabled`	`boolean`	الافتراضي `false`.
`name`	`LocalizedValue`	مطلوبة.
`spaceScope`	`DataReactionSpaceScope`	مطلوبة.
`loopThreshold`	`number`	اختيارية؛ الافتراضي `0` (تنفيذ حلقة المُطلِق مرة واحدة).
`relatedEntities`	`DataReactionRelatedEntities[]`	اختيارية، تقبل null.
`sourceEntity`	`DataReactionSourceEntity | null`	اختيارية. في التفاعلات المقيّدة بعلاقة، تكون `sourceEntity.sourceId` **مركّب علاقة** – `entityId:fieldId[:relatedEntityId]` – يحلّله التحزيم ويعيد تخطيطه وحدةً واحدة (`libs/relation-composite.ts`)؛ فهو معرّف مورد لا معرّف مُدخل.
`type`	`DataReactionTriggerTypeEnum`	مطلوبة، رقمية؛ الافتراضي `0`.
`updateIdentifier`	`string`	*يُجرَّد* من الحزم.
`validations`	`DataReactionValidation | null`	اختيارية.
`creationInfo`، `updateInfo`	`ActionInfo`	*تُجرَّدان* من الحزم.

تُركت `actions` و`eventConfig` عمدًا بعمق الملامح: فالتفاعل شجرة «حدث-شرط-إجراء» مفرداتُ إجراءاتها واسعة وقابلة للتوسّع، وجدولٌ سطحي يضلّل أكثر مما يفيد. استعمل التفاعلات والأتمتة [بناء التطبيقات] لما *تفعله* الإجراءات، وتصديرًا حقيقيًا لشكلها المسلسل. وسلوك استيراد واحد جدير بالمعرفة عند التوليد: يُبقي محلِّل التفاعلات على `id` في جسم الإنشاء عمدًا – فالمنصة تحترم معرّف تفاعل يزوّده العميل لأن الرموز المُشيرة إلى الذات داخل إعدادات الإجراءات تُعاد كتابتها مسبقًا على أساس المعرّف الهدف المخطَّط (`services/packaging-service/src/resource-parsers/data-reaction.parser.ts`).

`Since 1.10` DataEntityAggregate

أحدث نوع قابل للتحزيم – وهو المورد الذي تقوم عليه قائمة **المحصّلات (Aggregators)** على الكيان: زوجٌ مسمّى من `(field, aggregationFunction)` تحسبه المنصّة خادميًا على مُدخلات ذلك الكيان. ونموذجه `services/data-service/src/models/data-entity-aggregate/data-entity-aggregate.model.ts`، ويسافر بوصفه

`resources/DataEntityAggregate/{normalizedName}.json` . ولمعرفة ما التجميعة وكيف يفعّلها المعرض، انظر تجميعات الكيان وتجميع المدخلات [بناء التطبيقات › المجاميع والتجميع].

الخاصية	النمط	ملاحظات
`specificationVersion`	`number`	مطلوب؛ الافتراضي `1` .
`dataEntity`	`UUID`	مطلوب – الكيان المالك. **غير قابل للتغيير بنيويًا** (انظر أدناه).
`normalizedName`	`string`	مطلوب، مُتحقَّق النمط. وهو مفتاح الهوية العابر للبيئات.
`name`	`NullableLocalizedValue`	اختياري – فالخلفية تسمح عمدًا بمحصّلة بلا اسم.
`field`	`string`	مطلوب – قيمة `fullNormalizedName` للحقل المُحصَّل.
`aggregationFunction`	`AggregationFunctionEnum`	مطلوب، عددي.
`sensitiveDataAccessPolicy`	`SensitiveDataAccessPolicyEnum`	اختياري؛ وغيابه يعني السياسة الافتراضية (المميَّزة).
`id`	`UUID`	يولَّد خادميًا في الطرفين، و**ممنوع عند الكتابة** (`forbiddenOnWrite = ['id']` في `resource-parsers/data-entity-aggregate.parser.ts`).

ولاحظ ما يغيب هنا قياسًا بكل نوع آخر في هذه الصفحة: لا يعلن هذا النموذج `creationInfo` / `updateInfo` أصلًا، فلا شيء يحذفه تجريدُ المفاتيح المُدارة خادميًا.

والهوية عبر البيئات هي `normalizedName` لا `id` . فالمستورِد يفهرس المحصّلة بـ `DataEntityAggregate::{entityId}::{normalizedName}` – والكيان المالك جزء من المفتاح، لأن `normalizedName` ليس فريدًا إلاّ *داخل* كيانه.

وهي ابنٌ للكيان، وذلك يُتتبَّع مستقلًا عن كونها مقيَّدة بنطاق الكيان. فتقييد النطاق يقرّر كيف يطابق التحقّق محصّلةً واردة على الهدف: بـ `normalizedName` داخل الكيان الهدف، لا على مستوى المساحة. أما كونها *ابنًا* للكيان فهو ما يقود قرار «رقِّع الابن القائم بدل إنشاء واحد جديد». وكلاهما مهمّ: فلو سلكت محصّلةٌ موجودة أصلًا تحت الكيان الهدف مسارَ الإنشاء، لرفضها حارسُ تفرّد `normalizedName` داخل ذلك الكيان ولأخفق المورد في الاستيراد.

و `dataEntity` غير قابل للتغيير بنيويًا عبر التجاوز – فلا يستطيع تجاوز الحمولة إعادة توجيه محصّلة إلى كيان آخر. أما `normalizedName` فليس عصيًّا على التغيير عمدًا: فإعادة تسمية محصّلة عبر تجاوز إجراء مدعوم.

أما حوافّها في رسم الاعتماديات – حافّة صلبة من الكيان، وأخرى ليّنة عمدًا من كل معرض فعّلها – فموثَّقة في رسم الاعتماديات.

حقيقة النموذج وحقيقة الحزمة

الجداول أعلاه هي **الخاصيات المعلنة في النماذج**، مقروءةً من الأصناف المصدرية المسمّاة في كل قسم، مع التثبّت من قواعد التجريد عند التصدير (`SERVER_MANAGED_KEYS` و `executionPlan` و `items` المجموعات) في `package-export.service.ts` و `parser-utils.ts` . وهي لم تُقارَن حقلًا حقلًا بكل أرشيف ممكن الإصدار، وافتراضاتُ النماذج تخبرك بما ستملؤه data-service، لا بما تحويه كل حزمة تاريخية. فإذا خالفت حمولةٌ تحوّلها جدولًا هنا، فخذ بالحمولة عند القراءة وبالنموذج عند الكتابة – وبلِّغ عن الفرق. أما أنواع الفئة الثانية التسعة فصدِّر عيّنة حقيقية منها؛ ولا تستقرئ.

مخطط التبعيات

تحتوي كل حزمة على ملفّ `dependencies.json`، وهو مصفوفة مُسلسَلة من **حواف التبعية** تسجّل، لكل مورد محزَّم، الموارد التي يعتمد عليها وموضع كل اعتماد داخل JSON الخاص به. وهذه الحواف ليست مجرّد توثيق، بل هي المُدخل الآلي الذي يوجّه أكثر جزأين تعقيدًا في الاستيراد: إذ تقرأ تمريرة **إعادة التخطيط** قيمة `path` من كل حافة لتحدّد بدقّة الحقل الذي يجب إعادة كتابته من معرّف بيئة المصدر إلى معرّف بيئة الهدف، ويقرأ **الترتيب الطوبولوجي** العلاقات التي تمثّلها الحواف ليحدّد ترتيب الإنشاء، بحيث يُكتب كل مورد دائمًا بعد جميع الموارد التي يحتاج إليها. وتُستورَد الحزمة حتى إن افتقرت مواردها إلى حواف مطابقة، لكن بترتيب اعتباطي، مع بقاء مراجعها المتقاطعة مشيرةً إلى معرّفات لا وجود لها في الهدف.

توثّق هذه الصفحة صيغة الحافة وكيف يستخدمها محرّك الاستيراد. وتغطّي نظرة التحزيم العامة سير العمل المرتبط بها وواجهة المخطط الحيّة التي تقدّمها `dependency-service`؛ أما صيغة الحزمة فتغطّي الأرشيف الذي يحتوي على `dependencies.json`.

من أين تأتي الحواف

تتولّى خدمة `dependency-service` إدارة المخطط. وتستخرج الحواف من لقطات الموارد باستخدام مخطط تصريحي لكل نوع مورد يفسّره محرّك للمخططات، إلى جانب مستخرِجات مخصّصة للبنى التي لا تستطيع القواعد التصريحية معالجتها. ثم تعيد بناء الحواف استجابةً للأحداث كلما تغيّرت الموارد. ويحدّد المعدود `ResourceType` أنواع عقد المخطط، ويضمّ ثماني عشرة قيمة معرَّفة. وهذه أنواع عقد *المخطط*، وليست مجموعة الموارد القابلة للتصدير: فـ `Dashboard` عقدة في المخطط، لكنها لا تُكتب أبدًا في مجلّد `resources/` بالحزمة (راجع صيغة الحزمة).

تحفظ الخدمة الحواف داخليًّا على هيئة مستندات MongoDB مسطّحة (`fromType` / `fromId` و `toType` / `toId`)، وتحمل هذه المستندات أيضًا بصمة اللقطة `resourceVersion` التي تضمن أن تعطي إعادة البناء النتيجة نفسها مهما تكرّرت. ولا يصل هذا الشكل المسطّح إلى الحزمة أبدًا؛ إذ يحتوي `dependencies.json` على **كائن النقل المتداخل** الذي تقدّمه الخدمة (`docToEdge`)، وتُلتقط حالته وقت التصدير لتمثيل إغلاق الحواف الصادرة للتحديد المُصدَّر: يجتاز المُصدِّر الحواف الصادرة من الجذور المحدّدة اجتيازًا بالعرض، **ولا توسّع الاجتيازَ إلا الحواف الصلبة**؛ أما الليّنة فتُسجَّل ولا تُتبَع.

تشريح الحافة

تحدّد الواجهة `DependencyEdge` عقد النقل. وفيما يلي حافتان من معرضٍ حقيقي، واحدة من كل نوع:

```
[
  {
    "from": { "type": "DataView", "id": "5f0c33…", "normalizedName": "open_tickets" },
    "to":   { "type": "DataEntity", "id": "a91d78…", "normalizedName": "ticket" },
    "kind": "MEMBERSHIP",
    "hard": true,
    "path": "dataEntity",
    "resolvedVia": "DIRECT_ID",
    "fromActive": true,
    "fromArchived": false
  },
  {
    "from": { "type": "DataView", "id": "5f0c33…", "normalizedName": "open_tickets" },
    "to":   { "type": "DataCollection", "id": "c22e41…", "normalizedName": "priorities" },
    "kind": "REFERENCE",
    "hard": false,
    "path": "dataSource.collection",
    "resolvedVia": "DIRECT_ID"
  }
]
```

الحقل	النمط	المعنى
`from`	`ResourceRef`	المورد الذي **يحمل** المرجع، أي المورد المعتمِد.
`to`	`ResourceRef`	المورد المعتمَد عليه.
`kind`	معدود	`MEMBERSHIP` أو `REFERENCE`؛ انظر أدناه.
`hard`	منطقي	يحدّد ما إذا كان وجود الهدف شرطًا لصلاحية المصدر. ويمنع أيضًا حذف الهدف في المخطط الحي.
`path`	نص	مسار JSON في **لقطة المورد المصدر** يحدّد موضع المرجع، مثل `"module"` أو `"columns[0].entityId"`.
`resolvedVia`	معدود	الاستراتيجية التي اكتُشفت بها الحافة؛ انظر أدناه.
`fromActive` / `fromArchived`	منطقي	علامتا دورة حياة المورد المصدر وقت الاستخراج.
`metadata`	كائن	سياق إضافي خاص بالنوع؛ اختياري.

مرجعا `from` و `to`

كلّ طرف كائن من نوع `ResourceRef`: يحدّد `type` (إحدى قيم `ResourceType`) و `id` هويته، ويُضمَّن فيه `normalizedName` والاسم المترجَم `name` للعرض وللمطابقة بالاسم عند التحقّق. وتصف العقدةَ ثلاث علامات اختيارية: `isImplicit` (مُدار من النظام وغير قابل للتصدير)، و `isExternal` (ينتمي إلى وحدة غير الوحدة موضع الاستعلام)، و `isService` (كيان من نوع الخدمة). أما العلامة الرابعة `withChildren` فلا معنى لها إلا في *طلبات* التصدير، حيث تطلب توسيع شجرة `MEMBERSHIP` الفرعية للمرجع.

`kind` – عضوية أم مرجع

للمعدود `DependencyKind` قيمتان فقط، ويُفرَض الالتزام بهما أيضًا بوصفهما قيمتَي معدود في المستند المخزَّن:

- **`MEMBERSHIP`** – المصدر *ينتمي إلى* الهدف: كيان إلى وحدته (`path: "module"`)، أو معرض إلى كيانه (`path:"dataEntity"`، وهي صلبة). ويجتاز `withChildren` حواف العضوية عندما يوسّع تصدير وحدة أو كيان ليشمل شجرته

الفرعية.

- **`REFERENCE`** – مرجع متقاطع من حمولة المصدر إلى مورد مستقل: `dataSource.collection` في معرض، أو الاستعلام الهدف لتفاعل، أو معرّف كيان في قطعة.

`hard` – ما يجب وجوده وما يجوز تصفيره

تمثّل الحافة **الصلبة** شرطًا للصلاحية: فلا يعمل المصدر دون هدفه. ويتبعها إغلاق التصدير، ويراعيها منطق التخطّي التسلسلي في الاستيراد (فإذا تُخطّي مورد أو أخفق، وُسِم بالحالة `cascade_skipped` أيضًا كلُّ مورد يعتمد عليه بحافة صلبة)، وتُبقيها تمريرة الفرز المقتصرة على الحواف الصلبة. أما الحافة **الليّنة** فتمثّل تفضيلًا مسجَّلًا: إن لم يكن للهدف تخطيط في مساحة الهدف، **صفّر الاستيرادُ القيمةَ عند `path` الحافة** بدلًا من ترك معرّف مصدر معلّق، مع استثناءات مقصودة مثل `dataSource.query` الذي يُترك كما هو كي تُخفق خدمة البيانات بخطأ ذي معنى، هو `QUERY_NOT_FOUND` ، بدلًا من خطأ في النمط.

و`Since dev` لم تعد فئة واحدة من الحواف الليّنة التي تشير إلى خارج الحزمة تُصفَّر بصمت: إذ يُعلَن عند النشر عن مرجع **قطعة الصفحة** إلى معرض أو مورد آخر خارج الحزمة، ويُعرض على المثبِّت بوصفه ربطًا اختياريًا؛ ولا يُصفَّر إلا إذا تُرك دون ربط (`isMappablePageWidgetReference` ؛ انظر قطع الصفحات التي تشير إلى خارج الحزمة).

وتُحدَّد الصلابة لكل قاعدة على حدة في مخططات الاستخراج، وتُعدَّل عندما يخالفها السلوك الفعلي: فحافة `dataSource.query` لمعرضٍ من نوع الاستعلام معلنة حاليًا **صلبة**، تحديدًا لأن اعتمادها حافةً ليّنة أنتج حزمًا استوردت المعرض دون استعلامه المصدر.

`path` – أين يقع المرجع

`path` مسار JSON داخل اللقطة المحزَّمة للمورد *المصدر*، بصيغة `lodash.get` / `set` ، مثل `"module"` أو `"dataSource.collection"` أو `"columns[0].entityId"` . وهو المعلومة الوحيدة التي تتيح إعادة التخطيط بدقّة: فلولاه لاضطر المستورِد إلى تخمين الحقول النصّية في الحمولة التي تمثّل معرّفات. وتشارك الحافة التي لا تحتوي على `path` في الترتيب والتخطّي التسلسلي، لكنها لا تتيح تحديد أي موضع لإعادة الكتابة.

`resolvedVia` – كيف اكتُشفت الحافة

يسجّل المعدود `DependencyResolution` استراتيجية الاستخراج. وله غرض معلوماتي عند الاستيراد، إذ تتعامل إعادة التخطيط مع جميع الاستراتيجيات بالطريقة نفسها، لكنه مفيد عند تدقيق سبب وجود حافة:

القيمة	الاستراتيجية
`DIRECT_ID`	تحمل الحمولة معرّف UUID للهدف مباشرةً عند `path` .
`NORMALIZED_NAME`	حُدِّد الهدف من اسم عند `path` ، لا من معرّف.
`REFERENCE_PROPERTY`	القيمة عند `path` كائن خاصية مرجعية (`{dataEntity, dataEntry}`): تستهدف الحافة الكيان، بينما يتكفّل مسار معالجة مراجع القيم بمعرّف المُدخل.
`PIPELINE_STAGE`	معلَنة في المعدود، لكن لا يوجد مستخرِج يُصدرها حاليًا.

عقدة المحصّلة وحافّتاها `Since 1.10`

يُعدّ `DataEntityAggregate` ، وهو المورد الذي تقوم عليه قائمة **المحصّلات (Aggregators)** في الكيان، أحدث أنواع العقد. وتمثّل الحافّتان اللتان أُضيفتا معه أوضح مثال عملي على التمييز بين الصلب والليّن أعلاه، إذ اختيرت صلابة كل منهما لسبب يناقض الآخر.



حافّة الكيان صلبة. فالمحصّلة تنتمي إلى كيان واحد بالضبط ولا معنى لها بدونه؛ لذا فإن الحافّة `DataEntityAggregate.dataEntity → DataEntity` من نوع `MEMBERSHIP` وموسومة بأنها صلبة. وتترتّب عليها الآثار المعتادة للصلابة: فتصدير كيان مع أبنائه يضمّ محصّلاته معه، ولا يمكن حذف الكيان الذي تعتمد عليه.

وحافّة المعرض ليّنة عمدًا. فالحافّة `DataView.viewConfiguration.aggregates[] → DataEntityAggregate` من نوع `REFERENCE`، وهي ليّنة عن قصد: إذ يجب أن يؤدّي حذف محصّلة إلى إزالتها بصمت من كل معرض فعّلها، ولو كانت الحافّة صلبة لمنعت ذلك الحذف، وهو نقيض السلوك المطلوب. وتتيح هذه الحافّة للاستيراد إعادة تخطيط معرّفات المحصّلات المفعَّلة إلى معرّفات محصّلات المساحة الهدف. وحين لا يتوفّر تخطيط لأحد تلك المعرّفات، تنطبق عليه قاعدة الحافّة الليّنة فيُسقَط من المصفوفة، فيعرض المعرض محصّلة أقلّ بدلًا من أن يُخفق.

وتجعل المصفوفة هذه الحافّة جديدة من حيث البنية: فعناصرها هي معرّفات الأهداف، وهو شكل لا تستطيع القواعد الأوّلية القائمة معالجته؛ إذ تنهي قاعدة المعرّف المباشر التنفيذ مبكرًا عند مصادفة مصفوفة، بينما تجتاز قاعدة التكرار على المصفوفات الكائنات. لذلك أُضيفت قاعدة أوّلية جديدة، هي `DirectIdArray`، تُصدِر حافّة لكل عنصر بمسار مفهرس وفق الصيغة المعتادة (`viewConfiguration.aggregates[0]`).

والعاقبة التشغيلية التي يسهل أن تفوتك: المخطط المبني قبل هذا التغيير لن يُظهر المحصّلات البتّة حتى يُعاد بناؤه. فالمخطط يُحدَّث تدريجيًا استجابةً لأحداث دورة حياة الموارد، وإضافة قاعدة لا تعيد كتابة الحواف التي حُسبت سابقًا. وإلى أن تُشغَّل إعادة البناء، يُغفل عرض تبعيات الكيان محصّلاته، ولا يُظهر المعرض أي تبعية للمحصّلات التي يعرضها؛ فيبدو أن الميزة غائبة، بينما المخطط لم يُحدَّث.

حواف أخرى أُضيفت في 1.10 `Since 1.10`

أضاف خط الإصدار 1.10 إلى المخطط نوعين جديدين من العقد وحواف لعدة إجراءات تفاعل. ويغيّر كل صف الموارد التي يضمّها التصدير والمراجع التي يعيد الاستيراد ربطها؛ وكما في حافّتَي المحصّلة أعلاه، **لا يُظهر المخطط المبني قبل التغيير أياً منها إلى أن يُعاد بناؤه**.

الحافة (مسار المصدر ← الهدف)	النوع	صلبة؟	معناها
`QuickFilterGroup.dataEntity → DataEntity`	عضوية	صلبة	مجموعة المرشّحات السريعة تنتمي إلى كيانها وتنتقل معه.
`TagGroup.dataEntity → DataEntity`	عضوية	صلبة	وينطبق ذلك أيضًا على مجموعة الوسوم.
`DataQuery.quickFilterGroup → QuickFilterGroup`	مرجع	ليّنة	حذف المجموعة يفكّ تجميع استعلاماتها فحسب.
`DataQuery.tag.tagGroup → TagGroup`	مرجع	ليّنة	حذف المجموعة يترك الوسم بلا مجموعة.
`CounterWidget.value.moduleId → DataModule`	عضوية	صلبة	أصبحت قطعة العدّاد تنتمي إلى وحدتها.
إجراء التفاعل *تنفيذ وظيفة*: `options.functionId → Function`	مرجع	صلبة	الوظيفة التي يشغّلها الإجراء.
إجراء التفاعل *تنفيذ إجراء مخصّص*: `options.customActionId → DataReaction`	مرجع	صلبة	التفاعل الذي يشغّله الإجراء؛ وترفض reaction-service الإنشاء إذا تعذّر حلّ معرّفه.
الإجراء نفسه: `options.entryToRun.relationId → DataRelation` و`options.entryToRun.targetEntityId → DataEntity`	مرجع	ليّنة	كلاهما اختياري.
الإجراء نفسه: `options.parameters[].value` ← مُدخلات	مرجع	ليّنة	قيم مُدخلات تخضع للقواعد نفسها المطبَّقة على إجراءي الإنشاء والتحديث.

الحافة (مسار المصدر ← الهدف)	النوع	صلبة؟	معناها
إجراء التفاعل *حذف مُدخل مؤرشف*: `options.relation.relationId` و `options.querySource`	مرجع	ليّنة	القواعد نفسها المطبَّقة على *أرشفة مُدخل*.
إجراءا التفاعل *منح إذن / سحب إذن*: `options.resource.entitlementId` ← وحدة أو كيان أو معرض أو صفحة أو تقرير	مرجع	صلبة	يُحدَّد نوع الهدف من نوع الاستحقاق؛ ولا توجد حافة لاستحقاق `System` .
الإجراءان نفسهما، مع صاحب صلاحية **ثابت**: `options.principal.source.value` ← كيان USER أو GROUP	مرجع	ليّنة	يُعاد ربطه عبر مسار مراجع القيم، كما في *تبديل تفعيل المستخدم*.
إجراء التفاعل *Send SMS* و*Send WhatsApp*، مع مستلم **ثابت** من مستخدم أو مجموعة: `options.recipients[].source.value` ← كيان USER أو GROUP	مرجع	ليّنة	تخدم قاعدة واحدة الإجراءين. وقبل أن تشمل *Send WhatsApp*، لم تكن تُنشأ حواف للمستلمين الثابتين لتفاعل واتساب، وكان التفاعل يغيب عن الإشعارات المصدَّرة. ويُعاد ربطه عبر مسار مراجع القيم.

ويرافق ذلك تصحيحان أصغر نطاقًا. فقد أصبحت حواف **القيمة الافتراضية** و**العملة** للحقول تسجّل مسار الحقل كاملًا، مما يتيح لإعادة الربط أن تعيد كتابة الحقل الذي يحمل المرجع بدقّة. وأُزيلت القاعدة القديمة لإجراء التفاعل *تحديث كيان بيانات*، لأن إجراءَي إنشاء كيان البيانات وتحديثه لم يعملا قط، وأُزيلا من المنتج.

كيف تقود الحواف إعادة التخطيط

عند التنفيذ، يجمّع المستورِد حواف الحزمة بحسب `from.id` ، فيمنح كل مورد قائمة حوافه الصادرة، ويبني جدول معرّفات يخصّص مسبقًا معرّف هدف لكل مورد سينشئه. وهذا ما يتيح إعادة كتابة المراجع *قبل* أن يوجد المورد المشار إليه فعليًا. ثم تمرّ `RemappingService.remap` على الحواف الصادرة لكل مورد:

1. انسخ الحمولة نسخًا عميقًا (وإن وُجد تجاوز محفوظ، فقد حلّ محلّ الأصل المحزَّم قبل هذه النقطة؛ إذ تُطبَّق التجاوزات *قبل* إعادة التخطيط).
2. لكل حافة تحتوي على `path` : اقرأ القيمة عند `path` ، وابحث عن `to.id` في جدول المعرّفات؛ فإن **طابقت** القيمة الحالية معرّف المصدر مطابقة تامة، فاكتب المعرّف المخطَّط مكانها عند `path` . أما القيمة غير المطابقة فتُسجَّل وتُتخطّى؛ إذ يمنع حارس المطابقة التامة إفساد حقل يشبه المعرّف لكنه ليس معرّفًا.
3. تُعامل كائنات الخاصية المرجعية معاملة خاصة: حين تكون القيمة عند `path` كائن `{dataEntity, …}` ، لا يُعاد تخطيط سوى العضو `dataEntity` فيه؛ أما معرّف المُدخل المضمّن فيتولّاه مسار معالجة مراجع القيم.
4. الحافة الليّنة التي لا يتوفّر تخطيط لهدفها **تُضبط القيمة عند `path` فيها على `null`** (عدا الاستثناءات أعلاه)؛ ويُعاد تخطيط معرّف المورد `id` نفسه أخيرًا.

ثم تعالج تمريرتان احتياطيتان محدودتا النطاق ما فات التمريرة المعتمدة على المسارات: تُعيد الأولى الكتابة بمطابقة تامة للمعرّفات التي تحدّدها الحواف الصادرة لهذا المورد (فتغطّي الحواف التي تغيّر فهرس المصفوفة المسجَّل فيها)، وتطبّق الثانية بدائل التبعيات غير المخطَّطة التي هيّأها المستخدم. وتستبدل كلتاهما السلسلة كاملةً عند تطابقها التام، ولا تستبدلان سلاسل جزئية أبدًا.

ويترتّب على ذلك لكل من ينتج حزمة أو يحوّلها: **المعرّف الذي تغيّره في حمولة بلا حافة مطابقة يبقى دون إعادة كتابة**، و**الحافة التي لم يعد `path` فيها يشير إلى المعرّف الذي تسمّيه تُتخطّى**؛ ويحدث ذلك بصمت وعن قصد.

كيف تقود الحواف ترتيب الاستيراد

يكتب المُصدِّر `orderedResources` في `index.json` بترتيب طوبولوجي، لكن المستورِد لا يعتمد على هذا الترتيب، بل يعيد الفرز في كل تنفيذ باستخدام حواف الحزمة، لأن الحزم الأقدم صُدِّرت دون فرز (الخطوة 4e).

ويستخدم الفرز خوارزمية Kahn، التي تعمل في ثلاث تمريرات كحدّ أقصى:

1. **كل الحواف، صلبها وليّنها.** هذا هو الترتيب المثالي: فحتى الأهداف الاختيارية تُكتب قبل الموارد التي تشير إليها، فيُحَلّ المرجع الليّن بدلًا من تصفيره.
2. **التراجع إلى الصلب وحده.** إن كشفت التمريرة الأولى دورة (موردان يشير كل منهما إلى الآخر بحافة ليّنة)، أُسقطت الحواف الليّنة وأُعيد الفرز؛ فكسر الدورة الليّنة آمن تحديدًا لأن المراجع الليّنة قابلة للتصفير وقت إعادة التخطيط.
3. **طيّ المكوّنات قوية الاتصال.** تُطوى الدورة *الصلبة* المتبقّية في مكوّن قوي الاتصال (خوارزمية Tarjan)، فيبقى المكوّن بكامله مرتّبًا قبل الموارد المعتمِدة عليه. أما الدورات الصلبة بين الكيانات فتُكسر عند التنفيذ بإنشاء أحد الأعضاء في صورة هيكل، ثم ترقيع المراجع العابرة للدورة بعد وجود جميع الأعضاء.

وتُستخدم علاقات الحواف نفسها، بعد عكس اتجاهها، لتشغيل **التخطّي التسلسلي**: فقبل أي كتابة، يبني المستورِد فهرس تبعيات عكسيًا من الحواف، ويسم كل مورد يعتمد بحافة صلبة على مورد متخطًّى أو فاشل بالحالة `cascade_skipped`، ويطبّق ذلك بصورة متعدّية (`buildReverseDeps` / `applyCascadeSkip`).

قواعد لمنتج الحزمة

يتلخّص ما سبق في أربع قواعد لمن يولّد `dependencies.json` أو يحوّله؛ أما المسارات العملية لذلك فترد في صفحة توليد حزمة:

1. **كل مرجع داخلي متقاطع يحتاج حافة**، تتضمّن `from` و`to` و`path` الدقيق للحقل الذي يحمل المرجع. فغياب الحافة يعني غياب إعادة الكتابة وضمان الترتيب.
2. **أعلن الصلابة بصدق.** الحافة الصلبة إلى مورد غائب عن الحزمة وبيئة الهدف معًا لا تؤدّي إلى إخفاق صريح، بل تتسبّب في تخطّي الشجرة المعتمِدة عليه تسلسليًا. أما الحافة الليّنة إلى مورد غائب فتُصفَّر.
3. **أبقِ `path` متزامنًا مع الحمولة.** إن نقل التحويلُ الذي تجريه عنصرًا في مصفوفة، وجب تحديث الفهرس المسجَّل في `path` تبعًا لذلك؛ وإلا تخطّى حارس المطابقة التامة إعادة الكتابة دون أن يُفشل الاستيراد.
4. **لا تتّكل على `orderedResources` وحدها.** يعيد المستورِد اشتقاق الترتيب من الحواف؛ لذا فإن اتساق مجموعة الحواف داخليًا أهم من الترتيب الذي كتبت به الفهرس.

توليد حزمة

لنبدأ بإجابة صريحة عن السؤال الذي يدفع القارئ عادةً إلى هذه الصفحة: **ملفّ JSON واحد قائم بذاته يحوي كل تعريف يحتاجه نظام عامل ليس شيئًا تنتجه أسس ولا تقبله.** فالحزمة ملفّ ZIP متعدّد الأجزاء يجب أن تتوافق أجزاؤه – البيان، والفهرس، وحواف التبعيات، وحمولات الموارد – بعضها مع بعض، ولا يوجد في المنصّة ما يُصدِّر أو يستهلك ملفًّا واحدًا يضمّ كل شيء (يحدّد `services/packaging-service/src/libs/zip-utils.ts` هذا العقد، وتوثّقه صفحة صيغة الحزمة). وأقرب ما يتاح إلى ملفّ JSON واحد هو آلية النقل المنفصلة للوحدات والكيانات في خدمة البيانات؛ إذ تستخدم ملفًّا واحدًا بالفعل، لكنه لا يحمل سوى الوحدات والكيانات – راجع نقل الوحدة بصيغة JSON. وقد أُطلق محرّك التحزيم الذي ينتمي إليه هذا العقد ضمن سلسلة إصدارات **1.8** (`Since 1.8`)؛ أما في عمليات النشر الأقدم فلا تتوافر هذه المسارات ولا واجهة التصدير.

تتوافر بدلًا من ذلك ثلاثة مسارات عملية لإنتاج حزمة أو تحويلها أو تخصيصها باستخدام معاملات. وتوثّقها هذه الصفحة بالترتيب الذي يُنصح باتّباعه عند المفاضلة بينها.

اختيار المسار

المسار	مدعوم؟	متى تلجأ إليه
1. تصدير › تحويل › إعادة استيراد	نعم – المسار الرئيس	تريد نسخة معدَّلة أو مصوغة في قالب من موارد موجودة بالفعل في بيئة فعلية.
2. تجاوزات الحمولات على جلسة استيراد	نعم – المدخل البرمجي المعتمَد	تريد حقن قيم خاصة بالبيئة في حزمة وقت الاستيراد، دون إنشاء نسخة متفرّعة من ملفّ ZIP.
3. التأليف اليدوي من الصفر	ممكن، غير مدعوم	تولّد موارد لم يسبق أن احتوتها أي بيئة، وتقبل التكاليف المبيّنة أدناه.

ويمكن الجمع بين المسارين 1 و2: فالمسار الآلي الأكثر متانة للترقية المتكرّرة بين البيئات هو تصدير حزمة مرجعية، ثم تخصيصها لكل بيئة هدف عبر واجهة التجاوزات.

المسار 1 – تصدير، تحويل، إعادة استيراد

هذا هو المسار الوحيد الذي يبدأ من ناتج مضمون الصلاحية. صدّر حزمة مرجعية من بيئة مصدر، وفكّ ضغطها، وعدّل JSON، ثم أعد بناء الأرشيف واستورد الناتج. ستحصل بذلك على ملفّات `manifest.json` و `index.json` و `dependencies.json` صحيحة، ولن تحتاج إلا إلى الحفاظ على توافقها مع التغييرات التي أجريتها.

رحلة الذهاب والإياب

نقاط النهاية أدناه هي مسارات التصدير والاستيراد في خدمة التحزيم؛ أما الأعراف – غلاف `{success, result}` والرموز وأشكال الأخطاء – فتغطّيها صفحة استدعاء واجهة HTTP API، ولن نكرّرها هنا.

اختيار الموارد من واجهة الإدارة، `Since 1.8` . قبل استدعاء نقطة نهاية التصدير أدناه، يختار المسؤول عادةً الوحدات أو الكيانات أو الصفحات المُراد تضمينها من شجرة خانات اختيار في شاشة التحزيم ضمن واجهة الإدارة. وقد أُصلِح خللٌ في دورة الإصدار هذه: كان إلغاء تحديد الفرع الأخير المتبقّي لحاوية يُلغي تحديد الحاوية نفسها أيضًا، مما كان يتعذّر معه تصدير «هذه الوحدة أو هذا الكيان وحده، بلا فروع». أما الآن فتتتبّع الشجرة حالة تحديد الحاوية ذاتها بصورة منفصلة عن حالتها المُشتقّة من وجود أيّ فرع محدَّد. وما زال تحديد حاوية أو إلغاء تحديدها يسري على شجرتها الفرعية كاملة، كما كان من قبل.

```
export BASE="https://acme.example.com"
export TOKEN="<your API token>"

# 1 — start an export job
curl -s -X POST "$BASE/api/packaging/v1/export" \
  -H "x-app-token: $TOKEN" -H "content-type: application/json" \
  -d '{"resources":[{"type":"DataModule","id":"<module-id>","withChildren":true}],
       "name":{"en":"crm-reference"}}'
# → {"success":true,"result":{"jobId":"export-<exportId>","exportId":"<exportId>"}}
#   (Since 1.10 the export is also kept as an export record: GET /v1/exports/<exportId>)

# 2 — poll until the job completes (note: the job's own result is nested
#     inside the envelope's result)
curl -s "$BASE/api/packaging/v1/export/export-<exportId>/status" -H "x-app-token: $TOKEN"
# → {"success":true,"result":{"status":"completed",
#      "result":{"fileId":"<fileId>","name":"crm-reference.zip", …}}}

# 3 — download the ZIP. The export stores it in storage-service as a
#     temporary file; the response is the raw file, not enveloped JSON.
curl -s "$BASE/api/storage/v2/files/download/<fileId>?temp=true" \
  -H "x-app-token: $TOKEN" -o package.zip

# 4 — unzip, transform the JSON under resources/, rezip

# 5 — upload the transformed archive, again as a temporary file
curl -s -X POST "$BASE/api/storage/v2/files?temp=true" \
  -H "x-app-token: $TOKEN" -F "file=@package-transformed.zip"
# → {"success":true,"result":[{"id":"<newFileId>", …}]}

# 6 — open an import session against it, then configure and execute
curl -s -X POST "$BASE/api/packaging/v1/import/validate" \
  -H "x-app-token: $TOKEN" -H "content-type: application/json" \
  -d '{"fileId":"<newFileId>","fileName":"package-transformed.zip"}'
# → {"success":true,"result":{"sessionId":"…","validationReport":{…},
#      "updateIdentifier":"…"}}
```

بعد الحصول على `sessionId`، اتّبع مسار الجلسة القياسي: هيّئ التخطيطات، ونفّذ الاستيراد، واستقصِ الحالة، كما هو موثَّق في صفحة واجهة الاستيراد وفي نظرة التحزيم العامة. والتصدير مهمّة تُنفَّذ في الخلفية؛ إذ يُدرجها `POST /v1/export` في الصفّ ويعيد معرّف مهمّة، ويُشرح هذا النمط العام في العمل الطويل الأمد. أما علامة `temp=true` في كلا استدعاءَي التخزين فتتوافق مع طريقة تخزين المنصّة نفسها لملفّات الحزم؛ إذ يمرّرها كل استدعاء تنزيل في التحزيم، ويرفع `services/packaging-service/src/services/package-export.service.ts` الملفَّ بعلامة الملفّ المؤقّت.

و`Since 1.10` أصبح `name` في التصدير كائنًا مترجمًا و**مطلوباً** (ويؤدّي غيابه إلى `400 EXPORT_NAME_REQUIRED`)؛ ويُسمّى ملفّ ZIP بقيمته الإنجليزية، وإلا فبالعربية. ويحتاج المستدعي إلى صلاحية Developer على كل جذر مُصدَّر، ولا يمكن تصدير الموارد التي تنتمي إلى حزمة مثبّتة أصلًا (`403 EXPORT_INSTALLED_RESOURCE`) — انظر صلاحية التصدير.

إبقاء الأجزاء متوافقة

يظلّ التحويل آمنًا ما دامت مُدخلات الأرشيف متوافقة فيما بينها. وفيما يلي القواعد، وكلٌّ منها مستمدّ من رماز جهة الاستيراد:

- **أسماء الملفّات من `nameMap` لا من الأسماء.** خريطة `nameMap` في `index.json` (`id → normalizedName`) هي المفتاح المرجعي لملفّات `resources/{Type}/{name}.json` ؛ إذ يُلحِق التصدير المعرّف بالأسماء المتصادمة، ويتحقّق المستورِد بالمعرّف من كل مورد يعثر عليه بالاسم. فإذا أعدت تسمية ملفّ أو غيّرت `normalizedName` لمورد دون تحديث `nameMap` ، تعذّر الوصول إلى المورد.
- **مكوّنات المسار صارمة.** يجب أن يطابق كل مكوّن النمط `^[a-zA-Z0-9_.-]+$` ؛ ويُعدّ اشتقاق اسم ملفّ من اسم ظاهر للمستخدم يحتوي على مسافات أو حروف غير لاتينية السبب الأكثر شيوعًا لرفض أرشيف أُعيد بناؤه عند الاستخراج.
- **المراجع المتغيّرة تحتاج حوافّ مطابقة.** إذا أضاف التحويل مرجعًا إلى معرّف أو نقله، وجب أن يحتوي `dependencies.json` على حافة يشير مسارها `path` إلى ذلك المرجع، وإلا فلن يُعاد تخطيط المعرّف أبدًا. وتشرح صفحة مخطط التبعيات بدقّة كيف توجّه الحواف إعادة الكتابة والترتيب.
- **الحقول البنيوية ركائز لا يُعبث بها.** فالحقول التي تشير إلى أصل المورد (`module` و `dataEntity` وأشباهها) هي المفاتيح التي تعتمد عليها إعادة التخطيط وحلّ النطاق؛ ونقل مورد بين أصلين يتطلّب إعادة تأليفه، ولا يقتصر على تعديل حقل.
- **يجب أن تبقى أعداد البيان صحيحة.** يُفحص شكل `resourceCount` و `resourceTypes` عند الاستخراج (`assertManifestShape`)، ويجب أن يكون `schemaVersion` إصدارًا مدعومًا – من `1.0` إلى `1.4` في الإصدار 1.10، و `1.0` أو `1.1` فقط في بيئة هدف أقدم. واحتفظ بالإصدار الذي وُسم به التصدير ما لم تُضِف الجزء الذي يصفه ذلك الإصدار أو تحذفه؛ انظر إصدارات المخطط.

المسار 2 – واجهة تجاوز الحمولات

هذه هي الطريقة المعتمَدة لتغيير *محتوى* الحزمة برمجيًا، وهي لدى معظم القرّاء الإجابة الفعلية عن سؤال «هل أستطيع توليد هذا؟». ففي جلسة استيراد نشطة، بين التحقّق والتنفيذ، يمكن قراءة حمولة كل مورد محزَّم واستبدالها عبر مسار واحد:

```
GET    /api/packaging/v1/import/{sessionId}/resource/{sourceType}/{sourceId}/payload
PATCH  /api/packaging/v1/import/{sessionId}/resource/{sourceType}/{sourceId}/payload
DELETE /api/packaging/v1/import/{sessionId}/resource/{sourceType}/{sourceId}/payload
```

`sourceType` هو اسم `ResourceType` للمورد، و `sourceId` معرّفه *في الحزمة*: أي معرّفات بيئة المصدر كما يسردها تقرير التحقّق للجلسة. ولا يُعدَّل ملفّ ZIP نفسه أبدًا؛ فالتجاوز يُحفظ في الجلسة (`ResourceOverride`) ويُطبَّق **بدل الحمولة المحزَّمة، قبل إعادة التخطيط**، عند التنفيذ. ولهذا الترتيب أهميته: فأنت تكتب التجاوز باستخدام معرّفات بيئة المصدر، تمامًا كما لو كان قد حُزِّم، ثم يعيد المحرّك كتابته كسائر الحمولات. وهكذا تُخصَّص حزمة مرجعية لكل بيئة – بإعادة توجيه نقطة نهاية، أو ضبط عتبة، أو حقن إعدادات خاصة بالقاطن – مع بقاء الناتج ملفًّا واحدًا دون نسخ متفرّعة.

قراءة حمولة مورد

يعيد `GET` الحمولة التي سيستخدمها التنفيذ فعلًا: التجاوز إن وُجد، وإلا فالحمولة الأصلية المحزَّمة المستخرَجة من ZIP. وتأتي الاستجابة بالشكل (`GetResourcePayloadResponse`):

الحقل	المعنى
`sourceType` و `sourceId` و `normalizedName`	هوية المورد في الحزمة.
`payload`	ما سيُستورَد: التجاوز حين تكون قيمة `isOverride` هي `true` ، وإلا فالحمولة الأصلية المحزَّمة.
`isOverride`	ما إذا كان هناك تجاوز نشط حاليًا.

الحقل	المعنى
`packagedPayload`	الحمولة الأصلية المحزَّمة، ولا تكون موجودة **فقط** إلا أثناء وجود تجاوز نشط، ليتمكّن العميل من المقارنة ومعاينة نتيجة إعادة التعيين دون استدعاء ثانٍ.
`editedBy` و `editedAt`	بيانات نسبة التجاوز النشط إلى صاحبه ووقت تعديله.

إذا كان المورد موجودًا في الجلسة وغائبًا عن الأرشيف، يعيد الطلب `404`؛ وتُرفض طلبات القراءة بالرمز `409` أثناء تنفيذ الاستيراد.

حفظ تجاوز

يتلقّى `PATCH` القيمة `{payload, updateIdentifier}` (بالشكل `SaveResourcePayloadRequest`) ويحفظ الحمولة في الجلسة بعد التحقّق منها. كما يعيد حفظُ تجاوز لمورد سبق أن **أخفق** (أو جرى تخطّيه تسلسليًا) حالته إلى الانتظار، لتشمله *إعادة محاولة* الجلسة؛ وبذلك تتيح واجهة التجاوزات دورة إصلاح للاستيراد الفاشل. وتُسجَّل كل عملية كتابة في سجلّ التدقيق مع الجلسة والمورد والكاتب وحجم الحمولة.

ويرفض المتحقّق ما يلي:

الشرط	الاستجابة
حمولة ليست كائن JSON صِرفًا، أو تحتوي على مفاتيح تلويث النموذج الأولي	`400`
حمولة مُسلسَلة تتجاوز 2 م.ب (`MAX_OVERRIDE_PAYLOAD_BYTES`)	`413`
حقل بنيوي تغيّر عن اللقطة المحزَّمة	`400`

ما لا يستطيع التجاوز تغييره

تُؤخذ بصمة الحقول البنيوية – الحقول التي تشير إلى الأصل وتعتمد عليها إعادة التخطيط وحلّ التبعيات – عند التحقّق (`structuralSnapshot`)، وتُقفل (`STRUCTURAL_IMMUTABLE`) في الملفّ نفسه). فتغييرها كان سيفصل الحمولة عن حواف تبعيات الحزمة دون تنبيه:

نوع المورد	الحقول الثابتة
`DataEntity`	`module`، `dataModule`
`DataView`	`module`، `dataModule`، `dataEntity`، `definingEntity`
`DataQuery`، `DataReaction`، `DataCollection`، `StatisticQuery`	`dataEntity`، `definingEntity`
`DataRelation`	`dataModule`، `module`
`Announcement`	`entityId`

فالتجاوز يتيح تعديل *المحتوى لا البنية*. ولتغيير موضع مورد، استخدم إعدادات تخطيط الجلسة؛ ولتغيير ما تحتوي عليه الحزمة، استخدم المسار 1.

مسح تجاوز

يزيل `DELETE` التجاوز عند تمرير `updateIdentifier` الحالي بوصفه معامل استعلام، فيعود التنفيذ التالي إلى استخدام الحمولة المحزَّمة. وقد صُمّم مسح تجاوز غير موجود بحيث لا يُحدث أي تغيير ويعيد `200`، فلا تحتاج عملية «إعادة التعيين» المتكرّرة في رماز العميل إلى التحقّق من وجوده مسبقًا.

التزامن وحالة الجلسة

تخضع العمليات الثلاث لضوابط الجلسة نفسها: يجب أن تكون الجلسة مملوكة لمساحة المستدعي، وتُرفض عمليات الكتابة بالرمز `409` حين تكون الجلسة في حالة `importing` أو `validating`. وتعتمد التعديلات القفل التفاؤلي: إذ يجب أن تتضمّن كل عملية كتابة آخر قيمة لـ `updateIdentifier` تلقّاها العميل، وتعيد كل استجابة رمزًا جديدًا يحلّ محلّ السابق ويجب على العميل اعتماده. أما عدم التطابق فيؤدّي إلى استجابة `409` تكون فيها قيمة `error.code` هي `STALE_UPDATE_IDENTIFIER`، لتمييزها عن استجابة `409` الخاصة بضوابط الحالة. لذا يعمل المولِّد الذي يستخدم هذه الواجهة في حلقة: يقرأ الرمز، ويكتب، ويعتمد الرمز المُعاد، ثم يكرّر.

المسار 3 – التأليف اليدوي من الصفر

يمكن بناء حزمة دون تصدير حزمة أولًا؛ فالاستيراد يثق بأي أرشيف يجتاز التحقّق، أيًّا كان مصدره. لكن هذا المسار **غير مدعوم**، ولا توفّر المنصّة أدوات لتأليف الحزمة؛ فأنت عمليًا تعيد كتابة المُصدِّر بنفسك.

ما يجب عليك إنتاجه

جميع أجزاء صيغة الحزمة، مع ضمان توافقها بعضها مع بعض:

- ملفّ `manifest.json` يجتاز فحص البنية `assertManifestShape` – أي يتضمّن `schemaVersion` مدعومًا (`'1.0'` / `'1.1'`) و`createdAt` و`resourceCount` و`resourceTypes` – إضافةً إلى حقول هوية المصدر إذا أردت أن تنطبق ملفّات تعريف التخطيط المخزّنة.
- ملفّ `index.json` يمثّل فيه `orderedResources` ترتيبًا طوبولوجيًا صحيحًا وفق الحواف الصلبة، وتحدّد فيه `nameMap` اسم كل ملفّ موجود فعلًا تحت `resources/`.
- ملفّ `dependencies.json` تتضمّن حوافه قيم `path` صحيحة لكل مرجع داخلي، وإلا فلن تُعاد أي عملية تخطيط أو ترتيب (مخطط التبعيات).
- حمولة واحدة لكل مورد تحت `resources/{Type}/{name}.json`، تطابق كلٌّ منها مخطط نوعها. وتوثّق صفحة مخططات الموارد المستوى الأول؛ أما بقية المستويات، فالطريقة الموثوقة هي تصدير عيّنة فعلية ونسخ بنيتها.
- ولا شيء غير ذلك: فالمستوى الأعلى للأرشيف مقيّد بقائمة سماح، ويؤدّي وجود مُدخل مجهول إلى فشل الاستخراج (`UNKNOWN_ENTRY`). والتزم بحدود الأمان: 200 م.ب بعد فكّ الضغط، و10000 مُدخل، ومكوّنات مسار آمنة، ومنع عبور المسارات.

شبكة أمانك الوحيدة

يفحص المتحقّق الصارم عند مرحلة الاستخراج *الوعاء*: وجود المُدخلات المطلوبة، وصلاحية JSON، وسلامة بنية البيان، والتزام المستوى الأعلى بقائمة السماح. وإخفاقاته الأربعة المصنَّفة – `MISSING_ENTRY` و`INVALID_JSON` و`SCHEMA_VIOLATION` و`UNKNOWN_ENTRY` – هي الأخطاء الوحيدة التي تشير مباشرةً إلى الأرشيف الذي أنشأته؛ ومن يطوّر مولِّده بالتجربة والتكرار، فعليه أن يتوقّع مصادفتها جميعًا.

أما ما **لا** يفحصه فهو التوافق المتبادل الذي يضمنه المُصدِّر عادةً. فوجود مُدخل في `index.json` دون ملفّه لا يؤدّي إلى فشل الاستيراد، بل *يُتخطّى* المورد وتُتخطّى تسلسليًا الموارد التي تعتمد عليه عبر حواف صلبة. والحافة التي تحمل قيمة `path` خاطئة لا تُطلِق خطأً، بل يُترك المرجع دون إعادة كتابة أو تنبيه. أما الحمولة المخالفة لمخطط موردها، فتظهر في صورة إخفاق لكل مورد على حدة صادر عن خدمة البيانات عند التنفيذ، بمصطلحات بيئة الهدف لا بمصطلحات المولِّد الذي أنشأته. وهذا يؤكّد مجدّدًا سبب تفضيل المسارين 1 و2: ابدأ من ناتج أنشأته المنصّة، وخصّصه عبر الواجهة المصمّمة لهذا الغرض.

واجهة الاستيراد

تشرح صفحة التحزيم والتوزيع ما يفعله محرّك الاستيراد في كل مرحلة، وتعرض صفحة تصدير التطبيقات واستيرادها وتثبيتها [بناء التطبيقات] المعالجَ كما يراه باني التطبيقات. أما هذه الصفحة فتوثّق الواجهة نفسها بوصفها **عقدًا على مستوى الاتصال** لعميل يدير الاستيراد برمجيًا: كل نقطة نهاية للجلسات، مع جسمَي الطلب والاستجابة، وآلة حالات الجلسة كما تفرضها الخدمة بدقّة، وقواعد التزامن التي تمنع عمليتَي كتابة متزامنتين من إفساد جلسة واحدة.

توفّر `packaging-service` كل ما يرد هنا خلف بادئة البوّابة `/api/packaging` ؛ وتُكتب المسارات أدناه ابتداءً من `/v1/…` . وتوثّق صفحة استدعاء واجهة HTTP البرمجية الأعراف المشتركة – التوجيه، والمصادقة، والتقسيم إلى صفحات، وغلاف الاستجابة `{success, result}` – فلا تُكرَّر هنا. وكل جسم استجابة في هذه الصفحة هو العضو `result` من ذلك الغلاف، بلا استثناء. ويخضع كل استدعاء لشرطين:

- أن يكون الطلب مُصادَقًا عليه – فجميع متحكّمات التحزيم محميّة بـ `@authenticate` باستراتيجية Auth2 – وأن يقدّمه **مسؤول المساحة** (`Since 1.10` : تتطلّب جميع مسارات الاستيراد والتنفيذ والتجاوز وملفات تعريف التخطيط منحة Admin على مستوى النظام، أو استدعاء خدمة؛ وما عدا ذلك يُرفض بالرمز `403`):
- ومساحة هدف. ولا ترسلها أنت؛ إذ تستخلص البوّابة المساحة من مضيف الطلب وتضبط `x-space-id` بنفسها، مستبدلةً أيّ قيمة يرسلها العميل (القاطن والهوية). فوجّه الطلب إلى مضيف القاطن الخاص بالمساحة التي تستورد إليها. ويُرفض الطلب الذي يصل إلى `packaging-service` دون هذه الترويسة – أي طلب تجاوز البوّابة – بالرمز `400` ، ويُرفض طلب الوصول إلى جلسة تخصّ مساحة أخرى بالرمز `403` .

الجلسات ليست دائمة. يحذف فهرس TTL أي جلسة لم تُحدَّث طوال **سبعة أيام**؛ لذا سيجد العميل جلسته قد حُذفت إذا تركها في منتصف التهيئة مدةً أطول من ذلك.

آلة الحالات

تأخذ قيمة `status` في الجلسة إحدى خمس حالات: `validating` و `validated` و `importing` و `completed` و `failed` (`IMPORT_SESSION_STATUSES`). ولا توجد حالة `uploaded` : تبدأ دورة حياة الجلسة بالحالة `validating` أثناء تحقّق المرحلة الأولى، وتكون عادةً في الحالة `validated` حين يراها العميل للمرة الأولى.

وتحدّد الخدمة هذه الانتقالات بجدول صريح، لا بمجرد عرف متّبع، ويُنتظر من كل مسار في الرماز يكتب حالةً أن يتحقّق من هذا الجدول (`assertValidTransition`). وفيما يلي جميع الانتقالات:

الانتقال	من	إلى	مَن يُطلقه
`VALIDATE`	`validating`	`validated`	اكتمال تحقّق المرحلة الأولى (`POST /v1/import/validate`)
`START_IMPORT`	`validated`	`importing`	حجز `POST /…/execute` للجلسة
`COMPLETE`	`importing`	`completed`	إتمام عامل الاستيراد معالجة جميع التخطيطات، سواء وقعت أخطاء على مستوى الموارد أم لم تقع
`FAIL`	`validating` ، `validated` ، `importing`	`failed`	خطأ يتعذّر التعافي منه أثناء التحقّق أو التنفيذ
`RETRY`	`failed` ، `completed`	`importing`	`POST /…/retry` ، أو استدعاء `execute` على جلسة في حالة نهائية

الانتقال	من	إلى	مَن يُطلقه
`RECOVER_STUCK`	`importing`	`failed`	استعادة العامل للجلسة العالقة في `importing` بعد انقضاء مهلة التعافي (15 دقيقة، `IMPORT_STUCK_THRESHOLD_MS`)
`CANCEL`	`validated` ، `importing`	`failed`	`POST /…/cancel`

تترتّب على ذلك ثلاث نتائج ينبغي فهمها قبل كتابة عميل:

- `completed` **لا تعني النجاح الكامل.** يحدث الانتقال `COMPLETE` بعد معالجة جميع التخطيطات، وقد تتخلّلها أخطاء على مستوى الموارد؛ وتبيّن `importResult` وقيم `status` لكل تخطيط النتيجة الفعلية. وتبقى الجلسة في `completed` قابلة لإعادة المحاولة.
- `failed` **هي أيضًا حيث يحطّ الإلغاء.** لا توجد حالة منفصلة للإلغاء؛ فالجلسة الملغاة تكون في `failed` ، وتحمل رسالة `errorMessage` تبدأ بـ `Cancelled:` .
- `failed` **ليست دائمًا نهاية سيّئة قاطعة.** إذا أنتج التحقّق تقرير موارد لكنه تعذّر عليه المتابعة، فإنه ينشئ مع ذلك جلسة في `validated` ، ويتضمّن التقرير الأخطاء الحاجبة. ولا تُنشأ الجلسة مباشرةً في `failed` إلا إذا تعذّر على التحقّق إنتاج تقرير أصلًا، بسبب حزمة غير قابلة للقراءة أو إصدار مخطط غير متوافق (`createSession`). وتقبل كلتا الحالتين `failed` و `completed` إعادة المحاولة.

إنشاء جلسة

`POST /v1/import/validate`

ملفّ ZIP الخاص بالحزمة **لا** يُرسَل إلى نقطة النهاية هذه. ارفعه أولًا إلى خدمة التخزين، ثم مرّر معرّف الملف الناتج؛ فتتولّى خدمة التحزيم تنزيله بنفسها.

حقل الجسم	مطلوب	المعنى
`fileId`	نعم – غيابه يؤدي إلى `400`	معرّف ملفّ ZIP المرفوع في خدمة التخزين
`fileName`	لا	اسم الملف الأصلي، ويُعاد في قوائم الجلسات
`importStrategy`	لا	الاستراتيجية الشاملة الأولية: `MERGE_NEW_ONLY` أو `OVERWRITE_ALL` أو `MANUAL` (الافتراضي)

الاستجابة هي `{sessionId, validationReport, updateIdentifier}` (`ImportValidateRequest` / `ImportValidateResponse`). احتفظ بقيمة `updateIdentifier` ، إذ يجب إرسالها مع كل كتابة لاحقة (انظر قسم التزامن). وتشرح صفحة التحزيم والتوزيع بنية تقرير التحقّق ومعناه، بينما تشرح صفحة إصدارات المخطط كيف تضبط قيمة `schemaVersion` في البيان هذا الاستدعاء. ويُوثَّق عضو واحد من التقرير في هذه الصفحة، لأن التعامل معه يتطلّب عملية اتصال مستقلة: `mappedFields` `Since 1.10` .

ينشئ التحقّق دائمًا جلسة **جديدة**؛ ولا يمكن إعادة التحقّق ضمن جلسة قائمة. لذا تعني إعادة رفع حزمة مصحّحة إنشاء جلسة جديدة وبدء التهيئة من الصفر.

و `Since 1.10` يبلّغ التحقّق أيضًا عن الحالات التالية بوصفها أخطاء حاجبة في التقرير:

- `TARGET_ARCHIVED` – مورد في الحزمة سيطابق موردًا هدفًا **مؤرشفاً**؛ فلا يُعامَل بوصفه موجودًا، ولا يُكتب فوق سجلّ مؤرشف أبدًا؛
- `TARGET_INSTALLED` – مورد سيكتبه الاستيراد ويملكه **تثبيت** حزمة؛ اشتقّ التثبيت أولًا؛

- رموز `INSTALL_*`، حين يكون ملف ZIP إصدار حزمة منشورًا، فتكون الجلسة جلسة تثبيت – انظر وضع التثبيت؛ وأصبح `INSTALL_NAME_COLLISION` يسرد **كل** تصادم (`resourceType` و`name` و`displayName` و`scope` بقيمة `space` أو `environment` و`targetId?` و`targetModuleId?`)، بدلًا من التصادم الأول وحده؛
- `MAPPED_FIELD_SOURCE_UNDECIDED` – حقل ربط يحتاج إلى قرار من العميل قبل التشغيل (انظر قرارات حقول الربط).

ويُرفض رفضًا قاطعًا ملفُّ إصدار منشور صُدِّر قبل اعتماد معرّفات الحزم، بالرمز `422 PACKAGE_FORMAT_OUTDATED`.

قراءة الجلسات

`GET /v1/import/sessions`

يسرد جلسات المساحة الحالية في صورة ملخّصات – `{sessions: [{id, status, sourceEnvironment?, packageDescription?, packageFileName?, resourceCount?, createdAt, updatedAt, createdBy?, updatedBy?, updateIdentifier?}]}` – مرتّبةً من الأحدث نشاطًا إلى الأقدم، تنازليًا بحسب `updatedAt`. ويُضبط التقسيم إلى صفحات بمعاملَي الاستعلام `skip` و`limit`؛ والقيمة الافتراضية لـ `limit` هي 50، ويُحصر نطاقها بين 1 و100، فتُخفَّض القيمة الأكبر بصمت.

`GET /v1/import/{sessionId}`

يُعيد مستند الجلسة كاملًا: البيان، وتقرير التحقّق، وتخطيطات الموارد، وتخطيطات مراجع القيم (مع الإبقاء على `entryData` المحزَّمة كاملةً – وهي الحمولة التي يعمل عليها محرّر الإنشاء التلقائي في المعالج)، وتخطيطات التبعيات غير المخطَّطة، والتجاوزات، والنتائج. وإذا كانت الجلسة في `validated` أو `failed`، تعيد الخدمة عند كل قراءة فحص مراجع الإنشاء التلقائي غير المحلولة، وتُدرج الأخطاء المستجدّة في التقرير المُعاد؛ وبذلك تعكس الاستجابة الحالة الحالية للمساحة الهدف، بدلًا من لقطة قديمة.

`GET /v1/import/{sessionId}/status`

نقطة نهاية خفيفة لاستقصاء حالة استيراد جارٍ: `{status, importResult?, errorMessage?, resourceMappings?}` (`ImportStatusResponse`). ولا تُضمَّن `resourceMappings` إلا بعد أن تغادر الجلسة طور التهيئة، أي أثناء `importing` وفي الحالات النهائية، ليتمكّن العميل المستقصي من عرض تقدّم كل مورد دون جلب الجلسة كاملةً.

تهيئة جلسة

تُقبل عمليات كتابة التهيئة حين تكون الجلسة في `validated` أو `failed`، وتُرفض في أي حالة أخرى بالرمز `409`. والاستثناء الوحيد هو الاستراتيجية الشاملة، التي تتطلّب `validated` حصرًا. وتخضع هذه العمليات جميعها للقفل التفاؤلي – انظر قسم التزامن.

`PATCH /v1/import/{sessionId}/config`

نقطة النهاية الموحّدة للحفظ التلقائي (`SaveImportConfigRequest`). جميع الأقسام اختيارية؛ فيرسل العميل ما تغيّر فحسب. والتحقّق هنا متساهل عمدًا، إذ تُفرَض القيود الصارمة وقت التنفيذ:

```
{
  "mappings": [
    {
      "sourceType": "DataEntity",
      "sourceId": "0b6d…",
      "normalizedName": "invoice",
      "action": "OVERWRITE",
      "targetId": "9f2a…"
    }
  ],
  "valueReferences": [
    {"entryId": "77c1…", "action": "REMAP", "remapTargetId": "a41b…"}
  ],
  "unmappedDependencies": [
    {"sourceId": "d305…", "targetId": "e77f…"}
  ],
  "mappedFieldDecisions": [
    {"entityId": "0b6d…", "fieldId": "f1…", "targetParentId": "9f2a…",
     "decision": {"kind": "dontCopy"}}
  ],
  "updateIdentifier": "58c0…"
}
```

ينبغي للعميل مراعاة سلوك الخادم التالي:

- قيم `status` و`errorMessage` و`isHardDep` في أي تخطيط مُرسَل **تُجرَّد**، لأن الخادم هو الذي يتحكّم بها؛ ويُضبط كل تخطيط محفوظ من جديد على `pending`.
- إذا أُرسل تخطيط دون `targetId` لمورد صنّفه التحقّق على أنه موجود، فإنه يستعيد تلقائيًا قيمة `targetId` التي حدّدها التحقّق؛ وبذلك لا يمكن لعملية حفظ أن تُسقِط رابط المصدر بالهدف بصمت.
- في `unmappedDependencies`، القيمة `targetId: null` **تمسح** هدفًا سبق اختياره.
- و`Since dev` قد يحمل صفّ في `unmappedDependencies` ضمن التقرير القيمة `hard: false`: لا تعتمد عليه إلا قطع الصفحات، وإذا تُرك دون ربط أُفرغت تلك القطع (وغياب الحقل `hard` يعني أن التبعية صلبة). راجع قطع الصفحات التي تشير إلى خارج الحزمة.
- تُدمَج خيارات مراجع القيم عنصرًا عنصرًا؛ ويُتجاهَل أي `entryId` غير موجود أصلًا في الجلسة، ويُتحقَّق من وجود أهداف إعادة التوجيه ومن كونها نشطة قبل اعتماد الحفظ.
- يُرفض بالرمز `400` أي حفظ من شأنه مسح مالك مطلوب لخطوة سير عمل على مورد سيكتبه الاستيراد – انظر مالكي خطوات سير العمل المطلوبين.
- `Since 1.10` تمثّل قيمة `mappedFieldDecisions` القائمة **الكاملة** لقرارات حقول الربط، وتحلّ محلّ القائمة المخزَّنة؛ ويؤدي العنصر غير السليم إلى `400 MAPPED_FIELD_DECISION_INVALID`. ويعيد الحفظ الذي يغيّر التخطيطات أو أهداف التبعيات غير المخطَّطة حساب حالات حقول الربط أيضًا، لأن أيًا من هذين التغييرين قد يغيّر الأب الهدف لحقل (انظر قرارات حقول الربط).
- الجسم الفارغ طلب صالح لا أثر له: لا يُكتب شيء ولا يُدوَّر الرمز، لكن الرمز يُفحَص مع ذلك، فيعرف العميل إن كانت نسخته متقادمة.

الاستجابة هي `{updatedAt, updateIdentifier, dependencyImpact?, valueReferenceErrors?, mappingSummary?, mappedFields?, canProceed?}` (`SaveImportConfigResponse`). تُعاد `dependencyImpact` مع تغييرات التخطيطات، وتبيّن

الموارد التي فقدت تبعيات صلبة وأعداد التخطّي المتسلسل المحدّثة. وتمثّل `valueReferenceErrors` إنذارات مبكرة من آلية حماية المراجع المتداخلة للإنشاء التلقائي: فقد نجح الحفظ، لكن هذه الأخطاء ستمنع التنفيذ. وتبيّن `mappingSummary` عدد أصحاب الصلاحيات الذين حُسمت مراجعهم والذين لم تُحسم مراجعهم بعد، عقب حفظ مراجع القيم. أما `mappedFields` و`canProceed` (`Since 1.10`) فتُعادان كلما أُعيد حساب حالات حقول الربط.

`PATCH /v1/import/{sessionId}/strategy`

الجسم هو `{strategy, updateIdentifier}` (`ApplyStrategyRequest`). يعيد حساب إجراء كل تخطيط وفق الاستراتيجية المختارة: تنشئ `MERGE_NEW_ONLY` الجديد وتتخطّى الموجود، وتنشئ `OVERWRITE_ALL` الجديد وتستبدل الموجود، وتترك `MANUAL` قرار كل مورد للعميل، مع **الحفاظ على إعادة تخطيط الوحدات التي اختارها المستخدم** (فتبقى وحدة `DataModule` التي خُطِّطت يدويًا إلى هدف مختلف كما هي بعد إعادة الحساب). يُعيد الجلسة المحدّثة كاملةً، ولا يُقبل إلا في `validated`.

`PATCH /v1/import/{sessionId}/suggestions/accept-exact`

الجسم هو `{updateIdentifier}` (`AcceptExactSuggestionsRequest`). يحوّل، في عملية كتابة واحدة، كل اقتراح لصاحب صلاحية ذي مطابقة تامّة لم يُتّخذ بشأنه إجراء بعد إلى إعادة توجيه نحو هدفه المقترح، وذلك عبر مسار الدمج والتحقّق نفسه الذي يسلكه `/config`. وتُتخطّى الاقتراحات التقريبية والملتبسة والمحلولة أصلًا دون أن تُعدّ فاشلة. يُعيد `{acceptedCount, mappingSummary, updateIdentifier}`؛ وإذا لم توجد اقتراحات مؤهّلة، فلا يكون للعملية أثر، مع فحص الرمز وإعادة `acceptedCount: 0`.

نقاط نهاية تجاوز الحمولة

تتيح `GET` / `PATCH` / `DELETE /v1/import/{sessionId}/resource/{sourceType}/{sourceId}/payload` قراءة حمولة JSON لمورد واحد، واستبدالها، ومسح تجاوزها للتنفيذ القادم. يجب أن تكون `sourceType` قيمةً حرفية من تعداد `ResourceType`، وإلا فالاستجابة `400`؛ ويجب أن يكون للزوج تخطيط في الجلسة، وإلا فالاستجابة `404`. يُعيد `GET` التجاوز الحالي، أو الأصل المحزَّم عند غياب التجاوز؛ وعند وجود تجاوز، يُعيد معه النسخة المحزَّمة للمقارنة. ويُحفظ التجاوز عبر `PATCH` بجسم `{payload, updateIdentifier}`؛ إذ يتحقّق من الحمولة (`400` لحمولة غير سليمة أو لحمولة تغيّر حقلًا بنيويًا – حيث يُفحص ثبات البنية بمقارنتها ببصمات كل مورد المحفوظة في `structuralSnapshot` للجلسة – و`413` لحمولة أكبر من اللازم)، ثم يخزّن التجاوز في الجلسة بوصفه `ResourceOverrideModel`، ويعيد أي تخطيط فاشل سابقًا إلى `pending` لتشمله إعادة المحاولة. وتخضع كل عملية كتابة لتجاوز للقفل التفاؤلي، وتُسجَّل في سجلّ التدقيق. أما `DELETE` فيأخذ `updateIdentifier` معاملَ استعلام، ويعيد المورد إلى الحمولة المحزَّمة، ولا يتغيّر أثره بتكراره. تُحجب القراءات أثناء `importing`، وتُحجب الكتابات أثناء `importing` أو `validating`.

وتشرح صفحة توليد حزمة متى تلجأ إلى هذه النقاط لمواءمة حزمة مع كل بيئة، بدلًا من إصلاح قيمة واحدة غير سليمة.

قرارات حقول الربط `Since 1.10`

حقل الربط هو `DataEntityField` يحمل `mapping.enabled`؛ وهو حقل ينسخ قيمةً من كيان أب، ويحدّد حقل ذلك الأب بمعرّف UUID في `mapping.sourceFieldId`. وتخصّ هذه المعرّفات البيئة المصدر، بينما يُخطَّط الكيان الأب عادةً إلى كيان موجود أصلًا في الهدف بدلًا من إنشائه من جديد؛ لذا لا يشير المعرّف المحزَّم إلى أي حقل هناك. ويعالج الاستيراد ذلك بمطابقة `fullNormalizedName` للحقل بدلًا من معرّفه؛ ويحفظ الملفّ الجانبي `field-mapping-hints.json` هذا الاسم لينتقل مع الحزمة بين البيئات.

وحين تتعذّر هذه المطابقة، **يقرّر العميل** المصدر الذي ينسخ منه الحقل، ولا يمكن تشغيل الجلسة حتى يُحسم كل حقل من هذا النوع. وتُرسَل القرارات ضمن عملية حفظ التهيئة المعتادة.

التقرير: `mappedFields`

ترد `validationReport.mappedFields?: MappedFieldStatus[]` مع `POST /v1/import/validate` وGET `/v1/import/{sessionId}`، وتغيب حين لا تتضمّن الحزمة حقل ربط سيكتبه الاستيراد. ويُعيد `PATCH /…/config` نسخة محدّثة منها. وتتضمّن حالة واحدة لكل حقل ربط على كيان سيكتبه الاستيراد (المخطَّط بالإجراء `CREATE` أو `OVERWRITE`، إضافةً إلى الحقول التي يضيفها توسيعُ التخطّي لكيان موجود). أما الحقول التي يكون أبوها كيان هوية أساسيًا (المستخدمون، والمجموعات، والوحدات التنظيمية، وساعات العمل)، فلا تُسرد أبدًا، لأن تلك المعرّفات متطابقة في كل مكان.

```
type MappedFieldChoice =
  | {kind: 'copyFrom'; targetFieldId: string}
  | {kind: 'dontCopy'};

interface MappedFieldStatus {
  entityId: string; entityName?: string;   // the entity carrying the field (source-environment ids)
  fieldId: string; fieldName: string; fieldLabel?: ILocalizedValue;
  state: 'waiting' | 'matched' | 'unmatched' | 'unnamed' | 'decided' | 'unavailable';
  updateMode: number;                      // 1 Never, 2 ParentOnly, 3 TwoWay (the default when absent)
  sourceParentId: string;                  // the parent in the package's environment
  targetParentId?: string; targetParentName?: string; targetParentLabel?: ILocalizedValue;
  expectedSource?: {name: string; label?: ILocalizedValue; dataType?: number};
  matchedField?: MappedFieldCandidate;     // the field found on the target parent
  decision?: MappedFieldChoice;            // the standing decision, if any
  candidates: MappedFieldCandidate[];      // may be EMPTY: then only dontCopy is offered
}

interface MappedFieldCandidate {
  id: string;                              // target-parent field UUID
  fullNormalizedName: string;
  name?: Record<string, string | null>;
  dataType: number;
}
```

`state`	المعنى	ما يفعله العميل
`waiting`	ستنشئ هذه الحزمة الأب؛ لذا لا تتوفّر حقول للاختيار منها بعد.	لا شيء.
`matched`	حقل المصدر المتوقَّع موجود على الأب الهدف، أو أن الأب المحزَّم سيُكتب.	لا شيء.
`unmatched`	تحدّد الحزمة اسم حقل المصدر، لكن الأب الهدف لا يحتوي على حقل بهذا الاسم.	**قرِّر.**
`unnamed`	لا تستطيع الحزمة تحديد اسم حقل المصدر (لا يوجد تلميح في الملف الجانبي، ولقطة الأب غير موجودة في الحزمة).	**قرِّر.**
`decided`	يوجد قرار صالح للأب الهدف الحالي.	لا شيء (ويمكن تغيير القرار).
`unavailable`	تعذّرت قراءة الأب الهدف.	أعد المحاولة لاحقًا؛ وترافق هذه الحالة رسالة تحذير `MAPPED_FIELD_SOURCE_UNAVAILABLE`.

أيّ الحقول يتأهّل مرشَّحًا

لا تشمل `candidates` جميع حقول الأب الهدف؛ إذ تخضع لثلاث قواعد إلزامية. وإتاحة مرشَّح يخالف أيًا منها تؤدي إما إلى بيانات خاطئة بصمت، وإما إلى فشل المورد فشلًا قاطعًا في data-service:

1. **نوع البيانات نفسه.** يجب أن تساوي قيمة `dataType` للمرشَّح نظيرتها في حقل الربط.

2. **أهداف الارتباط يجب أن تكون مجموعة جزئية.** في حقل الارتباط (`dataType` `7`)، يجب أن يكون كل كيان يستطيع *المرشَّح* الإشارة إليه ضمن الكيانات التي يقبلها الحقل *المحلي* أصلًا. فالمجموعة الأوسع تسمح بنسخ مرجع إلى حقل لا يقبله، بينما يرفض اشتراط المساواة التامّة، خطأً، مرشَّحًا آمنًا ذا مجموعة أضيق. ويُرفض المرشَّح الذي لا تُعرف مجموعة أهدافه، وإذا لم تُعرف مجموعة أهداف الحقل المحلي، أُفرغت القائمة كلها؛ فالقاعدة تعتمد الرفض عند الشك.
3. `TwoWay` **يجب أن يتّفق على الإلزامية.** حين يكون `updateMode` هو `TwoWay` (`3`)، يجب أن تساوي قيمة `validations.required` للمرشَّح نظيرتها في حقل الربط، لأن data-service يرفض المورد بأكمله عند اختلاف الإلزامية في حقل مصدر ثنائي الاتجاه.

وتُستبعد استبعادًا تامًا الحقول المخصّصة للعرض فقط (`options.displayOnly.enabled`). ويُعامَل الربط المحزَّم الذي **لا** تُحدَّد له قيمة `updateMode` بوصفه `TwoWay` ؛ وهذا تشديد متعمّد لشروط الترشيح، لتجنّب إتاحة مرشَّح سيتسبّب لاحقًا في فشل المورد. وقد تكون `candidates` فارغة بالفعل.

حفظ القرارات، وما يحجب التشغيل

أرسل القرارات مع `PATCH /v1/import/{sessionId}/config` في `mappedFieldDecisions` بوصفها **القائمة الكاملة** التي تحلّ محلّ القائمة المخزَّنة؛ لذا أرسل كل قرار تريد الإبقاء عليه، لا القرارات التي تغيّرت وحدها:

```
{
  "mappedFieldDecisions": [
    {"entityId": "0b6d…", "fieldId": "f1…", "targetParentId": "9f2a…",
     "decision": {"kind": "copyFrom", "targetFieldId": "c7e3…"}},
    {"entityId": "0b6d…", "fieldId": "f2…", "targetParentId": "9f2a…",
     "decision": {"kind": "dontCopy"}}
  ],
  "updateIdentifier": "58c0…"
}
```

ويؤدي العنصر غير السليم إلى `400 MAPPED_FIELD_DECISION_INVALID` . وتضيف الاستجابة `mappedFields` (الحالات المعاد حسابها) و `canProceed` . ويرتبط القرار بقيمة `targetParentId` التي اتُّخذ لها، ولا يبقى صالحًا إلا ما دام ذلك هو الأب الهدف للحقل وما دام الاختيار متاحًا (`dontCopy` دائمًا؛ و `copyFrom` للحقل المطابَق أو لمرشَّح مسرود فقط). لذلك فإن الحفظ الذي يغيّر الهدف الذي يُخطَّط إليه الأب – عبر تخطيط مورد أو هدف تبعية غير مخطَّطة – يُسقط القرارات التي لم تعد صالحة، وتتضمّن استجابته الحالات الجديدة بالفعل.

ما يحجب التشغيل. يُعدّ كل حقل في `unmatched` أو `unnamed` بلا قرار **خطأً** تحقّق بالرمز `MAPPED_FIELD_SOURCE_UNDECIDED` ("Choose what '*entity.field*' copies from before installing".). وتُسرد الأخطاء العشرون الأولى كاملةً، ثم يُضاف عنصر ملخّص واحد، وتبقى `canProceed` بقيمة `false` حتى تُحسم جميعها. ويعيد التنفيذ وإعادة المحاولة الفحص، ويرفضان التشغيل بالرمز `409 MAPPED_FIELD_SOURCE_UNDECIDED` (مع تسمية عشرة حقول على الأكثر)، أو بالرمز `503 MAPPED_FIELD_SOURCE_UNAVAILABLE` حين تتعذّر قراءة أب.

ما يفعله القرار عند التنفيذ. يضبط `copyFrom` قيمة `mapping.sourceFieldId` على الحقل المختار. ويضبط `dontCopy` القيمة `mapping.enabled = false` ؛ وإذا كان `updateMode` هو `Never` (`1`) أو `ParentOnly` (`2`) وكان الحقل يحمل كتلة `locked` ، فإنه يمسح `locked.enabled` أيضًا، وإلا لاستُورد الحقل مقفلًا دون مصدر يملؤه. أما الحقل الذي لا قرار له، فتُجرى له مطابقة الاسم: تعيد المطابقة كتابة `sourceFieldId` عند نجاحها، وعند إخفاقها تترك الربط كما صُرّح به، مع تحذير استيراد يفيد بأنه لن ينسخ حتى يُصلح. ولا يُعطَّل شيء أبدًا دون قرار.

ولا يمكن لعملية تثبيت من المتجر أو تحديث منه تجري دون واجهة أن تتّخذ القرار نيابةً عن المستخدم: إذ يُعاد حقل الربط غير المحسوم ضمن `choices` غير المحلولة (`kind: 'mappedField'`)، ويُرفض التشغيل حتى يُحسم في صفحة الاستيراد.

التنفيذ والإنهاء

`POST /v1/import/{sessionId}/execute`

بلا جسم. تُجرى فحوص الحماية بالترتيب التالي:

1. إذا كانت الجلسة **ملغاة**، رُفض الاستدعاء بالرمز `409 IMPORT_SESSION_CANCELLED` (`Since 1.10`) – ابدأ استيرادًا جديدًا. وإلا، فإن كانت الجلسة أصلًا في `failed` أو `completed` ، **يُفوَّض الاستدعاء إلى إعادة المحاولة**: وبذلك يجوز للعميل استخدام execute بمعنى «شغّلها»، دون تتبّع أيّ العمليتين تلزمه.
2. يجب أن يكون لكل تبعية غير مخطَّطة يتطلّبها مورد سيكتبه الاستيراد فعلًا (إجراؤه `CREATE` أو `OVERWRITE`) هدف، وإلا فالاستجابة `400` . ولا تحجب التنفيذَ التبعياتُ التي لا تحتاج إليها إلا موارد متخطّاة. وفي أحدث نسخة من فرع التطوير (`1.11.0-alpha.4`)، لا يميّز هذا الفحص بعدُ الصفّ الاختياري `hard: false` ، مع أن صفحة الاستيراد تسمح بـ**التنفيذ** وهو غير مربوط – راجع التنبيه تحت قطع الصفحات التي تشير إلى خارج الحزمة.
3. `Since 1.10` يجب حسم كل حقل ربط يحتاج إلى قرار، وإلا فالاستجابة `409 MAPPED_FIELD_SOURCE_UNDECIDED` ، أو `503 MAPPED_FIELD_SOURCE_UNAVAILABLE` حين تتعذّر قراءة أب هدف (انظر قرارات حقول الربط).
4. يجب تحديد إجراء لكل مرجع قيمة يستخدمه مورد سيُكتب فعليًا. وتُطبَّق الخيارات الافتراضية الآمنة أولًا: يُضبط المرجع الموجود في الهدف على `KEEP` ، والمرجع ذو الحمولة المحزَّمة على `AUTO_CREATE` . وما بقي بلا حلّ يؤدي إلى `400` ؛ وبدءًا من `Since 1.10` يكون رمزه `VALUE_REFERENCE_ACTION_UNSET` . ويُرفض إجراء `AUTO_CREATE` للمرجع المعني وحده إذا طابق مفتاحُ هويته (البريد الإلكتروني للمستخدم أو اسم المستخدم مثلًا) مُدخلًا موجودًا في الهدف، بالرمز `VALUE_REFERENCE_IDENTITY_COLLISION` ، بدلًا من إنشاء صاحب صلاحية مكرّر.
5. `Since 1.10` يجب أن تتوفّر لدى القاطن سعة كافية لكل مُدخل سينشئه التشغيل: تستعلم خدمة التحزيم من data-service مسبقًا، وترفض التشغيل بالرمز `403 ENTRY_CAP_EXCEEDED` ("past its entries limit. Archive or delete… entries, or upgrade the plan, then install again."). ولا يكون قد كُتب شيء في تلك المرحلة. ويُسجَّل أي إخفاق آخر في هذا الفحص، ويستمرّ التشغيل، لأن كل عملية إنشاء تظلّ خاضعةً للفحص على حدة.
6. تُحجز الجلسة بانتقال **ذرّي** `validated → importing` . وإذا سبق استدعاء execute متزامن هذا الاستدعاء، أو تغيّرت حالة الجلسة، فالنتيجة `409` ، ولا يحدث تنفيذ مزدوج أبدًا.

الاستجابة هي `{jobId, sessionId}` (`ImportExecuteResponse`): أُدرجت المهمّة في الصفّ، ولم تُنجَز بعد. استقصِ `/status` حتى تصل آلة الحالات إلى `completed` أو `failed` .

`POST /v1/import/{sessionId}/retry`

بلا جسم. لا يُقبل إلا في `completed` أو `failed` ، وإلا فالاستجابة `409` . و `Since 1.10` لا يُقبل أبدًا على جلسة ملغاة (`409 IMPORT_SESSION_CANCELLED`). ويحجز الجلسة ذرّيًا كما يفعل execute. و `Since 1.10` يعيد فحص قرارات حقول الربط وحدّ المُدخلات تمامًا كما يفعل execute؛ ويُحسب الحدّ على الصفوف التي ستُعاد إتاحتها عند إعادة الضبط، فتُشمل المُدخلات التي لم يحتج إليها إلا مورد متخطّى تسلسليًا. وتُعاد التخطيطات الفاشلة والمتخطّاة تسلسليًا إلى `pending` ، مع استعادة إجراء التخطيطات المتخطّاة تسلسليًا الأصلي بعد إعادة حسابه وفق استراتيجية الجلسة، وتُعاد إتاحة مراجع القيم الفاشلة أو المتخطّاة. ولا يُعاد استيراد الموارد المستوردة أصلًا؛ فإعادة المحاولة لا تشمل إلا ما لم يكتمل. يُعيد `{jobId, sessionId}` .

`POST /v1/import/{sessionId}/cancel`

بلا جسم. يُقبل في `validated` أو `importing`، وتؤدي أي حالة أخرى إلى `400`. تُنقَل الجلسة ذرّيًا إلى `failed` برسالة خطأ `Cancelled: …`، وتُزال المهمّة من الصفّ؛ وإذا كان العامل قد بدأ التنفيذ بالفعل، فإنه يكتشف علامة الإلغاء بين معالجة مورد وآخر، فيُجهض الاستيراد. والموارد المكتوبة قبل الإجهاض **تبقى**: فالإلغاء يوقف الاستيراد ولا يتراجع عنه. و `Since 1.10` أصبح الإلغاء **نهائياً** أيضًا: تسجّل الجلسة `cancelledAt`، ويُرجِع استدعاء التنفيذ أو إعادة المحاولة عليها `409` `IMPORT_SESSION_CANCELLED`. يُعيد `{outcome, state?}`.

وأُضيف الحقل `importWarnings` في 1.10 إلى نتيجة التشغيل المنتهي (`importResult` في `/status`). ويبيّن هذا الحقل التغييرات التي أجراها الاستيراد بدلًا من أن يفشل بسببها:

- **استبدال الفهارس. تستبدل** الفهارسُ المخصّصة للكيان المستورد فهارسَ الهدف. ويُسقَط الفهرس الذي يتعذّر على الهدف فهرسة مسار حقله، مع تحذير مثل `Dropped index "<name>": field path <path> not found`، بدلًا من إفشال الكيان.
- **الملفات التي لم تُنقل.** كان المورد الذي لا يزال يحمل معرّف ملف من القاطن المصدر (لتعذّر تنزيله أثناء التصدير أو رفعه أثناء الاستيراد) يفشل بالرمز `FILE_EXPIRED`؛ أما الآن فيُزال الملف من المورد – بتفريغه، أو إسقاطه من مصفوفة، أو إزالته بوصفه صورة مصغّرة – مع تحذير واحد لكل مستند. ويُحتفظ بمعرّف الملف إذا أمكن العثور عليه في تخزين الهدف (عند الاستيراد داخل القاطن نفسه).
- **حقول ربط تُركت كما صُرّح بها.** إذا لم يكن لحقل الربط قرار، وظلّت مطابقة الاسم عاجزةً عن العثور على مصدره وقت التشغيل، بقي ربطه كما صُرّح به، مع تحذير يفيد بأنه لن ينسخ حتى يُصلح.

`DELETE /v1/import/{sessionId}`

يحذف جلسة متروكة. يُرفض بالرمز `409` حين تكون الجلسة في `importing`؛ ويأخذ `updateIdentifier` معاملَ استعلام، كي لا يستطيع تبويب متقادم حذف جلسة عدّلتها جهة أخرى.

مستند الجلسة

يُعيد `GET /v1/import/{sessionId}` الكائن `ImportSession` (المحفوظ في مجموعة MongoDB المسمّاة `ImportSession`). وفيما يلي أهم حقوله:

الحقل	الغرض
`id`	معرّف UUID v4 (يُولَّد عند التحقّق).
`spaceId`	المساحة الهدف.
`status`	`ImportSessionStatus` – انظر آلة الحالات.
`packageFileId` / `packageFileName`	مرجع ملف ZIP في خدمة التخزين (يُعاد تنزيله عند التنفيذ).
`manifest`	بيان الحزمة بعد تحليله.
`validationReport`	ناتج المرحلة الأولى.
`resourceMappings`	قرارات كل مورد (انظر `ResourceMapping` أدناه).
`valueReferenceMappings`	قرارات المراجع على مستوى المُدخلات.
`unmappedDependencyMappings`	أهداف التبعيات الخارجية غير الموجودة في الحزمة.
`resourceOverrides`	الحمولات التي عدّلها المستخدم، وتُطبَّق عند التنفيذ.
`structuralSnapshot`	بصمات الحقول الثابتة لكل مورد.

الحقل	الغرض
`importResult`	النتيجة الإجمالية للمرحلة الثانية.
`importJobId`	معرّف مهمّة BullMQ (لغرض الإلغاء).
`importStrategy`	الاستراتيجية الشاملة؛ وقيمتها الافتراضية `MANUAL`.
`updateIdentifier`	رمز التزامن التفاؤلي (UUID؛ يُدوَّر عند كل كتابة – انظر أدناه).

التعدادات

التعداد	القيم
`ImportSessionStatus`	`validating`، `validated`، `importing`، `completed`، `failed`
`ImportStrategy`	`MERGE_NEW_ONLY` (إنشاء الجديد، وتخطّي الموجود)، `OVERWRITE_ALL` (إنشاء الجديد، واستبدال الموجود)، `MANUAL` (الافتراضي؛ إبقاء قرار كل مورد كما هو)
`MappingAction`	`CREATE`، `OVERWRITE`، `SKIP` – ويظلّ الإجراء `SKIP` على مورد موجود يسجّل تخطيط معرّفه، فتُحَلّ إليه مراجع الموارد الأخرى (وهذا هو **تعيين** في تطبيق الويب)
`ResourceImportStatus`	`pending`، `completed`، `failed`، `skipped`، `cascade_skipped`
`ValidationStatus`	`NEW`، `EXISTING`
`ValueReferenceAction`	`KEEP`، `REMAP`، `NULL`، `AUTO_CREATE`
`EntryReferenceSourceStatus`	`FOUND`، `NOT_FOUND`، `ARCHIVED`
`ModuleConflictResolution`	`KEEP`، `MOVE`

`ResourceMapping`

القرار الذي يحرّره المستخدم لكل مورد: `sourceType`، و`sourceId`، و`normalizedName`، و`displayName?`، و`action`، و`targetId?` (معرّف الهدف الموجود، أو قيمة تُملأ بعد CREATE)، و`status?`، و`errorMessage?`، و`isHardDep?`، و`moduleConflictResolution?` (`KEEP` / `MOVE` لتعارض وحدة أُعيد تخطيطها)، و`entityId?`، و`moduleId?`.

`ImportResult`

يُحفظ بعد المرحلة الثانية، ويُعيده `GET /…/status`: `created`، و`overwritten`، و`skipped`، و`cascadeSkipped`، و`errors`، و`resources[]` (`{sourceId, targetId, type, action}`)، و`importErrors[]` (`{sourceId, type, message}`)، و`valueReferenceResults[]`، و`importWarnings` (`Since 1.10`، انظر أعلاه)، إضافةً إلى عدّادات مراجع القيم `entriesAutoCreated` و`entriesAutoCreateFailed` و`entriesKept` و`entriesRemapped` و`entriesNulled`.

التزامن: معرّف التحديث

تحمل كل عملية كتابة تغيّر الجلسة رمز تزامن تفاؤليًا. يُنشأ `updateIdentifier` عند إنشاء الجلسة و**يُدوَّر عند كل كتابة ناجحة**؛ ويجب أن تُرسل كل عملية كتابة الرمز الذي رآه العميل آخر مرة، ثم تُعيد الاستجابة الرمز الجديد الذي يجب على العميل اعتماده.

يؤدي عدم التطابق إلى `409`، وتكون قيمة `error.code` فيه `STALE_UPDATE_IDENTIFIER`. ويُميَّز هذا الرمز عمدًا عن رموز `409` الصادرة عن بوّابة الحالة، مثل «cannot modify config for session in status: importing»؛ فهي تشترك في رمز HTTP، لكنها تختلف في الرمز المميِّز للخطأ. عند `STALE_UPDATE_IDENTIFIER`، أعد تحميل الجلسة ودَع المستخدم يوفّق بين التغييرات؛ وعند `409` صادر عن بوّابة الحالة، أعد قراءة الحالة واتّبع آلة الحالات.

وتفحص الكتابات عديمة الأثر – حفظ تهيئة فارغ، أو accept-exact بلا اقتراح مؤهّل للقبول، أو حذف تجاوز غير موجود – الرمز مع ذلك، لكنها **لا** تدوّره؛ وبذلك تبقى حالة القفل لدى العميل متّسقة عبر هذه العمليات.

بوّابات الحالة في لمحة

نقطة النهاية	الحالات المقبولة	عند المخالفة
`PATCH /…/config`	`validated` ، `failed`	`409`
`PATCH /…/suggestions/accept-exact`	`validated` ، `failed`	`409`
`PATCH /…/strategy`	`validated`	`409`
`GET /…/resource/…/payload`	جميع الحالات عدا `importing`	`409`
`DELETE /…/resource/…/payload` / `PATCH`	جميع الحالات عدا `importing` ، `validating`	`409`
`POST /…/execute`	`validated` (تُفوَّض الحالات النهائية إلى إعادة المحاولة؛ ولا تُقبل جلسة ملغاة أبدًا)	`409`
`POST /…/retry`	`completed` ، `failed` (ولا تُقبل جلسة ملغاة أبدًا)	`409`
`POST /…/cancel`	`importing` ، `validated`	`400`
`DELETE /…/{sessionId}`	جميع الحالات عدا `importing`	`409`

لا تخضع القراءات (`GET` للقوائم والجلسة والحالة) لبوّابة حالة، وإنما لفحص ملكية المساحة وحده.

إصدارات المخطط

يحمل بيان كل حزمة قيمة `schemaVersion`، وهي إصدار **صيغة الحزمة**، المستقلّ عمدًا عن إصدار المنصّة التي أنشأت الحزمة. وتوثّق هذه الصفحة الإصدارات المتاحة، وما يقبله المستورِد منها، وفحوص الإصدار الثلاثة المستقلة التي تُجرى عند التحقّق، وما ينبغي لمُنتِج الحزم كتابته. أمّا الصيغة نفسها فموثّقة في صيغة الحزمة. وقد أُطلق محرّك التحزيم الذي ينتمي إليه هذا العقد ضمن سلسلة إصدارات **1.8** (`Since 1.8`)؛ ولا تتوافر هذه المسارات وواجهة التصدير في بيئات النشر الأقدم.

الإصدارات المدعومة

المرجع هو قائمة سماح مغلقة:

```
export const SUPPORTED_SCHEMA_VERSIONS = ['1.0', '1.1', '1.2', '1.3', '1.4'] as const;
```

الإصدارات `1.2` و `1.3` و `1.4` كلها `Since 1.10` . أمّا بيئات نشر **ما قبل 1.10** (1.8 أو أقدم) فلا تزال قائمة السماح فيها تقتصر على عنصرين، هما `['1.0', '1.1']` ، وترفض أي حزمة تعلن أيًّا من الإصدارات الثلاثة. وهذا الفرق بين الإصدارات هو موضوع ما الذي يضيفه 1.2 وما الذي يضيفه 1.3 و1.4، وهو سبب عدم إعلان الحزمة إصدارًا أحدث إلا عند الضرورة؛ إذ ينبغي أن يظلّ بالإمكان استيراد الحزم المُصدَّرة من 1.10 فصاعدًا بسلاسة في بيئة لم تُرقَّ بعد إلى إصدار أحدث من 1.8.

وعند الكتابة، تتولّى `pickSchemaVersion` اختيار الإصدار، فتُرجِع **أدنى إصدار يصف محتويات الحزمة فعلًا**: `1.4` حين تحمل ملف README، وإلا `1.3` حين تحمل كتلة `package` ، وإلا `1.2` حين تحمل الملفّ الجانبي لحقول الربط، وإلا `1.1` . وتمثّل `Constants.SCHEMA_VERSION = '1.4'` أعلى إصدار تستطيع الخدمة *إنتاجه*؛ ويؤكّد اختبار وحدة أنه مدرَج أيضًا في قائمة السماح، لضمان ألا ترفض الخدمة الحزم التي صدّرتها بنفسها. ولا يحمل **التصدير العادي** ملف README ولا كتلة `package` (`POST /v1/export` لا يقبل README `Since 1.10`)، لذا يكون دائمًا `1.1` أو `1.2` ؛ أمّا `1.3` و `1.4` فمصدرهما إصدارات الحزم المنشورة.

الإصدار	الوضع	ما هو
`1.0`	مقبول	التخطيط الأصلي
`1.1`	مقبول – ولا يزال الإصدار الذي يَسِم به التصدير أي حزمة بلا حقول ربط ولا كتلة `package` ولا ملف README	يضيف حقلَي هوية المصدر إلى البيان
`Since 1.2` `1.10`	مقبول من 1.10 فصاعدًا؛ **مرفوض على نشر ما قبل 1.10**	يضيف الملفّ الجانبي `field-mapping-hints.json` ؛ ولا تُوسَم به إلا حزمة تحمله فعلًا
`Since 1.3` `1.10`	مقبول من 1.10 فصاعدًا؛ **مرفوض على نشر ما قبل 1.10**	يضيف كتلة `package` في البيان و `hashes.json` و `package-references.json`؛ ولا يُوسَم به إلا إصدار حزمة منشور
`Since 1.4` `1.10`	مقبول من 1.10 فصاعدًا؛ **مرفوض على نشر ما قبل 1.10**	يضيف `README.md` و `readme/` ؛ وتُوسَم به أي حزمة تحمل ملف README – وهي عمليًا إصدار منشور يحمل هذا الملف
أي شيء آخر	مرفوض	خطأ حاجب `INCOMPATIBLE_SCHEMA` عند التحقّق

لاحظ أن هذه قائمة سماح، **لا** فحص نطاق: فالبيان المكتوب يدويًا الذي يعلن إصدارًا غير مدرَج يُرفَض تمامًا كما يُرفَض بيان يعلن `0.9` ، حتى إن بدا الإصدار «أحدث». كما يشترط مدقّق الحزم الصارم أن تكون `schemaVersion` **سلسلة نصّية**: فالرقم المجرّد في JSON يُخفق في فحص بنية البيان بمخالفة `SCHEMA_VIOLATION` قبل أن تبدأ مقارنة الإصدارات أصلًا.

ما الذي يضيفه 1.2 Since 1.10

> **منذ 1.10.** أُطلق `1.2` ضمن مسار التطوير الذي أصبح الإصدار 1.10. وفي بيئات نشر **ما قبل 1.10** تبقى قائمة السماح `['1.0', '1.1']`، وتُرفَض أي حزمة تعلن `1.2` بخطأ حاجب `INCOMPATIBLE_SCHEMA`.

يضيف `1.2` **الملفّ الجانبي** `field-mapping-hints.json`، وهو مُدخل اختياري واحد في المستوى الأعلى من ZIP، ولا يضيف شيئًا غيره. ويتيح للاستيراد حلّ قيمة `sourceFieldId` لحقل الربط (map field) بالاسم حين لا تكون لقطة الكيان الأب نفسها ضمن الحزمة. ولم تُضَف أي خاصية جديدة إلى حمولة أي مورد، ولم يُضَف حقل إلى البيان ولم يُحذف منه؛ والمُدخل نفسه موصَّف في صيغة الحزمة.

ويُعلَن هذا الإصدار **مشروطًا**: فلا تعلنه إلا حزمة تحمل الملفّ الجانبي فعلًا (`buildPackageManifest` عبر `pickSchemaVersion`). والقاعدة هي: *أعلِن أدنى إصدار يصف محتويات هذه الحزمة فعلًا* – `1.2` حين يكون الملفّ الجانبي الذي بُني للتوّ غير فارغ (ولا توجد كتلة `package` ولا ملف README)، و `1.1` فيما عدا ذلك.

وهذا الشرط هو جوهر التصميم، لأن `schemaVersion` بوّابة في **الاتجاهين** معًا:

الحزمة	البيئة الهدف	النتيجة
`1.0` أو `1.1`	بيئة تدعم `1.2`	**تُقبل كما هي.** فلا يُقرأ أي ملفّ جانبي، ويُستمدّ اسم مصدر حقل الربط من لقطة الكيان الأب داخل الحزمة – وهو سلوك ما قبل `1.2` بعينه.
`1.2` – أي التي تحمل الملفّ الجانبي	بيئة أقدم من `1.2`، أي بيئة تعمل بالإصدار 1.8 أو أقدم	**تُرفَض رفضًا قاطعًا.** يصدر الخطأ الحاجب `INCOMPATIBLE_SCHEMA` مع `canProceed: false`، قبل إجراء أي بحث في المساحة الهدف.
أي حزمة **بلا** حقول ربط صدّرتها بيئة تدعم `1.2`	أي بيئة	تُوسَم بالإصدار `1.1`، فيظلّ بالإمكان استيرادها في بيئة أقدم من `1.2` تمامًا كما كان من قبل.
`1.2`	بيئة تدعم `1.2`	تكتمل دورة التصدير والاستيراد: يُنقل الملفّ الجانبي، ويُحلّ حقل الربط بالاسم حتى حين لا يكون كيانه الأب ضمن الحزمة.

ولو أُعلن `1.2` دون شرط، لتعذّر استيراد *كلّ* حزمة مُصدَّرة – بما في ذلك الغالبية التي لا تحوي حقول ربط أصلًا – في أي بيئة لم تُرقَّ بعد. وهذا ما يوضّحه الصفّ الثالث، وهو سبب اتخاذ القرار لكل حزمة على حدة.

أمّا الحزمة التي تحمل الملفّ الجانبي فعلًا، فرفضها في الصفّ الثاني صحيح ولا مفرّ منه، ولا يمثّل ثغرة: فدون هذا الإعلان، كانت الخدمة الأقدم سترفض الحزمة نفسها على أي حال عبر `validatePackageStrict`، لكن بخطأ مُبهَم هو `UNKNOWN_ENTRY` عند المُدخل غير المعروف في المستوى الأعلى. أمّا `INCOMPATIBLE_SCHEMA` فيوضّح السبب على الأقلّ.

والقاعدة التشغيلية التي تتبع ذلك: رقِّ البيئة الهدف قبل بيئة المصدر. فبيئة المصدر التي تدعم `1.2` تستطيع مواصلة التصدير إلى بيئة هدف أقدم ما دامت حزمها لا تحمل حقول ربط؛ لكن لا توجد آلية في جهة الهدف تستطيع فرض هذا الشرط، وبمجرّد دخول حقل ربط واحد إلى حزمة يصبح تصديرها غير قابل للاستيراد هناك. لذا فإن ترقية البيئات الهدف أوّلًا هي الترتيب الوحيد الذي يضمن عدم حجب الاستيراد.

ما الذي يضيفه 1.3 و1.4 Since 1.10

ينتمي كلا الإصدارين إلى إصدارات الحزم المنشورة والمتجر داخل التطبيق، ويتبعان القاعدة الشرطية نفسها التي يتبعها `1.2`.

- `1.3` يضيف كتلة `package` الاختيارية في البيان ومُدخلَين في المستوى الأعلى، هما `hashes.json` و`package-references.json`. ولا يعلنه إلا **إصدار حزمة منشور**: فالتصدير العادي لا يحمل كتلة `package` أبدًا. وعند الاستيراد، تحوّل هذه الكتلة الجلسة إلى تثبيت.

- `1.4` يضيف `README.md` ومجلد الصور `readme/`، مع `manifest.readme` الذي يشير إليهما. و**أي** حزمة تحمل ملف README تعلن `1.4` . ولا يُكتب هذا الملف إلا في إصدار منشور؛ إذ لا يقبل `POST /v1/export` العادي ملف README (فلا حقل `readme` فيه)، مع أن المستورِد يقرأ أي ملف README يجده في ملف مبني يدويًا.

وهذه المُدخلات موصَّفة في صيغة الحزمة. وكما في `1.2` ، ترفض بيئة الهدف الأقدم من 1.10 هذه الحزم رفضًا قاطعًا بخطأ حاجب `INCOMPATIBLE_SCHEMA` ؛ ودون الإعلان، كانت سترفضها على أي حال عند مصادفة المُدخلات المجهولة في المستوى الأعلى.

وتطبّق خطوة التحقّق نفسها بوّابة أخرى على ملفات إصدارات الحزم: فالملف الذي تحمل فيه كتلة `package` قيمة `id` لا تمثّل معرّف `pkg_` صالحًا – أي ملف نُشر قبل وجود معرّفات الحزم – يُرفض بالرمز `422 PACKAGE_FORMAT_OUTDATED` ، ويجب تصديره من جديد من بيئته. وهذا ليس فحصًا لـ `schemaVersion` ؛ فقد يعلن ملف كهذا إصدارًا مدعومًا.

ما الذي أضافه 1.1

أضاف حقلين اختياريين في البيان: `sourceEnvironmentId` و `sourceSpaceId` . ويشكّلان مع المساحة الهدف مفتاح **ملفّ تعريف تخطيط** مثبَّت، يحفظ كيفية حلّ استيراد سابق من المصدر نفسه للمستخدمين والمجموعات والوحدات التنظيمية. وينتهي البحث عن الملفّ المخزَّن مبكرًا حين يغيب أي من الحقلين (`resolveStoredMappingProfiles`).

والنتيجة هي العودة بسلاسة إلى السلوك البديل دون إخفاق: إذ تُستورَد حزمة `1.0` بلا مشكلة، لكن لا يُملأ لها مسبقًا أي تخطيط محفوظ لأصحاب الصلاحيات؛ فيعود كل صاحب صلاحية إلى المطابقة بالمفتاح الطبيعي، ولا تُطبَّق عليها القرارات المحفوظة من الاستيرادات السابقة.

ثلاثة فحوص للإصدار، ثلاث درجات خطورة

يُجري التحقّق ثلاثة فحوص مستقلة للإصدار، تختلف درجات خطورتها.

صيغة الحزمة: `schemaVersion`

حاجب. تُنتج القيمة غير المدعومة الخطأ `INCOMPATIBLE_SCHEMA` ، وينتهي التحقّق مبكرًا دون تصنيف الموارد أو البحث في المساحة الهدف. وتُنشأ الجلسة مع ذلك لإتاحة فحص الإخفاق، لكنها تنتقل مباشرةً إلى الحالة `failed` بتقرير موارد فارغ و `canProceed: false` .

إصدار المصدر: `sourceVersion`

تحذير، وفي اتجاه واحد فقط. يسجّل حقل `sourceVersion` المعلوماتي في البيان إصدارَ المنصّة الذي صدّر الحزمة. فإن كان *أحدث* من بيئة الاستيراد، أضاف التحقّق التحذير غير الحاجب `NEWER_SOURCE_VERSION` ، إذ قد تشير الحزمة إلى قدرات لا تتوافر في البيئة الأقدم. وتقتصر المقارنة على `major.minor.patch` ؛ وتُهمَل وسوم ما قبل الإصدار، ولا يُنتج الإصدار الذي يتعذّر تحليله أي تحذير أصلًا. أما الحزمة *الأقدم* من البيئة فلا تثير أي تحذير، فهذا هو الاتجاه المدعوم.

عقد كل مورد: `specificationVersion`

تحذير لكل مورد. قد تحمل حمولة كل مورد محزَّم قيمة `specificationVersion` خاصة بها؛ وتُعامَل القيمة الغائبة على أنها `1` . ويعلن كل محلّل موارد إصدارات المواصفة التي يدعمها، ويدعم كل محلّل حاليًا `[1]` فقط (وهو الافتراضي المشترك، ولا يتجاوزه أيّ محلّل). وإذا أعلن مورد إصدارًا آخر، صدر التحذير `UNSUPPORTED_SPEC_VERSION` مع تحديد ذلك المورد؛ ويستمرّ التحقّق، لكن قد يفشل المورد عند التنفيذ.

مصفوفة التوافق

ما تقوله الحزمة	حكم المستورِد	الأثر
`schemaVersion: "1.4"` `Since 1.10`	مقبول من 1.10 فصاعدًا؛ **محجوب** في بيئات نشر ما قبل 1.10	يُتحقَّق من ملف README ويُحفظ مع سجلّ الاستيراد
`schemaVersion: "1.3"` `Since 1.10`	مقبول من 1.10 فصاعدًا؛ **محجوب** في بيئات نشر ما قبل 1.10	تحوّل كتلة `package` الاستيرادَ إلى تثبيت
`schemaVersion: "1.2"` `Since 1.10`	مقبول من 1.10 فصاعدًا؛ **محجوب** في بيئات نشر ما قبل 1.10	يُقرأ الملفّ الجانبي، فيُحلّ حقل الربط بالاسم دون لقطة كيانه الأب
`schemaVersion: "1.1"`	مقبول	السلوك الكامل، بما فيه إعادة استخدام ملفّات تعريف التخطيط المخزَّنة
`schemaVersion: "1.0"`	مقبول	تُستورَد الحزمة؛ ولا تُطابَق ملفّات التعريف أبدًا – ويُحلّ أصحاب الصلاحيات بالمفتاح الطبيعي وحده
`schemaVersion:` أي شيء آخر	**محجوب**	`INCOMPATIBLE_SCHEMA`؛ تُنشأ الجلسة في الحالة `failed` مع `canProceed: false`
`sourceVersion` أحدث من البيئة	تحذير	`NEWER_SOURCE_VERSION`؛ يستمرّ الاستيراد على مسؤوليتك
`sourceVersion` أقدم أو مساوٍ أو غائب	لا تحذير	الاتجاه المدعوم
`specificationVersion` لمورد ≠ 1	تحذير	`UNSUPPORTED_SPEC_VERSION` على ذلك المورد
كتلة `package` بقيمة `id` سابقة لمعرّفات الحزم `Since 1.10`	**محجوب**	`422 PACKAGE_FORMAT_OUTDATED`؛ صدّر الملف من جديد

الترحيل الأمامي

لا توجد خطوة ترحيل، ولا تُجرى أي عملية تحويل للترقية في أي موضع؛ فلا شيء يعيد كتابة حزمة من قيمة `schemaVersion` إلى أخرى، سواء عند الاستيراد أو التصدير. وتُقرأ حزمة `1.0` **كما هي**، وكل ما أضافته الإصدارات من `1.1` إلى `1.4` اختياري. ويؤدّي غياب أيٍّ من هذه الإضافات إلى تعطيل ميزة بعينها – إعادة استخدام ملفّات التعريف عند `1.1`، وحلّ حقل الربط بالاسم عند `1.2`، ووضع التثبيت عند `1.3`، وملف README عند `1.4` – دون إفشال الاستيراد. وهذا هو المقصود بالتصميم الموثَّق عند قائمة السماح نفسها: كل إصدار مدرَج مقبول «كي يظلّ بالإمكان استيراد الحزم الأقدم دون تغيير». فإصدارات الصيغة تسير في اتجاه واحد: تتقدّم البيئات، وتبقى الحزم قابلة للاستيراد.

أمّا إرشادات مُنتِج الحزم فموجزة:

- **اختم بأدنى إصدار يصف ما كتبتَه فعلًا.** لا تضمّن `field-mapping-hints.json` وسِم الحزمة بالإصدار `"1.1"`؛ وضمّن الملفّ الجانبي وسِمها بالإصدار `"1.2"`؛ وضمّن ملف README وسِمها بالإصدار `"1.4"`. أمّا وسمها بإصدار أحدث ممّا تتطلّبه محتوياتها فلا يحقّق أي فائدة، ويمنع استيرادها في كل بيئة لم تُرقَّ بعد.
- **لا تكتب كتلة `package` يدوياً.** حزمة `1.3` إصدار منشور: تحسب بيئة النشر تجزئاتها وتجزئتها الجذرية ومراجعها بين الحزم، ويتحقّق منها التثبيت. لذا انشر من منطقة الحزم.
- ضمّن `sourceEnvironmentId` و`sourceSpaceId` إذا أردت أن تستفيد عمليات الاستيراد ضمن مسار عملك من تخطيطات أصحاب الصلاحيات المحفوظة.
- لا تخترع إصدارًا أعلى للإيحاء بـ«محتوى أحدث». فموضع إصدار المنصّة الذي صدّر الموارد هو `sourceVersion`؛ أمّا `schemaVersion` فيصف تخطيط الحاوية وحده، وتحجب القيمة المجهولة الاستيراد حجبًا تامًّا.

نقل الوحدة بصيغة JSON

> **⚠ مسار قديم**
>
> هذه آلية قديمة وأضيق نطاقًا، أُبقيت لأغراض التوافق. لنقل تطبيق بين البيئات، استخدم التحزيم، أو استخدم من واجهة المستخدم تصدير التطبيقات واستيرادها وتثبيتها [بناء التطبيقات]. وانظر متى تستخدم التحزيم بدلًا منه.

إلى جانب نظام التحزيم القائم على ZIP، توفّر `data-service` آلية نقل ثانية أقدم: تصدير **الوحدات والكيانات ملفَّ JSON واحدًا** ثم تطبيق ذلك الملف في مساحة أخرى. وتكمن جاذبية هذه الآلية في اعتمادها على ملف واحد، لكنها تنقل *تعريفات الوحدات والكيانات* والقليل جدًّا مما عداها. توثّق هذه الصفحة صيغة الملف ونقاط النهاية الخاصة بهذه الآلية، وتوضّح ما لا تنقله، لتحدّد متى يكون نظام التحزيم الأداة المناسبة بدلًا منها.

لا يشترك النظامان في الرماز أو الصيغ أو نقاط النهاية. فالتحزيم يعمل في `packaging-service` تحت `/api/packaging`؛ أما هذا المسار فيعمل في `data-service` تحت البادئة الأساسية `/api`. وتشابه صيغتَي URL بينهما (`/v1/import/…` في كليهما) مجرد مصادفة، ولا يدلّ على توافق بينهما.

ملف واحد، نوعان

يتّبع الملف النموذج `ImportExportModel`:

الحقل	النمط	المعنى
`specificationVersion`	رقم	إصدار صيغة الملف. يكتب المصدِّر الحالي `1`؛ ولا يتغيّر مسار الاستيراد بحسب هذه القيمة
`kind`	`"DataEntity"` | `"DataModule"`	ما ينقله الملف (`ImportExportKindEnum`)
`dataModules`	مصفوفة، اختيارية	تعريفات الوحدات – تَرِد في ملفات نوع الوحدة
`dataEntities`	مصفوفة	تعريفات الكيانات

كل كيان في `dataEntities` هو `FullDataEntity`: تعريف الكيان كاملًا، بما يشمل الحقول، وسير العمل، والكيانات الداخلية، وإعدادات النموذج، والمشاركة، إلى جانب مصفوفات مضمّنة يملؤها المصدِّر: `triggers` (تفاعلات الكيان)، و`queries`، و`views`، و`lists` (المجموعات)، و`relations`. لكن وجود هذه المصفوفات لا يعني أن الاستيراد سيعيد إنشاء محتوياتها؛ انظر *ما الذي لن يحمله* أدناه.

وتعتمد قابلية النقل على **الأسماء المعيارية**: يستبدل المصدِّر معرّفات مصادر حقول مراجع الكيانات بأسماء الكيانات المعيارية، ويحوّل عدّادات الترقيم التلقائي إلى خيارات مضمّنة، ويسجّل، عند تصدير الوحدات، الوحدة التي ينتمي إليها كل كيان باسمها المعياري (`exportEntities`).

التصدير

تعيد كلتا نقطتَي التصدير **تنزيل ملف**، وهو مرفق من النوع `application/json` يحمل اسم أول مورد مصدَّر. وتكتب كل منهما الاستجابة مباشرة، لذا يكون جسم الاستجابة مستند JSON وحده، من دون غلاف `{success, result}` المستخدم في

استجابات الواجهة البرمجية العادية (انظر استدعاء واجهة HTTP البرمجية). أما نقطتا الاستيراد أدناه فتعيدان استجابتين بالغلاف المعتاد.

`POST /v1/export/entity`

الجسم: `{"dataEntities": [{"id": "…"}, …]}` . تصدّر نقطة النهاية الكيانات المحدّدة في ملف من نوع `kind: "DataEntity"`. ويُفحَص إذن التحرير لكل كيان؛ وعند الرفض، تُعاد الحالة `403` مع الرمز `CANNOT_EXPORT_ENTITY` .

`POST /v1/export/module`

الجسم: `{"dataModules": [{"id": "…", "dataEntities": [{"id": "…"}, …]}, …]}` . تصدّر نقطة النهاية الوحدات المحدّدة مع الكيانات المدرجة لكل منها، في ملف من نوع `kind: "DataModule"` يتضمّن `dataModules` و `dataEntities` معًا. ويجب أن تكون كل وحدة موجودة في المساحة التي يصدر منها الاستدعاء، ويُطبَّق فحص الإذن على كل كيان هنا أيضًا.

الاستيراد

صُمّم الاستيراد ليجري على مرحلتين، ويُمرَّر الملف بالمرجع: ارفع ملف JSON إلى خدمة التخزين أولًا، ثم مرّر معرّفه في المسار. ويُرفَض الملف الذي يتعذّر تحليله بالرمز `IMPORT_FILE_IS_INVALID` ؛ أما إذا كانت قيمة `kind` خارج قيم التعداد، فيُرفَض بالرمز `FILE_KIND_NOT_SUPPORTED` .

`POST /v1/import/{fileId}` – التشغيل التجريبي

يحلّل الملف ويبيّن ما *سيحدث عند تطبيقه*، دون كتابة أي شيء. الجسم `{"moduleId": "…"}` – وهو اختياري في المخطط، لكنه مطلوب عمليًا لملف نوع الكيان: فهذا الملف لا يتضمّن وحدات خاصة به، ومن دون تحديد وحدة هدف، لا يستطيع الاستيراد تحديد موضع إنشاء الكيانات، فيفشل بخطأ يشير إلى عدم وجود الوحدة (`prepareImportData`). أما في ملف نوع الوحدة، فتُتجاهَل قيمة `moduleId` في الجسم، إذ تُؤخَذ الوحدات من الملف.

الاستجابة `ImportResultDto` : ملخّصات لكل كيان ووحدة – `{resourceTempId, normalizedName, name, alreadyExists, allowOverride}` – و `dependencyMap` لتبعيات الكيانات داخل الملف، ومصفوفة `errors[]` من `{resourceTempId, path, errorCode, errorMessage}`. وتُعلَّم الأسماء المعيارية المكرّرة داخل الملف (`ENTITY_DUPLICATED_INSIDE_IMPORT_FILE` و `MODULE_DUPLICATED_INSIDE_IMPORT_FILE`)، وكذلك مراجع الكيانات التي يتعذّر حلّها.

`POST /v1/import/{fileId}/confirm` – التطبيق

يتّبع المسار النمط نفسه، ويستخدم الجسم نفسه، مع شرط إضافي واحد: يتطلّب التأكيد رخصةً تشمل الإدارة؛ ففحص الحزمة لا يغيّر شيئًا، أما تطبيقها فيغيّر التهيئة (`403 LICENSE_ADMIN_FEATURE_REQUIRED` بخلاف ذلك). ويعيد التأكيد إجراء تحليل التشغيل التجريبي، ثم **يرفض كليًّا** (`IMPORT_FILE_IS_INVALID`) إذا كشف التحليل عن أي خطأ، أو عن أي مورد متصادم لا يُسمح باستبداله؛ فلا يُطبَّق الملف جزئيًا، ولا يُستثنى مورد بعينه.

وعند متابعة التنفيذ، ينشئ الوحدات أو يحدّثها، ثم ينشئ الكيانات أو يستبدلها بترتيب التبعيات (`sortEntitiesByDependencies` ؛ وتؤدي دورة مرجعية إلى إيقاف العملية بالرمز `CIRCULAR_DEPENDENCY_DETECTED`)، ويعيد إنشاء التفاعلات المضمّنة في `triggers` لملف **نوع الكيان** حصرًا. ويجري إنشاء الكيانات واستبدالها بصورة غير متزامنة عبر ناقل أوامر المنصّة، لذا تؤكّد استجابة HTTP قبول المجموعة كاملة، ولا تؤكّد اكتمال إنشاء كل كيان أو استبداله.

و `Since 1.10` يُجرى فحصان إضافيان قبل كتابة أي شيء، ويؤدي الإخفاق في أيّ منهما إلى رفض التأكيد **بأكمله**:

- **الموارد المثبّتة** – تُرفض الكتابة فوق وحدة أو كيان مصدره حزمة مثبّتة، وكذلك إنشاء كيان داخل وحدة مثبّتة، بالرمز `403` `PACKAGE_RESOURCE_READ_ONLY`. اشتقّ نسخة من التثبيت أولًا؛ انظر الموارد المثبّتة.
- **سقف الكيانات** – عندما تحدّد باقة القاطن سقفًا لعدد الكيانات، يُرفض الاستيراد الذي سينشئ كيانات تتجاوز العدد المتاح ضمن هذا السقف بالرمز `403 ENTITY_CAP_EXCEEDED` .

قواعد التصادم

تُطابَق الموارد الموجودة باستخدام `normalizedName` داخل المساحة الهدف، وتحدّد راية `allowOverride` في تقرير التشغيل التجريبي وحدها إمكانية الاستبدال (`isEntityExistsResult` / `isModuleExistsResult`):

الحالة	قابل للاستبدال؟
الكيان موجود، **بلا مُدخلات**، واسمه غير محجوز	نعم – يُستبدل التعريف في مكانه
الكيان موجود **وله مُدخلات**	لا – يُمنَع الاستيراد
اسم الكيان كلمة محجوزة (`switch` ، `new` ، `class` ، …)	لا
الوحدة موجودة	نعم – يُحدَّث الاسم والوصف والأيقونة والمشاركة
الوحدة هي الوحدة الأساسية	لا

وقد تكون قاعدة المُدخلات غير متوقّعة: فهذه الآلية لا تستطيع تحديث *تعريف* كيان إلا ما دام فارغًا في المساحة الهدف. وبمجرد وجود بيانات فعلية، لا يُجري نقل JSON أي تغيير عليه؛ أما تطوير مخطط كيان يحتوي على بيانات فمسألة أخرى لا تصلح لها هذه الأداة.

ما الذي لن يحمله

تبدو صيغة الملف أشمل مما ينفّذه الاستيراد. وفي الرماز الحالي:

- **المعارض والاستعلامات والمجموعات والعلاقات** – يضمّنها التصدير في الملف، لكن الاستيراد لا يعيد إنشاءها مطلقًا؛ فهو ينشئ الوحدات والكيانات والتفاعلات فحسب، ومسار إنشاء الكيان يحمل تعريف الكيان وحده. لذا تصل المصفوفات المضمّنة ضمن الحمولة من دون أن تُعالَج.
- **التفاعلات – تُنقل نقلًا منقوصًا.** يعيد تأكيد ملف نوع الكيان إنشاء التفاعلات المضمّنة؛ أما **تأكيد ملفّ نوع الوحدة فيُسقِطها**، إذ لا يُرسِل مساره التفاعلات المحضَّرة مطلقًا. لذا، إذا احتجت إلى نقل التفاعلات مع وحدة، فلن يحقّق لك هذا المسار ذلك.
- **المُدخلات** – تُنقَل التعريفات فقط، ولا تُنقَل البيانات مطلقًا. ولا تتضمّن الصيغة أصلًا أي بنية تمثّل مُدخلًا.
- **كل ما يقع خارج ثنائية الوحدة/الكيان** – لوحات المعلومات، والتقارير، والوظائف، وأزرار النظام، والاستعلامات الإحصائية، وقطع العدّادات، والإعلانات، والصفحات/المعارض العامة: لا تمثيل لها في الصيغة إطلاقًا.
- **الهوية والأذونات** – لا إعادة ربط لأصحاب الصلاحيات، ولا مراجع للقيم، ولا منح للأذونات خارج حقل `sharing` المضمّن في تعريفَي الوحدة والكيان.
- **الإجراءات** – لا جلسة، ولا خطوة تهيئة، ولا إجراءات لكل مورد، ولا تجاوزات للحمولة، ولا إعادة محاولة. فالتأكيد عملية واحدة: إما أن يطبّق كل شيء، وإما أن يرفض كل شيء.

متى تستخدم التحزيم بدلًا منه

استخدم مسار JSON هذا ضمن نطاقه المحدّد: نسخ *هياكل* الوحدات والكيانات سريعًا بين المساحات في ملف واحد، سواء لنقل مخطط أولي، أو قالب لكيانات فارغة، أو تحديث تعريف ما دام الكيان في المساحة الهدف فارغًا.

والجأ إلى نظام التحزيم متى انطبق أيّ مما يلي: ضرورة نقل المعارض أو الاستعلامات أو التفاعلات مع الوحدات أو التقارير أو أي نوع آخر من الموارد؛ أو الحاجة إلى إعادة ربط مراجع المستخدمين والمجموعات والمُدخلات بدلًا من الاعتماد على المصادفة؛ أو حاجة موارد الهدف الموجودة إلى قرارات إنشاء/استبدال/تخطٍّ لكل مورد؛ أو حاجتك إلى تقرير تحقّق تستند إليه، أو تجاوزات للحمولة، أو إعادة محاولة، أو أثر تدقيق. أما سؤال الملف الواحد في ذلك النظام – «هل أحصل على JSON واحد فيه كل شيء؟» – فتجيب عنه صفحة صيغة الحزمة بوضوح: حزمة الحلّ الكاملة ملف ZIP متعدّد الملفات، ومسار JSON هذا ليس بديلًا عنها.

الحزم والمتجر والمتجر داخل التطبيق

هذه الصفحة موجّهة إلى **المُكامِلين والمهندسين**: نموذج الوصول، والمسارات، ورموز الأخطاء، وقواعد الخادم التي تقوم عليها الحزم، والإصدارات المنشورة، وصندوق المراجعة، والاتصال بالمتجر، والمتجر داخل التطبيق.

تعمل من الواجهة؟

إنشاء الحزم والإصدارات، وملء صفحة المتجر، والإرسال للمراجعة، والمراجعة، وتثبيت التطبيقات وتحديثها واشتقاقها وإلغاء تثبيتها – كل ذلك مشروح لبُناة التطبيقات في تصدير التطبيقات واستيرادها وتثبيتها [بناء التطبيقات].

Availability

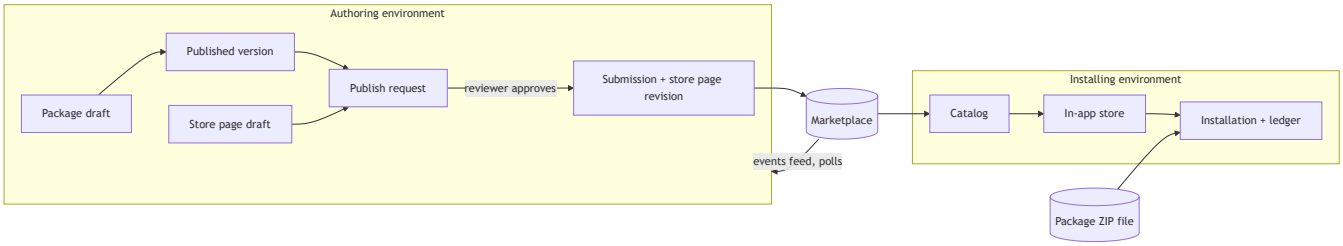
Since 1.10. وتذكر بعض إضافات 1.10 اللاحقة الإصدار التصحيحي الذي تصل فيه (بلاغات المشكلات من 1.10.1؛ وإشعار إيقاف النشر والتنبيه إلى ملف README من 1.10.2). ويضيف خط التطوير غيرها، وهي موسومة بـ `Since dev` حيث توصف: رفض الحفظ القديم، وقطع الصفحات التي تعرض عرضاً من خارج الحزمة، وصور الويب الملصقة في ملف README.

ينقل سير العمل القائم على الملفات الموارد بين البيئات في ملف ZIP مجهول الهوية: تصدّره، ثم تسلّمه، ثم تستورده. ويبني خط الإصدار 1.10 فوقه طبقة ثانية لبُناة التطبيقات الذين يريدون شحن **قالب** تستطيع بيئات أخرى تثبيته وتتبّعه:

- **حزمة** – هوية نشر دائمة ومسمّاة تملك وحدة واحدة أو أكثر؛
- **إصدارات منشورة** منها – نسخ من ملف ZIP للحزمة غير قابلة للتغيير، مرقّمة وفق semver؛
- **صفحة متجر** – بطاقة التطبيق في المتجر، تُصاغ مسوّدتها بجانب الحزمة؛
- **صندوق مراجعة** داخلي، فلا يصل شيء إلى المتجر دون موافقة شخص ثانٍ؛
- **اتصال بالمتجر** يسجّل البيئة لدى متجر أسس، وهوية **ناشر** مستقلة للبيئات التي تنشر؛
- **متجر داخل التطبيق** يتصفّح فهرس المتجر ويثبّت منه؛
- **دفتر تثبيت** لدى الجهة المستقبِلة يقفل ما أنشأه التثبيت إلى أن يشتقّه أحدٌ عن قصد.

التثبيت في جوهره ما زال استيراداً – الجلسة نفسها ومحرّك إعادة الربط نفسه – يجري في *وضع التثبيت* الذي يضيف فحوص الإصدار والتبعيات والملكية. وهذه الصفحة هي جانب التصميم والتكامل من هذه الطبقة: نموذج الوصول، والمسارات، والقواعد التي يفرضها الخادم. أما إضافات صيغة ملف الحزمة التي تعتمد عليها (كتلة `package` في البيان، و`hashes.json` ، و`package-references.json`، وملف README) فموصوفة في صيغة الحزمة؛ وآليات التصدير والاستيراد والمطابقة التي تحتها موصوفة في الحزم والتوزيع.

> ⚠ **معطّل حتى يفعّله المشغّل**
>
> اتصال المتجر **معطّل افتراضياً**. لا تتواصل خدمة الحزم مع أي متجر إلا بعد ضبط عناوين المتجر لديها؛ ومخطط النشر (chart) يأتي بالقيمة `marketplace.enabled: false` . وإلى أن يفعّله المشغّل يُرجِع كل مسار من مسارات المتجر `503` `MARKETPLACE_DISABLED`، ويُخفي تطبيق الويب كل واجهات المتجر (المتجر وصفحة المراجعات ومحرّر صفحة المتجر). أما الحزم والإصدارات والتثبيت من *ملف* حزمة فتعمل بدونه.



المصطلحات

المصطلح	ما هو
الحزمة	هوية النشر لقالب: معرّف دائم `pkg_<ULID>` ، واسم معروض ووصف مترجمان، والوحدات التي تملكها، ونطاق semver لكل حزمة تعتمد عليها. الحزمة بيانات مسوّدة في بيئة التأليف، وليست ملفاً.
إصدار الحزمة	نسخة منشورة واحدة غير قابلة للتغيير: ملف ZIP في التخزين مع `sha256` و `rootHash` القانوني، ولقطة لوحداتها وتبعياتها، وحالة (`published` أو `deprecated`)، وإرسالها إلى المتجر بعد أن يُرسَل.
صفحة المتجر	مسوّدة بطاقة الحزمة في المتجر: اسم المتجر، والمواضيع، ورابط التوثيق، وملاحظة الصيانة، والأيقونة، ولقطات الشاشة. ووصفها هو وصف الحزمة نفسه.
طلب النشر	عنصر مراجعة داخلي يحمل إصداراً أو صفحة متجر نحو المتجر. ويمكن أن ينتقل الإصدار وصفحة متجره معاً بوصفهما **مجموعة** واحدة.
اتصال المتجر	تسجيل البيئة لدى المتجر. واحد لكل بيئة، ويُحفظ في المساحة الأساسية (ROOT). ويوجد بناشر أو بدونه.
الناشر	هوية المتجر التي تظهر تحتها التطبيقات المنشورة: اسم وعنوان متجر. ويُعَدّ منفصلاً عن الاتصال، ولا تحتاج إليه إلا البيئة التي تنشر.
التثبيت	سجلّ الجهة المستقبِلة لإصدار حزمة واحد مثبّت في مساحة، مع دفتر بكل مورد أنشأه.
سجلّ الاستيراد	صف في تاريخ أي استيراد – استيراد عادي، أو ملف حزمة، أو تثبيت من المتجر.
المتجر	تجربة التصفح والتثبيت داخل التطبيق فوق فهرس المتجر. ولا يُخلط بينه وبين *مخزن* البيانات (مخازن البيانات [بناء التطبيقات › مشاركة مُدخلات الكيان بين المساحات]).

كل ما يتواصل مع المتجر – الاتصال، وطلبات النشر، وصفحة المتجر، والفهرس – متاح **في المساحة الأساسية (ROOT) فقط**. أما الحزم والتثبيتات وسجلّات الاستيراد فلكل مساحة.

نموذج الوصول

مجموعات المتجر الثلاث

حقوق المتجر هي العضوية في ثلاث مجموعات مستخدمين نظامية، يزرعها data-service في كيان مجموعات المستخدمين في المساحة الأساسية (الترحيل `1.10.0-a37-marketplace-permission-groups`):

الإذن	معرّف المجموعة	ما يتيحه للعضو
`publish`	`MARKETPLACE_PUBLISHERS`	إعداد النشر، وإعداد طلبات النشر وسحبها، وإلغاء نشر التطبيق وإعادة نشره، وقراءة شروط الناشر.
`review`	`MARKETPLACE_REVIEWERS`	قراءة صندوق المراجعة، والموافقة على الطلب أو إرجاعه مع ملاحظة، وقبول شروط الناشر.

الإذن	معرّف المجموعة	ما يتيحه للعضو
`install`	`MARKETPLACE_INSTALLERS`	تصفح الفهرس، وفتح صفحة التطبيق، وربط البيئة، والتثبيت من المتجر.
`update`	`MARKETPLACE_INSTALLERS`	المجموعة نفسها التي لـ `install`؛ وتحكم تحديث التطبيق المثبّت في مكانه.

ويدمج الترحيل `1.10.0-a40-merge-marketplace-updaters` مجموعة سابقة هي `MARKETPLACE_UPDATERS` في مجموعة المثبّتين، ولا يحذف المجموعة القديمة إلا بعد أن تفرغ.

النشر والمراجعة يحتاجان إلى المجموعتين، حتى للمسؤولين. فمسؤول النظام (`AdminAccess` على مستوى النظام) لا يملك دون الانتماء إلى مجموعة إلا `install` و`update`؛ أما `publish` و`review` فلا يأتيان إلا من العضوية في المجموعة. ويملك استدعاء الخدمة الداخلي الأذونات الأربع. وتُخزَّن العضوية مؤقتاً لكل قاطن ومستخدم مدة **60 ثانية**؛ ومسؤول النظام الذي يُرفض يعيد قراءة مجموعاته مرة واحدة دون الذاكرة المؤقتة، فلا يضطر المسؤول الذي أضاف نفسه للتو إلى مجموعة إلى الانتظار. وغياب الإذن يُرجِع `403 MARKETPLACE_PERMISSION_REQUIRED`.

وبعض العمليات محجوزة لمسؤولي النظام أياً كانت المجموعة: **إعادة تسمية الناشر**، و**تغيير عنوان المتجر**، وإكمال تثبيت من المتجر ما زالت فيه قرارات غير محسومة.

النشر استحقاق في الباقة

يجوز لأي بيئة أن تثبّت؛ أما النشر فلا يجوز إلا لبيئة تتضمن باقتها الاستحقاق `marketplacePublish`. ويقرؤه العملاء بوصفه `publishInPlan` في `GET /v1/marketplace/me`. وخارج الباقة يرفض الخادم بالرمز `403` `MARKETPLACE_PUBLISH_NOT_IN_PLAN` في هذه العمليات تحديداً:

- إعداد طلب نشر (لإصدار، أو لصفحة متجر، أو لكليهما بوصفهما مجموعة)؛
- الموافقة على طلب نشر أو على مجموعة؛
- إعداد النشر، وإعادة تسمية الناشر، وتغيير عنوان المتجر؛
- إلغاء نشر تطبيق أو إعادة نشره.

ويبقى مفتوحاً: الربط، والتصفح، والتثبيت والتحديث، وفحص عنوان متجر، ومسارات الشروط، وسحب الطلب أو إرجاعه، وكل القراءات. لذا تستطيع بيئة بلا هذا الاستحقاق أن تتصل *للتثبيت فقط*. ويُظهر تطبيق الويب للناشرين والمراجعين إشعاراً حين لا يكون النشر ضمن الباقة؛ أما الخلفية الأقدم من أن تُبلغ عن `publishInPlan` فتُعامَل على أنها تسمح به.

أين يُصرَّح بكل مسار

المسارات	من يحق له استدعاؤها
`/v1/packages/**`	مستخدم مصادَق عليه، مع **Developer أو أعلى على كل وحدة في الحزمة** (`403 PACKAGE_DEVELOPER_REQUIRED`). وتحتاج مسارات الإرسال أيضاً إلى المساحة الأساسية وإذن متجر.
`/v1/packages/{id}/listing` (القراءة والحفظ والصور)	المساحة الأساسية. **يكتب مسوّدةَ صفحة المتجر أيُّ مطوّر للحزمة** – دون حاجة إلى مجموعة متجر؛ ويجوز لحامل `review` قراءتها أيضاً. أما الإرسال وإلغاء النشر وإعادة النشر فتحتاج إلى `publish`.
`/v1/publish-requests/**` `/v1/publish-request-groups/**`	المساحة الأساسية، مع `review` (أو `publish` للسحب، و`publish` أو `review` للتحديث).
`/v1/marketplace/connection` (القراءة والربط)	المساحة الأساسية، مع `install` أو `publish` – فيستوفيه كل مسؤول عبر `install`.
`/v1/marketplace/me`	المساحة الأساسية؛ ويُرجِع أذونات المستدعي نفسه.

المسارات	من يحق له استدعاؤها
`/v1/marketplace/publisher` (الإعداد) و `/namespaces/{name}` (فحص عنوان)	المساحة الأساسية، مع مسؤول نظام أو `publish` .
`/v1/marketplace/publisher` (إعادة التسمية) و `/namespace` (تغيير العنوان)	المساحة الأساسية، لمسؤول النظام فقط.
`/v1/marketplace/slugs/{slug}`	المساحة الأساسية، و `publish` ، ووجود ناشر على الاتصال.
`/v1/marketplace/terms`	القراءة: `publish` أو `review` . والقبول: `review` .
`/v1/marketplace/catalog/*` و `/publishers/*`	المساحة الأساسية، مع `install` . و `/topics` يقبل `publish` أو `install` .
`/v1/installations/**`	مسؤول المساحة (منحة Admin على مستوى النظام في المساحة)؛ والإزالة ومعاينتها تحتاجان إلى **مسؤول البيئة** – أي مسؤول نظام يستدعي من المساحة الأساسية (ROOT). ومسارات التحديث تحتاج إلى مسؤول البيئة **و** `update` . و `POST /v1/installations/origins` مفتوح لأي مستخدم في المساحة، ليعرف المصمّم أن المورد مقفل قبل محاولة حفظه.
`/v1/imports/**`	مسؤول المساحة؛ و `POST /v1/imports/origins` مفتوح لأي مستخدم في المساحة.

وتقوم هذه القواعد إلى جانب قواعد 1.10 لسير العمل القائم على الملفات: فالتصدير العادي يُصرَّح به بـ صلاحية Developer على الجذور المصدَّرة، وكل مسار استيراد لمسؤول المساحة فقط.

هوية الحزمة

معرّف الحزمة هو `pkg_` يتبعه ULID من 26 حرفاً بترميز Crockford base32 (`^pkg_[0-9A-HJKMNP-TV-Z]{26}$`). وهو دائم ويسافر في بيان كل إصدار منشور، وبه تتعرّف بيئة أخرى على «الحزمة نفسها بإصدار أحدث».

- **الوحدة تنتمي إلى حزمة واحدة على الأكثر.** تُرفَض حزمة ثانية تطالب بها (`409 MODULE_IN_OTHER_PACKAGE` ، ويفرضه فهرس فريد فلا ينجح حفظان متزامنان معاً).
- **الاسم المختصر في المتجر** (store slug) هو اسم الحزمة في المتجر: أحرف لاتينية صغيرة وأرقام وشَرطات، يبدأ بحرف أو رقم، وطوله 50 حرفاً على الأكثر. يُختار عند إرسال أول إصدار للمراجعة و**يُقفل حين يوافق عليه المتجر**: وبعدها يُرجِع تغييره `409 SLUG_LOCKED` . والاسم الذي تستخدمه حزمة أخرى يُرجِع `409 SLUG_TAKEN` ، والاسم الذي يحجزه طلب مفتوح `GET /v1/marketplace/slugs/{slug}?packageId=` ويفحص `409 STORE_SLUG_IN_USE_BY_OPEN_REQUEST` . التوفّر مسبقاً (`available` ، مع `TAKEN` أو `INVALID` أو `LOCKED`).
- **الملفات الأقدم.** يُرفَض عند التحقق ملفٌّ إصدار منشور صُدِّر قبل وجود معرّفات الحزم، بالرمز `422 PACKAGE_FORMAT_OUTDATED` – صدّره من جديد من بيئته. ويمنح ترحيل data-service ذو الاسم `1.10.0-a39-package-identity` كل حزمة موجودة سابقاً معرّف `pkg_` .

تأليف الحزم والإصدارات

مسوّدات الحزم

المسار	الغرض
`GET /v1/packages`	حزم المساحة التي يحق للمستدعي تأليفها، الأحدث تعديلاً أولاً. ويحمل كل ملخّص حالته في المتجر: الاسم المختصر وهل هو مقفل، وهل اسم المتجر مقفل، وهل أُلغي نشر التطبيق، وهل **يملكه ناشر آخر**، وآخر حالة متجر معروفة.

المسار	الغرض
`POST /v1/packages`	إنشاء مسوّدة: `displayName` (مترجم، مطلوب)، و`description`، و`moduleIds`، و`pageIds` (الصفحات العامة)، و`dependencyRanges` (معرّف حزمة ← نطاق semver). ولا تُشترط أي من القائمتين عند الحفظ؛ أما الحزمة الفارغة فتُرفض عند إنشاء إصدار (`PUBLISH_EMPTY`).
`GET /v1/packages/{id}`	المسوّدة بوحداتها وصفحاتها محلولة (`missing: true` للمحذوف منها) وبأسماء الحزم التي تعتمد عليها.
`PUT /v1/packages/{id}`	استبدال المسوّدة. و`Since dev` يأخذ `updateIdentifier` الذي قرأه العميل آخر مرة ويرفض القديم منه – راجع رفض الحفظ القديم.
`DELETE /v1/packages/{id}`	فقط ما دامت الحزمة بلا إصدار (وإلا `409 PACKAGE_HAS_VERSIONS`).

رموز التحقق عند الحفظ: `DISPLAY_NAME_REQUIRED`، و`DEPENDENCY_RANGE_INVALID`، و`MODULE_NOT_FOUND` و`PAGE_NOT_FOUND` (`404`)، و`DESCRIPTION_TOO_LONG` . **وصف الحزمة هو أيضاً وصف صفحة المتجر**، ولذلك يُحدّ كل لسان بـ 120 حرفاً عند كتابته (`400 DESCRIPTION_TOO_LONG`). أما الوصف الأطول المخزَّن قبل وجود هذا الحد فيبقى إلى أن يُحرَّر ذلك اللسان.

رفض الحفظ القديم `Since dev`

يستبدل `PUT` المسوّدة المسوّدةَ كلها، ويستبدل `PUT` ملف README نصَّ Markdown كله، فكان شخصان (أو تبويبان) يحرّران الحزمة نفسها يكتب أحدهما فوق الآخر دون أن ينتبه. ويضيف خط التطوير إلى الاثنين رمزاً للتزامن المتفائل:

- **الرمز.** تحمل كل حزمة `updateIdentifier` يُصدَر عند إنشائها و**يتجدّد مع كل حفظ للمسوّدة وكل حفظ لنص Markdown في ملف README – ومعهما وحدهما**. أما الكتابات في الخلفية (استطلاع المتجر، والنشر، ومسوّدة صفحة المتجر) وإضافة صورة إلى README أو حذفها فلا تغيّره. والحزمة المنشأة قبل وجود الرمز لا رمز لها حتى أول حفظ.
- **أين ينتقل.** يُعاد في قائمة الحزم وتفاصيل الحزمة وجواب `PUT`، وفي ملف README للمسوّدة (`GET /v1/packages/{id}/readme`). ويُقبَل في جسم `PUT /v1/packages/{id}` و`PUT /v1/packages/{id}/readme` (`{markdown, updateIdentifier?}`).
- **العقد.** الرمز **المُرسَل** الذي لم يعد مطابقاً يُرفض بـ `409 STALE_UPDATE_IDENTIFIER` («…غيّره حفظ آخر؛ أعد تحميلها ثم أعد تعديلك»). ويُفحص عند قراءة الحزمة ثم مرة أخرى داخل الكتابة، فلا يمرّ حفظ يقع بين الاثنين. أما الرمز **المحذوف** من الطلب فلا يُفحص، فتبقى العملاء التي لا ترسله تعمل. ويُجيب كل `PUT` بالرمز **الجديد**؛ ويُجيب `PUT` ملف README بنص Markdown والرمز اللذين كتبهما هو، لا بما كتبه حفظ آخر.
- **في تطبيق الويب.** ترسل صفحة الحزمة آخر رمز قرأته، ولا تعتمد رمزاً جديداً إلا إذا جاء التجديد من حفظها هي. فيُرفض الحفظ التالي من تبويب أقدم بدل أن يكتب فوق غيره بصمت. وتطلب الرسالة من المستخدم إعادة التحميل ثم إعادة التعديل.

نشر إصدار

للنشر تشغيل تجريبي. يسرد `GET /v1/packages/{id}/publish-preview?version=&minPlatformVersion=` كل المشكلات دون نشر شيء، ومعها أعداد الموارد، والحزم التي اكتُشف الاعتماد عليها (بنطاق caret افتراضي)، والتبعيات الخارجية، والإصدار التالي المقترح للتصحيح/الثانوي/الرئيسي (`1.0.0` لأول نشر)، و`suggestedBump` المحسوب من مقارنة المحتوى بأحدث إصدار، و`canPublish` .

ويُرجِع `POST /v1/packages/{id}/versions` مع `{version, releaseNotes, minPlatformVersion?}` الاستجابة **`202`** `{jobId}` ويجري بوصفه مهمة تصدير: استعلم `GET /v1/export/{jobId}/status`، وتحمل نتيجته حينها كتلة `publish` (`packageId` و`versionId` و`version` و`sha256` و`rootHash`).

قواعد الإصدار:

- الإصدار **semver صارم**، ويجب أن يكون **أعلى من كل إصدار سُجِّل للحزمة**، بما فيها الإصدارات الموقوفة والتجريبية؛
- **ملاحظات الإصدار مطلوبة**؛
- يأخذ `minPlatformVersion` افتراضياً إصدار المنصة في البيئة نفسها، ولا يجوز إلا **رفعه** – يمكن تضييق الدعم ولا يمكن توسيعه؛
- «الأحدث» هو أعلى إصدار *منشور غير تجريبي*.

ويُرفَض النشر برمز أول مشكلة ما دامت أي من هذه قائمة:

المشكلة	معناها
`PUBLISH_EMPTY`	لا تضمّ الحزمة وحدة ولا صفحة.
`PUBLISH_MODULE_MISSING` و`PUBLISH_PAGE_MISSING`	حُذفت إحدى وحداتها أو صفحاتها.
`PUBLISH_PAGE_NOT_GLOBAL`	صارت صفحة في الحزمة تنتمي إلى وحدة أو كيان؛ ولا تحمل الحزمة إلا الصفحات العامة.
`PUBLISH_MODULE_IN_OTHER_PACKAGE`	تطالب حزمة أخرى بإحدى الوحدات.
`PUBLISH_MODULE_INSTALLED` و`PUBLISH_MODULE_FORKED` و`PUBLISH_PAGE_INSTALLED` و`PUBLISH_PAGE_FORKED`	جاءت وحدة أو صفحة من تثبيت (مشتقّ أو غير مشتقّ)؛ فلا يمكن إعادة نشر قالب مثبّت على أنه لك.
`PUBLISH_SYSTEM_FUNCTION_REFERENCE`	يستدعي مورد في الحزمة وظيفة لا تنتمي إلى أي وحدة.
`PUBLISH_PACKAGE_DEPENDENCY_UNSUPPORTED` و`PUBLISH_DEPENDENCY_CYCLE` و`PUBLISH_DEPENDENCY_FORKED` و`PUBLISH_DEPENDENCY_RANGE_INVALID` و`PUBLISH_DEPENDENCY_REFERENCE_UNSUPPORTED`	اعتماد على حزمة أخرى لا يمكن التعبير عنه: غير مدعوم، أو دائري، أو مشتقّ، أو بنطاق غير صالح، أو بنوع مرجع لا يعبر بين الحزم.
`PUBLISH_VERSION_INVALID` و`PUBLISH_VERSION_NOT_INCREASING`	الإصدار ليس semver صارماً، أو ليس أعلى من كل الإصدارات المسجّلة.
`PUBLISH_MIN_PLATFORM_INVALID`	الحد الأدنى لإصدار المنصة غير صالح أو أدنى من المسموح.
`PUBLISH_RELEASE_NOTES_REQUIRED`	لا توجد ملاحظات إصدار.
`PUBLISH_RESERVED_FIELD_PREFIX`	اسم حقل يستخدم البادئة المحجوزة `x_`.

مسارات أخرى للإصدارات: `GET /v1/packages/{id}/versions` (الأعلى أولاً) و`/versions/{version}`؛ و`/versions/{version}/download` (ملف ZIP مرفقاً)؛ و`POST /versions/{version}/deprecate` مع `message` مترجمة، و`/undeprecate`؛ و`GET /v1/packages/{id}/compare?from=&to=`، حيث يمكن أن تكون `to` القيمة `current` (الوحدات الحيّة) – ويبلّغ عن الموارد المضافة والمحذوفة والمتغيّرة وعن `suggestedBump`.

التبعيات الخارجية. كل ما تشير إليه الحزمة ولا تحمله يُعلَن في بيان الإصدار تبعيةً خارجية، يربطها المثبِّت بشيء يخصّه. و`Since dev` يشمل ذلك أيضاً العرض (أو أي مورد آخر) الذي تعرضه **قطعة في صفحة** من خارج الحزمة: يُعلَن دون أن يصير تبعية حزمة، وربطه اختياري – راجع قطع الصفحات التي تشير إلى خارج الحزمة.

وتعتمد المقارنة والسلامة كلتاهما على **تجزئات محتوى قانونية**: يُجزَّأ كل مورد مع استبعاد المفاتيح التي يديرها الخادم والمفاتيح الخاصة بالبيئة، واستبدال المعرّفات بمفاتيح نسبية داخل الحزمة، فيُعطي المحتوى نفسه التجزئة نفسها في أي بيئة.

أما حذف إصدار، وتاريخ الحزمة، وسجلّات التصدير فتنتمي إلى سير العمل القائم على الملفات، وهي موصوفة في الحزم والتوزيع. وتهمّ هنا قاعدة واحدة: **حذف إصدار أو تصدير يزيل سجلّه فقط – ويبقى ملف ZIP في التخزين**، لأن النسخ الاحتياطية للقاطن تسرد الملفات المخزَّنة دون أن تنسخها.

ملخّص المحتويات

يحمل كل بيان مُصدَّر – تصديراً عادياً كان أو إصداراً منشوراً – ملخّصاً `contents`: أعداداً لكل نوع مورد، وقائمة `modules[]` بكيانات كل وحدة وعدد حقولها (من دون الحقول الضمنية)، وخانة `other` للموارد الواقعة خارج أي وحدة مُصدَّرة، و`truncated:true` حين يُبلَغ حدّ (**100 وحدة و500 كيان**). ولا تُحفظ من الأسماء إلا `en` و`ar`. ويخزّن التثبيت هذا الملخّص، ويعرضه المتجر بعنوان *ما بداخلها*.

ملف README

للحزمة ملف README في المسوّدة – نص Markdown مع صور – يُحرَّر عبر `GET|PUT /v1/packages/{id}/readme` و`POST|GET|DELETE /v1/packages/{id}/readme/images[/{name}]`. ويجمّده النشر داخل ملف ZIP للإصدار. والحدود:

الحد	القيمة
نص Markdown	256 KiB
كل صورة	2 MiB؛ بصيغة PNG أو JPEG أو WebP أو GIF
الصور	50 صورة على الأكثر، و20 MiB إجمالاً
أسماء الصور	أحرف لاتينية صغيرة وأرقام و`.` و`_` و`-`، و100 حرف على الأكثر

يُقرأ README الإصدار المنشور من `GET /v1/packages/{id}/versions/{version}/readme`؛ ويحتفظ الاستيراد بالـ README الذي وصل معه في `GET /v1/imports/{id}/readme`.

يحتفظ الإصدار بملف README الذي أُنشئ معه. لا يمكن إضافة README إلى إصدار بعد إنشائه – فذلك يتطلب إصداراً جديداً – وتستخدم صفحة المتجر ملف README وصفاً مفصّلاً للتطبيق. وتقول صفحة الحزمة ذلك قبل أن يوجد الإصدار (راجع في منطقة الإدارة). ويخضع `PUT` ملف README لرفض الحفظ القديم `Since dev`.

صفحة المتجر

صفحة المتجر (بطاقة التطبيق في المتجر) مسوّدة مستقلة على الحزمة، تُحرَّر عبر `GET|PUT /v1/packages/{id}/listing` و`POST /v1/packages/{id}/listing/images` (`kind` = `icon` أو `screenshot`)، وتُجمَّد في طلب نشر خاص بها عبر `POST /v1/packages/{id}/listing/publish-requests`. وحقولها: **اسم المتجر** مترجماً، و**المواضيع**، و**رابط التوثيق**، وعلامة **الصيانة** مع ملاحظة مترجمة، و**الأيقونة**، و**لقطات الشاشة** مع تعليقاتها.

- **المواضيع تحلّ محلّ الحقول القديمة: المشكلة والميزات وجهة الاتصال والفئة.** تُنزع هذه الحقول الأربعة من أي مسوّدة مخزَّنة؛ والمسوّدة المحفوظة قبل وجود المواضيع تأخذ فئتها القديمة موضوعاً وحيداً لها. يُقصّ الموضوع ويُحوَّل إلى أحرف صغيرة وتُستبدل المسافات أو `_` فيه بـ `-`؛ ثم يجب أن يكون من 1 إلى 35 حرفاً أو رقماً أو شَرطة، يبدأ بحرف أو رقم. وتُسقط المدخلات غير الصالحة والمكرّرة عند إرسال الصفحة، والحد الأقصى **10** مواضيع. والحفظ الذي لا يحمل `topics` يُبقي المواضيع المخزَّنة.
- **الوصف هو وصف الحزمة.** ليس لصفحة المتجر وصف خاص بها: فهي تعرض دائماً وصف الحزمة، ويُتجاهل أي وصف يُرسَل مع الحفظ. ويظل الإرسال يفحص حدّ الـ 120 حرفاً ويبلّغ عن المشكلة تحت `description`. ولا تُرسَل صفحة متجر اعتُمدت من قبل دون وصف، لأن المراجعة الجديدة ستمحو وصف المتجر.
- **من يكتبها.** أي مطوّر للحزمة، دون مجموعة متجر. أما إرسالها فيحتاج إلى `publish`.

وحدودها مطابقة لحدود المتجر: اسم المتجر 60 حرفاً، والوصف 120، ورابط التوثيق 2048 (http أو https)، وملاحظة الصيانة 2000، وتعليق لقطة الشاشة 140، و8 لقطات شاشة على الأكثر. والصور بصيغة PNG أو JPEG أو WebP حتى 2 MiB؛ والأيقونة مربّعة من **256 إلى 1024** بكسل؛ ولقطات الشاشة بنسبة 16:9 (بهامش 2 في المئة) ومن **720×1280 إلى 2160×3840**. ولا يفحص الحفظ إلا الشكل والحدود، فيمكن الإبقاء على صفحة نصف مكتوبة؛ أما الإرسال فيفحص الحقول المطلوبة (`422` `LISTING_INVALID` مع سرد المشكلات). وما إن تُعتمد مراجعة لصفحة المتجر حتى **يُقفل الاسم** (`NAME_LOCKED`).

ويُخرج `POST .../listing/unpublish` التطبيق من المتجر ويعيده `.../republish` . وكلاهما يحتاج إلى `publish` وإلى استحقاق الباقة؛ ولا تحكمهما شروط الناشر – فالموافقة وحدها هي التي تحكمها.

طلبات النشر وصندوق المراجعة

لا يذهب شيء إلى المتجر مباشرة. يُعِدّ الناشر طلباً عبر `POST /v1/packages/{id}/versions/{version}/submit` (مع `storeSlug` إلى أن يُقفل الاسم المختص)، ثم يوافق عليه **مراجع في البيئة نفسها**؛ وعندها فقط ترفع المنصة أي شيء.

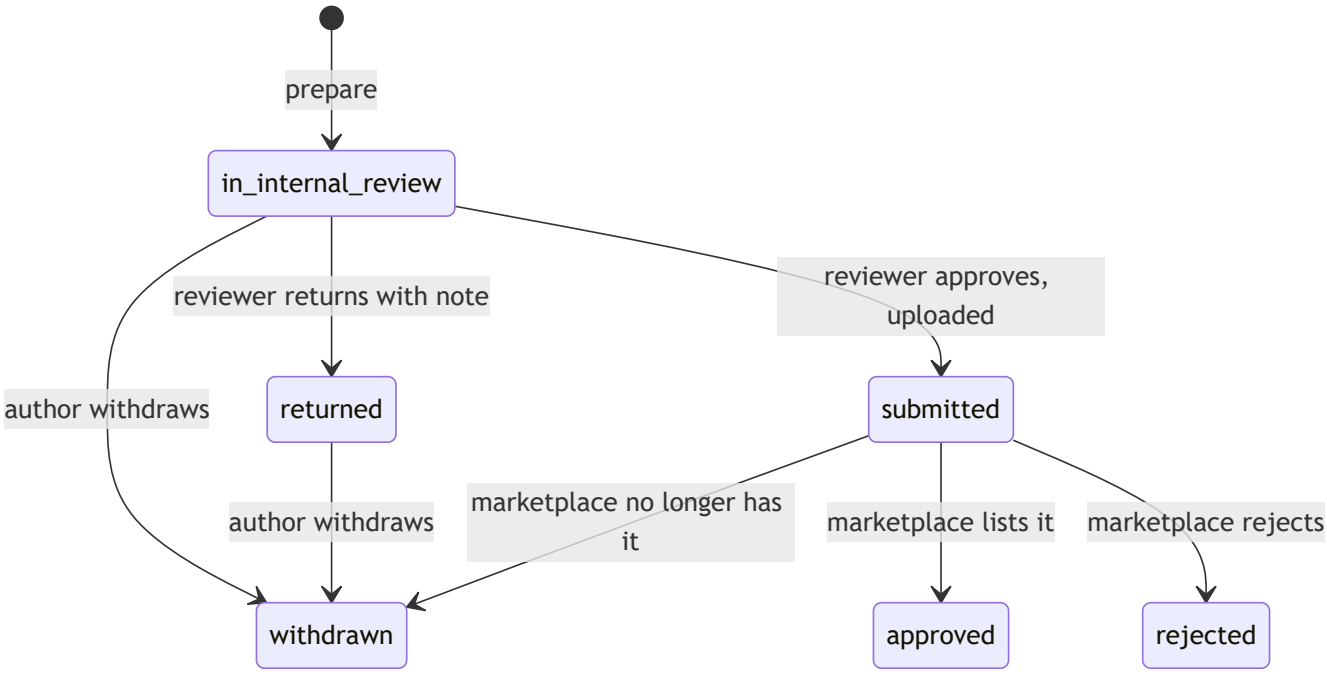
ويفحص الإعداد بهذا الترتيب:

1. إذن `publish` ؛
2. استحقاق الباقة (`marketplacePublish`)؛
3. **وجود ناشر على الاتصال الحيّ** – `409 MARKETPLACE_PUBLISHER_REQUIRED` («This environment has no publisher in the marketplace yet. Set up publishing first.»). فالاتصال وحده لا يكفي؛ انظر الربط وإعداد النشر.

ويُرفض الطلب كذلك ما دام ناشر آخر في المتجر يملك التطبيق، ولإصدار موقوف، ولإصدار له إرسال سابق إلى المتجر. والإصدار الذي **رفضه** المتجر لا يُرسَل مرة أخرى: انشر إصداراً جديداً.

شروط الناشر تُطلب عند الموافقة لا عند الإرسال. تفحص الموافقة `review` ، ثم الباقة، ثم الشروط: فحين لا تكون الشروط الحالية مقبولة يكون الجواب `409 MARKETPLACE_TERMS_REQUIRED` ، ويطلب مربع حوار الموافقة من المراجع قبولها (`POST` `/v1/marketplace/terms`، ويحتاج إلى `review`).

الحالات



للطلب نوع `kind` – `version` أو `listing` – ولا يمكن أن يكون مفتوحاً (`in_internal_review` أو `submitted`) في الوقت نفسه إلا طلب واحد لكل إصدار ولكل صفحة متجر (`409 LISTING_REQUEST_OPEN` ؛ والانتقال من حالة خاطئة هو `409 PUBLISH_REQUEST_STATE`).

المسار	يحتاج إلى	الأثر
`GET /v1/publish-requests?state=&skip=&limit=`	`review`	كل الطلبات في البيئة، الأحدث أولاً، مع عدد موارد كل طلب.
`GET /v1/publish-requests/{id}`	`review`	الطلب مع حزمته وملخّص الإصدار وملف README وصفحة المتجر المجمّدة، والمجموعة كاملة للطلب المنتمي إلى مجموعة.
`POST /v1/publish-requests/{id}/approve`	`review` ، والباقة، والشروط	يرفع إلى المتجر ويسجّل الإرسال. ويمكن استعادة مطالبة تُركت أكثر من 15 دقيقة.
`POST /v1/publish-requests/{id}/return`	`review`	يُرجِع الطلب مع ملاحظة من 1 إلى 1000 حرف.
`POST /v1/publish-requests/{id}/withdraw`	`publish`	المؤلف أو مسؤول نظام، من `in_internal_review` أو `returned` – أو طلب `submitted` لم يعد المتجر يملكه.
`POST /v1/publish-requests/{id}/refresh`	`publish` أو `review`	يعيد قراءة حالة المتجر.

ويسرد `GET /v1/packages/{id}/publish-requests` طلبات حزمة واحدة (`publish` أو `review`)، ويقرأ المراجع صور README وصور صفحة المتجر في الطلب عبر `GET /v1/publish-requests/{id}/readme/images/{name}` و `.../listing/images/{imageId}` . ويُسجَّل المراجع الذي وافق، ومعه هل كان هو المؤلف.

مراجعة واحدة للإصدار وصفحة متجره

يمكن أن ينتقل الإصدار وصفحة المتجر بوصفهما **مجموعة** واحدة، تُراجَع ويُبتّ فيها بخطوة واحدة:

المسار	الأثر
`POST /v1/packages/{id}/publish-requests/first-publish`	`{versionId, storeSlug?}` . **أول نشر يرسل الاثنين معاً** – الإصدار وصفحة المتجر لحزمة لم تُنشر قط. ويُرفض بالرمز `409 ALREADY_LIVE` بعد أن يُقفل الاسم المختصر أو اسم المتجر أو تُعتمد أي طلبات للحزمة.
`POST /v1/packages/{id}/publish-requests/together`	الزوج نفسه، سواء نُشر التطبيق من قبل أم لا. ولا يُرفض إلا ما دام *ذلك* الإصدار أو صفحة المتجر ينتظر قراراً.
`POST /v1/publish-request-groups/{groupId}/approve`	يوافق على الإصدار ثم على صفحة متجره. وصفحة المتجر التي تفشل تبقى قيد المراجعة ويمكن الموافقة عليها وحدها.
`POST /v1/publish-request-groups/{groupId}/return`	يُرجِع الطلبات التي ما زالت قيد المراجعة بملاحظة واحدة.
`POST /v1/publish-request-groups/{groupId}/withdraw`	يسحب الطلبات التي لم يُبتّ فيها بعد؛ ويجب أن يكون كل طلب للمستدعي ما لم يكن مسؤول نظام.

ويجتاز الطلبان الفحوص نفسها قبل تخزين أي منهما، فالفحص الفاشل لا يخزّن شيئاً. وإرجاع المجموعة أو سحبها لا يحرّك أياً من طلباتها إذا كان أحدها قيد الموافقة منفرداً في تلك اللحظة. ويتلقّى المرسل إشعاراً واحداً عن المجموعة كلها.

سحب طلب لم يعد المتجر يملكه

حين يُعاد قراءة طلب `submitted` ويجيب المتجر بأنه *غير موجود*:

- ما دام موجز الأحداث قيد الاستخدام، فالموجز – لا الرمز 404 – هو الذي يقرّر: يُعلَّم الطلب بأنه زال لكنه يبقى `submitted` ، ويجوز لشخص أن **يسحبه**. والسحب يغلقه بالحالة `withdrawn` مع السبب `MARKETPLACE_SUBMISSION_GONE`

(للإصدار) أو `MARKETPLACE_REVISION_GONE` (لصفحة المتجر). ولا يُعلَّم طلب صفحة المتجر ما دام التطبيق مُزالاً من المتجر:

- ومن دون الموجز يُغلق الطلب مباشرة بالسبب نفسه.

وسبب ثالث، `MARKETPLACE_APPLICATION_MOVED`، يغلق الطلبات المرسلة لتطبيق انتقل إلى ناشر آخر.

الإشعارات

مسار المراجعة هو أول منتِج لـ **إشعارات داخل التطبيق تحمل رابطاً** – فقد اكتسب كيان الإشعارات حقلاً `link` (الترحيل `1.10.0-a43-add-notification-link`) لا يجوز أن يشير إلا إلى داخل التطبيق. وتُكتب الإشعارات بالإنجليزية والعربية، في المساحة الأساسية. وخطوات المراجعة الداخلية داخل التطبيق فقط؛ أما **قرارات المتجر وتغيّرات المتجر وتغيّرات الناشر فتُرسَل أيضاً بالبريد الإلكتروني** (رسالة بالإنجليزية والعربية يفتح رابطها صفحة الحزمة). وكل إشعار يُبلغ أيضاً صفحات الحزم المفتوحة بأن تعيد التحميل.

الحدث	من يُشعَر	بريد	الرابط
أُرسل للمراجعة	أعضاء `MARKETPLACE_REVIEWERS` باستثناء المؤلف. وإن لم يوجد مراجع، يُبلَّغ مسؤولو النظام بالحاجة إلى مراجع.	لا	`/admin/packages/reviews/{requestId}` (الحاجة إلى مراجع: صفحة مجموعات المستخدمين)
وافق عليه مراجع وأُرسل	المؤلف، ما لم يكن هو من وافق	لا	`/admin/packages/build/{packageId}`
أُرجِع مع ملاحظة	المؤلف، ما لم يكن هو من أرجعه؛ ورسالة واحدة للمجموعة	لا	الرابط نفسه
وافق عليه المتجر أو رفضه	المؤلف وكل أعضاء مجموعتَي الناشرين والمراجعين؛ ويحمل الرفض سببه	نعم	صفحة الحزمة، أو صفحة متجرها حين يخصّ القرار صفحة المتجر
تغيّرت حالة المتجر (لم يُنشر بعد، أو منشور، أو أُزيل مع سببه، أو عاد إلى المتجر، أو أُلغي نشره)	الجمهور نفسه	نعم	الرابط نفسه
تغيّر عنوان المتجر أو أُضيف، أو انتقلت تطبيقات إلى الناشر أو منه	الناشرون والمراجعون	نعم	`/admin/packages/build`
أنهى المتجر الاتصال	الناشرون والمراجعون	نعم	`/admin/packages/build`
أبلغ عميل عن تثبيت أو تحديث فاشل لأحد تطبيقاتك `Since 1.10` (من 1.10.1)	الناشرون والمراجعون	نعم	`/admin/packages/build/{packageId}`

والروابط التي كُتبت قبل إعادة تسمية صفحة «البناء» (`/admin/packages/mine/...`) يعيد تطبيق الويب توجيهها إلى `/admin/packages/build/...`.

الاتصال بالمتجر

الإعداد

تقرأ خدمة الحزم ثلاثة عناوين للمتجر: أصل الواجهة البرمجية، والأصل الذي يجب أن تمرّ منه عمليات نقل ملفات الحزم، وأصلاً اختيارياً للويب. ويجب أن يكون كلٌّ منها أصل `https` مجرّداً (ويُقبل `http://localhost` خارج بيئة الإنتاج). وفي الإنتاج تعني القيمة غير المضبوطة أو المشوّهة أن المتجر **معطّل** – `503 MARKETPLACE_DISABLED` على كل مسار من مساراته. ويحتاج تطبيق الويب إلى صلاحية عنوانَي الواجهة البرمجية والويب للمتجر كليهما قبل أن يعرض أي واجهة للمتجر. وكيفية تفعيل المشغّل للاتصال، وأي متجر يشير إليه، من إعدادات النشر؛ وعلى مستوى المخطط (chart) تكون القيمة الافتراضية لـ `marketplace.enabled` هي `false`.

ويأتي **الاسم الذي تُسجَّل البيئة وتنشر تحته** من القاطن: ففي الخلية متعددة القاطنين هو الاسم المعروض للقاطن في سجلّ القاطنين؛ وفي النشر بالوضع الفردي هو الإعداد `TENANT_DISPLAY_NAME`، ويرجع عند غيابه إلى المضيف المأخوذ من `APP_BASE_URL`. والنشر الذي لا مضيف صالحاً له يُرجِع `422 BASE_URL_INVALID`.

تسجيل البيئة

لا يحتاج الاتصال إلى **حساب في المتجر** ولا إلى تسليم بين موقعين. يسجّل `POST /v1/marketplace/connection` *البيئة*، تحت القاطن: فيرسل معرّف البيئة، والاسم المعروض للقاطن واسم الشركة، وعنوانَي الحزم والويب للبيئة، وإصدار المنصة، وإصدارات مخطط الحزم التي تدعمها. ويجيب المتجر بمفتاح اتصال يُخزَّن مشفّراً، وبمفاتيح التوقيع المستخدمة للتحقق من التنزيلات. والربط متساوي الأثر (يُرجَع الاتصال الحيّ كما هو)، ولا يتصل شيء تلقائياً.

- **الربط لا يُنشئ ناشراً.** التسجيل الجديد **للتثبيت فقط**: فيكون الاتصال حيّاً بلا ناشر ولا عنوان متجر، ويعمل المتجر للتثبيت. ولا تستعيد ناشرها عند الربط إلا البيئة التي كانت ناشراً لدى المتجر من قبل. أما النشر فخطوة مستقلة – إعداد النشر.
- `GET /v1/marketplace/connection` يُرجِع الاتصال المخزَّن – `status` بالقيمة `active` أو `unreachable` أو `revoked`، والناشر، وعنوان المتجر، والنطاقات، و`registrationConflict`، و`connectedAt`، و`namespaceLocked` (لا يمكن تغيير عنوان المتجر بعد اعتماد أي شيء)، و`publishingRevoked` (`null`، أو سبب المتجر وتاريخه ما دام قد أوقف النشر) – ومعه، ما دام لا يوجد اتصال حيّ، معاينة لما سيشاركه الاتصال. ويُرجِع أيضاً **اقتراحاً للناشر** ما دامت البيئة بلا ناشر: آخر اسم ناشر بعد الإلغاء، وإلا فاسم البيئة، والجزء الأول من المضيف عنواناً للمتجر حين يكون عنواناً صالحاً.
- **حين تفشل المعاينة** (مثلاً لأن مضيف القاطن لا يشكّل عنوان https بسيطاً)، تحمل الاستجابة `registrationError: {code}` بدل إخفاء الاتصال، وتظل البطاقة تعرض **الاتصال**، الذي يبلّغ عن الخطأ كاملاً.
- **مسجّلة سابقاً.** إذا أبلغ المتجر أن البيئة مسجّلة من قبل، يُعلَّم الاتصال بـ `registrationConflict` ولا يُعاد تلقائياً.
- **حدود المعدّل.** تظهر حدود معدّل الطلبات لدى المتجر بالرمز `429 RATE_LIMITED` مع ترويسة `Retry-After`.

الربط وإعداد النشر

الربط وإعداد النشر **خطوتان منفصلتان**:

1. **الربط** – `POST /v1/marketplace/connection`. يحتاج إلى `install` أو `publish`، فكل من يحق له التثبيت يستطيع الربط، وكل مسؤول يستطيعه. والتثبيت من المتجر يربط البيئة أولاً حين لا تكون متصلة بعد (انظر التثبيت من المتجر يربط أولاً).
2. **إعداد النشر** – `POST /v1/marketplace/publisher` مع `{publisherName, namespace?}`، على الاتصال الحيّ. يحتاج إلى مسؤول نظام أو حامل `publish`، وإلى استحقاق الباقة. تختار البيئة **اسم الناشر** (من 1 إلى 60 حرفاً، بلا أحرف تحكّم أو تنسيق) و**عنوان المتجر** اختيارياً؛ وقد يمنح المتجر عنواناً غير المقترح. والاسم غير الصالح يُرجِع `422 PUBLISHER_NAME_INVALID` (وكذلك حين يرفض المتجر الاسم، مع رسالته)، والعنوان غير الصالح `422 NAMESPACE_INVALID`، والاتصال الذي لا يملك حقوق النشر `403 PUBLISH_NOT_ALLOWED`. والاتصال الذي له ناشر من قبل يُرجَع كما هو.

وبعد الإعداد **لا يغيّر ما اختير إلا مسؤولو النظام**:

المسار	الأثر
`PUT /v1/marketplace/publisher`	`{name}`. يعيد تسمية الناشر؛ ولا يمسّ عنوان المتجر.
`PUT /v1/marketplace/namespace`	`{name}`. يغيّر عنوان المتجر ما دام المتجر يسمح بذلك (`namespaceLocked` بالقيمة false). والعناوين من 1 إلى 50 حرفاً لاتينياً صغيراً أو رقماً أو شَرطة، تبدأ بحرف أو رقم.

المسار	الأثر
`GET /v1/marketplace/namespaces/{name}`	يفحص عنواناً: `available`، مع `TAKEN` أو `INVALID` أو `LOCKED` (مسؤول أو `publish`).
`GET /v1/marketplace/terms` و`POST /v1/marketplace/terms`	شروط الناشر وقبولها (`{version}`، ويجب أن يكون الإصدار الحالي لدى المتجر). القراءة بـ `publish` أو `review`؛ والقبول بـ `review`.

الإلغاء والنبضة والإبلاغ

- **الإلغاء.** حين يُلغي المتجر الاتصال تُنهيه البيئة محلياً ويُرجِع الاستدعاء `409 MARKETPLACE_NOT_CONNECTED`، ليعرض المتجر خيار الاتصال من جديد. ويُبلَّغ الناشرون والمراجعون. والربط من جديد يسجّل البيئة مرة أخرى تحت **آخر اسم ناشر** ويبدأ موجز الأحداث من جديد؛ أما التطبيقات المنشورة تحت الناشر القديم فتبقى معه، فيبلّغ المتجر حينها عن مالك آخر، ويُرفض إرسال تلك الحزم.
- **النبضة.** ترسل مهمة يومية إصدار المنصة وإصدارات المخطط المدعومة.
- **الإبلاغ.** تُبلَّغ عمليات التثبيت والإزالة والاشتقاق إلى المتجر في الخلفية.

إيقاف المتجر للنشر `Since 1.10`

يستطيع فريق المتجر إيقاف ناشر واحد عن النشر **دون إنهاء الاتصال**. ويصل ذلك على خط 1.10 بعد 1.10.1 (ويُشحن من **1.10.2**). ويبقى تثبيت التطبيقات وتحديثها متاحاً؛ ولا يتوقف إلا النشر.

- **مصدره.** قد يذكر وصف المتجر للاتصال أن النشر موقوف، مع سبب وتاريخ. وكل تحديث للاتصال يخزّن ذلك في `publishingRevoked` بالشكل `{reason, revokedAt}` (ويُخزَّن السبب أو التاريخ الغائب نصاً فارغاً)، أو `null` حين لا يكون النشر موقوفاً. والمتجر الأقدم من هذه الميزة لا يرسله أبداً.
- **أين يظهر.** يُرجِع `GET /v1/marketplace/connection` دائماً `connection.publishingRevoked`. ويُرجِع `GET /v1/marketplace/me` القيمة `publishingRevoked` من آخر تحديث مخزَّن (`null` لمستدعٍ لا يملك أياً من `install` و`publish` و`review`)؛ واستدعاء `me` على اتصال قديم يبدأ تحديثاً في الخلفية (مرة في الدقيقة على الأكثر)، فتعلم صفحة الحزمة بالإيقاف دون أن يفتح أحد بطاقة الاتصال.
- **يُمسح** حين يُلغى الاتصال وحين يبدأ تسجيل جديد، فلا يُظهر إعادة الربط إيقافاً قديماً أبداً.
- **لا تفرضه المنصّة.** لا ترفض خدمة التحزيم إرسالاً أو موافقة أو إعادة تسمية بسببه: فالمتجر يرفض كل إرسال ما دام النشر موقوفاً، ويظهر رفضه بالرمز المعتاد `502 MARKETPLACE_REFUSED`. ولا يعلن الإيقافَ حدثٌ ولا إشعار؛ وتراه صفحة الحزمة المفتوحة حين تعيد التحميل (وهي تعيده مع كل تغيير في الحزم).
- **في تطبيق الويب.** تعرض صفحة الحزمة تحذيراً بتاريخ الإيقاف، وسبب المتجر كما كُتب، و«لا يمكن إرسال إصدارات جديدة ولا تغييرات صفحة المتجر حتى يعيد المتجر النشر. يبقى تثبيت التطبيقات وتحديثها متاحًا.» ويُعطَّل *إرسال للمراجعة*. وتعرض بطاقة الاتصال وسم *النشر موقوف* (وسببه في التلميح) والإشعار نفسه، وتتوقف عن عرض إعادة التسمية وتحرير عنوان المتجر.

الأحداث وحالة المتجر

لا يستدعي المتجر البيئة أبداً، فالبيئة هي التي تقرأ منه. وتُجدوَل أربع مهام خلفية لكل قاطن حين ينجح الربط، ومن جديد عند بدء التشغيل لكل قاطن له اتصال حيّ:

المهمة	الدورة	ما تفعله
النبضة	24 ساعة	تبلّغ عن إصدارات المنصة والمخطط.

المهمة	الدورة	ما تفعله
قراءة الأحداث	60 ثانية	تقرأ موجز أحداث المتجر وتطبّقه. ولا تعمل إلا بعد أن يصير للاتصال ناشر.
استطلاع القرارات	15 دقيقة	شبكة الأمان خلف الموجز: تعيد قراءة الطلبات المرسلة وحالات المتجر.
فحص التحديثات	6 ساعات	تحدّث، لكل تثبيت في المساحة الأساسية، هل أُدرج إصدار أحدث.

موجز الأحداث

الموجز مؤشّر (cursor) على أحداث المتجر الخاصة بهذا الناشر. ويُحفظ المؤشّر وصحة الموجز على الاتصال؛ ولا يعمل إلا تشغيل واحد لكل قاطن في وقت واحد بفضل عقد إيجار (lease)، ويُحفظ المؤشّر بعد كل خطوة، فتعيد الخطوةَ الفاشلةَ المحاولةُ التالية (وتُتخطّى بعد عشر إخفاقات متتالية). وأول قراءة، أو العودة بعد تعذّر الموجز، أو صفحة *resync*، تبدأ من رأس الموجز وتُجري مروراً كاملاً للّحاق بدل إعادة تشغيل الفجوة. وما دام الموجز قيد الاستخدام وحديثاً (قُرئ خلال آخر 5 دقائق)، لا يفحص استطلاع القرارات إلا حالات المتجر المستحقة ويُجري مروراً كاملاً كل 6 ساعات؛ ومن دون موجز حديث يستطلع ما يصل إلى 50 طلباً مرسلاً في كل مرة، كما كان من قبل.

الحدث	الأثر
بُتّ في إرسال أو في مراجعة لصفحة المتجر	يصير الطلب المطابق ذو الحالة `submitted` إما `approved` أو `rejected`. ويُبلَّغ الجمهور (مرة واحدة للمجموعة).
أُزيل التطبيق من المتجر	تُسجَّل حالة المتجر مع **سبب** المتجر ويُبلَّغ «أُزيل من المتجر». ويبقى السبب على الحزمة ما دامت مُزالة.
أُعيد التطبيق إلى المتجر	تُسجَّل الحالة الجديدة ويُبلَّغ «عاد إلى المتجر»، مع ما بقي ناقصاً.
تغيّر عنوان المتجر أو أُضيف	يُحدَّث الاتصال ويُبلَّغ الناشرون.
انتقلت تطبيقات إلى هذا الناشر	تُزال علامة «يملكه ناشر آخر» عن تلك الحزم ويُبلَّغ الناشرون.
انتقلت تطبيقات إلى ناشر آخر	تُعلَّم تلك الحزم بأن ناشراً آخر يملكها، وتُغلق طلباتها المرسلة (`MARKETPLACE_APPLICATION_MOVED`)، ويُبلَّغ الناشرون.
أرسلت بيئة عميل **بلاغاً عن مشكلة** `Since 1.10` (من 1.10.1)	يُبلَّغ الناشرون والمراجعون داخل التطبيق وبالبريد (راجع ما يتلقّاه الناشر). ويُتخطّى البلاغ الذي أسقط المتجر تفاصيله؛ وحين يتعذّر إيصال أي شيء تفشل الخطوة، فتعيدها الجولة التالية دون تكرار.

حالة المتجر على الحزمة

تحتفظ كل حزمة بآخر حالة متجر أبلغ عنها المتجر، وتُرجَع بوصفها `store` في كل ملخّص حزمة (فارغة إلى أن تُقرأ أول مرة): `live` أو `maintenance` أو `unpublished` أو `delisted` أو `not_live`. وفي الحالة `not_live` تسرد الخطوات التي ما زالت ناقصة – الإصدار، أو صفحة المتجر، أو كلاهما – وكلٌّ منها إما ناقص أو قيد المراجعة؛ وتحمل الإزالة سببها. وتكتب هذه الحالةَ فحوصُ المتجر في استطلاع القرارات، والقراراتُ، والأحداثُ أعلاه، وكذلك إلغاءُ النشر أو إعادته فور حدوثهما.

ولا يُعلَن إلا التغيّر الحقيقي، وليس كل تغيّر: فلا يُعلَن النشر إلا حين رأت البيئة الحالة السابقة غير المنشورة؛ والإزالة أو إلغاء النشر الذي يُرى أول مرة صامت، وإلغاء النشر الذي قامت به هذه البيئة نفسها لا يُعلَن. والخطوة التي صارت ناقصة للتو تُعلن «لم يُنشر في المتجر بعد» مع ما يجب إرساله.

تطبيق يملكه ناشر آخر

تسأل البيئة المتجر عن مالك كل معرّف حزمة. وحين يملكه **ناشر آخر** – مثلاً بعد أن أعادت هذه البيئة الربط تحت ناشر جديد، أو بعد نقل التطبيق – تُعلَّم الحزمة، و**يُرفض كل إرسال** بالرمز `409 PACKAGE_OWNED`: إعداد إصدار أو مجموعة أو صفحة متجر، ثم مرة أخرى قبيل الرفع عند الموافقة. وحين لا يستطيع المتجر الإجابة تقرّر العلامة المخزَّنة. وتُزال العلامة حين يعود التطبيق، أو حين يجيب المتجر بأن هذا الناشر (أو لا أحد) يملكه.

التثبيت

للتثبيت طريقان: من **المتجر**، أو باستيراد **ملف إصدار منشور** (ملف ZIP يحمل بيانه كتلة `package`) عبر معالج الاستيراد المعتاد. وينتهي كلاهما في وضع التثبيت نفسه وسجلّ التثبيت نفسه.

مسارات المتجر

كلها تحتاج إلى المساحة الأساسية وإذن `install` (و`/topics` يقبل `publish` أيضاً).

يمكن تصفّح المتجر دون اتصال. حين لا تكون البيئة متصلة – لم تتصل قط، أو أُلغي اتصالها – يُقرأ الفهرس والناشرون وصفحات التطبيقات دون هوية من الفهرس العام للمتجر، مع إرسال إصدار المنصة وإصدارات المخطط المدعومة في كل قراءة. ولا يحتاج إلى اتصال حيّ إلا التثبيت.

المسار	السلوك
`GET /v1/marketplace/catalog?q=&cursor=&limit=&publisher=&topic=&sort=&lang=`	صفحة من عناصر المتجر (24 افتراضياً، و50 على الأكثر). ويرشّح `publisher` حسب عنوان المتجر، و`topic` حسب موضوع واحد؛ وقيمة `sort` هي `name` (الافتراضية) أو `newest` أو `updated` أو `installs`، و`lang` (`en` أو `ar`) هي اللغة التي يرتّب بها الترتيب حسب الاسم. ويبيّن كل عنصر هل هو **مثبّت** هنا (التثبيت، والإصدار، والحالة، والمصدر، و`updateAvailable`)، وهل هو **محجوب** (المنصة أقدم من المطلوب، أو الحزمة غير متوافقة)، و**عدد تثبيتاته** حين يحصيه المتجر – أي عدد البيئات الحيّة التي تثبّته الآن. والمعامل غير الصالح يُرجِع `400 STORE_REQUEST_INVALID`، والمؤشّر القديم `400 STORE_CURSOR_INVALID`.
`GET /v1/marketplace/publishers`	ناشرو المتجر – عنوان المتجر والاسم وعلامة التوثيق وعدد التطبيقات. وتكون فارغة ما دام المتجر معطّلاً أو متعذّر الوصول.
`GET /v1/marketplace/topics?q=&limit=`	المواضيع المستخدمة في صفحات المتجر المنشورة، الأكثر استخداماً أولاً، بوصفها اقتراحات (`q` بادئة).
`GET /v1/marketplace/catalog/{publisher}/{slug}`	صفحة التطبيق: حقول صفحة المتجر ومواضيعها، وملف README، وأحدث إصدار (الحد الأدنى للمنصة، والمحتويات، وأنواع الموارد، والتبعيات مع الإصدار المثبّت محلياً، والتبعيات الخارجية)، وقائمة الإصدارات. و`404 STORE_APP_NOT_FOUND` حين لا يكون التطبيق في المتجر.
`POST /v1/marketplace/catalog/{publisher}/{slug}/install-check`	الجسم `{version}`. هل يُثبَّت هذا الإصدار الآن؟ انظر فحص التثبيت من المتجر.
`POST /v1/marketplace/catalog/{publisher}/{slug}/install`	الجسم `{version}`. **`202 {installationId}`** حين يبدأ التثبيت؛ و** `200 {sessionId}` ** حين يلزم أن يتخذ شخص قرارات في صفحة الاستيراد. ويحتاج إلى اتصال حيّ (`409 MARKETPLACE_NOT_CONNECTED`).
`GET /v1/marketplace/packages/{packageId}/versions/{version}/readme/images/{name}`	صورة من README إصدار في المتجر، عبر وسيط.

والتثبيت من المتجر متساوي الأثر لكل مستخدم وحزمة وإصدار مدة 24 ساعة: فالتكرار يُرجِع التثبيت الموجود، والمحاولة المتزامنة تُرجِع `409 IDEMPOTENCY_KEY_IN_PROGRESS`.

فحص التثبيت من المتجر

يجيب `install-check` عن سؤال «هل يُثبَّت هذا الإصدار الآن؟» دون تثبيت أي شيء: يجلب الحزمة ويتحقق منها تماماً كما يفعل التثبيت، ويهيّئها، ويطبّق الإعدادات الافتراضية الآمنة، ثم يتخلّص من الجلسة دائماً. ويبيّن الجواب:

الحقل	معناه
`installable` و `refusal`	هل سيُثبَّت، أو أول رفض قاطع. والتطبيق **المثبّت هنا من قبل** يُرفض بالرمز `INSTALL_ALREADY_INSTALLED` – ويعرض المتجر التحديث بدلاً منه.
`collisions`	العناصر التي **تحجب** التثبيت: كل اسم متعارض مع نوع مورده، وهل يتعارض في هذه المساحة أو في مكان آخر من البيئة (فاسم كيان البيانات يجب أن يكون فريداً على مستوى البيئة).
`choices`	القرارات غير المحسومة التي سيتعيّن على المثبّت اتخاذها.
`publishedHere`	هذه البيئة بنت الحزمة ونشرتها، فمواردها موجودة هنا أصلاً.
`forkedHere`	**اشتقّت** هذه المساحة تثبيتاً سابقاً للتطبيق؛ فصارت عناصره ملكاً للمساحة، وتحتفظ بأسمائها، وتحجب التثبيت الجديد إلى أن تُعاد تسميتها أو تُحذف.

ولا يستطيع مستخدم بلا صلاحيات إدارية إكمال تثبيت فيه قرارات غير محسومة، فتعرض صفحة التطبيق القرارات بدل تفعيل زر **تثبيت**.

التثبيت من المتجر يربط أولاً

في صفحة التطبيق يجري فحص التثبيت عند فتح الصفحة. وعند الضغط على **تثبيت**، وإن لم تكن البيئة متصلة بعد، تطلب الصفحة من المستخدم التأكيد في مربع حوار *الاتصال بالمتجر* الذي يبيّن ما يشاركه الربط؛ ثم تربط البيئة وتُجري التثبيت. والربط بهذه الطريقة لا يُنشئ ناشراً.

ما يحدث حين تضغط «تثبيت»

تنزّل المنصة الإصدار المدرَج (200 MiB على الأكثر)، وتتحقق من توقيعه و `sha256` الخاص به وتجزئة بيانه مقابل عنصر الفهرس الموقَّع (`422 MARKETPLACE_SIGNATURE_INVALID` و `MARKETPLACE_HASH_MISMATCH` و `MARKETPLACE_PACKAGE_TOO_LARGE`)، ثم تهيّئه جلسة استيراد في وضع التثبيت بإعدادات افتراضية آمنة – إذ تُقبل اقتراحات المطابقة التامة للمراجع القيمية.

وإذا بقيت قرارات – تبعيات خارجية، أو أصحاب صلاحيات، أو حزم تعتمد عليها لا تحسمها الإعدادات الافتراضية – تُسلَّم الجلسة إلى **مسؤول نظام** لإكمالها في صفحة الاستيراد؛ ويُرفض أي مثبّت آخر بالرمز `409 UNRESOLVED_CHOICES` الذي يسرد تلك القرارات. أما الرفض القاطع (حزمة خاطئة، أو منصة أقدم من المطلوب، أو تعارض أسماء، أو تبعية مفقودة أو خارج النطاق) فيُرفع قبل كتابة أي شيء. ويُسجَّل ناشر المتجر على التثبيت وعلى سجلّ الاستيراد.

وضع التثبيت

يصير الاستيراد تثبيتاً حين يحمل الحزمة المتحقَّق منها كتلة `manifest.package` . ويضيف وضع التثبيت هذه الفحوص إلى التحقق:

الرمز	متى يُرفع
`INSTALL_MANIFEST_INVALID`	كتلة `package` مشوّهة.
`INSTALL_PLATFORM_TOO_OLD`	قيمة `minPlatformVersion` للحزمة أعلى من هذه المنصة.
`INSTALL_DEPENDENCY_MISSING` و `INSTALL_DEPENDENCY_OUT_OF_RANGE`	حزمة تعتمد عليها غير مثبّتة في هذه المساحة، أو مثبّتة بإصدار خارج النطاق المعلن (وتُحتسب الإصدارات التجريبية).
`INSTALL_REFERENCE_UNRESOLVED`	مرجع بين حزمتين لا يُحَلّ هنا.
`INSTALL_ALREADY_INSTALLED`	الحزمة مثبّتة في هذه المساحة من قبل – حدّثها في مكانها بدلاً من ذلك.
`INSTALL_NAME_COLLISION`	اسم مورد موجود هنا، أو يقع في مساحة أخرى من البيئة.

أما `INSTALL_VERSION_DEPRECATED` والتبعيات الخارجية غير المعلنة فتحذيرات لا أخطاء. والتثبيت **الفاشل** للإصدار *نفسه* تتسلّمه جلسة جديدة، وهذا ما يجعل إعادة التشغيل آمنة.

وأثناء التثبيت تُنشئ كل مطابقة موارد جديدة؛ ويُرفض مسبقاً كلٌّ من `TARGET_ARCHIVED` و`TARGET_INSTALLED`، وكذلك المرجع القيمي المتروك بلا إجراء. ويُقيَّد كل مورد مكتوب في دفتر التثبيت، ويُحفظ له **صف قفل**. وتوقف التثبيتَ كتابةٌ في الدفتر تظل تفشل، ويحتفظ التثبيت الفاشل بصفوف القفل، فيبقى ما أنشأه محمياً.

حالات التثبيت هي `installing` و`active` و`updating` و`failed` و`forked` و`uninstalling` و`uninstalled`.

المسار (لكل مساحة، لمسؤول المساحة)	الغرض
`GET /v1/installations?status=&skip=&limit=`	القائمة (50 افتراضياً، و100 على الأكثر)؛ تقبل `status` قائمة مفصولة بفواصل، والقيمة المجهولة `400` `INVALID_STATUS_FILTER`.
`GET /v1/installations/{id}`	التثبيت بموارده و**سلامتها** (`ok` أو `changed` أو `missing` مقارنةً بالتجزئة المشحونة)، و`platformAdjusted` (طبّع الخادم الهدف المورد عند كتابته)، والتبعيات، والمعتمِدين عليه، وملخّص المحتويات.
`GET /v1/installations/{id}/status`	استعلام خفيف: `{id, status, lastError?}`.
`POST / GET /v1/installations/{id}/problem-report`	`Since 1.10` (من 1.10.1). معاينة بلاغ عن تثبيت فاشل من المتجر أو تحديث متوقف وإرساله إلى ناشر التطبيق – راجع الإبلاغ عن تثبيت أو تحديث فاشل.
`POST /v1/installations/origins`	`{targetIds}` (من 1 إلى 500)؛ أي تثبيت يملك كل هدف، وهل هو `readOnly`.

الموارد المثبّتة للقراءة فقط. يرفض data-service وreaction-service تحديث مورد يقفله تثبيت أو حذفه بالرمز `403` `PACKAGE_RESOURCE_READ_ONLY` («fork it to edit») – لكل مستدعٍ، بمن فيهم مسؤولو النظام؛ ولا تُستثنى إلا كتابات packaging-service نفسها. ويدوم القفل إلى أن يُشتقّ التثبيت. والاستيراد العادي الذي سيكتب فوق مورد كهذا يُمنع بالرمز `TARGET_INSTALLED`. ويرفض التصدير العادي الموارد المثبّتة كذلك (`403 EXPORT_INSTALLED_RESOURCE`)، ولا يمكن نشر وحدة مثبّتة ضمن حزمتك.

تحديث التطبيق المثبّت في مكانه

يُحدَّث تطبيق المتجر المثبّت **في مكانه** إلى إصدار أحدث مدرَج – دون إزالة ثم إعادة تثبيت. وتحتاج مسارات التحديث إلى مسؤول البيئة في المساحة الأساسية مع إذن `update` (مجموعة المثبّتين):

المسار	الأثر
`POST /v1/installations/{id}/update-sessions`	`{version}`. يهيّئ الإصدار الأحدث جلسةَ تحديث: `201 {sessionId}`، أو `200 {sessionId, choices}` حين تبقى قرارات. ويُرفض بالرمز `409 UPDATE_NOT_NEWER` أو `409 INSTALLATION_FORKED` أو `409 UPDATE_IN_PROGRESS`.
`GET /v1/installations/{id}/update-sessions/{sessionId}`	معاينة التحديث، ومعها ملاحظات الإصدار من الفهرس بين الإصدار المثبّت والإصدار الهدف.
`POST /v1/installations/{id}/update`	`{sessionId}`. `202`: يُجري التحديث (متساوي الأثر لكل مستخدم وتثبيت وجلسة).
`POST /v1/installations/{id}/update/retry`	`202`: يعيد محاولة تحديث فاشل.
`POST /v1/installations/updates/check`	يحدّث «يتوفر تحديث» الآن (مرة في الدقيقة على الأكثر؛ وتعمل مهمة الساعات الست على أي حال).

يقبل التحديثَ التثبيتُ ذو الحالة `active` أو `failed` – والتثبيت الفاشل يُكمله إصدار أحدث مُصلَح. وتأتي `installed.updateAvailable` في المتجر من الفحص نفسه. أما كيفية تخطيط التحديث وكيفية حمل تغييرات العميل نفسه

عبره فتنتمي إلى آليات التحديث في الحزم والتوزيع. وفي تطبيق الويب تُفتح معاينة التحديث عند `/admin/packages/installed/installations/{id}/update`.

الاشتقاق والإزالة

- **الاشتقاق** – `POST /v1/installations/{id}/fork` مع `{confirm: true}` – يفصل تثبيتاً *نشطاً* في مكانه. فتصبح موارده قابلة للتحرير وتحتفظ بمصدرها، لكنها لم تعد تتبع الحزمة، ولا تتلقّى تحديثات، ولا يمكن تصديرها أو نشرها أبداً. ويُرفض الاشتقاق ما دام تثبيت آخر يعتمد على هذا التثبيت (`409 FORK_BLOCKED_BY_DEPENDENTS`)، وما لم يكن التثبيت نشطاً (`409 INSTALLATION_STATUS_CONFLICT`).
- **معاينة الإزالة** – `GET /v1/installations/{id}/uninstall-preview` (مسؤول البيئة) – تَعُدّ ما سيُحذف لكل نوع، والمُدخلات لكل مخزن، والحزم التي تعتمد على هذه الحزمة، وموارد المستدعي نفسه التي ستتعطّل.
- **الإزالة** – `POST /v1/installations/{id}/uninstall` مع `{confirm: true}` (مسؤول البيئة) – مسموحة من `active` أو `failed` أو `forked` أو `uninstalling`. تحذف مهمة خلفية موارد التثبيت ومُدخلاتها بترتيب التبعيات وتنتهي بالحالة `uninstalled`. وتُستبقى الوحدة التي ما زالت تضم كيانات **أنشأها مستخدموك**، وتُرفض الإزالة ما دامت عناصر من خارج الحزمة تشير إلى وحداتها. وإذا فشلت المهمة بقيت الحالة `uninstalling` مع `lastError.code = UNINSTALL_FAILED`، واستدعاء الإزالة مرة أخرى يستأنفها.

ويتعامل تطبيق الويب مع الإزالة كما يتعامل مع حذف وحدة: يسرد مربع الحوار الروابط القادمة من **خارج** الحزمة التي يجب إزالتها أولاً – ويجيب dependency-service عن ذلك بالمسار `GET /v1/external-dependents/{type}/{id}` – ويُبقي زر **الإزالة** معطّلاً إلى أن يُفصل كل رابط ويُحدَّد مربع التأكيد.

الإبلاغ عن تثبيت أو تحديث فاشل إلى ناشره `Since 1.10`

يستطيع المسؤول الذي فشل تثبيته من المتجر، أو توقّف تحديثه من المتجر، أن يرسل إلى ناشر التطبيق **بلاغاً عن مشكلة** من صفحة التثبيت. ويصل ذلك من **1.10.1**.

المسار	الأثر
`GET /v1/installations/{id}/problem-report`	المعاينة: ما سيُرسَل بالضبط، ليراجعه المسؤول أولاً.
`POST /v1/installations/{id}/problem-report`	يرسله ويُجيب بـ `{at, by, errorAt}`، ويُخزِّن ذلك أيضاً على التثبيت باسم `lastProblemReport`.

يحتاج المساران إلى **مسؤول البيئة** في المساحة الأساسية وإلى **اتصال حيّ بالمتجر** (وإلا `409 MARKETPLACE_NOT_CONNECTED`). ولا يحكمهما `publish` ولا استحقاق النشر في الباقة: فالإبلاغ عن مشكلة جزء من التثبيت.

ما يمكن الإبلاغ عنه. تثبيت من المتجر فقط (`source: marketplace`) لم يكن آخر خطأ فيه إلغاءً (رمزاً ينتهي بـ `_CANCELLED`)، وهو إما `failed` (مشكلة تثبيت) أو `updating` مع تحديث معلّق (تحديث غير نشط). وما عدا ذلك يُجيب بـ `409 PROBLEM_REPORT_NOT_AVAILABLE`.

المعاينة تحمل معرّف الحزمة وإصدارها (وفي التحديث: الإصدار الهدف والإصدار المثبّت)، والخطأ (الرمز والرسالة والوقت)، و**الموارد التي فشلت** في التشغيل – النوع ومعرّف المصدر والاسم والرسالة، مقروءةً من جلسة الاستيراد لذلك التشغيل – مع عدد ما لم يتّسع له البلاغ، و `detailsExpired: true` حين لا تعود تلك الجلسة موجودة (فلا يُبلَّغ حينها إلا الخطأ العام)، والمستلِم (ناشر التطبيق كما يعرفه المتجر الآن)، واسم المستدعي وبريده معاينةً لبيانات التواصل (`null` إن لم يكن له بريد)، و `reportKey` جديداً.

الحدود (يُقصّ النص عند حدّ حرف): رسالة الخطأ 1000 حرف، و**50** مورداً، ورسالة المورد 500، واسم المورد 300، والملاحظة **2000**.

الإرسال. الجسم `{note, shareContact, errorAt, recipientPublisherId, reportKey}`، والخمسة كلها مطلوبة (ويجوز أن تكون `note` بالقيمة `null`). ولا يثق الخادم بنسخة العميل أبداً: يعيد قراءة التثبيت ويعيد بناء البلاغ.

1. قيمة `errorAt` التي لم تعد آخر خطأ في التثبيت تُجاب بـ **`409 REPORT_STALE`** – تغيّر الخطأ منذ المعاينة، فعاين مرة أخرى.
2. `shareContact: true` لمستدعٍ بلا بريد يُجاب بـ **`400 CONTACT_UNAVAILABLE`**.
3. يذهب البلاغ – مع الملاحظة (تُرسَل الملاحظة الفارغة بالقيمة `null`) وبيانات التواصل حين تُشارَك فقط – إلى المتجر، و`reportKey` مفتاح عدم التكرار.
4. عند النجاح يُخزَّن `lastProblemReport` دون المساس بـ `updatedAt` للتثبيت. ولا شيء يمنع بلاغاً ثانياً؛ ويعدّ تطبيق الويب البلاغ قائماً ما دامت قيمة `errorAt` فيه مساوية لآخر خطأ في التثبيت.

رفض المتجر يمرّ بالحالة `409` مع رسالة المتجر: `NOT_INSTALLED` (لا سجلّ لدى المتجر بأن هذه البيئة نزّلت التطبيق)، و`RECIPIENT_CHANGED` (للتطبيق ناشر جديد – عاين مرة أخرى)، و`PUBLISHER_UNREACHABLE` و`PUBLISHER_CANNOT_RECEIVE` (لا يستطيع الناشر تلقّي البلاغات بعد). ويصير حدّ معدّل المتجر **`429 RATE_LIMITED`** مع `details.retryAfterSeconds` (قيمة المتجر، وإلا 60).

وتحمل ملخّصات التثبيت الآن أيضاً `publisher` (`namespace` و`name?`) و`lastProblemReport`.

ما يتلقّاه الناشر

يصل البلاغ إلى بيئة الناشر عبر موجز أحداثها. فتُبلِغ تلك البيئة كل عضو في `MARKETPLACE_PUBLISHERS` و`MARKETPLACE_REVIEWERS`، داخل التطبيق وبالبريد، مع رابط إلى صفحة الحزمة. والعنوان «بلاغ عن مشكلة: *التطبيق الإصدار*»؛ ويذكر النص التثبيت أو التحديث الفاشل برمزه ورسالته، والموارد التي فشلت (أسطراً كاملة فقط، ثم «…و N أخرى»)، وسطراً يقول إن التفاصيل انتهت صلاحيتها حين يكون ذلك، وملاحظة المُبلِّغ، و«الرد على *الاسم <البريد>*» أو أن المُبلِّغ لم يشارك بيانات التواصل، ومعرّف التثبيت مع إصدار منصة المُبلِّغ. ويُحدّ النص بـ 4000 حرف، وتُنقّى كل قيمة مُبلَّغ عنها. وعلى خلاف إشعارات المتجر الأخرى لا يجعل صفحات الحزم المفتوحة تعيد التحميل.

في تطبيق الويب. في صفحة التثبيت يرى مسؤول البيئة ***إبلاغ الناشر*** حين يكون التثبيت من المتجر وناشره معروفاً، ولم يكن آخر خطأ فيه إلغاءً، وقد فشل أو توقّف تحديثه. ويحمّل مربع الحوار المعاينة، ويسرد الخطأ والموارد التي فشلت، ويعرض ملاحظة اختيارية للناشر ومربع اختيار «اسمح *للناشر* بالرد عليّ على *البريد*» (حين توجد بيانات تواصل فقط)، ثم يرسل. ويقول حدّ المعدّل متى تُعاد المحاولة؛ ويعيد تغيُّرُ المستلِم أو قِدَمُ الخطأ أو غيابُ بيانات التواصل تحميلَ المعاينة. وبعد الإرسال تعرض الصفحة أن البلاغ أُرسل إلى *الناشر* في *التاريخ* بواسطة *الشخص* ما دام البلاغ قائماً.

سجلّات الاستيراد

`GET /v1/imports?skip=&limit=&source=&q=&state=` هو تاريخ الاستيراد في المساحة (50 افتراضياً، و100 على الأكثر). وقيمة `source` هي `marketplace` أو `package-file` أو `import`؛ و`q` للبحث (100 حرف على الأكثر)؛ و`state` ترشّح على الخادم، قبل التقسيم إلى صفحات وفي المجموع:

- `installed` – الاستيرادات العادية، والتثبيتات التي حالة تثبيتها `installing` أو `active` أو `updating` أو `failed` أو `forked` (فالتثبيت الجاري أو الفاشل له موارد في المساحة، فيُعدّ مثبّتاً)؛
- `failed` – التثبيتات الفاشلة فقط؛

- `uninstalled` – التثبيتات التي حالتها `uninstalled` أو `uninstalling`.

وتُرجِع القيم المجهولة `400 INVALID_STATE_FILTER` أو `400 INVALID_SOURCE_FILTER`. ويحمل كل سجلّ حالة التثبيت، وناشر المتجر لتثبيت المتجر، ومن استورده، والأعداد لكل نوع. ويسرد `GET /v1/imports/{id}/resources` ما كتبه الاستيراد، ويبيّن `POST /v1/imports/origins` من أين جاء كل مورد هدف – والأحدث استيراداً هو الغالب – مع البيئة والمساحة والمستخدم المصدِّرين مأخوذة من كتلة `origin` في الحزمة.

في منطقة الإدارة

مكان الحزم تحت `/admin/packages`؛ وتعيد عناوين `/admin/packaging` السابقة التوجيه إليه مع الحفاظ على سلسلة الاستعلام، فتصل الإشارات المرجعية والروابط القديمة إلى مكانها الصحيح. وصفحة الحزم المؤلَّفة هي **البناء** عند `/admin/packages/build`؛ وعنوانها السابق `/admin/packages/mine` (وكل ما تحته، ومنه روابط الإشعارات القديمة) يُعاد توجيهه إليها.

المسار	ما هو	من يراه
`/admin/packages/store` (و `/{publisher}/{slug}/`)	**المتجر**: الفهرس وصفحة التطبيق مع تثبيت أو فتح أو تحديث أو إزالة	المتجر مفعّل؛ وداخل الصفحة: المساحة الأساسية و `install`
`/admin/packages/installed` (و `/import` و `/installations/{id}` و `/installations/{id}/update` و `/{importId}`)	**المثبتة**: تاريخ الاستيراد، ومعالج الاستيراد، وصفحة التثبيت، ومعاينة التحديث	`AdminAccess` على مستوى النظام
`/admin/packages/build` (و `/exports` و `/export/{packageId}` و `/{packageId}/store-page`)	**البناء**: حزمك المؤلَّفة وتصديراتك (تبويبان)، وصفحة التصدير، وصفحة الحزمة (المحتويات والإصدارات)، ومحرّر صفحة متجرها	من يملك Developer على وحدة، أو حامل `publish`؛ وصفحة التصدير تحتاج إلى Developer على وحدة؛ ومحرّر صفحة المتجر يحتاج إلى تفعيل المتجر
`/admin/packages/reviews` (و `/{requestId}`)	**المراجعات**: صندوق المراجعة والطلب	`review`؛ والمتجر مفعّل

وفي القائمة تضم مجموعة **الحزم** أربعة عناصر بهذا الترتيب: **المتجر** (حين يكون المتجر مضبوطاً وللمستخدم `install`)، و**المثبتة**، و**البناء** (وتسميته *البناء*، لا «حزمي»)، و**المراجعات** (حين يكون المتجر مضبوطاً وللمستخدم `review`). ويتبع «البناء» و«المراجعات» مجموعاتِ المتجر لا حقَّ وصول، فيراهما الناشر أو المراجع الذي لا يملك منحة على مستوى النظام. وحين يتعذّر عرض المتجر تبيّن الصفحة السبب بهذا الترتيب: المساحة الأساسية فقط، أو التحميل جارٍ، أو تعذّر الوصول إلى المتجر، أو لا إذن تثبيت (مع رابط إلى صفحة مجموعات المستخدمين). و**لا توجد حالة «غير متصل»**: فالتصفّح لا يحتاج إلى اتصال.

بعض السلوكيات التي يجدر معرفتها حين تبني فوق هذا:

- **تصفّح المتجر.** البحث، ومرشّحا **الناشر** و**الموضوع** السريعان، والترتيب – **الاسم**، **الأحدث**، **المحدَّثة مؤخرًا**، **الأكثر تثبيتًا** – كلها محفوظة في العنوان (`q` و `publisher` و `topic` و `sort`)، فيحملها الرابط. وتبيّن البطاقات عدد البيئات التي ثبّتت التطبيق.
- **صفحة التطبيق** تتبع تخطيط صفحة المتجر نفسه (نظرة عامة مع README، ولمحة سريعة، وما بداخلها، ولقطات الشاشة، والتفاصيل، وسجلّ التغييرات). ويجري فحص التثبيت عند فتحها؛ وتُعرض حالة رفضه وتعارضاته ورسالتا «اشتُقّ هنا» و«نُشر هنا»، مع *تحقّق مرة أخرى*.
- **سير عمل نشر واحد في صفحة الحزمة.** المحتويات ← الإصدار ← صفحة المتجر ← *إرسال للمراجعة*، الذي يرسل الإصدار وصفحة المتجر معاً. وزرّا إلغاء النشر وإعادة النشر في الترويسة. وتنتظر الصفحة إعداد النشر (فغياب الناشر يحجب خطوات المتجر ويشير إلى الإعداد)، وتعرض حالة المتجر وسبب الإزالة، وتعرض إشعار «يملكه ناشر آخر» (دون عرض أي إرسال)، وتعيد التحميل حين تتغيّر الحزمة في مكان آخر، وتعرض **سحب** الطلب الذي لم يعد المتجر يملكه.

ويحفظ محرّر صفحة المتجر أو يلغي من شريطه الخاص قبل الإرسال، ويشرح لماذا قد يختلف اسم المتجر عن اسم الحزمة.

- **إعداد النشر** بطاقة في صفحة «البناء». يُدخل فيها مسؤول أو حامل `publish` اسم الناشر وعنوان متجر اختيارياً (يُفحص أثناء الكتابة)؛ ويستطيع المسؤول لاحقاً إعادة تسمية الناشر أو تغيير العنوان في مكانه. والمستخدم الذي لا يحق له الإعداد يُطلب منه أن يسأل ناشراً أو مسؤولاً. وبعد الإلغاء تقول البطاقة إن التطبيقات الموجودة تبقى مع الناشر القديم، وتعرض الاتصال.
- **الشروط عند الموافقة.** يطلب مربع حوار الموافقة من المراجع قبول شروط الناشر حين لا تكون مقبولة بعد.
- **يمكن للمطوّرين التصدير دون منحة على مستوى النظام.** من يملك Developer على وحدة يصل إلى «البناء» وصفحة التصدير، ولا تعرض شجرة التصدير إلا الجذور التي يستطيع ذلك المستخدم تطويرها – وهي القاعدة نفسها التي يطبّقها الخادم.
- **الموارد المقفلة تفصح عن ذلك.** تُظهر الوحدة أو الكيان القادم من تثبيت إجراءَي **تخصيص** و**عرض التثبيت** في الترويسة. ويعرض «تخصيص» السبل التي تُبقي التثبيت سليماً – أضف حقلك الخاص، أو أخفِ ما لا تستخدمه، أو ابنِ معرضك الخاص، أو اسأل الناشر – ثم الاشتقاق ملاذاً أخيراً.
- **المثبتة** تُرشَّح من شريط الأدوات – قائمتا *الحالة* و*المصدر* بجانب البحث والترتيب (راجع تحسينات المتجر و«المثبتة») – وتبيّن من ثبّت كل حزمة (*ثبّته*) وناشرها لتثبيتات المتجر (*من*). والتطبيق المثبّت من المتجر الذي أُدرج له إصدار أحدث يعرض **تحديث** في صفحة تثبيته وفي صفحته في المتجر؛ وكلاهما يفتح معاينة التحديث. والتثبيت من المتجر الذي فشل أو توقّف تحديثه يعرض **إبلاغ *الناشر*** في صفحة تثبيته `Since 1.10` (من 1.10.1؛ راجع الإبلاغ عن تثبيت أو تحديث فاشل).
- **إنشاء حزمة** معروض لمطوّري الوحدات في ترويسة المتجر وحالته الفارغة (ويفتح «البناء» ونموذج الإنشاء مفتوح)، و**كيف تعمل الحزم** يفتح دليلاً قصيراً في لوحة المساعدة.
- **ملف README قبل الإصدار** `Since 1.10` (من 1.10.2). لأن الإصدار يحتفظ بملف README الذي أُنشئ معه، تقول صفحة الحزمة ذلك قبل أن يوجد الإصدار. فلمن يستطيع التحرير يعرض قسم README تنبيهاً بأن الإصدار التالي سيحتفظ بنسخة من ملف README هذا حين يوجد، وتحذيراً «أضف ملف README قبل إنشاء إصدار» حين لا يوجد؛ ويقول مربع حوار *إنشاء إصدار* «لن يحتوي هذا الإصدار على ملف README» مع زر **إضافة README** يغلق مربع الحوار ويفتح محرّر README. **ولا يظهر تحذير غياب README إلا بعد التأكد من غيابه** – بقراءة أعادت نص Markdown فارغاً أو *غير موجود*، أو بحفظ – فالملف الذي تعذّر تحميله فحسب لا يثير تحذيراً.
- **صور الويب الملصقة تُنسخ إلى الحزمة** `Since dev` . لصق صورة من الويب (`http(s):`) أو صورة `data:` في محرّر README يجعل المتصفح ينزّلها ويخزّنها صورةَ README عادية، فتنتقل مع الحزمة وتُحسب ضمن حدود صور README. ولا يرسل التنزيل بيانات اعتماد ولا مُحيلاً، ويتوقف بعد 15 ثانية، ويحدّد النوع من البايتات (PNG أو JPEG أو GIF أو WebP) لا من النوع الذي يعلنه الخادم. وإذا فشل النسخ – لغياب الوصول عبر المصادر، أو انقطاع الاتصال، أو حالة خطأ، أو نوع غير معروف، أو رفض الرفع – تصير الصورة رابطاً إلى مصدرها ويقول المحرّر إنه تعذّر نسخها («نزّلها ثم استخدم إضافة صورة بدلًا من ذلك»). وصور الويب الموجودة أصلاً في README تُفتح في المحرّر روابطَ كما يراها القرّاء، ويُبقيها الحفظ روابط.
- **إيقاف النشر** `Since 1.10` (من 1.10.2): تعرض صفحة الحزمة وبطاقة الاتصال إيقاف المتجر للنشر وسببه وتتوقفان عن عرض الإرسال – راجع إيقاف المتجر للنشر.

تحسينات المتجر و«المثبتة» `Since 1.10`

هذه التحسينات في تطبيق الويب جزء من إصدار 1.10:

- **ترشيح «المثبتة» من شريط الأدوات بدل التبويبات**: قائمة *الحالة* (المثبتة، والفاشلة، والمُزالة) وقائمة *المصدر* بجانب البحث والترتيب. وتبدأ القائمتان فارغتين فتظهر كل الحالات والمصادر؛ ولا يوجد خيار *الكل*، ويعرض الترتيب حقول ترويسات الأعمدة.
- **مربّع أيقونة واحد لتطبيقات المتجر** – مربّع بإطار تملؤه الصورة، ويعرض الحرف الأول من الاسم حين لا توجد أيقونة أو يتعذّر تحميلها – في قائمة المتجر، وترويسة صفحة التطبيق (الأيقونة والاسم معاً)، ومعاينة المتجر في صفحة «البناء». وتملأ بطاقة المتجر الصفحة مهما قصرت القائمة.
- **شروح في خطوتَي الإعداد والربط**: تقول بطاقة *إعداد النشر* ما ملف الناشر وما الغرض من كل حقل، وتعاين بطاقة تطبيق «من» الاسم المكتوب وعنوان صفحة الناشر؛ ويقول مربع حوار الربط عند التثبيت لماذا يربط التثبيت البيئة وإن الربط لا ينشر شيئاً.
- **إصلاحات أصغر**: مربع بحث مضغوط للمرشّحات السريعة محاذٍ لشريط الأدوات؛ والترتيب المفرد يحتفظ بحقله بعد *إعادة الضبط*؛ وإعادة التحميل التي تجد تطبيق المتجر قد زال تعرض حالة «غير موجود» في المتجر.

رموز الأخطاء

رموز المتجر التي قد يصادفها العميل:

الحالة	الرموز
`400`	`MARKETPLACE_ORIGIN_INVALID` و `STORE_REQUEST_INVALID` و `STORE_CURSOR_INVALID` و `UPDATE_REQUEST_INVALID` و `DESCRIPTION_TOO_LONG` و `CONTACT_UNAVAILABLE` (بلاغ المشكلة، من 1.10.1)
`403`	`MARKETPLACE_PERMISSION_REQUIRED` و `MARKETPLACE_PUBLISH_NOT_IN_PLAN` و `PUBLISH_NOT_ALLOWED` و `PACKAGE_DEVELOPER_REQUIRED` و `EXPORT_INSTALLED_RESOURCE` و `PACKAGE_RESOURCE_READ_ONLY` (reaction-serviceg data-service)
`404`	`STORE_APP_NOT_FOUND` و `STORE_VERSION_NOT_FOUND`
`409`	`MARKETPLACE_NOT_CONNECTED` و `MARKETPLACE_CONNECTION_CHANGED` و `MARKETPLACE_TERMS_REQUIRED` و `MARKETPLACE_PUBLISHER_REQUIRED` و `PACKAGE_OWNED` و `ALREADY_LIVE` و `UNRESOLVED_CHOICES` و `IDEMPOTENCY_KEY_IN_PROGRESS` و `PUBLISH_REQUEST_STATE` و `STORE_SLUG_IN_USE_BY_OPEN_REQUEST` و `LISTING_REQUEST_OPEN` و `FORK_BLOCKED_BY_DEPENDENTS` و `INSTALLATION_STATUS_CONFLICT` و `UPDATE_NOT_NEWER` و `UPDATE_BREAKS_DEPENDENTS` و `UPDATE_IN_PROGRESS` و `UPDATE_SESSION_EXPIRED` و `INSTALLATION_FORKED` : ورموز بلاغ المشكلة `PROBLEM_REPORT_NOT_AVAILABLE` و `REPORT_STALE` ، ومن المتجر `NOT_INSTALLED` و `RECIPIENT_CHANGED` و `PUBLISHER_UNREACHABLE` و `PUBLISHER_CANNOT_RECEIVE` (من 1.10.1)؛ و `STALE_UPDATE_IDENTIFIER` عند حفظ قديم لمسوّدة حزمة أو ملف README (`Since dev`)
`422`	`MARKETPLACE_SIGNATURE_INVALID` و `MARKETPLACE_HASH_MISMATCH` و `MARKETPLACE_PACKAGE_TOO_LARGE` و `MARKETPLACE_ORIGIN_NOT_ALLOWED` و `IDEMPOTENCY_KEY_REUSED` و `PACKAGE_FORMAT_OUTDATED` و `PUBLISHER_NAME_INVALID` و `NAMESPACE_INVALID` و `BASE_URL_INVALID`
`429`	`RATE_LIMITED` (مع `Retry-After` ؛ وفي بلاغ المشكلة مع `details.retryAfterSeconds`)
`502`	`MARKETPLACE_UNREACHABLE` و `MARKETPLACE_REFUSED` (مع `details.upstreamCode`)
`503`	`MARKETPLACE_DISABLED` و `MARKETPLACE_STORAGE_UNAVAILABLE` و `MARKETPLACE_KMS_UNAVAILABLE`

وبعض حالات الرفض تأتي من المتجر نفسه وتُمرَّر برسالته – ومنها `SLUG_TAKEN` و `SLUG_LOCKED` و `NAME_LOCKED` و `NAMESPACE_TAKEN` و `NAMESPACE_LOCKED` و `LISTING_INVALID` و `MEDIA_INVALID` و `REVISION_PENDING` و `NOT_LIVE` و `README_INVALID` و `MANIFEST_INVALID` و `ENVIRONMENT_REGISTERED` .

هذه المسارات موثّقة هنا يدوياً: فخدمتا packaging-service وdependency-serviceg لا تنشران وصف OpenAPI، ولذلك لا تظهران في المرجع المولَّد للخلفية.

الوظائف: شغّل رمازك الخاص

هذه الصفحة موجّهة إلى المطوّرين الذين يحتاجون إلى منطق تعجز الأدوات عديمة الرماز عن التعبير عنه: استدعاء خدمة خارجية، أو إجراء عملية حسابية، أو تنفيذ تحقّق مخصّص. تكتب وظيفة صغيرة بلغة TypeScript وترفعها، فتتولّى المنصة بناءها ونشرها واستدعاءها عند تشغيل أتمتة، أو حفظ مُدخل، أو وصول طلب.

متى تستخدم وظيفة

تحتاج إلى…	استخدم
الاستجابة لحدث بإجراءات مدمجة (إنشاء مُدخل، أو إرسال بريد إلكتروني، أو استدعاء عنوان URL)	تفاعلًا [بناء التطبيقات › التفاعلات والأتمتة] – دون رماز
حقل محسوب، أو قيمة ديناميكية، أو رمز قالب	محرّك التعابير المدمج – دون نشر
رمازك الخاص، أو استدعاءات عبر الشبكة لخدمات خارجية، أو حسابات أكثر تعقيدًا	**وظيفة**

يمكن للتفاعل استدعاء وظيفة حين يحتاج إلى منطق مخصّص، وبذلك يكمّل أحدهما الآخر.

كيف تعمل

- الوظيفة سجلّ في **الإدارة › التطبيقات › الوظائف**، ومصدرها ملف ZIP ترفعه.
- عند حفظها، تبني المنصة المصدر وتنشره بوصفه خدمةً بلا خادم داخل العنقود، فتنتقل **الحالة** من **جاري التوريد** إلى **تم التحقق** (أو **فشل البناء**)، ويُملأ حقل عنوان URL الخاص بها. ولا يمكن استدعاؤها إلا بعد ذلك.
- تستدعيها المنصة عبر HTTP بصيغة CloudEvents، وتقرأ JSON الذي تُعيده.
- تؤدّي إعادة رفع المصدر إلى إعادة بنائها، وتؤدّي أرشفة السجلّ إلى إلغاء نشرها.

تُحدَّد للوظيفة في **النوع** إحدى القيم الأربع التالية، وهي تحدّد الجهة التي تستدعيها والبيانات التي تمرّرها إليها:

النوع	متى تُستدعى	ما تتلقّاه	ما تُعيده
إجراء مخصص للتفاعل	عبر إجراء **تنفيذ وظيفة** في تفاعل	`{ context, parameters }`	أي كائن JSON؛ ويمكن للإجراءات اللاحقة استخدامه
مدقق مستقل	عند حفظ مُدخل أو تعديله أو أرشفته	`{ dataEntry, existingEntry }`	`{ success, errors?: [{ errorCode }] }`
مدقق القيمة	عند حفظ قيمة حقل	`{ value }`	`{ success, errors?: [{ errorCode }] }`
معترضة	قبل معالجة طلب وارد مطابق	طريقة الطلب (method) ومساره واستعلامه وترويساته وجسمه	طلبًا أُعيدت صياغته، أو استجابة خاصة بها

اكتب وظيفة

الوظيفة مشروع TypeScript من مشاريع Knative Functions. وتكون بنية مجلّدها على النحو الآتي:

```
my-function/
├── func.yaml        # required
├── params.yaml      # optional: the parameters a reaction can map
├── package.json
├── tsconfig.json    # compiles src/ to build/
└── src/
    └── index.ts     # exports handle()
```

`func.yaml`

```
specVersion: 0.36.0
name: order-total          # replaced by the platform's deployed name (Since 1.10)
runtime: typescript
build:
  builder: pack
  buildEnvs:
    - name: BP_NODE_RUN_SCRIPTS
      value: build
```

تنشر المنصة وظيفتك باسم تشتقّه من السجلّ: لذا لا تحدّد قيمة `name` هنا الاسم الذي تُنشر به الوظيفة `Since 1.10`. كما تقصر المنصة إتاحة كل وظيفة على العنقود: فلا تحصل أبدًا على عنوان URL عام لها.

`params.yaml`

صرّح بالمُدخلات التي يربطها الباني في إجراء **تنفيذ وظيفة**:

```
version: "1"
parameters:
  - name: amount
  - name: currency
```

تقرأ المنصة الأسماء عند نشر الوظيفة وتعرضها في الإجراء. وتصف هذه الأسماء المُدخلات، لكن المنصة لا تتحقّق منها.

`package.json`

```
{
  "name": "order-total",
  "version": "1.0.0",
  "scripts": {
    "build": "npx -p typescript tsc",
    "start": "faas-js-runtime ./build/index.js"
  },
  "dependencies": {
    "cloudevents": "^9.0.0",
    "faas-js-runtime": "^2.4.0"
  },
  "devDependencies": {
    "typescript": "^5.5.2"
  }
}
```

المعالج: إجراء مخصّص لتفاعل

تتلقّى هذه الوظيفة مبلغًا وعملة يربطهما التفاعل، وتطبّق ضريبة القيمة المضافة، وتُعيد نتيجة يمكن للإجراءات اللاحقة في التفاعل نفسه استخدامها:

src/index.ts

```
import { Context } from 'faas-js-runtime';

interface Input {
  context?: unknown;                      // the event that triggered the reaction
  parameters?: { amount?: string | number; currency?: string };
}

const handle = async (context: Context) => {
  // The platform sends a CloudEvent; depending on its mode the payload is the body
  // itself or the body's `data` property.
  const body = (context.body ?? {}) as { data?: Input } & Input;
  const input: Input = body.data ?? body;

  const amount = Number(input.parameters?.amount);
  if (!Number.isFinite(amount)) {
    return { success: false, message: 'amount must be a number' };
  }

  const vat = Math.round(amount * 0.05 * 100) / 100;
  return {
    success: true,
    message: 'VAT calculated',
    result: { net: amount, vat, gross: amount + vat, currency: input.parameters?.currency ?? 'AED' },
  };
};

export { handle };
```

- **أعِد كائن JSON.** تحلّل المنصة استجابة الوظيفة بوصفها JSON، وتُعامل الاستجابة المرسَلة نصًّا عاديًا على أنها خطأ.
- **حافظ على ثبات البنية.** `Since 1.10` يمكن للإجراءات اللاحقة والتفاعلات المتسلسلة ربط `success` و`message` وأي مفتاح ضمن `result` (مثل `result.gross`)؛ انظر مخرجات «تنفيذ وظيفة» [بناء التطبيقات › التفاعلات والأتمتة]. فإذا غيّرت اسم مفتاح، تعطّلت عمليات الربط تلك.
- **أبلِغ عن الإخفاق في الجسم.** يُسجَّل الإجراء على أنه نُفِّذ كلما أعادت وظيفتك استجابة؛ لذا أعِد `success: false` لإبلاغ الإجراءات اللاحقة بالإخفاق.

المعالج: مدقّق

يُعيد المدقِّق المستقل رموز أخطاء، وتعرض المنصة الرسالة التي أدخلتها لكل رمز في قائمة **الأخطاء** الخاصة بالوظيفة:

src/index.ts

```
import { Context } from 'faas-js-runtime';

const handle = async (context: Context) => {
  const body = (context.body ?? {}) as any;
  const { dataEntry } = body.data ?? body;

  const end = new Date(dataEntry?.value?.end_date);
  const start = new Date(dataEntry?.value?.start_date);
  if (end < start) {
    return { success: false, errors: [{ errorCode: 'END_BEFORE_START' }] };
  }
  return { success: true };
};

export { handle };
```

ويعمل مدقِّق القيمة بالطريقة نفسها، ويتلقّى `{ value }`.

ابنِها وجرّبها محليًا

```
npm install
npm run build
npm start          # serves the function on http://localhost:8080

curl -s -X POST http://localhost:8080 \
  -H 'Content-Type: application/json' \
  -d '{"parameters": {"amount": 200, "currency": "AED"}}'
```

ويستجيب وقت التشغيل أيضًا لفحوص السلامة في `/health/readiness` و`/health/liveness`.

حزّمها

أنشئ ملف ZIP يحتوي في جذره على `func.yaml`، إلى جانب `package.json` و`src/`. ولا تُدرج `node_modules` و`build`؛ فالمنصة تثبّت الاعتماديات وتبني الوظيفة عند نشرها.

```
cd my-function && zip -r ../order-total.zip . -x 'node_modules/*' 'build/*'
```

لا يُنشر إلا ما يُرفع بصيغة ZIP، ويُتجاهَل أي مصدر آخر دون الإبلاغ عن خطأ.

أضِف الوظيفة

1. افتح **الإدارة › التطبيقات › الوظائف**، ثم اختر **أنشئ وظيفة**.
2. أدخل **الاسم**، واختر **النوع** و**نطاق الوظيفة** (**النظام** أو **الوحدة** أو **الكيان**).
3. أبقِ **نوع التوريد** على **داخلي** (الخيار الافتراضي، مع خيار ZIP)، وارفع ملف ZIP في **ملف الوظيفة**.
4. احفظ. يعرض حقل **الحالة** القيمة **جاري التوريد** أثناء البناء.
5. عندما تصبح الحالة **تم التحقق** ويظهر **رابط الوظيفة**، تكون الوظيفة جاهزة. وإذا ظهرت **فشل البناء**، فاقرأ **خطأ التوريد**، وأصلح المصدر، وارفع ملف ZIP جديدًا.

وللمدقِّق، أضِف كل رمز خطأ تُعيده الوظيفة، مع رسالته، تحت **الأخطاء**.

تحتاج إلى مستوى **إدارة** على النظام لإدارة الوظائف.

استدعِ وظيفة من تفاعل

1. تأكّد من أنّ الوظيفة في حالة **تم التحقق**.
2. يفتح الباني تبويب **التفاعلات** في الكيان، ويضيف تفاعلًا، ويختار الحدث.
3. يضيف إجراء **تنفيذ وظيفة**، ويختار الوظيفة، ويربط قيمة بكل معامل من `params.yaml` .

تجد شرحًا للتفاعلات وإجراءاتها في التفاعلات والأتمتة [بناء التطبيقات].

ويمكنك أيضًا تشغيل وظيفة جرى توريدها من رمازك الخاص:

```
curl -s -X POST "https://acme.example.com/api/functions/v1/<function-id>/execute" \
  -H "Authorization: Bearer <access-token>" \
  -H "Content-Type: application/json" \
  -d '{"parameters": {"amount": 200, "currency": "AED"}}'
```

يتطلّب الاستدعاء بيانات اعتماد ، ويُعيد JSON الذي تُعيده وظيفتك، أو `{ "success": false, "error": … }` إذا فشلت الوظيفة أو لم تُعِد JSON. ويُرفض استدعاء الوظيفة المفقودة أو غير النشطة أو غير الموزَّدة بالرمز `404` أو `400` . ولسرد الوظائف المتاحة في نطاق ما، استدعِ `GET /api/functions/v1/{scope}` (`SYSTEM` أو `MODULE` أو `ENTITY` ، مع `?scopeId=` في النطاقين الأخيرين)؛ ولا تُسرد إلا الوظائف الموزَّدة النشطة.

المدقّقات

لا تعمل وظيفة المدقِّق إلا عند ربط الباني لها بكيان:

- يتحقّق **مدقق مستقل** من قيمة المُدخل كاملةً. ويُحدَّد وقت تشغيله – قبل حفظ المُدخل أو تعديله أو أرشفته – عند ربطه. ولا يسري إلا على مُدخلات الكيان الواقع في نطاقه.
- يتحقّق **مدقق القيمة** من قيمة حقل واحد عند حفظ المُدخل.
- يوقف الحفظَ أولُ مدقِّق يُعيد أخطاء. ويُبحث عن كل `errorCode` في قائمة **الأخطاء** الخاصة بالوظيفة لعرض رسالته؛ وفي مدقِّق القيمة يُربط الخطأ بالحقل.

معترِضات الطلبات

تعمل وظيفة **معترضة** قبل معالجة طلب وارد يطابق مسارها وطريقة HTTP الخاصة بها. ويمكنها أن تُعيد طلبًا معدَّلًا – في الجسم أو الترويسات أو الاستعلام – أو أن تستجيب للطلب بنفسها، فلا يصل إلى المنصة أبدًا. وتقتصر المعترِضات على مرحلة ما قبل المعالجة، وهي ليست وسيلة للتحكّم في الوصول: فلا تعتمد عليها لرفض الطلبات.

كيف تُنشر وظيفتك `Since 1.10`

- **وظائف كل قاطن معزولة.** في النشر متعدّد القاطنين، يملك كل قاطن مساحته الخاصة في العنقود، فلا تتعارض أبدًا وظيفتان تحملان الاسم نفسه وتخصّان قاطنين مختلفين.
- **الاسم المنشور يأتي من السجلّ** لا من `func.yaml` : صيغة مختصرة من الاسم المعياري للوظيفة مع جزء من معرّفها، مثل `order-total-e93838fe1c2d` . ويبقى كما هو عند إعادة تسمية الوظيفة.

- **الأرشفة تحذف الوظيفة المنشورة.**
- **يؤمّن المشغّل البناءَ واستدعاءاته الراجعة.** تُضبط طريقة مصادقة المنصة لدى باني الوظائف وطريقة التحقّق من استدعاءات الحالة الراجعة منه ضمن إعدادات النشر – انظر الوظائف في النشر متعدّد القاطنين، ولتفاصيل الآليات، انظر الوظائف – التفاصيل الداخلية.

استدعِ المنصة من وظيفتك

`Since 1.10` تتلقّى كل وظيفة عنوان القاطن الخاص بها في متغيّرات بيئتها:

المتغيّر	القيمة
`OSOS_API_BASE_URL`	واجهة API الخاصة بالقاطن، مثل `https://acme.example.com/api`
`APP_ID`	معرّف القاطن

كوِّن العناوين دائمًا باستخدام `OSOS_API_BASE_URL`؛ ولا تكتب مضيفًا ثابتًا في الرماز أبدًا. وإذا لم يكن للقاطن مضيف عام، تُزوَّد وظيفته دون هذا المتغيّر.

على خط الإصدار 1.10، **لا تملك الوظيفة بيانات اعتماد خاصة بها**، ولا تحفظ المنصة لها **أي أسرار**: فلا شيء في بيئتها يتيح لها استدعاء واجهة API، ولا يتضمّن سجلّ الوظيفة قائمة أسرار. ويتاح كلاهما على خط التطوير، كما هو موضّح أدناه.

رمز API الخاص بها `Since dev`

على خط التطوير، تتلقّى كل وظيفة أيضًا `OSOS_API_TOKEN`، وهو رمز API الخاص **بمستخدم التطبيق التابع لها**. وبذلك تستدعي الوظيفة واجهة API بهويتها الخاصة:

```
const res = await fetch(`${process.env.OSOS_API_BASE_URL}/v2/entities/orders/entries/count`, {
  headers: { 'x-app-token': process.env.OSOS_API_TOKEN ?? '' },
});
const { result } = await res.json();
```

- **يبدأ مستخدم الوظيفة بلا أي وصول.** ويمنحه مسؤول القاطن أذونات الوصول التي يحتاجها تحديدًا، كأي مستخدم تطبيق – انظر استخدام رمز API، وللمسؤول أنشئ رمز API – الخطوة 2 [الإدارة › أنشئ رمز API لعملية تكامل].
- لا تغيّر إعادة رفع ملف ZIP الرمز.
- حين يعيد المسؤول توليد الرمز، يُمرَّر الرمز الجديد مباشرةً إلى الوظيفة، فتُعاد تشغيلها به؛ ويظلّ القديم صالحًا طوال مهلة السماح المعتادة. ولا يُرسل الرمز بالبريد الإلكتروني أبدًا.
- تؤدّي أرشفة الوظيفة إلى أرشفة مستخدمها، فيتوقّف الرمز عن العمل (بعد انقضاء مدة الخبء القصيرة في بوّابة API).

الأسرار `Since dev`

على خط التطوير، يتضمّن سجلّ الوظيفة قائمة **أسرار**: صفًّا لكل سرّ بمفتاح وقيمة. وتُمرَّر كل قيمة إلى الوظيفة العاملة بوصفها متغيّر بيئة يحمل اسم مفتاحها:

```
const apiKey = process.env.PAYMENT_API_KEY;
```

- **تُخزَّن القيم مشفّرة** وتُعرض مقنّعة. ولا يستطيع أحد قراءتها مجددًا، بمن فيهم المسؤولون.
- **غيّر قيمة** بإدخال قيمة جديدة؛ وحفظ القيمة المقنّعة دون تغيير يُبقيها كما هي. و**احذف** سرًّا بإزالة صفّه.

- **يسري التغيير على مراجعة جديدة.** تُحدِّث المنصة السرّ وتعيد تشغيل الوظيفة دون إعادة بنائها؛ وتحتفظ النسخة العاملة بالقيمة القديمة حتى ذلك الحين.
- **تحمل الحزم الأسماء لا القيم.** يحتفظ تصدير الوظيفة بمفاتيح أسرارها ويستبعد القيم. وبعد تثبيت حزمة، أدخل القيم في البيئة المستهدفة؛ فالوظيفة المثبَّتة تتيح لك تغيير قيم الأسرار الموجودة فقط.

ويُرفض الحفظ بالرمز `422` عند مخالفة إحدى القواعد:

الرمز	متى
`FUNCTION_SECRET_KEY_INVALID`	المفتاح ليس اسمًا صالحًا لمتغيّر بيئة (أحرف وأرقام و `_` ، ولا يبدأ برقم).
`FUNCTION_SECRET_KEY_DUPLICATE`	صفّان يستخدمان المفتاح نفسه.
`FUNCTION_SECRET_KEY_RESERVED`	المفتاح تضبطه المنصة: `APP_ID` ، و `OSOS_API_BASE_URL` ، و `PORT` ، و `K_SERVICE` ، و `K_CONFIGURATION` ، و `K_REVISION` .
`FUNCTION_SECRET_SYSTEM_MANAGED`	أضفت صفّ `OSOS_API_TOKEN` أو غيّرته أو أزلته.
`FUNCTION_IDENTITY_SYSTEM_MANAGED`	ضبطت هوية الوظيفة.

وتُفحص حدود الحجم عند نشر الوظيفة، لا عند الحفظ: الحدّ الأقصى لكل `KEY=value` هو 131071 بايت، وللأسرار مجتمعةً 1 MiB، ولا يُسمح بمحرف NUL. فإذا جرى تجاوز هذه الحدود، أو تعذّر تسليم الأسرار، يُظهر **خطأ التوريد** أحد الرموز التالية:

الرمز	المعنى
`IDENTITY_UNAVAILABLE`	تعذّر إنشاء مستخدم الوظيفة؛ ولم يُنشر شيء.
`SECRETS_SYNC_FAILED`	تعذّر تسليم الأسرار قبل النشر؛ ولم يُنشر شيء.
`SECRETS_UPDATE_FAILED`	تعذّر تسليم تغيير في الأسرار إلى الوظيفة العاملة؛ وتعيد المنصة المحاولة.

ولا تتضمّن أيٌّ من هذه الرسائل قيمة سرّ.

الوظائف مقابل محرّك التعابير

للمنطق الذي لا يحتاج إلى نشر – الحقول المحسوبة، والقيم الديناميكية، والأهداف الشرطية، ورموز القوالب – استخدم محرّك التعابير المدمج الذي يعمل في بيئة معزولة. إذ يمكن للحقل أن يستمدّ قيمته من تعبير أو مؤشِّر، مع إعادة حسابه عند فتح النموذج، أو عند الحفظ، أو عند تغيّر المُدخل، أو عند تغيّر حقول مختارة. ويتوفّر استدعاءان لحلّ التعابير من الرماز:

- `POST /api/v1/entities/{entityId}/expressions/resolve` – تعبير على مستوى الكيان؛
- `POST /api/v1/entities/{entityId}/{entryId}/tokens/resolve` – رموز لمُدخل واحد.

معلومات مفيدة

- **TypeScript وZIPg فقط.** لا تُنشر اللغات الأخرى ولا أنواع المصادر الأخرى (مثل مستودع GitHub).
- **لا يُعاد النشر إلا بمصدر جديد.** لا يؤدّي تعديل الحقول الأخرى في سجلّ الوظيفة إلى إعادة بنائها؛ ويؤدّي تغيير الأسرار إلى إعادة تشغيلها دون إعادة بنائها `Since dev` .
- **النشر غير متزامن.** لا يمكن سرد الوظيفة ولا تشغيلها حتى تصبح حالتها **تم التحقق**.
- **الوظائف خاصة بالعنقود.** ولا يمكن الوصول إليها إلا عبر المنصة.
- **العزل يأتي من وقت التشغيل بلا خادم**، لا من المنصة؛ فالوظيفة تعمل بوصفها خدمةً مستقلة.

استكشاف الأخطاء وإصلاحها

لا تبلغ الحالة أبدًا تم التحقق. قد لا يكون باني الوظائف في المنصة مُعدًّا في هذا النشر؛ وفي هذه الحالة لا يترتّب على حفظ الوظيفة أي إجراء. راجع المشغّل لديك.

الحالة فشل البناء. اقرأ **خطأ التوريد**. من الأسباب الشائعة: غياب `func.yaml` عن جذر ملف ZIP أو عدم صلاحيته، أو استخدام وقت تشغيل غير `typescript`، أو وجود خطأ بناء في رمازك. شغّل `npm run build` محليًا أولًا.

يعمل التفاعل لكنّ الإجراءات اللاحقة لا تتلقّى شيئًا. تحقّق من أنّ معالجك يُعيد JSON (لا نصًّا)، ومن أنّ المفاتيح التي ربطها الباني تحت `result` موجودة في استجابتك.

للتعمّق أكثر

للبُناة: التفاعلات والأتمتة [بناء التطبيقات] · للمشغّلين: الوظائف في النشر متعدّد القاطنين · لمهندسي المنصة: الوظائف – التفاصيل الداخلية

الملفات: قواعد التخزين

هذه الصفحة موجّهة إلى المطوّرين الذين يرفعون الملفات ويُرفقونها ويُتيحونها عبر الـ API. وتشرح القواعد التي تحكم تلك الاستدعاءات: مدّة بقاء الملف، وما يحدث حين يُحفظ مُدخل يستخدمه، والملفات المقبولة، وكيف تُطبَّق حدود التخزين، وكيف تعرض ملفًا في متصفح دون رمز وصول.

أمّا الاستدعاءات نفسها — الرفع، والإرفاق بمُدخل، والتنزيل، وأخطاؤها — فتجدها في التعامل مع الملفات. ابدأ من هناك.

كيف تُحفظ الملفات {#how-file-image-document-fields-use-storage}

لا يحتوي حقل الملف أو الصورة أو المستند على البايتات نفسها أبدًا، بل يحتوي على **الواصف** الذي تُعيده عملية الرفع:

```
{ "id": "5f0e9b2a-…", "name": "invoice.pdf", "mimeType": "application/pdf", "size": 20481 }
```

والمعرّف `id` هو مرجع الملف في جميع المواضع: في قيم المُدخلات، وعمليات التنزيل، والروابط الموقّعة. ارفع الملف أولًا، ثم ضع الواصف في المُدخل.

الملفات المؤقّتة والدائمة

	مؤقّت (`?temp=true`)	دائم (الافتراضي)
مدّة البقاء	يُحذف تلقائيًا بعد **يوم واحد** ما لم تحتفظ به عملية حفظ	يبقى حتى تحذفه المنصة أثناء التنظيف
استخدمه لـ	ملف توشك أن تُرفقه بمُدخل	ملف تديره بنفسك
التنزيل	الاستدعاء نفسه، والمعرّف نفسه	الاستدعاء نفسه، والمعرّف نفسه

ارفع المرفقات ملفاتٍ مؤقّتة `Since 1.10` . فحين تحفظ لاحقًا مُدخلًا يشير إلى الملف، **تُبقيه** المنصة؛ فيصبح دائمًا بالمعرّف نفسه، وتسجّل المنصة أنّ المُدخل يستخدمه. أمّا الملف المؤقّت الذي لا تحتفظ به أيّ عملية حفظ فيُحذف بعد يوم.

ما يفعله الحفظ بالملفات `Since 1.10`

حين يُحفظ مُدخل — أو كيان، أو معرض، أو صفحة، أو تقرير، أو تفاعل — يحتوي على واصفات ملفات، تفحص المنصة كل ملف **جديد** يُضاف إلى ذلك المُدخل:

- يُحتفَظ بالملف المؤقّت ويصبح دائمًا.
- يُسجَّل أنّ المُدخل يستخدم الملف الدائم.
- تفشل عملية الحفظ إذا لم يعد الملف موجودًا (انظر أدناه).

أمّا الملفات التي كان المُدخل يحتوي عليها أصلًا فلا تُفحص مجدّدًا؛ لذا لا تتسبّب أبدًا في فشل إعادة حفظ مُدخل قديم.

أخطاء الحفظ

الحالة / الرمز	المعنى	ما العمل
`400 FILE_EXPIRED`	ملف تشير إليه عملية الحفظ غير موجود، ويكون عادةً ملفًا رُفع مؤقّتًا ومضى عليه أكثر من يوم. ويسرد `details` كل ملف مع حقله (`{fileId, fieldName}`) أو موضعه (`{fileId, path}`)، وسببًا.	ارفع الملف من جديد، ثم أعد الحفظ.

الحالة / الرمز	المعنى	ما العمل
`503` `STORAGE_SERVICE_UNAVAILABLE`	تعذّر الوصول إلى تخزين الملفات أثناء عملية حفظ تُضيف ملفًا جديدًا. أمّا عملية الحفظ التي لا تُضيف ملفًا جديدًا فتُنفَّذ.	أعِد المحاولة مع زيادة مدّة الانتظار بين المحاولات.

الملفات المقبولة

تخضع عمليات الرفع عبر الـ API لثلاثة فحوص:

1. **نوع الملف، بحسب الامتداد.** تُرفض الملفات التي لا امتداد لها. وتُقبل افتراضيًا الامتدادات التالية (ويمكن للمشغّل لديك تغيير القائمة):

```
.jpg .jpeg .png .gif .bmp
.txt .csv .pdf .md .rtf .html
.doc .docx .xls .xlsx .ppt .pptx .odt .ods .odp
.mp3 .mp4 .avi .mov .wmv .flv .mkv .webm .wav .m4a .3gp
.psd .ai .indd .xd .fig
.css
.ttf .otf .woff .woff2
.zip .rar
```

2. **المحتوى يطابق النوع** `Since 1.8` . في الملفات الثنائية – الصور، وPDF، وملفات Office، والوسائط، وPhotoshop، والخطوط، والأرشيفات – تقرأ المنصة بداية الملف وترفضه إن لم يطابق محتواه امتداده. أمّا الصيغ النصية (`.txt` و `.csv` و `.md` و `.html` و `.rtf` و `.css`) وبعض صيغ التصميم (`.ai` و `.indd` و `.xd` و `.fig`) فتُفحص بالامتداد فقط.

3. **لا JavaScript في ملفات PDF** `Since 1.8` . يُرفض ملف PDF الذي يحتوي على إجراء JavaScript.

الرفض	المعنى
`400 INVALID_FILE_TYPE`	الامتداد غير مقبول، أو لا يوجد امتداد أصلًا.
`400 INVALID_FILE_CONTENT`	المحتوى لا يطابق الامتداد.
`400 PDF_JAVASCRIPT_NOT_ALLOWED`	ملف PDF يحتوي على JavaScript.

ولا تفرض المنصة نفسها حدًّا لحجم الملف الواحد؛ لكن قد تفرضه الشبكة التي تسبق بيئة النشر لديك. وتُرسل الملفات عند تنزيلها دائمًا كمرفقات، ولا تُعرض أبدًا بوصفها محتوى نشطًا.

حدود التخزين `Since 1.10`

إذا كانت خطة القاطن تفرض حدًّا للتخزين، يُرفض الرفع بالرمز **`403 STORAGE_CAP_EXCEEDED`** حين يكون القاطن قد بلغ الحدّ بالفعل، أو حين تؤدّي عملية الرفع إلى تجاوزه، حتى لو حدث ذلك في منتصف رفع ملف كبير؛ وعندها يُزال الملف الجزئي. ويتضمّن `details` القيمتين `capBytes` و `usedBytes` . وتُحتسب الملفات المؤقّتة أيضًا ضمن المساحة المستخدمة. ولا تُجدي إعادة المحاولة؛ إذ يجب زيادة حدّ التخزين للقاطن. وتجد طريقة قراءة حدود القاطن في حدود الخطة.

عرض ملف في متصفح دون رمز وصول `Since 1.6`

يستطيع المتصفح تحميل ملف باستخدام `<img src>` عادي أو رابط، إذا زوّدته **رابطًا موقّعًا** قصير الصلاحية:

1. اطلب الروابط باستخدام بيانات اعتمادك المعتادة، لما يصل إلى 100 معرّف ملف في الطلب الواحد:

```
curl -s -X POST "https://acme.example.com/api/storage/v2/files/links" \
  -H "Authorization: Bearer <access-token>" \
  -H "Content-Type: application/json" \
  -d '{"ids": ["5f0e9b2a-…", "a71c…"]}'
```

وتكون الاستجابة `[{ id, url, expiresAt }]`. وتُستبعَد المعرّفات غير الموجودة دون إشعار. أمّا المعرّفات غير الصالحة (التي لا تتكوّن من 1 إلى 64 محرفًا من الحروف أو الأرقام أو `_` أو `-`)، أو الطلبات التي تتجاوز 100 معرّف، فتُرفض بالرمز `400 INVALID_FILE_IDS`.

2. استخدم كل `url` مباشرةً؛ فهو لا يحتاج إلى بيانات اعتماد، ويظلّ صالحًا حتى `expiresAt` – أي لمدّة تتراوح بين ساعة وساعتين وفق الإعدادات الافتراضية.

الاستجابات التي يُعيدها الرابط:

- `200` مع الملف. تُرسل الصور والفيديو والصوت وPDF للعرض ضمن الصفحة، بينما تُرسل ملفات SVG وسائر الملفات للتنزيل. ويمكن تخزين الاستجابات في الخبء حتى انتهاء صلاحية الرابط، ويُجاب عن الطلب المتكرّر الذي يحمل القيمة نفسها لـ `ETag` بالرمز `304`. ولا تُدعَم نطاقات البايتات؛ لذا استخدم التنزيل المعتاد للتنقّل داخل الفيديو.
- `403 LINK_INVALID` (رابط عُبث به) أو `403 LINK_EXPIRED`؛ و `404` إذا لم يعد الملف موجودًا.
- `503 FILE_LINKS_NOT_CONFIGURED` من كلا الاستدعاءين إذا لم يُهيّئ المشغّل لديك الروابط. استخدم حينها التنزيل المعتاد.

وتتّبع استجابات الرفض هذه بنية الأخطاء الخاصة بتخزين الملفات: اقرأ الرمز من `result.details.error`، لا من `error.code` (انظر الأخطاء من خدمة التخزين).

وتكون روابط الملف الواحد متطابقة ضمن نافذة التوقيع نفسها، مما يتيح للمتصفحات الاحتفاظ بها في الخبء عبر عمليات تحميل الصفحات. واطلب روابط جديدة قبيل `expiresAt`.

> **⚠ الرابط الموقّع صلاحية لحامله**
>
> الرابط الموقّع يحمل صلاحيته بنفسه. فكل من يحمل الرابط يستطيع جلب الملف حتى تنتهي صلاحيته. لا تسجّل الروابط في السجلّات، ولا تضعها في عناوين قابلة للمشاركة، ولا تُسلّمها إلى أطراف ثالثة.

الملفات المفقودة تُجاب بـ `404` `Since 1.6`

يُجاب عن طلب تنزيل ملف أو الوصول إليه برابط موقّع بالرمز `404` إذا كان سجلّه موجودًا لكنّ بايتاته المخزَّنة لم تعد موجودة، كما يحدث مع المعرّف غير المعروف.

التنظيف التلقائي `Since 1.10`

تستطيع المنصة حذف الملفات التي لم تعد مستخدمة في أيّ موضع، مع مراعاة الضوابط التالية:

- لا يُنظر أبدًا إلا في الملفات التي **أبقاها حفظ** (رُفعت مؤقّتة ثم جرى الاحتفاظ بها عند الحفظ). أمّا الملفات المرفوعة مباشرةً إلى التخزين الدائم فلا تُنظَّف تلقائيًا أبدًا.

- لا يُنظر في الملف إلا بعد أن يتوقّف كل مُدخل كان يستخدمه عن استخدامه، وتنقضي مهلة سماح (7 أيام افتراضيًا).
- يجب أن يؤكّد كل مُدخل سبق أن استخدم الملف أنّه لم يعد يستخدمه. وعند وجود أيّ شكّ، يُحتفَظ بالملف.
- يحتفظ المُدخل بكل ملف وارد في تاريخه؛ لذا لا تُحذف تلك الملفات ما دام المُدخل موجودًا.

ويحدّد المشغّل لديك ما إذا كان التنظيف يحذف الملفات فعليًا أم يكتفي بالإبلاغ عنها؛ وتأتي المنصة بإعداد افتراضي يقتصر فيه التنظيف على الإبلاغ.

تنزيل تصدير القاطن `Since 1.10`

يُنزَّل تصدير بيانات القاطن (انظر واجهة النسخ الاحتياطي) عبر **رابط أحادي الاستخدام** تُصدره المنصة على مضيف القاطن نفسه، وتكون صلاحيته من 5 إلى 10 دقائق. ولا يحتاج إلى بيانات اعتماد؛ فالرابط نفسه يمثّل بيانات الاعتماد. ويُجاب عن طلب استخدام رابط سبق استخدامه بالرمز `410 LINK_USED`، وعن طلب استخدام رابط منتهي الصلاحية أو معدَّل بالرمز `403 LINK_EXPIRED` أو `403 LINK_INVALID`، وعن طلب تصدير غير معروف بالرمز `404 EXPORT_NOT_FOUND`. ويُنزَّل التصدير في أرشيف gzip.

ويظلّ الملف المخزَّن موجودًا عند حذف إصدار الحزمة أو سجلّ التصدير المرتبط به.

> **للتعمّق أكثر**
>
> للمشغّلين: مرجع التكوين – storage-service · إعدادات كانس الملفات · لمهندسي المنصة: تخزين الملفات – التفاصيل الداخلية

صيغ المعايير ومسارات الحقول

هذه الصفحة موجّهة إلى **المُكامِلين** الذين يكتبون المعايير أو مسارات الحقول مباشرةً – عبر الواجهة البرمجية، أو حزمة، أو حمولة يحرّرونها يدويًا. أما البُناة الذين ينشئون الشروط في المصمّم فلا يكتبون هذه المسارات أبدًا؛ فالمصمّم يكتبها عنهم. تظهر شروط المعايير على خمسة أسطح متمايزة في واجهات أسس البرمجية: **معايير المعرض**، و**الاستعلامات المحفوظة**، و**مسارات التجميع في الاستعلامات الإحصائية**، و**معايير التشغيل في التفاعلات**، و**مسارات القيم** `Dynamic` **في إجراءات التفاعلات**. يفسّر كل سطح سلسلة مسار الحقل على طريقته الخاصة، والصيغ متقاربة بما يكفي لتبدو قابلة للتبادل، ومختلفة بما يكفي لتوقعك في الخطأ. هذه الصفحة هي المرجع الوحيد لأيّ صيغة يتطلّبها كل سطح، ولماذا، وكيف يبدو الاختيار الخاطئ أثناء التشغيل.

> ⚠ **المسارات الخاطئة تفشل بصمت – في كل مكان تقريبًا**
>
> لا يتحقق أيّ سطح من أسطح المعايير أو التجميع من مسار الحقل مقابل الكيان. المسار ذو الصيغة الخاطئة لا يُرفض أبدًا:
>
> - **معايير المعرض والاستعلامات المحفوظة** تُعيد HTTP `200` مع **مصفوفة مدخلات فارغة**.
> - **الاستعلامات الإحصائية** تُعيد **دلوًا واحدًا** بقيمة `category: null` (ودوال من نمط `Sum` تُنتج `0`).
> - **معايير التشغيل في التفاعلات** لا تتطابق أبدًا – يُتخطّى التشغيل ولا يبقى سوى سطر واحد في سجلّات الخدمة.
>
> الاستثناء الوحيد هو مسار القيمة `Dynamic` في إجراء التفاعل، الذي يفشل بصوت مسموع – يُخفق الإجراء ويُسجَّل الفشل في السجل التاريخي لتنفيذ التشغيل (انظر بصمت أم بصوت مسموع).

الصيغ حسب السطح

تُخزَّن بيانات حقول المُدخل تحت الخاصية `value` في كل مستند مُدخل، ولذلك تتطلّب معظم الأسطح الاسم المعياري الكامل للحقل مسبوقًا بـ `value.` – لكن ليس كلّها، وليس لمسارات أوصاف المُدخل.

السطح	الصيغة المطلوبة	مثال على المسار	ما يُنتجه الاختيار الخاطئ
معايير المعرض (`filter.criteria` + المرشحات السريعة)	`value.<fullNormalizedName>`؛ أوصاف المُدخل دون بادئة (`state`، `creationInfo.doneAt`)	`value.amount`	HTTP `200` مع مصفوفة مدخلات فارغة
الاستعلامات المحفوظة (`DataQuery.criteria`، بما فيها كمصدر بيانات معرض)	نفس صيغة معايير المعرض – تُدمج في المرشِّح ذاته عند التنفيذ	`value.status.id`	HTTP `200` مع مصفوفة مدخلات فارغة
تجميع الاستعلام الإحصائي (`aggregation.field`، `aggregation.groupByField`)	`value.<fullNormalizedName>`؛ استثناءات الأوصاف دون بادئة (`creationInfo.doneAt` / `doneBy`)، `value.updateInfo.doneAt` / `doneBy`)؛ عبور العلاقات: `value.<field>.:relation.value.<field>`	`value.department`	دلو واحد بقيمة `category: null`؛ و`Sum` / `Average` على المسار الغائب تُنتج `0` / `null`
معايير التشغيل في التفاعلات (`DataReaction.criteria`)	`value.<field>` (الصيغة القديمة) أو الصيغة القياسية `scope.entries.<entityNormalizedName>.value.<field>` – **وليس المسار المجرّد أبدًا**	`value.priority`	المعايير لا تتطابق أبدًا؛ يُتخطّى التشغيل بصمت

السطح	الصيغة المطلوبة	مثال على المسار	ما يُنتجه الاختيار الخاطئ
مسارات القيم `Dynamic` في إجراءات التفاعلات	الصيغة القياسية `scope.entries.<entityNormalizedName>.value.<field>`؛ وتُوسَّع تلقائيًا `value.<field>` وأسماء الحقول المجرّدة المعروفة	`scope.entries.task.value.assignee`	**يفشل الإجراء بخطأ مُسجَّل** في سجل تنفيذ التشغيل

القاعدة العملية: اكتب البادئة `value.` دائمًا. فهي مطلوبة على الأسطح الثلاثة الأولى ومقبولة (تُوسَّع تلقائيًا) على سطحَي التفاعلات كليهما. أما السطح الوحيد في التفاعلات الذي تكون فيه الأسماء المجرّدة هي الصيغة الفعلية فليس مسار معايير أصلًا: قائمة مراقبة الحقول `eventConfig.fields` لحدث `DataEntryUpdated`، التي تُطابِق قيم `fullNormalizedName` المجرّدة.

لماذا يقرأ كل سطح المسارات على طريقته

معايير المعرض والاستعلامات المحفوظة

عند جلب مدخلات معرض ما، تدمج خدمة البيانات `filter.criteria` من الطلب، ومعايير مصدر البيانات من نوع الاستعلام في المعرض، وأيّ استعلام مرشِّح سريع نشط، في تعبير ترشيح **واحد**، ثم تسلّمه إلى محرك الاستعلام **دون أيّ نطاق حقول**. يستخدم المحرك سلسلة `field` الخاصة بالشرط **حرفيًا** كمفتاح مطابقة، وتُنفَّذ مرحلة `$match` الناتجة مباشرة على مستندات المدخلات المخزّنة – التي تعيش كل قيم حقولها تحت الخاصية `value` في المستند. هذه هي الصيغة كلّها: `value.<fullNormalizedName>` يصل إلى بياناتك، بينما تقع أوصاف المُدخل مثل `state` أو `creationInfo.doneAt` في المستوى الأعلى من المستند ولذلك تُكتب دون البادئة.

ولا شيء يفحص المسار عند الحفظ أيضًا: التحليل الذي يجري عند حفظ معايير الاستعلام يتحقق من **المعامِلات فقط** – يُرفض المعامِل المجهول أو تعبير الحقل غير المدعوم، أما المسار السليم نحويًا والموجَّه إلى المستوى الخاطئ فيمرّ بسلام ثم لا يطابق شيئًا لاحقًا.

مسارات التجميع في الاستعلامات الإحصائية

يشغّل المسار `GET /v2/entities/{entity}/statistic-queries/{id}/execute` التجميع المخزَّن. داخليًا، تنزع الخدمة البادئة `value.` من `aggregation.field` / `aggregation.groupByField` **فقط للبحث عن الحقل على الكيان** – وتحتفظ بسلسلة المستدعي الخام كمسار البحث وتُقحمها حرفيًا في مراحل خط الأنابيب `$unwind` / `$group` / `$addFields`. ولأن مستندات المدخلات تعشّش بياناتها تحت `value`، يجب أن تكون المسارات مسبوقة بـ `value.` لتصيب أيّ شيء؛ وعقد الواجهة الخلفية نفسه لهذه المسارات هو `value.<fieldName>` (أو `creationInfo.<doneAt>` للأوصاف). يستخدم عبور العلاقات العلامة `:relation` هكذا: `value.<field>.:relation.value.<field>`. والأوصاف هي الاستثناء مجددًا، وتُعالَج دون البادئة: `creationInfo.doneBy` / `updateInfo.doneBy` و `creationInfo.doneAt` / `updateInfo.doneAt`.

مثال عملي، مع قيم دالة التجميع الرقمية التي يتطلّبها النموذج (`Count = 0`، `Sum = 1`، `Min = 2`، `Max = 3`، `Average = 4`):

```
{
  "aggregation": {
    "aggregationFunction": 1,
    "field": "value.amount",
    "groupByField": ["value.department"]
  }
}
```

اكتب `"field": "amount"` بدلًا من ذلك فيُجمِّع خط الأنابيب على `$amount` – وهو مفتاح غائب عن كل مستند – فتنهار كل المدخلات في دلو واحد بقيمة `category: null` ويكون المجموع `0`، دون أيّ خطأ في أيّ مكان.

وثمة فخّ مجاور على السطح نفسه: يجب أن يكون `source.store` هو **الاسم المعياري** للكيان. يُبحث عن المخزن بالحقل `normalizedName` فقط، وعند التنفيذ تُستخدم السلسلة حرفيًا كاسم مجموعة MongoDB. تمرير **معرّف** كيان صحيح أو اسم عرض يفشل عند الحفظ برسالة مضلّلة `Entity <store> not found` (الرمز `ENTITY_NOT_FOUND`) – فالكيان موجود، لكنه غير موجود تحت `normalizedName`. (كما أن نمط المخطط الخاص بالمخزن يرفض شرطات UUID والمسافات رفضًا مباشرًا.)

معايير التشغيل في التفاعلات

معايير التفاعلات **لا** تُقيَّم على مستند المُدخل الحي. يُسطَّح سياق التشغيل إلى مُدخل صناعي يحمل كائن `value` فيه مفاتيح على شكل `scope_entries_<entityNormalizedName>_value_<field>`: يُدرَج ذلك المُدخل في مجموعة مؤقتة، وتُعاد كتابة مسارات المعايير لتطابقه – إذ يُحوَّل كل من `scope.entries.<entity>.value.<path>` والصيغة القديمة `value.<field>` إلى المفاتيح المسطَّحة.

أما المسار **المجرّد** فلا تمسّه إعادة الكتابة، ثم يقتطع منطق نطاق الحقول في المحرك أول مقطع من المسار – وهو ما يترك للمسار المجرّد أحادي المقطع **السلسلة الفارغة**، يصبح الشرط مطابقة على `""`، وهي لا تطابق شيئًا، فيُتخطّى التشغيل. ولذلك فإن صيغة معايير التفاعلات هي فعليًا مرساة `value.` ذاتها كما في المعارض: اكتب `value.<field>`، أو الصيغة الكاملة `scope.entries.<entityNormalizedName>.value.<field>`.

وإن كنت قد رأيت أسماء حقول مجرّدة «تعمل» في التفاعلات، فذلك كان قائمة المراقبة `eventConfig.fields` لحدث `DataEntryUpdated` – وهي قائمة قيم `fullNormalizedName` مجرّدة تحدّد *أيّ تغييرات الحقول تُطلق الحدث*، وليست مسار معايير.

مسارات `Dynamic` في إجراءات التفاعلات

قيمة `Dynamic` في إجراء التفاعل لا تُحلّ عبر MongoDB إطلاقًا: بل هي عملية قراءة خصائص على **سياق التشغيل في الذاكرة**. يقبل المحلّل ثلاث صيغ – القياسية `scope.entries.<entityNormalizedName>.value.<field>`، والقديمة `value.<field>` (تُوسَّع تلقائيًا إلى الصيغة القياسية)، والاسم المجرّد `<fullNormalizedName>` عندما يطابق حقلًا معروفًا. البادئة `value.` تعمل هنا دائمًا إذًا، لكن هذا هو السطح الوحيد الذي لا تكون فيه مطلوبة.

حقول التصنيف والمُحوَّل تُطابِق معرّف الخيار

تحمل القيمة المخزَّنة لحقل التصنيف كائن الخيار، وتقارن المعايير مقابل `id` الخاص به:

- **In / Not in** مع كائنات خيارات: تعيد حالة التصنيف الصريحة في المحرك كتابة مفتاح المطابقة إلى `value.<field>.id` وتحوّل كل وسيط إلى `id` الخاص به، فيُترجم إلى `{"value.<field>.id": {"inq": ["<optionId>", …]}}` (و `nin` لـ Not in).
- **Equal / Not equal** مع وسيط كائني يتوسّعان لكل مفتاح: يصبح `{"id": "<uuid>"}` هو `{"value.<field>.id": {"eq": "<uuid>"}}`.
- **تجميعات الاستعلام الإحصائي** تُلحق `.id` بمسار التجميع لحقول التصنيف **والمُحوَّل**، ثم تحلّ معرّف كل دلو في النتيجة رجوعًا إلى لصيقته المعروضة – خيارات التصنيف من قائمة خيارات الحقل المضبوطة، وقيم المُحوَّل من المجموعات الداخلية للكيان.

معرّف الخيار `id` سلسلة نصية عادية مطلوبة في المخطط؛ وهو UUID بعُرف الواجهة الأمامية لا بإلزام المخطط. طابِق مقابل **معرّف الخيار**، وليس اللصيقة أبدًا. مثال على شرط *In*:

```
{
  "field": "value.priority",
  "args": [{ "id": "0b9f3c2e-…" }, { "id": "6d41a8f0-…" }]
}
```

يُترجم إلى `{"value.priority.id": {"inq": ["0b9f3c2e-…", "6d41a8f0-…"]}}`.

`In` / `NotIn` مع قيمة مرجع أو تصنيف مفردة `Since 1.10`

يُطابق `In` و`NotIn` حقل **المرجع** على `<field>.dataEntry` وحقل **التصنيف** على `<field>.id`، باختزال كل وسيط إلى ذلك المعرّف. وقبل 1.10 لم يكن الاختزال يجري إلّا حين تكون القيمة المحلولة **قائمةً** أصلًا. فالكائن المفرد – معامل مملوء من مُدخل اختيار أحادي (`status: {id: 'doing'}`)، أو معيار تفاعل مثل `assignee In scope.me()` – كان يبقى كاملًا داخل `$in` فلا يُطابق شيئًا، دون أي خطأ.

وصار محرّكا الاستعلام كلاهما (محرّك خدمة البيانات ومحرّك خدمة التفاعلات) يترجمان الزوج بالطريقة نفسها: **تُلفّ القيمة في قائمة أولًا**، ثم يحوّل العنصر الحامل لـ `dataEntity` + `dataEntry` المفتاح إلى `.dataEntry`، والعنصر الحامل لـ `id` يحوّله إلى `.id`. ويُختار التحويل بفحص **القيم** لا نوع بيانات الحقل – فالمعيار المكتوب يدويًا بسلسلة معرّف مجرّدة ما زال يُقارَن حرفيًا. وفي معايير التفاعلات لم يعد عنصر `null` في قائمة مراجع يرمي استثناءً.

مصدر القيمة `Expression` وشكل نتيجته `Since 1.10`

يحمل وسيط المعايير من نوع **تعبير** (`FilterExpressionValueTypeEnum.FieldExpression`، `2`) **سلسلةً** نصية لتعبير JavaScript؛ وتُرفض القيمة غير النصية عند الحفظ. ويقيّمه المحرّكان كلاهما تعبيرًا عامًّا – **بلا مُدخل في النطاق** – ويضعان **النتيجة الخام** في الشرط. و**لا تحويل حسب نوع الحقل**، فلا يُطابق إلّا حين يُعيد الشكل الذي يخزّنه الحقل.

ويعرض منشئ الاستعلامات مصدر التعبير لأنواع الحقول التالية، ولكلٍّ منها النتيجة التي يجب أن ينتجها (تظهر تلميحًا تحت المربّع):

نوع الحقل	يجب أن يُعيد التعبير
نص	`string`
رقم	`number` – `10` لا `'10'`
تاريخ / تاريخ ووقت / وقت	`string` بالصيغة المخزّنة (مثل `yyyy-MM-dd`)، لا كائن `Date`
منطقي	`boolean`
مرجع	`{dataEntity, dataEntry}` – مثل `scope.me()`، لا معرّفًا مجرّدًا

ويقبل `In` / `NotIn` النوع نفسه أو قائمة منه. ولا تعرض حقول التصنيف والنص المترجم وJSON / الجغرافيا والملف والأيقونة والكتلة هذا المصدر. والشكل الخاطئ مجرّد **تحذير** في المحرّر – إذ يفحص المحرّر نوع التعبير مقابل الأنواع المقبولة، لكن **زرّ التطبيق لا يُحجب أبدًا** – فيمكن حفظ تعبير غير مطابق ثم لا يُطابق شيئًا بصمت، تمامًا كمسار الحقل الخاطئ. وتستخدم لوحة **الترشيح** في مدخلات المستخدم النهائي محرّر التعبيرات نفسه ذا الإكمال التلقائي الذي في منشئ الإدارة؛ ولا تُحمَّل رماز المحرّر إلّا حين يُعرض صفّ تعبير.

مؤشرات التاريخ في معايير التفاعلات

تحلّ معايير التفاعلات الرمزين `@today()` و `@now()` **كليهما** – ويحتوي نطاق التعبيرات أيضًا على `@StartOfDay` و`@StartOfWeek` و`@StartOfMonth` و`@StartOfYear` . ما يختلف بين الرمزين الرئيسيين هو **شكل القيمة المُعادة**، وذلك الاختلاف هو الفخّ الحقيقي:

الرمز	هل يُحلّ في معايير التفاعلات؟	يُعيد	استخدمه مع
`@now('±N<unit>')`	نعم	**تاريخًا ووقتًا** كاملًا بصيغة ISO	مقارنات الحقول من نوع DateTime فقط
`@today('±N<unit>')`	نعم	سلسلة **تاريخ فقط** (بداية اليوم المحلي، مع إزاحة `+2h` مثبّتة في الرماز قبل التنسيق)	الحقول من نوع التاريخ فقط

ولأن `@now()` يُنتج تاريخًا ووقتًا كاملًا، فإن مقارنة مساواة أو نطاق مقابل قيمة مخزَّنة **من نوع التاريخ فقط** لا تتطابق أبدًا – وهو ما يظهر تمامًا كأن «`@now()` لا يعمل في معايير التفاعلات»، مع أن الرمز يُحلّ دون مشكلة. استخدم `@today('±Nd')` للحقول من نوع التاريخ فقط، واحتفظ بـ `@now()` لمقارنات DateTime.

قواعد التلاعب بالمقدار، المشتركة بين الرمزين:

- **يجب** أن يكون المقدار مُوقَّعًا: `'+1d'` ، `'-3w'` . المقدار **غير المُوقَّع** مثل `'1d'` يُتجاهل بصمت ويُعاد التاريخ الأساسي.
- الوحدات: `d` أيام، `w` أسابيع، `M` أشهر، `y` سنوات، `h` ساعات، `m` دقائق، `s` ثوانٍ.

المؤشر الذي يفشل تقييمه **ليس** صامتًا: يرمي تنفيذ التعبير استثناء `ExpressionExecutionException` .

بصمت أم بصوت مسموع: أين يظهر المسار الخاطئ

السطح	فحص وقت الحفظ	المسار الخاطئ أثناء التشغيل	أين يظهر
معايير المعرض / الاستعلام المحفوظ	التحقق من المعامِلات فقط – لا خطأ مسار سيّئ أبدًا	`$match` على مفتاح غير موجود ← HTTP `200` ومدخلات فارغة	لا مكان
تجميع الاستعلام الإحصائي	إخفاقات البحث عن الحقل تُلتقط وتُسجَّل فحسب	دلو واحد بقيمة `category: null` ؛ و`Sum` لحقل غائب ← `0`	لا مكان – شكل الاستجابة يبدو سليمًا
معايير التشغيل في التفاعلات	–	المعايير لا تتطابق أبدًا؛ يُتخطّى التشغيل	سطر واحد في سجلّات الخدمة فقط؛ لا سجلّ في تاريخ التشغيل
مسار `Dynamic` في إجراء التفاعل	–	يُرمى `Path "<path>" does not exist in the context.`؛ وتُغلَّف إخفاقات الحلّ في `INVALID_ENTRY_FIELD`	يُسجَّل في **السجل التاريخي لتنفيذ التشغيل** وفي سجل أخطاء الخدمة – ولا يظهر شيء للطلب الأصلي

السجل التاريخي لتنفيذ التشغيل هو المكان **الوحيد** الذي يظهر فيه مسار `Dynamic` الخاطئ؛ ومن وجهة نظر المستدعي، حتى الفشل المسموع يبدو صامتًا.

كل ما في هذه الصفحة سلوك قائم منذ زمن طويل، لم يتغيّر من خط 1.2 حتى خط التطوير الحالي، باستثناء القسمين الموسومين `Since 1.10` (`In` / `NotIn` مع قيمة مفردة ومصدر القيمة `Expression`)، ولكلٍّ منهما شارته.

استدعاء واجهة HTTP

تُوجَّه هذه الصفحة إلى المطوّرين الذين يستدعون واجهة أسس البرمجية من الرماز، وهي تتناول القواعد المشتركة بين جميع الاستدعاءات: إلى أين تُرسل الاستدعاء، وكيف تُجري المصادقة، وكيف تقرأ الاستجابة وأخطاءها، وكيف تقسّم النتائج إلى صفحات وتصفّيها، وأين تجد المواصفة المقروءة آليًا. اقرأها مرة واحدة قبل كتابة عميل؛ أمّا دليل البدء السريع مع الواجهة الخلفية فيرشدك إلى إجراء أول استدعاء ناجح في غضون دقائق.

أول استدعاءين لك

اقرأ صفحة من المُدخلات، ثم أنشئ مُدخلًا. استبدل المضيف والرمز واسم الكيان (الاسم النموذجي للكيان، مثل `employees`):

```
export BASE="https://acme.example.com"
export TOKEN="<your-api-token>"

# Read: the first 20 active employees in HR, newest first
curl -s -G "$BASE/api/v2/entities/employees/entries" \
  -H "x-app-token: $TOKEN" \
  --data-urlencode 'filterCriteria={"limit":20,"skip":0,"order":["value.hireDate DESC"],"criteria":{"operator":"Equa
l","field":"value.department","args":[{"type":0,"value":"HR"}]}}'

# Write: create one record
curl -s -X POST "$BASE/api/v1/entities/employees/entries" \
  -H "x-app-token: $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"value": {"name": "Lina Haddad", "department": "HR"}}'
```

تَرِد الاستجابة لكلا الاستدعاءين ضمن الغلاف المعتاد: `{ "success": true, "result": … }` . وتشرح بقية الصفحة كل جزء من ذلك.

إلى أين تُرسل الطلبات

أرسل كل استدعاء إلى **مضيف القاطن نفسه**، بمسارات تبدأ بـ `/api` . تتحقّق بوّابة الـ API على ذلك المضيف من بيانات اعتمادك، وتحدّد القاطن من اسم المضيف، وتفحص أذونات الوصول، ثم تمرّر الاستدعاء إلى الجزء المناسب من المنصة. ولا تستدعِ أي خدمة داخلية مباشرةً، ولا ترسل معرّف قاطن أبدًا.

وتحدّد بادئة المسار المجال:

بادئة المسار	المجال
`/api`	منصة البيانات الأساسية: الكيانات، والمُدخلات، والمعارض، والاستعلامات، والأذونات
`/api/comments`	التعليقات
`/api/notification`	الإشعارات ورموز الإشعارات الفورية
`/api/timeline`	منشورات الشريط الزمني
`/api/workflow`	تنفيذ سير العمل وملكيته
`/api/reaction`	قواعد الأتمتة (التفاعلات)
`/api/functions`	الوظائف على الخادم

بادئة المسار	المجال
`/api/integrations`	تكاملات الأطراف الثالثة
`/api/dependency`	مخطط الاعتماديات
`/api/packaging`	الحزم، والتصدير والاستيراد، وطلبات النشر، والمتجر داخل التطبيق، والتثبيتات
`/api/logs`	سجلّات البريد الإلكتروني والرسائل القصيرة وWhatsApp (للقراءة فقط)، والـ webhook الخاص بحالة WhatsApp `Since 1.10`
`/api/storage`	الملفات – انظر التعامل مع الملفات
`/api/rtc`	رموز المكالمات الصوتية والمرئية

إذا كانت العملية موثّقة ضمن مجال غير منصة البيانات الأساسية – في مرجع نقاط النهاية أو في مستند OpenAPI الخاص بذلك المجال – فإن مسارها يكون نسبيًا إلى بادئة ذلك المجال. فعملية سرد قوالب WhatsApp مثلًا، `GET /v1/whatsapp/templates` في مجال الأتمتة، تُستدعى بالمسار `/api/reaction/v1/whatsapp/templates` .

الاستدعاءات المقيّدة بمعرض. تتوفّر لكثير من عمليات القراءة والكتابة صيغة تمرّ عبر معرض: `/api/v1/v/{viewId}/…` . وتُفحص إعدادات الوصول الخاصة بالمعرض أولًا؛ لذا يعني الرمز `403` على مسار كهذا عادةً أنّ المعرض لا يسمح بتلك العملية، لا أنّ بيانات اعتمادك خاطئة.

وتكون أجسام الطلبات والاستجابات بصيغة JSON (`Content-Type: application/json` ، `Accept: application/json`)، باستثناء عمليات رفع الملفات وتنزيلها.

الإصدارات

يَرِد الإصدار في المسار بعد البادئة مباشرةً، وهو يخصّ كل عملية على حدة، لا المجال كله. وكثيرًا ما تتوفّر للمورد الواحد إصدارات `v1` و `v2` و `v3` جنبًا إلى جنب:

```
GET  /api/v1/entities/{entity}/entries          (deprecated)
GET  /api/v2/entities/{entity}/entries
GET  /api/v3/entities/{entity}/entries
POST /api/v1/entities/{entity}/entries
GET  /api/v1/entities/{entity}/entries/{id}
```

- تُوسم الإصدارات القديمة بأنها مُهمَلة في مستند OpenAPI، ولا تُزال.
- يضيف الإصدار الأحدث عادةً قدرات جديدة؛ فلا تفترض أنّ الإصدارين `v1` و `v2` من العملية «نفسها» يُعيدان البنية نفسها.
- استخدم أعلى إصدار يقدّمه المورد ما لم تحتج إلى بنية قديمة.

المصادقة

أرسل أحد نوعَي بيانات الاعتماد مع كل استدعاء:

بيانات الاعتماد	الترويسة	مصدرها
رمز API	`x-app-token: <token>`	رموز الـ API
رمز وصول OAuth	`Authorization: Bearer <token>`	المصادقة OAuthg

وللمفاضلة بينهما، راجع اختيار بيانات الاعتماد.

قراءة الاستجابة

تُغلَّف الاستجابات الناجحة بغلاف موحّد:

```
{ "success": true, "result": { "count": 42 } }
```

وتكون قيمة `result` مساوية لـ `null` حين لا تُعيد العملية شيئًا. وتَرِد بعض الاستجابات دون غلاف:

الطلب	الجسم عند النجاح
`GET` على مسار `/api/v3/…` `Since 1.10`	**الحمولة المجرّدة**، مثل `{ "entries": [ … ], "tags": [ … ] }`
`POST` و `PATCH` و `PUT` و `DELETE` على `/api/v3/…`	مغلَّفة
كل استدعاءات JSON من `v1` و `v2` وغير المرقّمة بإصدار	مغلَّفة
تنزيلات الملفات والتدفقات الأخرى، وفحوص السلامة	البايتات الخام أو التدفق
مستندات OpenAPI (`/openapi.json…` و `/openapi.yaml…`)	المستند المجرّد

وتُغلَّف الأخطاء دائمًا، بما فيها أخطاء عمليات القراءة على `/v3` .

> **انزع الغلاف بحسب البنية لا بحسب الإصدار**
>
> في عمليات النشر بالإصدارات من 1.8 حتى 1.8.21، تبقى استجابات القراءة على `/v3` مغلَّفة. لذا ينبغي للعميل الذي يتعامل مع أكثر من سلسلة إصدارات أن يستخرج `result` من الغلاف **فقط حين يحمل الجسم** `"success": true` . وهذا بالضبط ما تفعله البوّابة الإلكترونية للمنصة.

معالجة الأخطاء

تتّبع جميع الأخطاء البنية نفسها:

```
{
  "success": false,
  "result": null,
  "error": {
    "code": "ENTITY_NOT_EXIST",            // stable, machine-readable
    "message": { "en": "…", "ar": "…" },   // localized text for people
    "serviceName": "…",                     // which part of the platform answered
    "traceId": "…"                          // quote this when you report a problem
  }
}
```

- **ابنِ منطق التفرّع على `error.code` ، لا على نص الرسالة أبدًا.** واستخدم رمز حالة HTTP للمعالجة العامة.
- **ليس `500` مؤقتًا دائمًا.** فكثير من حالات الرفض الحتمية لا تزال تُعيد `500` . راجع الرمز في الأخطاء وإعادة المحاولة قبل إعادة المحاولة.
- تتضمّن بعض الأخطاء كائن `details` . ففي حال إخفاق التحقّق (`VALIDATION_FAILED`)، يحدّد هذا الكائن المسار الذي أخفق التحقّق منه والقاعدة، وكذلك القيم المسموح بها إذا كان الاختيار غير صالح.
- تؤدي القيمة المكرّرة في حقل فريد إلى استجابة `400 UNIQUE_VALUE` ، مع مسار الحقل والقيمة في `details` .
- تؤدي بيانات الاعتماد التي لا تستطيع المنصة ربطها بمستخدم إلى استجابة `401` .
- تُستثنى الملفات من هذه البنية؛ انظر التعامل مع الملفات.

حالات الرفض المتعلقة بالقاطن والسعة `Since 1.10`

الحالة	`error.code`	متى	ما العمل
`404`	`SPACE_ERROR`	المضيف الذي استدعيته لا يطابق أي مساحة.	استخدم مضيفًا مسجَّلًا للمساحة.
`404`	`TENANT_NOT_RESOLVED`	في النشر متعدّد القاطنين: المضيف لا يطابق أي قاطن نشط.	استخدم مضيف القاطن نفسه.
`429`	`RATE_LIMITED` + `Retry-After`	في النشر متعدّد القاطنين: استنفد القاطن حصّته من الطلبات في الثانية أو في الدقيقة. ويحدّد `details` القيمتين `window` و `limit`؛ وعادةً ما تتضمّن الاستجابة ترويسات `RateLimit-*`.	انتظر عدد الثواني المحدّد في `Retry-After`.
`503`	`RATE_LIMIT_UNAVAILABLE` + `Retry-After: 1`	تعذّر على مقيِّد المعدّل احتساب الطلب، فرفضه.	أعِد المحاولة بعد ثانية واحدة.
`503`	`TENANT_IN_MAINTENANCE` + `Retry-After`	في النشر متعدّد القاطنين: تجري استعادة القاطن من نسخة احتياطية أو نقله. وتُرفض جميع الاستدعاءات باستثناء قراءات الحالة في واجهة النسخ الاحتياطي.	انتظر عدد الثواني المحدّد في `Retry-After` (من 1 إلى 3600، والقيمة الافتراضية 30)، ثم أعِد المحاولة.
`400`	`SPACE_CONTEXT_MISSING`، `SPACE_NOT_RESOLVED`	فقط عند استدعاء خدمة مباشرةً دون المرور ببوّابة الـ API.	أجرِ الاستدعاء عبر بوّابة الـ API.

تعامَل مع أي `503` مصحوب بـ `Retry-After` بالطريقة نفسها: انتظر المدّة التي يحدّدها، ثم أعِد المحاولة. ويتضمّن جسم استجابة `TENANT_IN_MAINTENANCE` قيمة `traceId` تساوي `NO-TRACE-ID` ورسالة عربية فارغة. ولتفاصيل ما يحدث لأحداث القاطن في أثناء ذلك، انظر نموذج البيانات والأحداث.

حالات رفض تستحق المعالجة

تُعيد حالات الرفض التالية، المتعلقة بمُدخلاتك أو بأذونات وصولك، رمز حالة 4xx ورمز خطأ ثابتًا. وقد أُتيحت الحالات الموسومة بـ `Since 1.8` أيضًا ضمن تحديثات 1.8؛ أمّا البقية فأُتيحت بدءًا من 1.10.

الحالة	`error.code`	أين	متى
`422`	`WORKFLOW_ACTION_EXECUTION_NOT_ALLOWED` `Since 1.8`	تنفيذ إجراء سير عمل	الإجراء مخفي على المُدخل. (واستخدام هذا الرمز بدلًا من `403` مقصود.)
`422`	`CURRENT_STEP_ALREADY_OWNED` `Since 1.8`	أخذ ملكية خطوة سير عمل	للخطوة الحالية مالك بالفعل.
`422`	`ENTITY_ARCHIVE`	`DELETE /api/v1/entities/{id}`، أو تحديث كيان مؤرشف أو نقله	الكيان مؤرشف بالفعل، وكثيرًا ما ينتج ذلك عن نقرة مزدوجة أو إعادة محاولة.
`400`	`INVALID_POINTER`	أي مرشّح أو معايير تتضمّن قيمة مؤشِّر، مثل `…/entries/count?whereCriteria=`	المؤشِّر ليس دالة مؤشِّر معروفة، أو لا يتّبع الصيغة `@name` أو `@name()` أو `@name('<amount>')`. وتُرفض `@Today` و`@Me` و`@none` وأي صيغة أخرى.
`400`	`DATA_VIEW_INVALID_ALLOWED_OPERATIONS`	حفظ معرض	تحرير المُدخلات مفعّل، لكن لم تُضبط أي عملية تحرير مدعومة.
`400`	`MISSING_REQUIRED_PARAMETER`	`POST /api/v1/entities/{entity}/entries`، و`POST …/entries/draft`، و`/api/v1/views/{id}/entries`، و`…/entries/draft`	لا يتضمّن الطلب أي جسم.
`400`	`MISSING_REQUIRED_PARAMETER` `Since dev`	`PATCH /api/v1/entities/{entity}/views/{id}`	حفظ معرض دون أي جسم للطلب. أمّا الجسم الجزئي فمقبول.

الحالة	error.code	أين	متى
403	ACCESS_OBJECT_NO_AUDIENCE	GET /api/v2/access-object/{id}	لا يسمح المعرض أو الصفحة أو التقرير بوصول الزوّار ولا المستخدمين المسجَّلين، فلا يستطيع أحد فتحه.
404	QUERY_NOT_FOUND	حفظ استعلام أو معرض أو خلاصة أو معرض وكيل يشير إلى استعلام باسمه	الاستعلام المحدّد بالاسم غير موجود. (أمّا قراءة معرض مخزَّن لم يعد استعلامه موجودًا فلا تزال تُعيد 500.)
404	–	GET /api/v1/spaces/{id}	المساحة غير موجودة.
400	FILE_EXPIRED	عمليات الحفظ التي تُرفق بها ملفات مؤقّتة مرفوعة: المُدخلات، والكيانات، والمعارض، والصفحات، والتقارير، والتفاعلات	انتهت صلاحية ملف مؤقّت قبل الحفظ. ويسرد details القيم {fileId, fieldName?, path?, reason} لكل ملف. أعِد رفع الملف، ثم أعِد الحفظ.
503	STORAGE_SERVICE_UNAVAILABLE	عمليات الحفظ نفسها، وقراءات الملفات	تعذّر الوصول إلى تخزين الملفات أثناء عملية حفظ تضيف ملفًا جديدًا. ويمكن إعادة المحاولة.
كما ورد من الجزء الآخر Since 1.8	رمز الجزء الآخر	أي عملية تستدعي جزءًا آخر من المنصة	تحتفظ الاستجابة ذات الرمز 4xx الواردة من ذلك الجزء بحالتها وجسمها.

للاطّلاع على حالات الرفض الناتجة عن حدود الخطة، انظر حدود الخطة. وللاطّلاع على الفهرس الكامل وإرشادات إعادة المحاولة لكل رمز، انظر الأخطاء وإعادة المحاولة.

تجنّب الكتابة فوق تغيير غيرك Since 1.10

يحمل كل مُدخل updateIdentifier يتغيّر مع كل عملية كتابة، وتُعيده عمليات القراءة مع المُدخل. أرسله مع التغيير الذي تجريه، فتُرفض الكتابة بالاستجابة 409 UPDATE_CONFLICT إذا غيّر شخص آخر المُدخل منذ قراءتك له.

هذا الفحص اختياري لكل طلب على حدة؛ فإذا أغفلت الرمز، جرت الكتابة كالمعتاد. ولا تتوفّر صيغة لإرساله عبر ترويسة (لا If-Match ولا ETag):

العملية	أين ينتقل الرمز
PATCH /api/v1/entities/{entity}/entries/{id}	الخاصّية updateIdentifier في الجسم
PATCH /api/v2/entities/{entity}/entries/{id}	معامل الاستعلام updateIdentifier (لأن الجسم مصفوفة JSON Patch)
PUT /api/v1/views/{id}/entries/{entryId}	الخاصّية updateIdentifier في الجسم

أمّا المُدخل الذي لم يحصل على رمز بعد، فلا تُرفض الكتابة إليه لهذا السبب أبدًا؛ بل يحصل على رمز في تلك العملية. ولخطوات القراءة ثم الاحتفاظ بالرمز ثم إعادة إرساله، إلى جانب جسم استجابة التعارض، انظر كتابة البيانات.

> **قيود معروفة**
>
> تقتصر الحماية على العمليات الثلاث المذكورة أعلاه. ولا يرسل عميل الويب الخاص بالمنصة الرمز عند التحرير المباشر، كما لا تغيّره بعض مسارات الكتابة الداخلية؛ لذا لا تُكتشف التعديلات المتزامنة الناتجة عن تلك المسارات.

التقسيم إلى صفحات والترشيح

توجد صيغتان للترشيح، لكل جيل من العمليات صيغته. وتستخدم الاستعلامات المحفوظة، والإحصاءات، ومرشّحات المعارض، وشروط التفاعلات نموذجَ الترشيح نفسه.

v1: المرشّح الكلاسيكي

تقبل عمليات السرد والعدّ في `v1` كائن `filter` بوصفه معامل استعلام (`?filter={…}` ، بصيغة JSON أو مرمَّزًا بترميز URL):

وتقبل عمليات العدّ `where` :

المفتاح	الغرض
`where`	شرط المطابقة
`fields`	تضمين الخصائص أو استبعادها
`order`	الفرز: `"<field> ASC"` أو `"<field> DESC"` ، أو مصفوفة منها
`limit`	حجم الصفحة (1 على الأقل)
`offset` / `skip`	عدد الصفوف التي يجري تخطّيها (اسمان مترادفان)
`include`	تحميل المُدخلات المرتبطة

عوامل `where` : `eq` ، و `neq` ، و `gt` ، و `gte` ، و `lt` ، و `lte` ، و `inq` (ضمن)، و `nin` (ليس ضمن)، و `between` ، و `like` ، و `nlike` ، و `regexp` ، مع التجميع باستخدام `and` / `or` .

```
GET /api/v1/entities/employees/entries?filter={"where":{"value.active":true},"limit":50,"skip":100,"order":["value.nam
e ASC"]}
```

v2 وv3: مرشّح المعايير {#structured-criteria-filter-v2-v3}

تقبل عمليات `v2` و `v3` كائن `filterCriteria` (وتقبل عمليات العدّ `whereCriteria`) ومعامل البحث النصي الحرّ `search` . ويحتفظ `filterCriteria` بالمفاتيح `limit` و `skip` و `offset` و `order` و `fields` و `where` و `include` ، ويضيف شجرة `criteria` . وتتضمّن كل عقدة في الشجرة الحقول الآتية:

الحقل	ملاحظات
`field`	الاسم النموذجي للحقل. ولا يُدرج في عقد `And` / `Or` .
`operator`	إلزامي. أحد العوامل الآتية: `And` ، `Or` ، `Equal` ، `NotEqual` ، `LessThan` ، `LessThanOrEqual` ، `GreaterThan` ، `GreaterThanOrEqual` ، `In` ، `NotIn` ، `Contains` ، `DoesNotContain` ، `StartsWith` ، `DoesNotStartWith` ، `Like` ، `IsNotLike` ، `IsEmpty` ، `IsNotEmpty` ، `ElemMatch` ، `Type` ، `Under`
`fieldOperator`	دالة تجميع اختيارية تُطبَّق على الحقل أولًا: `Size` ، `Sum` ، `Avg` ، `Min` ، `Max` ، `First` ، `Last` ، `Any`
`args`	إلزامي. عقد فرعية (في `And` / `Or`) أو قيم: `{ "type": <value type>, "value": … }`

```
// GET /api/v2/entities/employees/entries?filterCriteria=…&search=cairo
{
  "limit": 50,
  "skip": 0,
  "order": ["value.hireDate DESC"],
  "criteria": {
    "operator": "And",
    "args": [
      { "field": "value.department", "operator": "Equal",
        "args": [ { "type": 0, "value": "HR" } ] },
      { "field": "value.salary", "operator": "GreaterThanOrEqual",
        "args": [ { "type": 1, "value": 5000 } ] }
    ]
  }
}
```

تحقّق من سلامة المرشّح المعقّد قبل حفظه باستخدام `POST /api/v1/queries/parser`، الذي يبيّن ما إذا كان التعبير سليم البنية. ويُعبَّر عن الأعداد والمجاميع والقيم الدنيا والقصوى والمتوسطات باستعلامات إحصائية. و`Since 1.8` تراعي عمليات القوائم الشجرية (`…/entries/tree`) أيضًا `where` في المرشّح. وترد قواعد مسارات الحقول في صيغ المعايير ومسارات الحقول، ويُشرح اختيار الصيغة في قراءة البيانات.

قواعد التقسيم إلى صفحات

- استخدم `limit` مع `skip` / `offset` للتقسيم إلى صفحات. ولا يُدعَم الترقيم بالعلامات (cursors).
- **لا يضع الخادم حدًّا أقصى لحجم الصفحة.** لذا اختر قيمة معقولة لـ `limit`.
- يتطلّب العدّ استدعاءً منفصلًا: `GET …/entries/count` (v1، `?where=`) أو `GET /api/v2/entities/{entity}/entries/count` (`?whereCriteria=` و`search`)، ويُعيد `{ "count": <n> }`.

معاملات القوائم الأخرى:

المعامل	ينطبق على	المعنى
`search`	v2، v3	بحث نصي حرّ في الحقول القابلة للبحث
`appliedQuickFilters`	السرد، والعدّ، ولوحة المهام	معرّفات المرشّحات السريعة المراد تطبيقها
`appliedTagsFilters`	v3، والشجرة، ولوحة المهام	معرّفات مرشّحات الوسوم؛ وتتضمّن الاستجابة خريطة `tags` مطابقة
`include_virtual`	السرد، والعدّ	تضمين المُدخلات الوهمية (المحسوبة)
`fields`	`GET …/entries/{id}`	إسقاط تُفصل عناصره بفواصل، مثل `?fields=value.name,value.email`؛ ويظلّ خاضعًا لقواعد الوصول إلى المعرض
`excludeResolved`	`GET …/entries/{id}`	تخطّي حلّ العلاقات والتصنيفات لتخفيف عملية القراءة

عمليات مفيدة

- **التغييرات المجمَّعة** `Since 1.4` – تقبل `PATCH /api/v1/entities/{entity}/bulk-update` و`…/bulk-upgrade` (وكذلك `v2`) قائمة صريحة من المعرّفات أو محدِّد «تحديد كل المطابق»، وتُعيد نتيجة لكل صف. و`Since 1.6` تتوفّر صيغة تُدرج العمل في الطابور ومسار ثانٍ، `PATCH /api/v2/entities/{entity}/bulk-update`. انظر كتابة البيانات.
- **التحقّق البشري** `Since 1.6` – قد يتطلّب إنشاء مُدخل اجتياز تحدّي تحقّق (للزوّار المجهولين).
- **الحقول المقنَّعة** `Since 1.8` – تُدرج استجابات قراءة المُدخلات الحقول التي لا يُسمح لك بقراءتها في `sensitiveMaskedFields`، مما يتيح لك عرضها مقفلة.
- **المُدخلات المجمَّعة** `Since 1.8` – في الكيان الذي يجمّع كيانات أخرى، تُستمدّ حالة القفل لكل حقل من المُدخل المصدري.

التفاعلات في وحدة `Since 1.8` – يسرد `GET /api/reaction/v1/reactions/module/{moduleId}` التفاعلات المعرَّفة على أي كيان ضمن وحدة. ويقبل `filter` اختياريًا (`where` و`fields` و`order` و`limit` و`skip` و`include`)، لتضييق النتائج بحسب `eventType` مثلًا. ويُدمج شرط `where` الذي تحدّده مع نطاق الوحدة، ولا يمكنك توسيع هذا النطاق. وإذا لم تتضمّن الوحدة أي كيانات، يُعيد الاستدعاء قائمة فارغة.

استدعاءات مساعدة صغيرة

الاستدعاء	ما يُعيده
`GET /api/v1/version`	إصدار الواجهة الخلفية
`GET /api/v1/icons`	الأيقونات المتاحة
`GET /api/v1/configs`	الإعدادات على مستوى المنظمة
`GET /api/v1/requests/{entity}/{id}/{mode}`	ما إذا كان يُسمح لك بتنفيذ `{mode}` على ذلك المُدخل

ويُشرح الشريط الجانبي للمعارض (`GET /api/v2/side-bar`) في الصفحات والقطع [بناء التطبيقات].

العمل طويل الأمد

تُعيد بعض العمليات معرّف المهمة فورًا؛ ثم تستطلع حالتها عبر استدعاء الحالة حتى تنتهي. ويمكنك، بدلًا من الاستطلاع، متابعة تقدّم المهمة مباشرةً عبر قناة الزمن الحقيقي.

العملية	البدء	الاستطلاع
مهمة خلفية عامة	`POST /api/v1/job/{messageId}` ← `202` (تُعيد `messageId`)	`GET /api/v1/job/{messageId}`
تصدير حزمة	`POST /api/packaging/v1/export` ← `202` { `jobId` }	`GET /api/packaging/v1/export/{jobId}/status`
استيراد حزمة	`POST /api/packaging/v1/import/{sessionId}/execute`	`GET /api/packaging/v1/import/{sessionId}/status`
إعادة بناء الاعتماديات كاملةً	`POST /api/dependency/v1/rebuild` ← `202` { `jobId` }	`GET /api/dependency/v1/rebuild/status/{jobId}`
نسخة احتياطية للقاطن `Since 1.10`	`POST /api/v1/admin/backups` ← `202`	`GET /api/v1/admin/backups/{backupId}`
استعادة القاطن `Since 1.10`	`POST /api/v1/admin/restores` ← `202`	`GET /api/v1/admin/restores/{restoreId}`
تصدير بيانات القاطن `Since 1.10`	`POST /api/v1/admin/exports` ← `202`	`GET /api/v1/admin/exports/{exportId}`

- لا يمكن تشغيل أكثر من عملية واحدة لإعادة بناء الاعتماديات كاملةً في الوقت نفسه؛ ويُجاب عن طلب العملية الثانية بـ `409` .
- يُجاب عن طلب استطلاع مهمة تصدير أو استيراد لمساحة أخرى بـ `403` .
- تَرِد بعض النتائج في صورة ملفات، لا بصيغة JSON: تصديرات البيانات (XLSX وPDF)، وCSS السمات، وملفات الترجمة، والخطوط؛ لذا تعامَل معها بوصفها تنزيلات.

وترد قيم الحالة، وسياسات الاحتفاظ والإلغاء لكل نوع من المهام في العمل طويل الأمد.

حدود الخطة `Since 1.10`

قد تفرض خطة القاطن حدودًا على المُدخلات والكيانات والمساحات والتخزين وعمليات تشغيل الأتمتة في الشهر. ولا يُطبَّق أي حدّ لم تُضبط قيمته.

ما يرفضه كل حدّ – تُعيد جميع حالات الرفض `403` ، مع القيم الخاصة بالقاطن في `details` :

`error.code`	يرفض	`details`
`ENTRY_CAP_EXCEEDED`	إنشاء مُدخل (مفرد، أو ضمن دفعة، أو مسودّة، أو عبر وكيل) متى بلغ المجموع الموزون الحدّ الصارم مع عدم السماح بالتجاوز. ولا تُرفض القراءة ولا التحرير ولا التصدير أبدًا.	`{cap, hardLimit, current}`

error.code	يرفض	details
ENTITY_CAP_EXCEEDED	إنشاء كيان يتجاوز الحدّ. ولا تُحتسب الكيانات الضمنية والأساسية.	{cap, current}
SPACE_CAP_EXCEEDED	إضافة مساحة تتجاوز الحدّ، بما في ذلك استعادة مساحة مؤرشفة. ولا تُحتسب المساحات المؤرشفة؛ أمّا المعطَّلة فتُحتسب. ولا تُرفض المساحة الافتراضية أبدًا.	{limit, current}
STORAGE_CAP_EXCEEDED	رفع ملف متى بلغ حجم البيانات المخزَّنة بالبايت maxStorageGb ، أو إذا كان الرفع سيؤدي إلى تجاوزه. وتُعيد خدمة تخزين الملفات الاستجابة وفق بنية الأخطاء الخاصة بها.	{capBytes, usedBytes}

ويُرفض تثبيت الحزمة قبل أن يبدأ إذا كان سيؤدي إلى تجاوز حدّ المُدخلات.

حدّ تشغيل الأتمتة الشهري ليس خطأ HTTP. فتشغيل التفاعل الذي يتجاوز الحدّ لا يتسبّب في فشل أي طلب، بل يُسجَّل بالحالة refused (الرمز REACTION_RUN_CAP_EXCEEDED ، مع limit و used و period)، ولا يُنفَّذ أيٌّ من إجراءاته. ويمكن العثور على هذه التشغيلات باستخدام outcome=refused في سجلّ التشغيل (قراءة البيانات)، أو باستخدام status=refused في سجلّات التفاعلات، أو من خلال refusedCount في قائمة التفاعلات. ولا تُقاس عمليات تشغيل الأتمتة في النشر وحيد القاطن.

النسخ الاحتياطي للقاطن واستعادته وتصدير بياناته Since 1.10

في النشر متعدّد القاطنين، يستطيع مسؤولو النظام لدى القاطن نفسه نسخ بياناته احتياطيًا واستعادتها وتصديرها عبر واجهة API (المرجع: BackupAdmin). ولا تتوفّر شاشة لهذه العمليات بعد.

- **من:** مسؤولو النظام لدى القاطن فقط.
- **أين:** في النشر متعدّد القاطنين فقط. وفي غيره، يُعاد لكل استدعاء 501 BACKUP_NOT_SUPPORTED_IN_SINGLE_MODE .
- **كيف:** يُعاد لكل طلب بدء 202 مع معرّف؛ ثم استخدم استدعاء GET المقابل للاستطلاع.

الاستدعاء	الجواب	ملاحظات
GET /api/v1/admin/backups	200 {items, quotas}	الأحدث أولًا. quotas : [{quota, used, limit, resetsAt}] لكلّ من backupsPerDay و restoresPerDay و dumpDownloadsPerDay (اليوم بحسب UTC)
GET /api/v1/admin/backups/{backupId}	200	{backupId, reason, state, createdAt, restorePoint?, sizeBytes?, expiresAt?}
POST /api/v1/admin/backups	202	ابدأ إنشاء نسخة احتياطية الآن. دون جسم للطلب.
POST /api/v1/admin/restores/preview	200	الجسم {backupId} أو {pointInTime} (ISO-8601). ويُعيد ما ستُجريه الاستعادة، والتحذيرات، و confirmationToken أحادي الاستخدام مع confirmationExpiresAt .
POST /api/v1/admin/restores	202	الجسم {backupId} أو {pointInTime} ، مع confirmationToken . ويدخل القاطن وضع الصيانة من هذه اللحظة.
GET /api/v1/admin/restores/{restoreId}	200	{restoreId, backupId, pointInTime?, state, step?, safetyBackupId?, startedAt, finishedAt?, error?, hint?, …} – استخدم هذا الاستدعاء للاستطلاع
POST /api/v1/admin/exports	202	ابدأ إنشاء أرشيف لتصدير بيانات القاطن. **حسابات المستخدمين غير مشمولة.**
GET /api/v1/admin/exports	200 {items}	تصديرات القاطن
GET /api/v1/admin/exports/{exportId}	200	{exportId, state, createdAt, finishedAt?, expiresAt?, sizeBytes?, error?, linksLeft, manifest}

الاستدعاء	الجواب	ملاحظات
`POST /api/v1/admin/exports/{exportId}/link`	`200 {url, expiresAt}`	رابط تنزيل أحادي الاستخدام على مضيف القاطن نفسه، صالح لمدة تصل إلى 10 دقائق؛ وبحدّ أقصى 5 روابط لكل تصدير

وأثناء الاستعادة، يُعيد القاطن `503 TENANT_IN_MAINTENANCE` لجميع الطلبات باستثناء طلبات `GET` على `/v1/admin/restores…` و `/v1/admin/backups…` و `/v1/admin/exports…`، مما يتيح مواصلة الاستطلاع.

الحالة	`error.code`
`400`	`VALIDATION_FAILED` (غياب كلٍّ من `backupId` و `pointInTime`، أو غياب `confirmationToken`)، و `CONFIRMATION_INVALID`، و `RESTORE_TARGET_INVALID`
`404`	`BACKUP_NOT_FOUND`، و `RESTORE_NOT_FOUND`، و `EXPORT_NOT_FOUND` – ويُعامَل معرّف يخصّ قاطنًا آخر على أنه غير موجود
`409`	`TENANT_IN_MAINTENANCE`، و `TENANT_NOT_BACKUPABLE`، و `TENANT_NOT_RESTORABLE`، و `TENANT_NOT_EXPORTABLE`، و `BACKUP_NOT_RESTORABLE`، و `POINT_IN_TIME_NOT_RESTORABLE`، و `RESTORE_IN_PROGRESS`، و `EXPORT_IN_PROGRESS`، و `EXPORT_NOT_READY`، و `EXPORT_LINKS_EXHAUSTED`
`429`	`BACKUP_QUOTA_EXCEEDED`، و `RESTORE_QUOTA_EXCEEDED`، و `EXPORT_QUOTA_EXCEEDED`، مع `Retry-After` الذي يحدّد المدّة المتبقية حتى تجديد الحصّة
`501`	`BACKUP_NOT_SUPPORTED_IN_SINGLE_MODE`
`502`	`BACKUP_ENGINE_FAILED`، و `EXPORT_LINK_FAILED`
`503`	`BACKUP_ENGINE_UNAVAILABLE`، و `EXPORT_ENGINE_UNAVAILABLE`، و `BACKUP_SERVICE_UNAVAILABLE`، مع `Retry-After: 60`

أمّا التفاصيل الخاصة بالمشغّل – الاحتفاظ والحصص اليومية – فترد في النسخ الاحتياطي للقاطن واستعادته وتصديره.

مجموعات أخرى من الـ API

إعدادات النظام

لا يتطلّب `GET /api/v1/system-settings/public` **أي بيانات اعتماد**. ويُعيد قيم إعدادات النظام الموسومة بأنها عامة للمساحة الحالية، بالشكل `{block: {field: value}}`. وقد يُرفض حفظ مُدخل إعدادات النظام بالحالة `422`: `SYSTEM_SETTINGS_SPACE_MISMATCH`، أو `SYSTEM_SETTINGS_TENANT_DEFAULT_ARCHIVE` (لا يمكن أرشفة المُدخل الخاص بالقاطن كله)، أو `SYSTEM_SETTINGS_DUPLICATE_PHONE_NUMBER` (يوجد مُدخل نشط آخر يستخدم رقم WhatsApp نفسه بالفعل). المرجع: SystemSettings. `Since 1.10`.

WhatsApp `Since 1.10`

الاستدعاء	من يحقّ له الاستدعاء	ملاحظات
`GET /api/reaction/v1/whatsapp/templates`	مستوى صلاحية العرض على النظام أو أعلى	القوالب المعتمدة لحساب WhatsApp التجاري الخاص بالمساحة، لإجراء **إرسال رسالة واتساب**: `[{name, language, category, bodyParameterCount}]`. وتُستبعد قوالب المصادقة. وتُعاد `409` `whatsapp_not_configured` حين لا يكون الحساب مُعدًّا؛ و `502` `whatsapp_provider_error` حين يرفض المزوّد الطلب (`details {metaCode, metaStatus}`). المرجع: WhatsAppTemplates.
`GET /api/logs/v1/whatsapp` و `GET /api/logs/v1/whatsapp/count`	كما في سجلّات البريد والرسائل القصيرة	سجلّ رسائل WhatsApp، مع `filter`. ولا يرد في أي مستند OpenAPI.
`GET` / `POST /api/logs/v1/whatsapp/webhook/{spaceId}`	عام – تستدعيه Meta	مصافحة الاشتراك من Meta (`GET`) والاستدعاء الراجع لحالة التسليم (`POST`).

ونقطة webhook عامة لأنّ Meta لا تستطيع تسجيل الدخول. وهي محميّة برمز التحقّق الخاص بالمساحة وبتوقيع:

- يجب أن يكون `spaceId` هو `ROOT` أو معرّف مساحة موجودة، وإلا تُعاد `404`.
- `GET`: يجب أن تكون قيمة `hub.mode` هي `subscribe` وأن يساوي `hub.verify_token` رمز التحقّق الخاص بالمساحة. وعند النجاح، تُعاد `200` مع قيمة `hub.challenge` وحدها كنصّ عادي؛ وفي غير ذلك تُعاد `403`.
- `POST`: يجب أن يتطابق جسم الطلب الخام مع التوقيع في ترويسة `X-Hub-Signature-256`، المنشأ باستخدام سرّ التطبيق الخاص بالمساحة. وتُعاد `401` عند غياب السرّ أو غياب التوقيع أو عدم صحّته؛ و`400` إذا كان JSON غير سليم البنية مع توقيع صالح؛ وفي غير ذلك تُعاد `200`. ولا تُسجَّل إلا حالات التسليم (`sent` و`delivered` و`read` و`failed`)؛ ولا تُسجَّل الرسائل الواردة.

وإعدادات WhatsApp هي حقول في كيان إعدادات النظام، تُحرَّر عبر محرّر المُدخلات أو عبر واجهة API. أمّا سلوك إجراء **إرسال رسالة واتساب** فترد تفاصيله في التفاعلات والأتمتة [بناء التطبيقات].

الحزم والمتجر

لا يتوفّر لمجال التحزيم (`/api/packaging`) مستند OpenAPI، لذا لا يرد في مرجع نقاط النهاية. وترد مساراته والجهات التي يحقّ لها استدعاء كلٍّ منها في الحزم والمتجر والمتجر داخل التطبيق. و`Since 1.10` أصبح يشمل أيضًا سجلّ الحزمة، وحذف إصدار غير منشور، وطلبات النشر الأولى والمجمَّعة (الموافقة والإرجاع والسحب)، وإعداد الناشر وإعادة تسميته، وقوائم الناشرين والموضوعات في المتجر، وفحص التثبيت، وتحديثات التثبيتات، وسجلّات التصدير (`GET /api/packaging/v1/exports`، و`GET /{id}/exports/…` `DELETE` – مع بقاء ملف التصدير عند حذف سجلّه؛ انظر سجلّات التصدير).

إعادة تسمية المعارض والصفحات المثبَّتة `Since 1.10`

يكون المعرض أو الصفحة المثبَّتة من حزمة للقراءة فقط عند استخدام `PATCH /api/v1/entities/{entity}/views/{id}` و`PATCH /api/v1/generic-views/{id}`، باستثناء النشر وإعادة التسمية المتاحة اعتبارًا من `Since 1.10`: إذ تُقبل `name` و`description` و`icon`، وتُحفَظ قيمها عند تحديث الحزمة. ويجب أن يتضمّن طلب إعادة التسمية `updateIdentifier` الحالي، وإلا أُعيدت `409 UPDATE_CONFLICT`. وتمسح `description: null` الوصف؛ أمّا `icon: null` فلا تمسح الأيقونة إلا مع معامل الاستعلام `clearIcon=true`. ويُرفض أي تغيير آخر.

ولا يمكن استدعاء مسارات الخدمات الداخلية للمنصة عبر بوّابة الـ API، ولا تُسرد هنا؛ لكن قد تظهر في آثار التتبّع.

المواصفات المقروءة آليًا

ينشر كل مجال في المنصة مستند OpenAPI 3 خاصًّا به؛ ولا يوجد مستند موحّد يجمع المجالات. إذا كان المشغّل لديك قد فعّل المستندات، فيمكنك الحصول عليها من بادئة المجال:

المجال	المستند	المستكشف
منصة البيانات الأساسية	`/api/openapi.json` (أو `.yaml`)	`/api/explorer`
أي مجال آخر	`/api/<area>/openapi.json`، مثل `/api/reaction/openapi.json`	`/api/<area>/explorer`

إليك ما ينبغي معرفته قبل توليد عميل من هذه المستندات:

- **تصف المستندات الغلاف.** `Since 1.8` يعلن كل مستند مخطّطات الغلاف وينصّ على قاعدته: ويصف مخطّط النجاح لكل عملية قيمة `result` الداخلية، فتتوافق الأنواع المولَّدة عندما يستخرج عميلك `result` من الغلاف. أمّا المستند المُصدَّر قبل 1.8 فيصف أجسامًا مجرّدة – فأعِد توليده.

- **بعض المجالات بلا مستند**: التحزيم، والسجلّات، والاعتماديات، والتكاملات. وترد مساراتها في هذه الصفحة وفي أدلّتها الخاصة.
- **العمليات الداخلية مستبعدة** من كل مستند.

للاطّلاع على طريقة توليد عميل وإبقائه محدَّثًا، راجع توليد عميل.

مستند لمعرض واحد `Since 1.10`

يمكن للمعرض نشر مستند OpenAPI 3.0 صغير خاص به. فعّل **وثيقة Swagger** في خيارات العرض للمعرض، ثم:

```
GET /api/v1/views/{viewId}/openapi.json
```

الجواب	المعنى
`200`	المستند (`application/json`)
`403` `VIEW_ACCESS_DENIED`	المعرض موجود، لكنك لا تستطيع فتحه
`404` `DATA_VIEW_NOT_EXIST`	المعرض غير موجود، أو كيانه مؤرشف

> ⚠ **الجواب غير مغلَّف، لكنّ العمليات التي يصفها مغلَّفة**
>
> حلّل هذه الاستجابة بوصفها مستند OpenAPI مجرّدًا — فلا `result` فيها. أمّا العمليات الخمس التي يصفها المستند **فتُجيب** داخل الغلاف، كسائر عمليات API، وينصّ المستند على ذلك.

يصف المستند خمس عمليات بالضبط، تُنفَّذ كلّها عبر المعرض:

العملية الموثّقة	ما تفعله
`GET /v1/v/{viewId}/v1/entities/{id}`	مخطّط الكيان
`GET /v1/v/{viewId}/v3/entities/{entity}/entries`	سرد المُدخلات
`GET /v1/v/{viewId}/v1/entities/{entity}/entries/{id}`	قراءة مُدخل واحد
`POST /v1/v/{viewId}/v1/entities/{entity}/entries`	إنشاء مُدخل
`PATCH /v1/v/{viewId}/v3/entities/{entity}/entries/{id}`	تحديث مُدخل

- أُدرج معرّف المعرض ومعرّف الكيان سلفًا في كل مسار، لذا يمكن نسخ الصفوف ولصقها مباشرةً؛ ولا يبقى سوى `{id}` الخاص بالمُدخل عنصرًا نائبًا.
- تتطلّب قراءة مخطّط الكيان بيانات اعتماد دائمًا، حتى في معرض مفتوح للزوّار. أمّا عمليات المُدخلات الأربع فتُتاح للزوّار المجهولين حين يسمح المعرض بذلك.
- يتبع مخطّط `value` للمُدخل الحقول المختارة وأنواعها في المعرض نفسه. ولا يُصفّى بحسب الحساسية: فالحقل الحسّاس الذي يعرضه المعرض يظهر باسمه ونوعه، ولا تظهر قيمته أبدًا.

عند تفعيل المفتاح، يستطيع أي مستخدم مسجَّل الدخول يمكنه فتح المعرض جلب مستنده؛ ويستطيع المستخدم الذي لديه مستوى مطوّر أو تعديل أو إدارة على النظام جلبه أيًّا كانت حالة المفتاح. ولا تعتمد على إمكان الوصول المجهول إلى المستند نفسه. وبوصفه مصدرًا لتوليد عميل، فإنّ هذا المستند أضيق نطاقًا وأكثر ثباتًا من المستند الكامل — انظر توليد عميل.

مرجع سريع

- **المضيف والمسارات:** مضيف القاطن نفسه؛ و `/api` لمنصة البيانات الأساسية، و `/api/<area>` للمجالات الأخرى؛ ويُحدَّد الإصدار في المسار؛ ويُستخدم JSON في الاتجاهين.
- **بيانات الاعتماد:** `x-app-token: <token>` أو `Authorization: Bearer <token>` . ولا تضبط ترويسات الهوية بنفسك أبدًا.
- **النجاح:** `{ "success": true, "result": … }` ، باستثناء `GET /api/v3/…` (حمولة مجرّدة `Since 1.10`)، والتنزيلات، ومستندات OpenAPI. انزع الغلاف عند وجود `success: true` .
- **الأخطاء:** `{ "success": false, "error": { "code", "message": {en, ar}, "traceId" } }` . حدّد التفرّع بناءً على `error.code` ؛ وقد يكون `500` دائمًا؛ و `503` مع `Retry-After` يعني الانتظار.
- **التقسيم إلى صفحات:** `limit` + `skip` / `offset` ؛ لا ترقيم بالعلامات؛ ولا حدّ أقصى يفرضه الخادم.
- **الترشيح:** `filter` / `where` (v1) أو `filterCriteria` / `whereCriteria` + `search` (v2، v3).
- **التزامن:** أعِد إرسال `updateIdentifier` الذي قرأته؛ وتُعاد `409 UPDATE_CONFLICT` عند عدم التطابق.
- **العمل طويل الأمد:** ابدأ، ثم استطلع الحالة باستدعاء مخصّص لها، أو تابع العمل مباشرةً.

> **للتعمّق أكثر**
>
> للمشغّلين: عمليات بوّابة الـ API · الحدّ الأمني · لمهندسي المنصة: واجهة HTTP البرمجية – التفاصيل الداخلية

المصادقة OAuthg

توجَّه هذه الصفحة إلى المطوّرين الذين يحتاجون إلى بيانات اعتماد تثق بها المنصة، سواء لتسجيل دخول المستخدمين إلى تطبيقهم أو لتمكين خادم أو سكربت من استدعاء الواجهة البرمجية. وتوضّح كيفية العثور على مزوّد OpenID الخاص بقاطنك، وتسجيل عميل، وتنفيذ تدفّق OAuth المناسب، واستدعاء الواجهة البرمجية بالرمز الناتج، والحفاظ على نشاط الجلسة.

يُثبت الرمز **مَن** يستدعي الواجهة البرمجية، أما الإجراءات المسموح بها للمستدعي فتُحدَّد على نحو منفصل، وفق أذونات ذلك المستخدم في القاطن.

اختر الطريقة

المستدعي	الطريقة
شخص يستخدم تطبيق الويب أو الجوال الخاص بك	رمز التخويل مع PKCE
خادمك حين يعمل باسمه	رمز API يملكه مستخدم تطبيق مخصّص. لا بيانات اعتماد العميل: فذلك الرمز بلا مستخدم، فلا يكاد يصل إلى شيء
سكربت أو خدمة أو أداة أتمتة تعمل دون تدخّل بشري	رمز API – وهو الخيار الأبسط
سكربت موثوق من الطرف الأول يحتفظ بكلمة مرور مستخدم	منحة كلمة المرور
زائر لم يسجّل الدخول، على صفحة عامة	رمز مجهول

وتشرح صفحة اختيار بيانات الاعتماد المفاضلة بين هذه الخيارات بمزيد من التفصيل.

اعثر على مزوّد OpenID الخاص بقاطنك

لكل قاطن مزوّد OpenID **خاص به**، له مُصدِره ومفاتيح توقيعه وعملاؤه؛ وتصل إليه عبر مضيف القاطن، فلا تمرّر معرّف القاطن مطلقًا.

1. اطلب من القاطن عنوان مضيفه، مثل `https://acme.example.com`.

2. اقرأ التهيئة العامة للقاطن، فهي تحدّد المزوّد (`authority`) وعميل الويب الذي تستخدمه بوابة القاطن نفسها:

```
curl -s https://acme.example.com/api/identity/v1/tenant-config
```

```
{
  "schemaVersion": 1,
  "appId": "<tenant-id>",
  "authority": "https://<authority-host>",
  "clientId": "<web-client-id>",
  "scope": "<scopes-the-web-client-requests>"
}
```

3. حمّل مستند الاكتشاف من المرجعية (authority)، واستخرج منه جميع نقاط النهاية، ولا تكتب المسارات ثابتةً في الرماز:

```
curl -s https://<authority-host>/.well-known/openid-configuration
```

تكتفي معظم مكتبات عملاء OpenID Connect / OAuth 2.0 بعنوان المرجعية، وتنفّذ الخطوة 3 بنفسها.

⚠ تحقّق من الرموز مقابل القاطن الذي تستدعيه

`Since 1.10` ، في النشر متعدّد القاطنين، يكون مُصدِر الرمز ومفاتيح التوقيع خاصَّين بالقاطن. فتحقّق من مطابقة `iss` في الرمز لمرجعية القاطن الذي تستدعيه، ولا تتحقّق منه مقابل مرجعية على مستوى الخلية مطلقًا. وبعد تدوير المفاتيح، يبقى المفتاح السابق في مجموعة مفاتيح القاطن 24 ساعة، لذا حدّث خبء المفاتيح لديك مرة في اليوم على الأقل. ويستجيب القاطن الذي لم تُهيَّأ هويته بعد بالحالة `503` .

اطلب تسجيل عميل

لا يتاح التسجيل الذاتي للعملاء. اطلب من مشغّل المنصة تسجيل عميل OAuth لتطبيقك، وحدّد له:

- هل هو عميل **عام** (تطبيق متصفح أو جوال، بلا سرّ، مع PKCE) أم عميل **سرّي** (خادم يستطيع الاحتفاظ بسرّ)؛
- أنواع المنح التي تحتاجها – رمز التخويل مع رموز التحديث، أو بيانات اعتماد العميل، أو منحة كلمة المرور؛
- **عناوين إعادة التوجيه** (redirect URIs) وعناوين إعادة التوجيه بعد تسجيل الخروج؛
- النطاقات التي تحتاجها، وهي عادةً `openid email profile offline_access` ؛
- مدة صلاحية أقصر أو أطول لرمز الوصول لهذا العميل، إن احتجت إلى ذلك.

يجب أن تتطابق عناوين إعادة التوجيه تمامًا، مع استثناء واحد: يجوز أن يمثّل حرف بدل واحد **نطاقًا فرعيًا كاملًا** في اسم المضيف، مثل `https://*.apps.example.com/auth-callback` . ويُرفض استخدام حرف البدل في المسار، أو استخدام أكثر من حرف بدل واحد، أو استخدامه لتمثيل جزء من تسمية مثل `*suffix.example.com` ، أو فوق لاحقة عامة مثل `*.co.uk` .

تتسلّم `client_id` ، وتتسلّم أيضًا `client_secret` إذا كان العميل سرّيًا. احتفظ بالسرّ على الخادم وحده. (يسجّل مشغّلو المنصة العملاء كما هو موضّح في خادم الهوية – عملاء OIDC.)

سجّل دخول المستخدمين (رمز التخويل مع PKCE)

هذا هو التدفّق المستخدم لأي تطبيق يسجّل فيه شخص دخوله. وتتولّى صفحة تسجيل الدخول المستضافة، نيابةً عنك، التعامل مع كلمات المرور، ومفاتيح المرور، وعوامل التحقق الثانية، ومزوّدي الهوية الخارجيين.

1. **أرسل المستخدم إلى نقطة نهاية التخويل** مع تحدّي PKCE:

```
https://<authority-host>/auth
  ?client_id=<client-id>
  &response_type=code
  &redirect_uri=https://app.example.com/auth-callback
  &scope=openid%20email%20profile%20offline_access
  &state=<random-state>
  &code_challenge=<S256-challenge>
  &code_challenge_method=S256
```

2. **تسلّم الرمز** على عنوان إعادة التوجيه الخاص بك، وتأكّد من تطابق `state` . ويظل الرمز صالحًا لمدة 10 دقائق.

3. **بادِله برموز** عند نقطة نهاية الرموز:

```
curl -s -X POST https://<authority-host>/token \
  -d grant_type=authorization_code \
  -d code=<code> \
  -d redirect_uri=https://app.example.com/auth-callback \
  -d client_id=<client-id> \
  -d code_verifier=<pkce-verifier>
```

ويُجري العميل السرّي المصادقة أيضًا باستخدام قيمة `client_secret` الخاصة به.

4. **استدعِ الواجهة البرمجية** برمز الوصول – راجع استدعاء الواجهة البرمجية.

يجب على العملاء العامّين استخدام PKCE. وما يزال التدفّق الضمني (implicit) مقبولًا للعملاء الأقدم؛ فلا تستخدمه في أي تطوير جديد.

ابدأ تسجيل الدخول بمزوّد محدّد `Since 1.10`

إذا كان القاطن يتيح تسجيل الدخول الخارجي (مثل مزوّد هوية وطني أو Microsoft Entra ID)، فأضف `idp_hint=<provider` إلى طلب التخويل لتتخطّى صفحة تسجيل الدخول وتنتقل مباشرةً إلى ذلك المزوّد. واستخدم الاسم الفريد للمزوّد، لا `name>` تسميته المعروضة؛ واطلب هذا الاسم من مدير القاطن. ويُتجاهَل الاسم غير المعروف أو غير النشط، فتظهر صفحة تسجيل الدخول المعتادة.

وحين يتيح نطاق مزوّدًا خارجيًا واحدًا فقط دون سواه، تُتخطّى صفحة تسجيل الدخول لجميع العملاء افتراضيًا. فلا تعتمد على ظهور صفحة تسجيل الدخول (مثلًا، لعرض رابط **إلغاء** فيها) في مثل هذا النطاق.

حين يُلغي المستخدم `Since 1.4`

إذا اختار المستخدم **إلغاء** في صفحة تسجيل الدخول، أو تركها دون تفاعل لمدة 10 دقائق، يتسلّم عنوان إعادة التوجيه الخاص بك `error=access_denied` مع قيمة `state` التي أرسلتها. فتعامل مع ذلك على أنه «رفض المستخدم»، لا على أنه إخفاق – كأن تعيد المستخدم إلى صفحتك الرئيسية.

رموز التحديث

تحصل على رمز تحديث حين تطلب النطاق `offline_access`، أو حين يكون عميلك عميل ويب عامًا. جدّد الرموز قبل انتهاء صلاحية رمز الوصول:

```
curl -s -X POST https://<authority-host>/token \
  -d grant_type=refresh_token \
  -d refresh_token=<refresh-token> \
  -d client_id=<client-id>
```

- يظل رمز التحديث صالحًا لمدة 14 يومًا، ويمكن تجديده لمدة إجمالية لا تتجاوز سنة واحدة.
- رموز التحديث **تُدوَّر**: فقد تتضمّن الاستجابة رمز تحديث جديدًا. فاحفظ دائمًا آخر رمز تسلّمته؛ إذ قد يتوقّف الرمز القديم عن العمل.
- يحتفظ رمز التحديث المُدوَّر لعميل الويب العام بما تبقّى من مدة صلاحية الرمز الذي حلّ محلّه، فلا يمكن تمديد جلسة المتصفح إلى ما لا نهاية.

أما كيفية تصميم عميل يحافظ على نشاط الجلسات رغم انتهاء الصلاحية والإبطال فموضّحة في دورة حياة الرمز.

من خادم إلى خادم: بيانات اعتماد العميل

يستطيع العميل السرّي الحصول على رمز يعمل باسم العميل نفسه، دون مستخدم:

```
curl -s -X POST https://<authority-host>/token \
  -u "<client-id>:<client-secret>" \
  -d grant_type=client_credentials
```

أرسل بيانات اعتماد العميل بالطريقة التي سُجّل بها عميلك – باستخدام مصادقة HTTP Basic كما في المثال، أو باستخدام `client_id` و`client_secret` بوصفهما حقلَي نموذج. ويظل الرمز صالحًا لمدة 10 دقائق، ما لم يُسجَّل عميلك بمدة صلاحية أخرى. ولا يرتبط الرمز بجلسة، لذا اطلب رمزًا جديدًا عند انتهاء صلاحيته.

ما يصل إليه الرمز: لا شيء تقريبًا. فهو يعرّف العميل لا مستخدمًا، فلا تمرّر البوابة معرّف مستخدم، وتعامل المنصة المستدعي بملف تعريف بلا أذونات. فيُصادَق الطلب، ثم تعود كل قراءة تقريبًا مرفوضةً أو فارغة. وعلى مقبس الوقت الفعلي يتصل الرمز، لكن الاشتراك في الإشعارات يُرفض، لأنه يحتاج إلى مستخدم. فإن كان خادمك يعمل باسمه، فاستخدم بدلًا منه رمز API يملكه مستخدم تطبيق مخصّص. وتشرح صفحة اختيار بيانات الاعتماد السبب.

السكربتات الموثوقة: منحة كلمة المرور

يجوز للعميل الموثوق من الطرف الأول أن يقدّم اسم المستخدم وكلمة مروره مباشرةً للحصول على رمز:

```
curl -s -X POST https://<authority-host>/token \
  -d grant_type=password \
  -d username=<username> \
  -d password=<password> \
  -d scope="openid email profile offline_access" \
  -d client_id=<client-id> \
  -d client_secret=<client-secret>
```

وتتضمّن الاستجابة `access_token` و`expires_in` و`token_type: Bearer` و`scope`، وكذلك `refresh_token` حين يُسمح به.

الإجابة	المعنى
`400 invalid_grant`، "invalid credentials provided"	اسم المستخدم أو كلمة المرور غير صحيح. ويؤدّي تكرار الإخفاق إلى قفل الحساب لمدة متزايدة.
`400 invalid_grant`، "account is locked"	الحساب مقفل. انتظر، أو اطلب المساعدة من أحد المديرين.
`400 invalid_scope`	طلبت نطاقًا غير مسموح به للعميل.

لا تستخدم هذه المنحة إلا حين يستطيع السكربت الاحتفاظ بكلمة المرور بأمان. فهي تتجاوز عوامل التحقق الثانية وتسجيل الدخول الخارجي، لذا فضّل تدفّق رمز التخويل كلما شارك شخص في عملية تسجيل الدخول.

استدعِ الواجهة البرمجية

أرسل رمز الوصول مع كل طلب إلى مضيف القاطن:

```
curl -s https://acme.example.com/api/v2/entities/<entity>/entries/count \
  -H "Authorization: Bearer <access-token>"
```

- الحالة `401` مع `AUTHENTICATION_ERROR` تعني أن الرمز مفقود أو منتهي الصلاحية أو لم يعد نشطًا: جدّده، أو أعد تسجيل دخول المستخدم.

- للتحديثات الحيّة عبر socket، استخدم رمز الوصول **نفسه**؛ راجع الزمن الحقيقي.
- إذا تضمّن الطلب ترويسة `x-app-token`، فتُستخدم قيمتها بدلًا من رمز الحامل (bearer).

أعمار الرموز

الرمز	العمر
رمز الوصول	ساعة واحدة، ما لم يُسجَّل عميلك بمدة صلاحية أخرى
رمز بيانات اعتماد العميل	10 دقائق، ما لم يُسجَّل عميلك بمدة صلاحية أخرى
رمز التخويل	10 دقائق
رمز الهوية (ID token)	ساعة واحدة
رمز التحديث	14 يومًا (رموز العملاء العامّين المُدوَّرة تحتفظ بما تبقّى من مدة صلاحيتها)
المنحة وجلسة تسجيل الدخول	14 يومًا
الرمز المجهول	10 دقائق

النطاقات والادعاءات

النطاق	الادعاءات الممنوحة
`openid`	`sub`
`email`	`email`، `email_verified`
`profile`	`preferred_username`
`offline_access`	رمز تحديث

وتُمنح أيضًا الادعاءات المخصّصة للمستخدم حين يعرّفها القاطن. ويمثّل `sub` معرّف المستخدم في القاطن.

الزوّار المجهولون `Since 1.10`

تستطيع صفحة المتصفح التي تعرض محتوى عامًا لزوّار لم يسجّلوا الدخول الحصول على **رمز مجهول** دون أي بيانات اعتماد:

```
curl -s -X POST https://acme.example.com/api/identity/v1/anonymous/token
```

```
{ "access_token": "<token>", "token_type": "Bearer", "expires_in": 600 }
```

لا يحمل الرمز إلا النطاق `anonymous`، فلا يتيح الوصول إلا إلى ما يسمح به القاطن للزوّار. ولا تضع سرّ عميل في رماز المتصفح مطلقًا؛ إذ يجب نقل التكاملات السابقة التي كانت تحصل على رمز مجهول باستخدام سرّ عميل نظام إلى نقطة النهاية هذه، لأن عميل النظام ذاك قد لا يكون موجودًا.

والتهيئة الخاصة بالقاطن الموضّحة في اعثر على مزوّد OpenID الخاص بقاطنك متاحة للعامة أيضًا، لذا يستطيع تطبيق المتصفح بدء العمل دون أي إعدادات خاصة بالنشر.

تحقّق من الرمز في خدمتك

إذا أنشأت خدمة تتسلّم رموز الوصول الصادرة عن المنصة، فلا تفكّ ترميزها بنفسك، بل تحقّق منها لدى المزوّد: استدعِ **نقطة نهاية الفحص (introspection)** المذكورة في مستند الاكتشاف باستخدام بيانات اعتماد عميلك:

```
curl -s -X POST https://<authority-host>/token/introspection \
  -u "<client-id>:<client-secret>" \
  -d token=<access-token>
```

(استخدم حقول النموذج بدلًا من مصادقة HTTP Basic إذا كان عميلك مسجّلًا بهذه الطريقة.) ولا تقبل الرمز إلا إذا تضمّنت الاستجابة `"active": true`، واستخدم قيمة `sub` فيها معرّفًا للمستخدم. وتحقّق من `exp` و `client_id` وفق متطلّبات خدمتك.

تسجيل الخروج والإبطال

- **تسجيل الخروج:** أرسل المستخدم إلى نقطة نهاية إنهاء الجلسة لدى المزوّد (من مستند الاكتشاف) مع `post_logout_redirect_uri` المسجّل لديك. وإذا كان المستخدم قد سجّل الدخول عبر مزوّد خارجي يدعم ذلك، فيُسجَّل خروجه من ذلك المزوّد أيضًا.
- **إبطال رمز:** استدعِ نقطة نهاية الإبطال (من مستند الاكتشاف) باستخدام الرمز وبيانات اعتماد عميلك.
- **تسجيل الخروج عبر القناة الخلفية** `Since 1.6`: إذا كان عميلك مسجّلًا بعنوان URI لتسجيل الخروج عبر القناة الخلفية، فيُخطره المزوّد عند انتهاء الجلسة.

العمل نيابةً عن مستخدم آخر

يستطيع الشخص العمل نيابةً عن زميله خلال فترة مأذون بها – وهذا هو **التفويض**، الذي يُعدّه المدير كما هو موضّح في التفويض [الإدارة]. ولاستخدام تفويض نشط في استدعاء، أضف معرّفه إلى الترويسة `X-Delegation-Id`. ويُسجَّل الإجراء على أنه نُفّذ لصالح الزميل وعلى يد الشخص، مما يحافظ على دقة السجلّ التاريخي. وخارج فترة التفويض، يُرفض الطلب بالحالة `403` `DELEGATION_EXPIRED`. والتفويض مخصّص لأشخاص ينوب بعضهم عن بعض، لا للتكاملات الآلية.

رموز API

للسكربت أو الخدمة التي لا تستطيع تنفيذ تدفّق OAuth، يُعدّ استخدام **رمز API** المُرسَل في الترويسة `x-app-token` الخيار الأبسط. فهو يعمل باسم مستخدم-تطبيق يُنشئه مسؤول القاطن – راجع إنشاء رمز API لتكامل [الإدارة › أنشئ رمز API لعملية تكامل] – واستخدمه كما هو موضّح في استخدام رمز API.

من المفيد معرفته

- **تتولّى صفحة تسجيل الدخول المستضافة عوامل التحقق الثانية والمزوّدين الخارجيين.** بحسب إعدادات القاطن، قد يُطلب من المستخدمين إدخال رمز من تطبيق مصادقة، أو رمز عبر الرسائل القصيرة أو البريد الإلكتروني أو `Since` `1.10` واتساب، أو استخدام مفتاح مرور. استخدم تدفّق رمز التخويل ودع الصفحة تعرض ما يتطلّبه القاطن، بدلًا من إنشاء صفحة تسجيل دخول خاصة بك.
- **القاطن قيد الصيانة** `Since 1.10` يستجيب بالحالة `503 TENANT_IN_MAINTENANCE` مع `Retry-After` لاستدعاءات تسجيل الدخول والرموز والتجديد، مثلًا أثناء استعادته من نسخة احتياطية. انتظر وأعد المحاولة، ولا تسجّل خروج المستخدم. راجع استدعاء واجهة HTTP البرمجية.
- **الرموز والقاطن.** يرتبط رمز الوصول الصادر عن تدفّق رمز التخويل أو تدفّق التجديد بمضيف تسجيل الدخول الخاص بالقاطن، لذا لا يمكن إعادة استخدامه مع قاطن آخر.

للتعمّق أكثر

للمديرين: الهوية ومزوّدو الهوية [الإدارة › طرق تسجيل الدخول والدخول الموحّد] · لمشغّلي المنصة: خادم الهوية · عمليات البوّابة – المصادقة · لمهندسي المنصة: الهوية وOAuth ورموز API – التفاصيل الداخلية

الزمن الحقيقي: الأحداث الحيّة والمكالمات

هذه الصفحة موجّهة إلى المطوّرين الذين يريدون تلقّي التغييرات عبر الدفع بدلًا من الاستقصاء، سواء لإبقاء مُدخل مفتوح محدّثًا، أو إظهار الإشعارات فور وصولها، أو متابعة مهمّة بدؤوها عبر HTTP. وتغطّي أيضًا رموز المكالمات الصوتية والمرئية.

القناة الحيّة **دفعٌ فقط، وبأفضل جهد**. فأنت تشترك وتتلقّى الأحداث؛ أما جميع عمليات الكتابة والقراءة الموثوقة، فتُجرى دائمًا عبر واجهة HTTP البرمجية.

الاتصال

تعمل القناة ببروتوكول **Socket.IO v4**، ولا يستطيع عميل WebSocket عادي الاتصال بها. استخدم مكتبة عميل Socket.IO v4، واتصل **بمضيف القاطن نفسه** عبر مسار Socket.IO الافتراضي:

```
import { io } from 'socket.io-client';

const socket = io('https://acme.example.com', {
  transports: ['websocket', 'polling'],
  // A function, so every reconnect sends a fresh token
  auth: (cb) => cb({ token: getAccessToken() }),
});

socket.on('connect_error', (err) => console.error('socket refused:', err.message));
```

- **بيانات الاعتماد هي رمز وصول OAuth** – وهو الرمز نفسه الذي تستخدمه لاستدعاءات HTTP – أو رمز مجهول في الصفحات العامة. **ولا يستطيع رمز API فتح مقبس (socket)**: فالخدمة التي تستدعي الواجهة البرمجية باستخدام `x-app-token` تحتاج إلى رمز OAuth لمستخدم كي تتصل بالمقبس. وللخدمة يعني ذلك منحة كلمة المرور لحساب خدمة مخصّص، ويجب أن يكون مستخدمًا عاديًا: فمستخدم التطبيق لا كلمة مرور له ولا يستطيع تسجيل الدخول. ولا يكفي رمز بيانات اعتماد العميل: فهو يفتح المقبس، لكنه بلا مستخدم، فيُرفض `subscribeToNotifications`.
- **تُرفض المصافحة التي لا تحمل رمزًا، وكذلك** المصافحة التي تحمل رمزًا غير صالح أو غير نشط (`Invalid token`).
- **يُفحص الرمز مرة واحدة، عند المصافحة.** كل إعادة اتصال تتطلّب مصافحة جديدة؛ لذا قدّم رمزًا جديدًا في كل مرة، كما في المثال.
- **اتصل بمضيف القاطن نفسه** `Since 1.10`. في بيئة نشر متعدّدة القاطنين، يتبع المقبس القاطن المرتبط بالمضيف الذي تتصل به. ويُرفض الاتصال بمضيف لا يطابق أي قاطن بالرسالة `Tenant not resolved`، ولا يُقبل رمز صادر عن قاطن آخر.

وتسري القواعد نفسها على العامل الذي يعمل دون واجهة وعلى عملية الخادم الخلفي. أما المتطلّبات الإضافية لهما (بيانات اعتماد ثانية، ومطابقة الحالة مع الحالة الفعلية)، فترد في الاشتراك في الأحداث.

الاشتراك

لا يتلقّى المقبس الجديد شيئًا حتى تشترك. ولكل اشتراك إقرار: **انتظره** قبل أن تفترض نجاح الاشتراك، لأن بعض محاولات الانضمام الفاشلة لا تُبلّغ عن أي خطأ.

ما تُرسله	الحمولة	الإقرار	ما تتلقّاه بعد ذلك
`subscribeToNotifications`	لا شيء – تُستمدّ هويتك من الرمز	`subscribedToNotifications {userId}`	إشعاراتك وأحداث اكتمال مهامّك
`entries/subscribe-to-entry` `Since 1.4`	`{entityId, entryId}`	`subscribedToEntry {entityId, entryId}`	التغييرات على ذلك المُدخل
`entries/unsubscribe-to-entry` `Since 1.4`	`{entityId, entryId}`	`unsubscribedFromEntry {entityId, entryId}`	–
`Since` `subscribeToJobExecution` `1.6`	`{messageId}`	`subscribedToJobExecution {messageId}`	اكتمال المهمّة
`unsubscribeFromJobExecution` `Since 1.6`	`{messageId}`	`unsubscribedFromJobExecution {messageId}`	–

يؤدّي غياب `messageId` أو `entityId` أو `entryId` إلى إطلاق حدث `error` (`{message}`) بدلًا من الاشتراك. كما يؤدّي استدعاء `subscribeToNotifications` على مقبس لم يُحدَّد المستخدم المرتبط به إلى إطلاق `error` وقطع الاتصال.

الأحداث التي تتلقّاها {#server-client-events}

الحدث	يصل حين تكون مشتركًا في	المعنى
`notification`	الإشعارات	أُنشئ إشعار لك؛ والحمولة هي سجلّ الإشعار.
`entries/value-updated`	المُدخل	تغيّرت قيم الحقول. و `Since 1.10` قد تحمل الحمولة قيمة `updateIdentifier` للمُدخل.
`entries/workflow-updated`	المُدخل	حدث تغيير في سير العمل: بلوغ خطوة أو مغادرتها، أو تنفيذ إجراء، أو تغيّر المالك، أو إضافة خطوة مخصّصة. ويحمل الحدث `eventName` .
`entries/comments-updated`	المُدخل	أُضيف تعليق أو عُدّل. ويحمل الحدث `eventName` .
`entries/archived` `Since 1.6`	المُدخل	أُرشف المُدخل.
`entries/restored` `Since 1.6`	المُدخل	استُعيد المُدخل.
`entries/virtual-entry-created` `Since 1.8`	المُدخل المصدر	أُنشئ منه مُدخل وهمي.
`entries/virtual-entry-updated` `Since 1.8`	المُدخل المصدر	حُدّث مُدخل وهمي أُنشئ منه.
`osos/infrastructure/job-executed`	المهمّة، **و**الإشعارات	انتهت مهمّة خلفية. وتحمل الحمولة `messages` و `result` .
`icons-updated`	– (كل مقابس القاطن)	تغيّرت مجموعة الأيقونات.
`request-entry-modified`	– (كل مقابس القاطن)	أُنشئ طلب أو حُدّث أو أُرشف: `{requestId, originalEntryReference, operation}` .
`Since` `packages/changed` `1.10`	– (كل مقابس القاطن)	تغيّرت حالة حزمة في المراجعة أو المتجر: `{ $name, $version: 1, packageId }` . أعد قراءة الحزمة عبر HTTP.
`error`	–	كان استدعاء الاشتراك أو إلغاء الاشتراك غير صالح.

وصفات

إبقاء مُدخل مفتوح محدّثًا

```
socket.emit('entries/subscribe-to-entry', { entityId, entryId });
socket.on('entries/value-updated',    (e) => refreshFields(e));
socket.on('entries/workflow-updated', (e) => refreshWorkflow(e));
socket.on('entries/comments-updated', (e) => refreshComments(e));

// when the screen closes:
socket.emit('entries/unsubscribe-to-entry', { entityId, entryId });
```

التنبيه حين يحفظ شخص آخر قبلك `Since 1.10`

قد يحمل `entries/value-updated` قيمة `updateIdentifier` الحالية للمُدخل. قارنها بالقيمة التي قرأتها:

- **لا رمز بعد، أو الرمز نفسه** – لم يتغيّر شيء بالنسبة إليك.
- **رمز مختلف** – حفظ شخص آخر بعد آخر قراءة أجريتها. أبلغ المستخدم، واعرض عليه إعادة التحميل، وامنع المحرّر المفتوح من الكتابة فوق النسخة الأحدث عند الحفظ.

وراعِ الأمرين الآتيين:

- هذا تنبيه لإحاطة المستخدم، وليس آلية الحماية نفسها. فالعميل الذي فاته الحدث يتلقّى مع ذلك `409` `UPDATE_CONFLICT` عند محاولة الكتابة نفسها.
- تؤدّي عملية الحفظ التي تجريها إلى عدّة عمليات كتابة لاحقة (الفهرسة، والأتمتة)، لكلٍّ منها رمز جديد. لذا تجاهل المقارنة ما دامت عملية الحفظ التي تجريها قيد التنفيذ، وإلا وجّهتَ تنبيهًا إلى الشخص الذي حفظ للتوّ.

متابعة مهمّة بدأتها عبر HTTP

```
// An HTTP call answered with a messageId
socket.emit('subscribeToJobExecution', { messageId });

socket.on('osos/infrastructure/job-executed', (evt) => {
  if (evt.messageId !== messageId) return;   // may also arrive via notifications
  confirmOverHttp(messageId);                // the status call is authoritative
});
```

اشترك فور بدء المهمّة، فالحدث الذي يقع قبل اشتراكك لا يُعاد بثّه. وتحقّق من الحالة النهائية باستدعاء حالة المهمّة؛ انظر العمل طويل الأمد.

قواعد التسليم

- **بأفضل جهد، ودون إعادة بثّ.** لا تُرسل أحداث المُدخلات إلا ما دام هناك مشترك، ولا يُحفظ شيء لعميل منقطع. لذا، بعد كل اتصال أو إعادة اتصال، اشترك من جديد وأعد قراءة البيانات التي تعرضها عبر HTTP.
- **الإشعارات لا تُرشَّح حسب المساحة.** تتلقّى إشعارات كل مساحة تنتمي إليها. و`Since 1.10` يُخفي عميل الويب في المنصّة إشعارات المساحات الأخرى بنفسه؛ أما العميل المخصّص، فعليه ترشيحها بحسب قيمة `value.space` في الإشعار.

- **الإجراءات المفوَّضة تُشعِر شخصين.** حين يصدر إشعار عن إجراء نُفّذ نيابةً عن شخص، يتلقّاه كلٌّ من المستلم والشخص الذي نفّذ الإجراء.
- **إلغاء الاشتراك سريعًا قد يحجب أحداث المُدخلات الوهمية.** تُرسل هذه الأحداث بعد نحو ثانية من التغيير، ولا تُرسل إلا إذا كنت لا تزال مشتركًا.

المكالمات الصوتية والمرئية

تُصدر المنصّة رموزًا لقنوات الصوت والصورة التي تشغّلها Agora، ولا تنقل الوسائط بنفسها. احصل على رمز عبر الواجهة البرمجية باستخدام بيانات اعتمادك المعتادة، ثم انضمّ إلى القناة باستخدام حزمة تطوير عميل Agora:

```
curl -s "https://acme.example.com/api/rtc/<channel>/publisher?expiry=3600" \
  -H "Authorization: Bearer <access-token>"
```

```
{ "success": true, "result": { "rtcToken": "<token>", "userId": "<your-user-id>" } }
```

الجزء	القيم
`<channel>`	اسم القناة. مطلوب.
الدور	`publisher` (يرسل الصوت والصورة) أو `audience` (يستقبل فقط). وتؤدّي أي قيمة أخرى إلى إرجاع `400`.
`expiry`	مدّة صلاحية الرمز بالثواني. اختياري، والقيمة الافتراضية `3600`.

يُصدَر الرمز لمعرّف المستخدم الخاص بك **أنت**؛ ولا يمكنك طلب رمز لمستخدم آخر. أما معرّف تطبيق Agora الذي يحتاج إليه العميل، فيُؤخذ من إعدادات المشغّل لديك.

> **للتعمّق أكثر**
>
> للعملاء الذين يعملون دون واجهة: الاشتراك في الأحداث · للمشغّلين: مرجع التكوين — web-socket-service · لمهندسي المنصّة: الزمن الحقيقي — التفاصيل الداخلية

الويب هوك والتكاملات

هذه الصفحة موجّهة إلى المطوّرين الذين يربطون أسس بأنظمة أخرى عبر الأحداث: لاستدعاء نظامك عند وقوع حدث في قاطن، وربط Zapier أو n8n أو الأدوات المماثلة، وتمكين تلك الأدوات من تشغيل الأتمتة في المنصّة.

مثال: عند اعتماد طلب شراء، أبلغ النظام المالي. يضيف باني التطبيق تفاعلًا إلى كيان طلبات الشراء يُشغَّل عند الاعتماد، ويستخدم الإجراء **استدعاء API** لإرسال المبلغ ومقدّم الطلب إلى نقطة النهاية لديك. أما مسؤوليتك بوصفك مطوّر تكامل فتشمل نقطة النهاية المستقبِلة والعقد المتّفق عليه.

وتتناول صفحتان مستقلتان مجالين مرتبطين: قراءة بيانات المنصّة من Grafana (لوحات معلومات Grafana `Since 1.6`)، وتشغيل رمازك الخاص داخل عملية (الوظائف).

اختر النهج

تريد أن…	استخدم
تستدعي نظامك حين يتغيّر مُدخل أو يتقدّم سير عمل	الإجراء «استدعاء API»
تشغّل رمازًا كتبته باستخدام بيانات المنصّة	الإجراء **تنفيذ وظيفة** – انظر الوظائف
تزوّد مهامّ Zapier (Zaps) بأحداث المنصّة	إشعار Zapier
تمكّن مهمّة Zap أو أداة أخرى من تنفيذ إجراء في المنصّة	إجراءات Zapier، أو واجهة HTTP البرمجية باستخدام رمز
تُبقي شاشةً محدّثة آنيًا	الزمن الحقيقي
تفحص الطلبات الواردة أو تعيد كتابتها قبل معالجتها	معترضات الطلبات

استدعِ نظامك حين يقع حدث

الاستدعاءات الصادرة **إجراءات تفاعل**. ولا توجد شاشة منفصلة للاشتراك في الويب هوك؛ بل يضيف باني التطبيق تفاعلًا إلى كيان، ويختار الحدث الذي يشغّله، ثم يُلحق به أحد هذين الإجراءين:

الإجراء	ما يفعله
استدعاء API	يرسل طلب HTTP – `GET` أو `POST` أو `PATCH` أو `PUT` أو `DELETE` – بالعنوان والترويسات والجسم التي يحدّدها الباني، مع تعبئتها بقيم من المُدخل. وتُعدّ الاستجابة `2xx` نجاحًا، وتُحفظ الاستجابة في السجل التاريخي للتشغيل.
إشعار Zapier	يرسل المُدخل إلى روابط Zapier (hooks) المشتركة (انظر أدناه).

إعداد التفاعل مهمّة باني التطبيق – انظر التفاعلات والأتمتة [بناء التطبيقات]. أما العقد الذي تتّفق عليه مع الباني فيحدّد الأسلوب والعنوان والترويسات وحقول الجسم والاستجابة التي تعيدها نقطة النهاية لديك.

صمّم نقطة النهاية المستقبِلة

- **تحقّق من هوية المستدعي بنفسك.** لا توقّع المنصّة الطلبات الصادرة. لذا اطلب من الباني إرسال سرّ تصدره أنت – في ترويسة `Authorization` أو `X-Api-Key` مثلًا – وتحقّق منه في كل طلب.
- **أبقِ السرّ خارج السجل التاريخي.** يُحفظ طلب كل تشغيل واستجابته في السجل التاريخي للتفاعل. وتُخزَّن القيم المرسلة في ترويسات بيانات الاعتماد المعروفة (`Authorization` و`Cookie` و`X-Api-Key`)، والقيم المقروءة من

الحقول الحسّاسة، وأي قيمة يضع عليها الباني علامة **محجوب في السجل التاريخي** بصيغة `****` `Since 1.6` – انظر حجب الأسرار في السجل التاريخي للتفاعلات [بناء التطبيقات › التفاعلات والأتمتة].

- **أجب سريعًا بـ `2xx`.** `Since 1.8` يجب أن يكتمل تبادل الطلب والاستجابة خلال 30 ثانية افتراضيًا، وتُسجَّل أي استجابة غير `2xx` بوصفها إخفاقًا.
- **وفّر عنوانًا عامًا.** تُرفض الاستدعاءات إلى الشبكات الخاصة والمحجوزة (نطاقات IP الداخلية، وعناوين البيانات الوصفية السحابية)، ما لم يسمح مشغّل المنصّة بتلك الوجهة. ولا يُتتبَّع أكثر من 5 عمليات إعادة توجيه، ويقتصر حجم كلّ من جسم الطلب وجسم الاستجابة على 10 MiB. و `Since 1.10` تُحجز بعض أسماء الترويسات للمنصّة وتُرفض عند حفظ التفاعل (`400 RESERVED_HEADER_NAME`): انظر استدعاء API: ما تسمح به المنصّة [بناء التطبيقات › التفاعلات والأتمتة].
- **اجعلها متساوية الأثر عند التكرار.** تعامل مع كل استدعاء على أنه إشعار بأن «هذا السجلّ تغيّر»، وأعد قراءة المُدخل عبر واجهة HTTP البرمجية إن احتجت إلى حالته الموثوقة.

وللاطّلاع بدقّة على ما أُرسل في الاستدعاء وما ورد في الاستجابة، يفتح مسؤول النظام التشغيل في **سجل التفاعلات** `Since 1.10` – انظر واجهة سجل التفاعلات البرمجية [بناء التطبيقات › التفاعلات والأتمتة].

موصِّل Zapier

يتيح الموصِّل لمهامّ Zap الاشتراك في أحداث المنصّة، وتلقّي المُدخلات عند وقوع تلك الأحداث، وإرسال إجراءات إلى المنصّة. وهو متاح منذ إصدارات قديمة تسبق سلسلة 1.2.

من المنصّة إلى Zapier {#outbound-flow-platform–zapier}

حين يشغّل تفاعلٌ الإجراء **إشعار Zapier**، ترسل المنصّة هذه الحمولة إلى كل عنوان hook مشترك:

```
{
  "dataEntity": { "id", "specificationVersion", "normalizedName", "name" },
  "dataEntry":  { "id", "value", "specificationVersion", "dataEntity",
                  "updateInfo", "creationInfo" },
  "trigger":    { "eventType", "actionOptions", "criteria" }
}
```

إدارة اشتراكات المشغّلات

يشترك تطبيق Zapier عبر روابط REST hooks ضمن `/api/integrations` :

الاستدعاء	الغرض	الجسم
`POST /api/integrations/v1/zapier/trigger/subscribe`	الاشتراك برابط hook	`{ hookUrl, zapName, triggerType, entity, entry? }`
`GET /api/integrations/v1/zapier/trigger/subscriptions`	سرد الاشتراكات	–
`DELETE /api/integrations/v1/zapier/trigger/unsubscribe/{subscription_id}`	إلغاء الاشتراك	–
`GET /api/integrations/v1/zapier/trigger/perform-list`	حمولة نموذجية لخطوة الاختبار في محرّر Zap	–

من Zapier إلى المنصّة {#inbound-flow-zapier–platform}

يُحدث إجراء Zap أثرًا في المنصّة عبر إرسال حدث. ويحتاج هذا الاستدعاء إلى رمز وصول OAuth (يستخدم عميل OAuth الخاص بـ Zapier في القاطن تدفّق رمز التفويض – انظر المصادقة وOAuth):

```
curl -s -X POST https://acme.example.com/api/integrations/v1/zapier/action \
  -H "Authorization: Bearer <access-token>" \
  -H "Content-Type: application/json" \
  -d '{"profile": {"entityId": "<entity-id>", "zapId": "<zap-id>",
       "event": {"eventType": 2, "output": {"fileUrl": "https://…"}}}}'
```

تشغّل المنصّة كل تفاعل نشط على ذلك الكيان يكون مشغّله **حدث من Zapier** بمعرّف Zap نفسه. فقيمة `zapId` تربط مهمّة Zap بتفاعلاتها. أما `output` فيحمل بيانات حرّة البنية من Zapier يمكن للتفاعلات استخدامها، وتأخذ `eventType` إحدى القيم التالية:

القيمة	حدث Zapier
`0`	إنشاء جدول بيانات Google
`1`	إضافة صفّ إلى جدول بيانات Google
`2`	إنشاء ملف PDF
`3`	إنشاء حدث في تقويم Google
`4`	تلقّي بريد في Gmail
`5`	إلغاء حدث في تقويم Google

يعيد الاستدعاء استجابة نجاح حتى لو تعذّر تمرير الحدث؛ لذا راجع السجل التاريخي للتفاعل للتأكّد من تشغيل التفاعلات.

قيود معروفة

- **البيانات النموذجية ثابتة.** يُعيد `perform-list` مُدخلًا اصطناعيًا واحدًا، لا بيانات حيّة.
- **لا يوجد رابط «التقاط» وارد.** يقبل `PATCH /api/integrations/v1/zapier/webhook/catch` الطلب دون تنفيذ أي إجراء. لذا استخدم استدعاء الإجراء أعلاه بدلًا منه.

معترضات الطلبات

يمكن لوظيفة منشورة أن تعمل قبل معالجة طلب وارد يطابق مسارًا وأسلوب HTTP محدّدين. وتستطيع:

- **إعادة كتابة الطلب** – تغيير جسمه أو ترويساته أو معاملات الاستعلام قبل مواصلة معالجته؛ أو
- **الإجابة عنه مباشرةً** – إعادة استجابة بحالة وجسم وترويسات تحدّدها الوظيفة، فلا يصل الطلب إلى المنصّة أبدًا.

تعمل المعترضات بترتيب أولويتها، وتقتصر على مرحلة ما قبل المعالجة؛ فلا يوجد معترض للاستجابات.

إعادة توجيه المعارض في عميلك الخاص

يستطيع باني التطبيق إعداد المعرض لتوجيه أشخاص مختلفين إلى وجهات مختلفة – مثلًا، الزوّار إلى صفحة عامة والموظفين إلى معرض داخلي. وحين يفتح عميلك الخاص معرضًا لمستخدم نهائي (`GET /api/v2/views/end-user/{id}`، وما يقابله في الصفحات)، تتضمّن الاستجابة `redirection`: إمّا `null`، وإمّا الوجهة `{ "type": <type>, "redirectTo": "<id or URL>" }`. **ويجب أن يتصرّف عميلك بناءً عليها**؛ فالمنصّة لا تعيد التوجيه من تلقاء نفسها.

`type`	الوجهة
`0`	عنوان URL ثابت
`1`	معرض بيانات
`2`	صفحة

type	الوجهة
3	تقرير

تُفحص القواعد بترتيب أولويتها (الرقم الأصغر أولًا)، وتُطبَّق أول قاعدة تنطبق على المستخدم: الجميع، أو المستخدمون المسجّلون، أو الزوّار، أو مستخدمون ومجموعات محدّدون بالاسم. والقاعدة المضبوطة على **مقيد** لا تنطبق إلا على المستخدمين والمجموعات المحدّدة فيها. وقواعد إعادة التوجيه متاحة منذ إصدارات قديمة.

أسطح تكامل أخرى

- **الروابط المختصرة.** تختصر المنصّة الروابط في رسائل البريد الإلكتروني والرسائل النصية التي ترسلها، بصيغة `https://<host>/sh/<code>`، وهي ليست واجهة برمجية للعملاء.
- **رموز QR.** يمكن للمعارض عرض رموز QR تولّدها وظيفة في المنصّة.
- **استيراد الدليل.** يمكن استيراد المستخدمين والمجموعات من Active Directory (LDAP) وMicrosoft Entra ID، ويُنفَّذ ذلك عبر مهام دفعية يشغّلها المسؤولون – انظر المستخدمون من دليل شركتك [الإدارة].
- **أحداث المنصّة.** توفّر منظومة الأحداث والأوامر الخاصة بالمنصّة نفسها أوسع إمكانات التكامل؛ انظر نموذج البيانات والأحداث.

> **للتعمّق أكثر**
>
> لبُناة التطبيقات: التفاعلات والأتمتة [بناء التطبيقات] · لمهندسي المنصّة: التكاملات – التفاصيل الداخلية

لوحات معلومات Grafana

Availability

.Since 1.6

تُشحَن مع أسس **إضافة مصدر بيانات لـ Grafana** – باسم `Apptomator` – تتيح لك بناء لوحات معلومات Grafana فوق بيانات أسس. وخاصيتها المميّزة أن المدير يبني لوحة المعلومات **مرّة واحدة**، فيرى كل مُشاهِد يفتحها الصفوف التي تسمح بها أذوناته هو في أسس وحدها.

وهذا ليس مرشّحاً تضبطه أنت. فالإضافة لا تحتفظ ببيانات اعتماد خاصة بها: إذ يسافر كل طلب عبر وسيط مصادر البيانات في Grafana ومعه رمز OAuth الخاص بالمُشاهِد المُسجَّل دخوله، فتُقيِّم البوّابة والأذونات [الإدارة › الأذونات والوصول] الطلبَ تماماً كما تفعل لذلك الشخص في بوّابة أسس الشبكية. فشخصان يفتحان اللوحة نفسها يحصلان على صفوف مختلفة، ولا يستطيع أيّ منهما توسيع ما يراه.

هذه الصفحة موجّهة لمن **يبني اللوحات**. أما تثبيت الإضافة، وتوصيل OAuth في Grafana مقابل خدمة الهوية في أسس، ونشر أرشيف الإضافة، فتغطّيها صفحة إضافة مصدر البيانات لـ Grafana في قسم التهيئة والصيانة.

ما الذي يمكنك بناؤه وما لا يمكن

تنفّذ الإضافة **استعلامات أسس المحفوظة** – وهي ليست بانيَ استعلامات ارتجالياً. فكل ما تستطيع اللوحة قراءته قد نُمذِج في أسس مسبقاً:

ما لديك	ما تستطيع اللوحة فعله
معرض بيانات [بناء التطبيقات › معارض البيانات وأوجهها]	قراءة مُدخلاته صفوفاً
استعلام بيانات [بناء التطبيقات › الاستعلامات والمرشّحات] محفوظ	تطبيقه كمرشّح سريع على تلك الصفوف
استعلام إحصائي محفوظ	إعادة القيمة المجمّعة التي يحسبها، جاهزة الشكل للرسم

ويترتّب على ذلك أمران يشكّلان طريقة تصميمك لـ Grafana:

- **المعرض إلزامي لكل لوحة.** فالمُدخلات لا تُقرأ إلّا عبر معرض، لأن خطة تنفيذ المعرض هي ما تُخوِّله البوّابة. ولا يوجد اختصار على مستوى الكيان – فإن كان للبيانات مكان على لوحة معلومات، فهي تحتاج معرضاً.
- **كل ما تريد تجميعه ينبغي أن يكون استعلاماً إحصائياً.** فالتجميع في أسس أرخص بكثير من سحب الصفوف إلى Grafana وتجميعها هناك، والنتيجة تصل جاهزة للرسم.

إعداد مصدر البيانات

يُنفَّذ مرّة واحدة لكل خادم Grafana، من **Connections › Data sources › Add new data source › Apptomator**.

الإعداد	القيمة
URL	عنوان بوّابة أسس الأساس – **نطاق العميل الحقيقي**، مثل `https://osos.example.com`
Forward OAuth Identity	**مُفعَّل**. وليس اختيارياً

الإعداد	القيمة
Save & test	يؤكّد العنوان وتمرير الرمز والبوّابة بنداء واحد

واثنان منها يستحقّان التأكيد:

- **يجب أن يكون العنوان مُضيفاً يُحَل إلى نطاق العميل.** فالبوّابة تحلّ النطاق والفضاء من ترويسة `Host`، وأي طلب لا يُحَل مُضيفه يُرفَض بـ `MISSING_DOMAIN_ID`. ولن ينفع عنوان داخلي ضمن العنقود.
- **مفتاح Forward OAuth Identity هو ما يجعل النموذج يعمل.** فبإطفائه لا يمرّر الوسيط أي رمز للمُشاهِد ويتوقّف نموذج أذونات كل مُشاهِد عن السريان بصمت – فاللوحات لا تُخفِق، بل تكفّ عن كونها شخصية فحسب.

ونمط الوصول مُثبَّت على `proxy` من قِبل الإضافة ولا يُعرَض كخيار. فالوصول المباشر (من المتصفّح) يتجاوز الوسيط كلّياً، فلن تُرفِق Grafana رمز المُشاهِد أبداً.

بناء لوحة

محرّر الاستعلام سلسلة تحاكي نموذج الوصول في أسس – وحدة، ثم معرض، ثم استعلام. و`Since 1.8` تبدأ السلسلة من **الوحدة**: اختفى منتقي Entity السابق، وأي لوحة مُحفَظة قبل هذا التغيير تستشفي وحدتها ذاتياً أول مرّة تُفتَح فيها محرّرها.

الحقل	المعنى
Type	`Rows (Data Query)` للبيانات على مستوى الصف، و`Statistic Query` لقيمة مجمّعة
Module	الوحدة التي تحمل المعرض المُراد القراءة منه – نقطة بداية السلسلة
View	**إلزامي.** المعرض الذي تُخوِّل البوّابة خطة تنفيذه، وهو ما يحدّد الكيان [بناء التطبيقات › إنشاء كيان وتصميمه] – فالكيان لا يُختار مباشرة أبداً؛ بل يسافر مع المعرض، وهو ما يُحفَظ في الاستعلام
Saved query	اختياري للصفوف (مرشّح سريع فوق المعرض)؛ و**إلزامي** للوحات الإحصاء
(الوسائط)	حقل إدخال لكل وسيط يُعلِنه الاستعلام المحفوظ. والقيم نص حر، فتعمل متغيّرات لوحة المعلومات مثل `$department`
Time field	للصفوف فقط. حقل `DateTime` / `Date` يُعرَض كعمود الزمن في الإطار – وكذلك لترشيح اللوحة بالمجال الزمني للوحة المعلومات
Limit	للصفوف فقط. الافتراضي 1000؛ ويقصّه الخادم إلى سقفه الخاص على أي حال
Fields	للصفوف فقط. أسماء الأعمدة المُبقاة مفصولة بفواصل؛ والفراغ يعني الكلّ
Metadata	للصفوف فقط، مُعطَّل افتراضياً. يُظهِر أعمدة المسك `_meta*`

و`Since 1.8`، تغيّر أيضاً من يستطيع بناء لوحة: صار الاكتشاف يمرّ عبر نقطة النهاية نفسها الخاصة بالمستخدم النهائي التي تستخدمها لوحة أسس نفسها، فصار وصول مستخدم أسس العادي إلى المعرض وحده كافياً لتأليف لوحة عاملة. أما سابقاً فكانت نداءات الاكتشاف تتطلّب رمزاً بنطاق إداري في أسس، فلا يستطيع بناء لوحة عاملة إلّا رمز مسؤول. وتبقى إذن التأليف في Grafana نفسها – من يملك فتح محرّر الاستعلام أصلاً – دون تغيير، وهي البوّابة المنفصلة القائمة أصلاً؛ انظر `analytics.oauth.authoringGroup` في إضافة مصدر البيانات لـ Grafana.

وحقول النص تُثبَّت عند مغادرة الحقل أو ضغط Enter، لا عند كل ضغطة مفتاح.

الوسائط ومتغيّرات لوحة المعلومات `Since 1.8`

تظهر وسائط الاستعلام المحفوظ كحقول إدخال، ويقبل كلٌّ منها متغيّر لوحة معلومات في Grafana. فوضع `$department` في صندوق وسيط يجعل اللوحة تتبع مُنتقي المتغيّرات في لوحة المعلومات.

وثمّة قاعدة مهمّة حين يكون الوسيط مرتبطاً بحقل **رقمي** أو **منطقي**: **القيمة الفارغة تُرسَل بمعنى «غير مضبوطة»، لا `0` ولا `false` أبداً**. وهذا مقصود – فتحوّل رقم فارغ إلى `0` بصمت كان سيرشّح البيانات إلى لا شيء ويبدو نتيجةً فارغة لا خطأً. فاترك الوسيط فارغاً لتعني «لا ترشّح على هذا».

> **تنبيه**
>
> على معرض مُفعَّل فيه **الترشيح أولاً**، تؤدّي كل قيمة وسيط وظيفتين: تُستخدَم وسيطَ استعلام *وتُحقَن* مطابقةً مساواة على الحقل ذي الاسم نفسه. وهذا سلوك المعرض نفسه لا الإضافة، لكنه يعني أن القيمة الواحدة تضيّق الصفوف مرّتين.

حقول الزمن ومُنتقي الزمن في لوحة المعلومات `Since 1.8`

من دون حقل زمن، لا أثر لمُنتقي الزمن في لوحة المعلومات على اللوحة. وضبطه يفعل ثلاثة أمور دفعةً واحدة: يصير عمود الزمن في الإطار، وتُفرَز الصفوف به، وتُضيَّق اللوحة إلى مجال المُنتقي على جانب الخادم.

وتعرض القائمة:

- حقول الكيان نفسه من نمط `Date` / `DateTime`؛
- حقول `Date` / `DateTime` **داخل الكتل**، وتظهر بالشكل `Basic info / Birthdate`؛
- تاريخَي التدقيق النظاميَّين، **Created at** و**Last updated**.

ولا يُعرَض `_metaData.createdAt` و`_metaData.updatedAt` **عن قصد**. فهما يحملان اللحظات نفسها، لكنّهما يُحسَبان عند قراءة المُدخَل لا يُخزَّنان، فلا يمكن لأي مرشّح أن يطابقهما – وعرضهما كان سيمنحك عمود زمن يتجاهله المُنتقي بصمت. أما **Created at** و**Last updated** فيأتيان من حقول مخزَّنة ويعملان فعلاً مع المُنتقي.

ويُنسَّق حدّا المجال ليطابقا طريقة تخزين الحقل لقيمه – فحقل `Date` يحمل `"2026-05-19"`، وحقل `DateTime` لحظة ISO كاملة – وتستخدم الحدود بالتاريخ فقط **المنطقة الزمنية للوحة المعلومات**، فتطابق الأيامُ ما يعرضه المُنتقي.

> **ملاحظة**
>
> اللوحة المحفوظة قبل وجود الترشيح بالمجال الزمني لا تحمل نمط الحقل، فتتجاهل المُنتقي بصمت. افتح محرّرها مرّة واحدة ويُملأ النمط تلقائياً.

كيف تُعرَض قيم أسس

معظم قيم حقول أسس كائنات، فتُسطَّح كلٌّ منها إلى خلية عرض:

القيمة	كيف تُعرَض
مرجع [بناء التطبيقات ، ربط الكيانات]	عنوانه الرئيسي المحلول، مثل `Super Admin`
تصنيف	تسميته المحلولة، مثل `مقبول`
ملف	`avatar.jpg [file]` – نائب؛ فالمرفقات لا تُعرَض
قيمة مُتَرجَمة	بلغة المُشاهِد في Grafana، مع التراجع إلى الإنكليزية
حقل متعدّد القيم	أجزاؤه موصولة
كتلة (`address: {city, country}`)	**عمود لكل حقل فرعي**: `address.city` و`address.country`

وتتوسّع الكتل [بناء التطبيقات › اختر نوع الحقل] إلى أعمدة تماماً كما تفعل في وجه الجدول في أسس. ويُحتفَظ باسم الكتلة بادئةً لأن أسماء أعمدة Grafana يجب أن تكون فريدة، فتبقى كتلتان لكلٍّ منهما `city` متمايزتين. أما الكتلة التي لا يمكن أن تصير أعمدة – المتداخلة داخل حقل متكرّر – فتتراجع إلى `city: Damascus, country: Syria` في خلية واحدة.

وتبقى حقول أسس الرقمية رقمية، فتظلّ قابلة للجمع. أما كل حقل قيمته كائن فيصير نصّاً، والنص لا يُعَدّ ولا يُجمَّع إلّا عدّاً وتجميعاً بالمفتاح – لا جمعاً حسابياً.

وتقع أعمدة `_meta*` – `_metaData` و`_metaOrder` و`_metaCurrentSteps` و`_metaWorkflowHistory` و`_metaAdhocSteps` – على كل مُدخَل وتتوسّع إلى أعمدة كثيرة، ولذلك تُخفى ما لم تُفعِّل **Metadata**. وتسمية أحدها في **Fields** تُظهِره على أي حال.

أعمدة التدقيق `Since 1.8`

> ⚠ **تغيير جذري – حذف `systemTimeField`**
>
> لم يعد خيار اللوحة `systemTimeField` موجوداً – **حُذِف لا أُهمِل فقط**. فأي لوحة معلومات استخدمته لإظهار تاريخ نظامي واحد كعمود تتوقّف عن العمل مع هذا الإصدار من الإضافة، ولا يوجد إعداد في محرّر الاستعلام يعيد إنتاج سلوك العمود الواحد القديم، لأن الأعمدة الأربعة أدناه غير مشروطة. **أعد تأليف أي لوحة كهذه لتقرأ `creationInfo.doneAt` / `updateInfo.doneAt` مباشرةً.**

يحمل كل إطار صفوف الآن أربعة أعمدة تدقيق، مُلحَقة **بعد** حقول المُدخَل نفسه كي لا تُزيح أبداً حقل Time field الذي اخترته من مقدّمة الإطار:

العمود	يُظهِر	النمط
`creationInfo.doneAt`	متى أُنشئ المُدخَل	زمن، دائماً
`creationInfo.doneBy`	من أنشأه – الاسم المعروض المحلول، أو مُعرِّف الفاعل الخام إن تعذّر الحلّ	نصّي
`updateInfo.doneAt`	متى آخر تحديث للمُدخَل	زمن، دائماً
`updateInfo.doneBy`	من حدّثه آخر مرّة – الاسم المعروض المحلول، أو مُعرِّف الفاعل الخام إن تعذّر الحلّ	نصّي

وهي غير مشروطة – لا إعداد يفعّلها – فتوزيع المُدخلات حسب يوم الإنشاء، أو عدّها مُجمَّعةً حسب المؤلِّف، لا يحتاج إلا تحويل **Group by**، دون اختيار Time field مسبقاً. سمِّ عموداً أو أكثر منها في **Fields** للإبقاء على تلك الأعمدة فقط في لوحة لا تريد الأربعة جميعاً. ويُسنَد لعمودَي `…doneAt` نمط الزمن سواء أكان أحدهما Time field المُختار أم لا، فتجميع أحدهما باليوم يبعد تحويل *Format time* واحداً فقط.

والمرجع الذي لم يعد يُحَل – فاعل `doneBy` مُحذوف، مثلاً – يتراجع الآن إلى مُعرِّفه الخام، مثل أي مرجع آخر غير محلول، لا إلى السلسلة الحرفية `"{}"` التي كان خللٌ في العرض يُظهِرها سابقاً في الخلية.

الرسم البياني

اللوحات ليست مقصورة على الجداول – فاختر أي تمثيل بصري. وما يحتاجه المخطّط عمودٌ رقمي وشيءٌ يُرسَم مقابله، ولذلك **يهمّ نمط الاستعلام الذي تختاره أكثر من التمثيل البصري**.

للقيم المجمّعة – استخدم استعلاماً إحصائياً. وهو أقصر طريق إلى لوحة أعمدة أو دائرية أو رقمية، لأن التجميع يقع في أسس. فيصل الإطار جاهز الشكل للرسم: عمود `category` (مفتاح التجميع)، وعمود لكل تجميع فرعي، وقيمة `value` رقمية. وجّه إليه **Bar chart** أو **Pie chart** فيُعرَض بلا أي تحويلات إطلاقاً.

للعدّ والجمع فوق الصفوف – استخدم استعلام صفوف مع تحويل. فالصفوف تعود غير مجمّعة، فأضف **Transform › Group by**؛ جمِّع بالعمود الذي تريده على المحور الأفقي، واضبط حقلاً آخر على *Calculate* ← *Count* (أو Sum/Mean). ثم بدّل التمثيل إلى Bar chart.

للاتجاهات عبر الزمن – استخدم استعلام صفوف مع حقل زمن. فيصير العمود عمود زمن حقيقياً في Grafana، وتُفرَز الصفوف به وتُضيَّق إلى مجال المُنتقي، فترسم لوحة **Time series** بالترتيب وتتبع المُنتقي. اقرنه بعمود رقمي، أو بـ **Group by** على فترة زمنية لعدّ المُدخلات يومياً.

وأمران جديران بالمعرفة:

- **المخطّطات متعدّدة السلاسل من استعلام إحصائي مُجمَّع فرعياً** تصل بصيغة طولية (`department`، `category`، `value`). ولوحة الأعمدة في Grafana تريد عموداً لكل سلسلة، فأضف **Transform › Partition by values** على عمود التجميع الفرعي، أو **Rows to fields**.
- **دمج استعلامين في مخطّط واحد** يتمّ بـ **Transform › Join by field** – فالإضافة لا تصل على الخادم. أضف كلا الاستعلامين هدفين منفصلين على اللوحة.

تضمين لوحة معلومات داخل أسس

يمكن عرض لوحة معلومات Grafana داخل أسس باستخدام صفحة [بناء التطبيقات › الصفحات والقطع] من نمط **مُضمَّنة**، وهي تعرض إطاراً `iframe` بسيطاً – فلا يلزم شيء خاص على جانب أسس.

الهدف	العنوان
لوحة معلومات كاملة بلا زخرفة Grafana	`<grafana>/d/<uid>/<slug>?kiosk`
لوحة مفردة	`<grafana>/d-solo/<uid>/<slug>?panelId=<n>`

ويتوقّف نجاح التضمين فعلياً على شرطين على جانب الخادم – سماح Grafana بالتأطير أصلاً، وكون الموقعين على الموقع نفسه كي تُرسَل كعكة جلسة Grafana. وكلاهما إعداد نشر؛ انظر إضافة مصدر البيانات لـ Grafana.

قيود تُصمَّم حولها

هذه خصائص النموذج لا عيوب – فخطّط لها لا حولها.

- **الاستعلامات المحفوظة حصراً.** فلا بانيَ استعلامات ارتجالياً. وإن احتاجت لوحة المعلومات قطعاً جديداً من البيانات، فنمذِجه معرضاً أو استعلاماً محفوظاً أولاً.
- **لا تنبيهات ولا تقارير مجدولة.** فقواعد تنبيه Grafana وتقاريرها تعمل **بلا جلسة مُشاهِد**، فلا هوية تُمرَّر ولا سبيل لاحترام أذونات كل مُشاهِد. وكل ما هو على شكل تنبيه ينتمي إلى آليات الإشعارات [الإدارة › الإشعارات وقوالب البريد الإلكتروني] ومراقبة الوقت/اتفاقية مستوى الخدمة [الإدارة › ساعات العمل والعطلات] في أسس نفسها.
- **سقف الصفوف** `Since 1.8` . يقصّ الخادم كل طلب إلى سقفه الخاص (10000 افتراضياً)، فلا يمكن للوحة أن تبثّ مجموعة كاملة أبداً. جمِّع بدلاً من ذلك.
- **يقصّ** `Fields` **الأعمدة بعد وصول الصفوف**، ولا يقلّل ما يُنقَل. فـ **المعرض** هو من يقرّر أي حقول تعود، فقلّص هناك إن كان الحجم مهمّاً.
- **لوحات الإحصاء لا تقبل وسائط زمن تشغيل.** فمسار تنفيذ الإحصاء لا يقبل سوى حقل الترشيح الخاص بالاستعلام. وإن وجب أن يتغيّر إحصاءٌ بحسب مُدخَل، فاجعل التغيّر جزءاً من تصميم الاستعلام.

- **المرفقات لا تُعرَض** – فالملفات تظهر `[file] <name>` .
- **مسارات المشاركة التي تُعرَض بلا جلسة – اللقطات ولوحات المعلومات العامّة – يجب أن تبقى معطّلة.** فهي ستقدّم بيانات مستخدم واحد لكل من يفتح الرابط. انظر صفحة التهيئة.

ذات صلة

- معارض البيانات والأوجه [بناء التطبيقات › معارض البيانات وأوجهها] – الوحدة التي تُنفَّذ عليها كل لوحة
- الاستعلامات [بناء التطبيقات › الاستعلامات والمرشّحات] – استعلامات البيانات والاستعلامات الإحصائية المحفوظة
- لوحات المعلومات والملخّصات وقطع العدّادات [بناء التطبيقات › الخلاصات ولوحات المعلومات والعدّادات] – لوحات أسس الداخلية
- التقارير [بناء التطبيقات] – مُخرَجات مجدولة على شكل مستند
- الويب هوك والتكاملات – أسطح التكامل الصادرة الأخرى
- إضافة مصدر البيانات لـ Grafana – تثبيتها وتشغيلها

نموذج البيانات والأحداث

للتكامل بفاعلية مع أسس، تحتاج إلى تصوّر واضح لكيفية تخزينها البيانات وتبادل الرسائل بين خدماتها. يصف هذا الدليل نموذج البيانات المتاح للمطوّرين و**مفردات ناقل الرسائل** التي يراقبها مطوّرو التكامل ويتفاعلون معها، بمستوى من التفصيل يتيح الاستناد إليه في التصميم والرجوع إليه عند كتابة المشغّلات/التفاعلات، أو الوظائف المخصّصة، أو التكاملات الخارجية (webhooks/Zapier) التي تستجيب للنشاط داخل القاطن.

تمثّل المفردات المستخدمة هنا العقد المشترك للمنصّة؛ فهي التعريف الموحّد الذي تستورده كل خدمة خلفية، وتُعلَن فيه جميع الأحداث والأوامر ورسائل socket ونماذج المجال. واحرص دائمًا على إجراء المطابقة باستخدام المعرّفات الواردة في الرسائل كما تظهر أدناه، لا أسماء الفئات.

النموذج الأساسي

يكاد كل شيء في أسس يكون **مدخل بيانات** يحمل حمولة `value` ديناميكية، ويحدّد **كيان بيانات** مخطّطه. ويقوم النموذج كلّه على بضعة مفاهيم متّسقة:

- **الكيانات والمدخلات.** يعرّف **كيان البيانات** مخطّطًا يشمل الحقول، والتحقّق، وسير العمل، والمشغّلات. أما **مدخل البيانات** فهو مُدخل واحد من ذلك الكيان، يحمل قيم حقوله إلى جانب حالته وتاريخه.
- **المراجع.** ترتبط المُدخلات بعضها ببعض عبر مؤشِّر مرجعي عام بالصيغة `{ dataEntity, dataEntry }`. وحيثما أشار مُدخل إلى آخر، فهذه هي البنية التي ستجدها.
- **النصّ المُترجَم.** الأسماء والنصوص الموجّهة للمستخدمين هي **سلاسل مُترجَمة** تحمل العربية والإنجليزية معًا (`{ ar, en }`). أرسِل الأسماء والتسميات بهذه البنية، وتوقّع استقبالها بها.

اللبنات الأساسية

المفهوم	الدور
كيان البيانات	تعريف المخطّط: الحقول، والتحقّق، والحالة (مسودّة / نشط / مؤرشف)، وسير العمل، والمشغّلات
مدخل البيانات	مُدخل ينتمي إلى كيان، ويحمل حمولة `value` وحالةً وتاريخًا
وحدة البيانات	تجمّع من الكيانات يحدّد محتوى الشريط الجانبي
معرض البيانات	طريقة عرض لكل كيان (جدول، لوحة مهام، تقويم، شريط زمني، وغيرها)
استعلام البيانات / الاستعلام الإحصائي	مرشّحات وتجميعات محفوظة
سير العمل	تعريف عملية (خطوات، ملكية، SLA) مرتبط بكيان
التفاعل	قاعدة أتمتة (حدث + شرط + إجراءات)
الوظيفة	رماز يعمل بلا خادم ويكتبه المستخدم
حقّ الوصول	إذن يحدّد مستوى الوصول (ممنوع / عرض / تعديل / مسؤول) إلى هدف
التفويض	منح صلاحية «العمل بالنيابة» خلال فترة زمنية محدّدة

تُنمذَج كائنات الهوية نفسها – المستخدمون، والمجموعات، والوحدات التنظيمية، والأدوار، والإسنادات – بوصفها كيانات بيانات؛ لذا تنطبق عليها المفاهيم ذاتها عند قراءتها أو كتابتها.

أنواع الحقول

يُختار نوع الحقل من مجموعة ثابتة: نصّ، رقم، تاريخ، وقت، تاريخ ووقت، منطقي (Boolean)، Blob، **كيان** (مرجع إلى مُدخل آخر)، أيقونة، **كتلة (Block)** (جدول فرعي)، JSON مُترجَم، GeoJSON، وJSON. ويمكن إدخال قيمة الحقل يدويًا، أو تعيين قيمة افتراضية لها، أو جلبها ديناميكيًا، أو حسابها من **تعبير**، أو تحديدها عبر **مؤشّر**، مع تحديد توقيت إعادة الحساب عند فتح النموذج أو الحفظ أو تغيّر الحقل. وترتبط خيارات الحساب بمحرّك التعابير الموصوف في دليل الوظائف والتكاملات.

شكل المرجع

تستخدم المراجع بين المُدخلات في جميع أجزاء أسس مؤشِّرًا يتألّف من جزأين، لا معرّفًا مجرّدًا:

```
export interface ReferenceProperty {
  dataEntity: Id;   // which entity/collection the target lives in
  dataEntry: UUID;  // which record within it
}
```

يحدّد `ReferenceProperty` كلاً من *نوع المُدخل* المُشار إليه و*المُدخل بعينه*. وعندما تكون قيمة حقل في مُدخل مرجعًا، فإنها تتّخذ هذا الشكل.

الكيان (المخطَّط)

يمثّل `DataEntity` تعريفًا لنوع من المُدخلات، وهو أقرب ما يكون إلى مخطَّط جدول أو مجموعة. وفيما يلي الحقول الرئيسية التي يقرؤها مطوّر التكامل:

الحقل	النوع	ملاحظات
`id`	`UUID | string`	معرّف الكيان.
`normalizedName`	`string`	اسم آلي ثابت؛ استخدمه لتحديد الكيان في الرماز بدلاً من `name` المُترجَم.
`name` / `description`	`LocalizedString`	تسميات بشرية، لكل لغة.
`type`	`DataEntityTypeEnum`	فئة الكيان (انظر الـ enum أدناه). `Custom = 0` كيان عادي معرَّف من قبل الباني.
`version`	`number`	نسخة المخطَّط، تُرفَع عند تغييرات الحقول التي تكسر التوافق. تسجّل المدخلات الـ `entityVersion` التي حُفِظت مقابلها.
`module`	`UUID`	الوحدة التي ينتمي إليها الكيان.
`fields`	`DataEntityField[]`	تعريفات الحقول.
`workflows`	`DataEntityWorkflow[]`	تعريفات سير العمل المرفقة بالكيان.
`state`	`DataEntityStateEnum`	`Active = 0` ، `Archived = 1` .
`isImplicit`	`boolean`	كيانات مملوكة للنظام (انظر قيم الـ enum `Core` ، `System` ، `Space` ، …).
`virtual`	`VirtualEntityOptions?`	موجود للكيانات المحسوبة/المُجمِّعة (بلا صفوف فعلية).

يُعرِّف `DataEntityField` حقلاً واحدًا من خلال `normalizedName` ، و`dataType` (`DataTypeEnum`)، و`renderOptions` ، و`validations` ، والخصائص الاختيارية `virtualField` ، و`mapping` ، و`sources` ، و`sensitiveAccess` ، و`valueValidators` . وعند الاستجابة لتغيير على مستوى الحقل، أجرِ المطابقة باستخدام `normalizedName` / `fullNormalizedName` الخاص بالحقل.

المُدخل

يمثّل `DataEntry<T>` مُدخلًا واحدًا من كيان، وهو الكائن الذي تتلقّاه مع معظم الأحداث:

الحقل	النوع	ملاحظات
`id`	`UUID`	معرّف المُدخل.
`dataEntity`	`UUID`	الكيان الذي ينتمي إليه هذا المُدخل.
`value`	`T extends DataEntryValue`	قيم الحقول، مفهرسة بحسب `normalizedName` لكل حقل. وتحمل حقول المرجع قيمًا من نوع `ReferenceProperty`.
`state`	`DataEntryStateEnum`	`Draft = 0`، `Active = 1`، `Archived = 2`.
`stateHistory`	`DataEntryStateHistoryRecord[]`	سجل تاريخي للتدقيق لانتقالات الحالة.
`lifeCycleStatus`	`LifeCycleStatusEnum?`	حالة دورة حياة سير العمل: `Failed = -1`، `Active = 0`، `Completed = 1`؛ `undefined` = ليس ضمن سير عمل.
`entryType`	`EntryTypeEnum?`	`Normal = 0`، `Virtual = 1`.
`virtual`	`boolean?`	`true` للمُدخلات المحسوبة التي لا وجود فعلي لها (ناتج كيان تجميعي/افتراضي).
`basedOn`	`ReferenceProperty[]?`	للمُدخلات الافتراضية، المُدخلات المصدرية التي اشتُقّ منها الحساب.
`entityVersion`	`number`	إصدار مخطَّط الكيان الذي حُفِظ هذا المُدخل وفقه.
`store`	`ReferenceProperty`	المخزن/القسم الذي يوجد فيه المُدخل.
`isImplicit`	`boolean`	مُدخل مملوك للنظام لا يمكن تعديله أو أرشفته.
`isInvalid` / `invalidReason`	`boolean` / `DataEntryInvalidReasonEnum | null`	يُضبَط عندما يجعل تغيير في المخطَّط المُدخل غير صالح: `Obsolete = 0`، `Invalid = 1`.
`creationInfo` / `updateInfo`	`ActionInfo`	هوية مَن أنشأ/حدّث المُدخل، ووقت الإنشاء/التحديث.

يحمل كائن `value` أيضًا بيانات وصفية لسير العمل ضمن مفاتيح محجوزة: `_metaCurrentSteps`، و`_metaWorkflowHistory`، و`_metaAdhocSteps`، و`_metaOrder`. وتعامَل مع أي مفتاح `_meta*` على أنه مفتاح يديره النظام.

> ملاحظة حول الحالات: `lifeCycleStatus` (`undefined` / `Active` / `Completed` / `Failed`) قيمة تُستخدم لتتبّع دورة حياة سير العمل المصاحب للمُدخل. وهي تختلف عن الحالات الموجّهة للمستخدم التي تُعرض للمُدخلات من نمط الطلب (مثل مسودّة / مفتوح / مكتمل / مُعلَّق، مع نتيجة نجَح / فشِل / مرفوض / انتهت المهلة / أُلغي / متقادم). وعند بناء أتمتة لمُدخل من نمط الطلب، اقرأ `lifeCycleStatus` لمعرفة حالة التتبّع، وحقل الحالة الخاص بالكيان لمعرفة الحالة المعروضة في الواجهة.

التفاعلات والمشغّلات وإعدادات الأحداث

التفاعل (ويُعرف أيضًا بالمشغّل، trigger) هو الكائن الذي يستخدمه الباني للتعبير عن القاعدة التالية: «عندما يقع *هذا* الحدث ويطابق *هذه* المعايير، شغّل *هذه* الإجراءات». ويُمثَّل بالنموذج `DataReaction`:

الحقل	النوع	ملاحظات
`eventType`	`EventTypeEnum`	الحدث الذي يستمع إليه هذا التفاعل.
`eventConfig`	`DataReactionEventConfig`	إعداد المطابقة الخاص بنوع الحدث (انظر أدناه).
`executorType`	`{ id: EventExecutorTypeEnum }?`	الهوية التي تُشغَّل بها الإجراءات: `SYSTEM`، `MANAGER`، `EXECUTOR`.
`criteria`	`FilterExpression?`	مرشّح يعتمد على قيم المُدخل المتأثّر.

الحقل	النوع	ملاحظات
`query`	`ReferenceProperty?`	استعلام محفوظ يحلّ محلّ `criteria` عند ضبطه.
`actions`	`DataReactionAction[]`	الإجراءات المطلوب تشغيلها، بالترتيب.
`spaceScope`	`DataReactionSpaceScope`	`{ rootSpace, currentSpace, otherSpaces }` – المساحات التي يعمل فيها التفاعل.
`type`	`DataReactionTriggerTypeEnum?`	`NormalTrigger = 0` ، `SLATrigger = 1` .
`sourceEntity`	`DataReactionSourceEntity?`	`Self` ، أو `Relation` ، أو `AbsoluteEntity` – الكيان الذي تُشغّل أحداثُه هذا التفاعل.
`loopThreshold`	`number?`	حماية من سلاسل التفاعلات غير المنضبطة.
`disabled`	`boolean?`	التفاعل موقوف مؤقتًا.
`relatedEntities`	`DataReactionRelatedEntities[]?`	الكيانات المرتبطة التي تُؤخذ تغييراتها في الحسبان أيضًا.

إعداد الحدث

`DataReactionEventConfig` نوع اتّحادي مُميَّز (discriminated union) مفهرس بحسب نوع الحدث. وفيما يلي أنواع الإعدادات التي يضبطها مطوّر التكامل غالبًا:

- `DataEntryCreatedEventConfig` – يُطلَق عند الإنشاء.
- `DataEntryUpdatedEventConfig` – `fields[]` (مطابقة قيم محدّدة من `fullNormalizedName` ؛ فارغة = أي حقل) و `relatedFields[]` للتغييرات المرصودة عبر المراجع.
- `DataEntryArchivedEventConfig` – يُطلَق عند الأرشفة.
- `CustomEventConfig` – إجراء مخصّص (المشغّل **إجراء مخصص** لدى الباني): حقل `customActionType` إلزامي (`UI` أو `External` أو `Local` – `CustomActionTypeEnum` ؛ وقد حلّ محلّ الراية التي أُوقف استخدامها `uiEvent` `Since` `1.8`)، وراية `contextual` (قيمتها الافتراضية `true` ، ولا تؤثّر في وقت التشغيل حتى الآن)، وإعداد اختياري لمربّع الحوار `askUser` (كيان مضمَّن، أو كيان خارجي ومعرض)، ورسائل `messages` للبدء/النجاح.
- `WorkflowStepReachedEventConfig` / `WorkflowStepExitedEventConfig` / `WorkflowActionExecutedEventConfig` – يُحدَّد نطاقها باستخدام `steps[]` (`workflowId` + `stepId` ، وقد يأخذ `stepId` القيمة الحرفية `'ADHOC_STEP'`)، وتُقيَّد إعدادات الخروج والإجراء كذلك بمطابقة `actions[]` .
- `SLATimeExceededEventConfig` – `occurrencesConfig` (`startingAfter` و `repeatAfter` بالثواني، و `repetitions` عدد) إلى جانب تقييد نطاق الخطوة.
- `TimerEventConfig` – `timerOptions` مع طابع زمني `oneTime` أو تكرار `repeat` يحدّده تعبير cron في `expression` (+ المنطقة الزمنية / البدء / الانتهاء).
- `ReactionSucceededEventConfig` / `ReactionFailedEventConfig` – تشغيل متسلسل بعد تفاعل آخر باستخدام `entityId` + `triggerId` .
- `FileGeneratedEventConfig` – تُجرى المطابقة باستخدام `template` .
- `EventAnnouncedEventConfig` – `entityId` + `announcementId` لحدث مُعلَن.

يرث كل إعداد من `BaseEventConfig` ، الذي يحمل `relatedInfo` (`relatedEntity` + `relatedFieldPath` ، وتكون قيمته null لكيان التفاعل نفسه) و `triggerContextId` . **Since 1.8**، أصبح `triggerContextId` ضمن إعداد الحدث (إذ يملك هذا الإعداد كيان السياق)، بعد نقله إليه من خيارات كل إجراء على حدة.

الإجراءات

يتضمّن `DataReactionAction` حقل `actionType` (`AutomatedActionTypeEnum`) وحقل `options` من نوع اتّحادي محدّد (`DataReactionActionOptions`). والإجراءات هي ما ينفّذه التفاعل؛ أمّا تعداد `actionType` فيفهرس ما تستطيع المنصّة تنفيذه تلقائيًّا: إنشاء/تحديث/أرشفة الكيانات والمُدخلات، ونقل خطوات سير العمل، وإرسال البريد/الرسائل النصية/الإشعارات/إشعارات push، واستدعاء واجهة برمجة تطبيقات خارجية (API)، وتشغيل وظيفة مخصّصة، وإعلان حدث، وإخطار Zapier، وغيرها. انظر جدول التعداد أدناه.

ناقل الرسائل

يعتمد أسس داخليًّا على ناقل رسائل (`@node-ts/bus`) وفق نمط **أمر/حدث (command/event)**:

- **الأوامر** طلبات بصيغة الأمر: «افعل هذا». ومن أمثلتها إنشاء المُدخلات أو تحديثها أو تعديلها جزئيًّا أو أرشفتها أو استعادتها أو تحديثها دفعيًّا؛ أو تنفيذ إجراء سير عمل أو تغيير خطوة سير عمل؛ أو جدولة مهمّة؛ أو إخطار تكامل خارجي.
- **الأحداث** وقائع: «حدث هذا». ومن أمثلتها إنشاء مُدخل أو تحديثه أو حذفه أو أرشفته أو استعادته؛ أو بلوغ خطوة سير عمل أو الخروج منها؛ أو تغيّر حقّ وصول؛ أو توليد رمز API؛ أو انتهاء مهمّة خلفية؛ أو إطلاق مهمّة مجدولة.

لا تحتاج إلى الناقل لاستخدام أسس، لكن الإلمام بمفرداته مفيد، إذ تشكّل أساس الواجهات التي *تتعامل معها* فعلًا: تشترك **التفاعلات** في أحداث المجال وتُصدر أوامر لتنفيذ إجراءاتها (واجهة الأتمتة دون رماز)، وتتمثّل **التحديثات الفورية** في إعادة نشر أسس أحداثًا مختارة من المجال إلى عميلك عبر Socket.IO.

تصنيف الرسائل

تُصدَّر ثلاث عائلات من الرسائل من `@osos/shared/messages` :

العائلة	الفئة الأساس	الاتجاه / الدلالة	مكان الإعلان
Events	`Event` (`@node-ts/bus-messages`)	«حدث شيء ما»: يُنشَر بعد وقوع حدث، ويُوزَّع (fan-out) على أي عدد من المشتركين. وهذه هي المفردات الأساسية التي يتفاعل معها مطوّر التكامل.	`messages/events`
Commands	`Command`	«افعل هذا»: تعليمة موجَّهة إلى الخدمة المسؤولة عن الإجراء، ويعالجها معالِج واحد فقط.	`messages/commands`
Socket messages	`BaseSocketMessage`	مظاريف تُنقَل إلى عملاء الواجهة المتّصلين عبر اتصال socket الآني (مثل اكتمال المهمّة/الأمر، وأحداث الواجهة المخصّصة).	`messages/socket-messages`

يحمل كل حدث وأمر معرّفًا نصيًّا ثابتًا وإصدار مخطَّط (schema version):

```
export class DataEntryCreatedEvent extends Event {
  static readonly NAME = 'osos/data-entry/data-entry-created';
  readonly $name = DataEntryCreatedEvent.NAME;
  readonly $version = 1;
  constructor(
    readonly dataEntry: DataEntry,
    readonly userId: string,
    readonly normalizedEntity: { normalizedName: string; name: LocalizedString; type: DataEntityTypeEnum; moduleId: st
ring },
    readonly valueDiff?: UpdateAdditionalData[],
    readonly isSystemModification?: boolean,
  ) { super(); }
}
```

- `$name` هو الاسم المنقول في الرسائل، ضمن فضاء أسماء بالصيغة `osos/<domain>/<event>`. اعتمد في المطابقة على هذه السلسلة النصية، ولا تعتمد على اسم الفئة مطلقًا.
- `$version` هو إصدار مخطَّط الحمولة. وحتى كتابة هذه السطور، يحمل **كل** حدث وأمر `$version = 1`.
- تمثّل معاملات constructor الحمولة، وجميع الحقول `readonly`.

> ملاحظة: لا تتطابق سلسلة `$name` واسم ملف الفئة دائمًا، وتتضمّن بعض الأسماء أخطاء إملائية تاريخية أُبقيت كما هي حفاظًا على توافق الرسائل المنقولة (`osos/scheduler/scheduled-job-triggerd`، `osos/reaction/event-announced`). انسخ دائمًا ثابت `NAME` كما يرد في المصدر تمامًا.

سمات الرسالة (بيانات المظروف الوصفية)

إلى جانب الحمولة، تحمل كل رسالة مُرسَلة مظروفًا من السمات. ويُعرَّف له شكلان في `@osos/shared`.

`BaseMessageAttributes`:

الحقل	النوع	المعنى
`messageId`	`string`	المعرّف الفريد لهذه النسخة من الرسالة.
`instanceId`	`string?`	يربط مهمّة/أمرًا مُرسَلًا برسالة اكتماله.

`BaseStickyAttributes` – تنتقل السمات *اللاصقة* (sticky) عبر سلسلة السببية (من الرسالة المُشغِّلة إلى كل رسالة تتسبّب فيها):

الحقل	النوع	المعنى
`userId`	`string?`	المستخدم الفاعل، الذي يُمرَّر معرّفه عبر سلسلة التفاعل بأكملها.
`spaceId`	`string?`	المساحة (النطاق الفرعي للقاطن) التي وقع فيها النشاط.

يستقبل المعالِج الحمولة والسمات معًا:

```
async handle(
  event: AccessRightChangedEvent,
  messageAttributes: MessageAttributes<BaseMessageAttributes>,
): Promise<void> { /* ... */ }
```

> تحمل أحداث كثيرة *أيضًا* `userId` في حمولتها الخاصة (مثل `DataEntryCreatedEvent.userId`). ويشير الرماز المصدري إلى أنّ `stickyAttributes.userId` يحلّ محلّه تدريجيًّا. وفي الوقت الحالي، استخدم `userId` الوارد في الحمولة متى وُجد، وارجع إلى السمة اللاصقة عند غيابه.

التعدادات الأساسية

تظهر هذه التعدادات ذات القيم الصحيحة/النصية (integer/string enums) في جميع حمولات الأحداث والنماذج. والقيم الرقمية هي القيم المنقولة فعليًّا؛ فلا تفترض ترتيبها، وانسخها كما هي.

`EventTypeEnum` – أنواع مشغّلات التفاعل

الاسم	القيمة
`DataEntryCreated`	0
`DataEntryUpdated`	1

الاسم	القيمة
`DataEntryArchived`	2
`WorkflowActionExecuted`	3
`WorkflowStepReached`	4
`WorkflowStepExited`	5
`Timer`	6
`CustomEvent`	7
`ReactionSucceeded`	8
`FileGenerated`	9
`ReactionFailed`	10
`EventAnnounced`	11
`DataEntryRestored`	12
`ZapierEvent`	20
`SlaTimeExceeded`	100

`AutomatedActionTypeEnum` – أنواع إجراءات التفاعل

الاسم	القيمة	الاسم	القيمة
`CreateDataEntity`	0	`MoveWorkflowToStep`	20
`UpdateDataEntity`	1	`ExecuteWorkflowAction`	21
`ArchiveDataEntity`	2	`ChangeCurrentStepOwner`	22
`CreateDataEntry`	10	`UserActivationToggle`	30
`UpdateDataEntry`	11	`Since 1.10` `SetPermission`	31
`ArchiveDataEntry`	12	`Since 1.10` `RemovePermission`	32
`CreateBlockItem`	13	`CallApi`	90
`UpdateBlockItem`	14	`SendNotification`	100
`DeleteBlockItem`	15	`SendEmail`	101
`DeleteArchivedEntry`	16	`SendSms`	102
		`GenerateFileFromTemplate`	103
		`ExecuteCustomFunction`	104
		`AnnounceEvent`	105
		`Since 1.8` `ExecuteCustomAction`	106
		`Since 1.10` `SendWhatsApp`	107
		`NotifyZapier`	200

يظلّ `CreateDataEntity` (0) و`UpdateDataEntity` (1) مُعلَنين، لكن `Since 1.10` يُرفَض عند الحفظ أي تفاعل يتضمّن أيًّا منهما، ويُتخطّى عند التشغيل التفاعل المخزَّن الذي يتضمّن أيًّا منهما – انظر أنواع الإجراءات [بناء التطبيقات › التفاعلات والأتمتة].

يتضمّن `SendWhatsApp` (107) `Since 1.10` الخيارات `SendWhatsAppActionOptions`: `recipients` (يُحدَّد المستلمون بالطريقة نفسها المتّبعة عند إرسال SMS)، و`templateName` (قالب Meta معتمد)، و`language` (رمز لغة القالب، مثل `en` أو

ar)، و parameters (TokenSource[] : يملأ العنصر ذو الفهرس *i* الموضع {{i+1}} ، وقد تكون المصفوفة فارغة). سلوك الإجراء عند تشغيله موثّق في التفاعلات والأتمتة [بناء التطبيقات]؛ وتُجلَب قائمة القوالب المعتمدة من واجهة قوالب WhatsApp.

تعدادات أخرى شائعة الظهور

Enum	القيم
DataEntryStateEnum	Archived = 2 ، Active = 1 ، Draft = 0
DataEntityStateEnum	Archived = 1 ، Active = 0
EntryTypeEnum	Virtual = 1 ، Normal = 0
LifeCycleStatusEnum	Completed = 1 ، Active = 0 ، Failed = -1 (إضافة إلى undefined = ليس ضمن سير عمل)
DataEntryInvalidReasonEnum	Invalid = 1 ، Obsolete = 0
EventExecutorTypeEnum	EXECUTOR ، MANAGER ، SYSTEM
DataReactionTriggerTypeEnum	SLATrigger = 1 ، NormalTrigger = 0
TriggerSourceEntityTypeEnum	AbsoluteEntity = 2 ، Relation = 1 ، Self = 0
FunctionTypeEnum	VALUE_VALIDATOR ، STANDALONE_VALIDATOR ، INTERCEPTOR ، REACTION_CUSTOM_ACTION
FunctionStatusEnum	DELETED ، PROVISIONED ، PROVISIONING ، BUILD_FAILED

التعداد DataEntityTypeEnum كبير؛ وأكثر القيم التي يصادفها مطوّرو التكامل هي Custom = 0 ، و Service = 2 ، و Request = 3، و Notification = 7 ، و Task = 8 ، و User = 9 ، و Group = 10 ، وقيم النطاق النظامي (System = 40 ، Core = 30 ، VirtualProxy = 50 ، VirtualAggregator = 52 ، Space = 60).

فهرس الأحداث

قائمة كاملة بالأحداث المنشورة، مجمّعة حسب المجال. يسرد عمود الحمولة معاملات constructor (الحمولة المنقولة)، وتشير ? إلى الحقول الاختيارية.

الكيانات (دورة حياة المخطّط)

Event ($name)	Payload
osos/data-entity/data-entity-created	userId ، dataEntity: DataEntity
osos/data-entity/data-entity-updated	dataEntity ، userId ، requiresReIndex? ، workflowStatus? ، entityChanges?[] ، rawDiff? ، isSystemModification?
osos/data-entity/data-entity-archived	dataEntity ، userId
osos/data-entity/data-entity-version-changed	dataEntity ، oldVersion ، newVersion ، userId
osos/data-entity/data-entity-deleted Since 1.10	dataEntity (يُقرأ قبل الحذف النهائي)، moduleNormalizedName ، userId – انظر أدناه ، owned: DeletedEntityOwnedIds
osos/data-entity-field-meta/data-entity-updated	dataEntity ، changedMeta (mainTitle / mainPicture)، userId

يبلّغ DataEntityUpdatedEvent.entityChanges[] عن التغييرات الهيكلية في الحقول باستخدام التعداد EntityChanges : FIELD_DELETED ، FIELD_ADDED ، FIELD_REQUIRED_VALIDATION_CHANGED ، FIELD_RENDER_OPTIONS_CHANGED ، FIELD_INTERNAL_COLLECTION_CHANGED ، VIRTUAL_ENTITY_OPTIONS_CHANGED ، TREE_STRUCTURE_DELETED . أمّا rawDiff فيمثّل الفرق الكامل الناتج عن مكتبة just-diff .

`DataEntityDeletedEvent` – الحذف النهائي لكيان `Since 1.10`

يُنشَر في ختام **الحذف النهائي لكيان** الذي يُجريه المدير (`DELETE /v1/entities/{id}/permanent`)، قبيل حذف مستند الكيان نفسه. ويسرد `owned` معرّفات العناصر التي حُذفت مع الكيان، مصنّفةً بحسب النوع: المعارض، والاستعلامات، والمجموعات، والاستعلامات الإحصائية، والعلاقات، والتجميعات، ومجموعات المرشّحات السريعة، ومجموعات الوسوم، والتقارير، وأزرار النظام، وقطع العدّادات، ولوحات المعلومات، والصفحات، والوظائف. ولا تُدرَج التفاعلات والإعلانات، إذ تعثر عليها الخدمات المعنيّة بها باستخدام معرّف الكيان.

- **يجب أن تكون معالجة المستهلكين متساوية الأثر (idempotent).** فقد تؤدي إعادة الحذف إلى نشر الحدث مرة أخرى.
- **لا يُنشر أيّ حدث أرشفة** بسبب الحذف النهائي، ولذلك لا تُطلَق تفاعلات الأرشفة الخاصة بالكيان.
- تسجّله خدمة التدقيق بالاسم `DATA_ENTITY_DELETED` .

المدخلات (دورة حياة المُدخل)

Event (`$name`)	Payload
`osos/data-entry/data-entry-created`	`isSystemModification?` , `valueDiff?` , `normalizedEntity` , `userId` , `dataEntry: DataEntry`
`osos/data-entry/data-entry-updated`	`valueDiff?` , `valueChanges?[]` , `metaDataChanges` , `normalizedEntity` , `userId` , `dataEntry` `refreshedEntities?` , `isSystemModification?`
`osos/data-entry/data-entry-deleted`	`dataEntry`
`osos/data-entry/data-entry-archived`	`userId` , `dataEntry`
`osos/data-entry/data-entry-restored`	`userId` , `dataEntry`
`osos/data-entry/data-entry-invalidated`	`normalizedEntity` , `userId` , `dataEntry`
`osos/data-entry/data-entry-permanently-deleted`	`entryNameBefore` , `stateBefore` , `userId` , `dataEntry`
`osos/data-entry/data-entries-archived`	`dataEntries: Pick<DataEntry,'id'>`
`osos/data-entry/data-entries-permanently-deleted`	`userId` , `items: PermanentlyDeletedEntrySnapshot[]`
`osos/data-entry/draft-data-entry-upserted`	`rawDiff` , `userId` , `dataEntry`
`osos/data-entry/draft-data-entry-deleted`	`userId` , `draftDataEntryId` , `dataEntityId`

يمثّل `DataEntryUpdatedEvent.valueChanges[]` (`IValueChangesEntryUpdated`) مجموعة التغييرات على مستوى الحقل: يتضمّن كل عنصر `fullNormalizedName` ، و `fieldId` اختياريًا، و `mapConfig` اختياريًا (مصدر/مرجع الحقل المُطابَق/ `updateMode`)، و `isTreeStructureField?` . وتُطابَق هذه التغييرات مع إعداد `fields[]` لتفاعل `DataEntryUpdated` . أمّا `refreshedEntities` فهي قائمة داخلية لمنع التكرار، يستخدمها معالِج تحديث المُجمِّع لتفادي إعادة تحديث كيان سبق تحديثه ضمن العملية نفسها. و**Since 1.6**، يُعلِن `DataEntryInvalidatedEvent` عن المُدخلات التي أبطلها تغيير في المخطَّط. أمّا `DataEntryRestoredEvent` وعائلة أحداث الحذف الدائم فهي **Since 1.8**.

الكتابة التي يُحذف مُدخلها قبل إعادة قراءته لا تنشر شيئًا `Since dev` . بعد الإنشاء أو التحديث أو التعديل الجزئي أو الترقية، تعيد data-service قراءة المُدخل لتُدرج نسخته المحلولة في `DataEntryCreatedEvent` / `DataEntryUpdatedEvent` . وكان

الحدث يُنشَر بقيمة غير معرَّفة للحقل `dataEntry` إذا سبق حذفٌ متزامن إعادةَ القراءة، فتُخفق معالجته لدى كل مستهلك وتُعاد المحاولة. أمّا الآن فتُبلَّغ عملية الكتابة على أنها إخفاق لا تُعاد محاولته، ولا يُنشر **أي** حدث إنشاء/تحديث؛ وللحذف حدثه الخاص `DataEntryDeletedEvent`. ولم يتغيّر عقد الحمولة (`dataEntry` حاضر دائمًا في الحدث الذي يُنشر)، لكن عملية الكتابة قد تنتهي الآن دون أي حدث، لذا لا تنتظر حدث تحديث لمُدخل ربما حُذف. كذلك تتخطّى audit-service أي حدث تحديث يصل دون `dataEntry` (من ناشر أقدم أو رسالة مُعادة)، مع إصدار تحذير.

المجموعات والعلاقات والمعارض والاستعلامات والوحدات

Event ($name)	Payload
`osos/data-collections/data-collection-created` / `-updated` / `-archived`	`dataCollection: DataCollections`، `userId` (`-updated` يضيف `rawDiff?`)
`osos/data-relation/data-relation-created` / `-updated` / `-deleted`	`dataRelation: IDataRelation`، `userId` (`-updated` يضيف `rawDiff?`)
`osos/data-view/data-view-created` / `-updated` / `-deleted`	`dataView: IDataViews`، `userId`، إضافة إلى (published/pinned)، `isSystemModification?`، `rawDiff?` في `-updated`، `permissions?`، `changedStatus?`
`osos/data-query/data-query-created` / `-updated` / `-deleted`	`dataQuery: DataQuery` (`-updated` يضيف `criteriaChanged: boolean`)، `userId`، `rawDiff?`
`osos/data-module/data-module-created` / `-updated` / `-deleted`	`dataModule: DataModule`، `userId` (`-updated` يضيف `changes?: Record<string, {oldValue,newValue}>`)
`osos/generic-view/generic-view-created` / `-updated` / `-deleted`	`genericView: IGenericView`، `userId`، `permissions?`، `changedStatus?`، `defaultPageForSpaces?`، `rawDiff?`
`osos/statistic-query/statistic-query-created` / `-updated` / `-deleted`	`statisticQuery: IStatisticQuery`
`osos/report/report-deleted`	`reportId`، `userId`

> التقارير ميزة نطاقُها **مقتصر على الويب**: فهي غير متاحة في تطبيق الجوال، ولذلك يُعامَل أي رابط عميق لتقرير يظهر لعميل جوال بوصفه غير مدعوم.

التفاعلات والمشغّلات والأحداث المخصّصة

Event ($name)	Payload
`osos/reaction/reaction-created` / `-deleted`	`dataReaction: DataReaction`، `userId`
`osos/reaction/reaction-updated`	`dataReaction`، `previousDataReaction`، `userId`، `rawDiff?`
`osos/reaction/announcement-created` / `-deleted`	`announcement: Announcement`، `userId`
`osos/reaction/announcement-updated`	`announcement`، `previousAnnouncement`، `userId`
`osos/reaction/event-announced`	`dataEntityId`، `dataEntry`، `announcementId`، `userId`، `announcementEntry?`
`osos/triggers/custom-event-triggered`	`dataEntityId`، `dataEntries: ReferenceProperty[]`، `userId`، `triggerId`، `timestamp`، `askUserValue?`، `askUserDataEntry?`
`osos/triggers/trigger-executed`	`dataEntityId`، `triggerId`، `triggerHistoryId`
`osos/triggers/email-generated`	`subject`، `body`، `to`، `cc`، `bcc`، `link?`، `attachments?`، `relatedTriggerExecutionId?`، `eventType?`، `from?`، `dataEntry?`

Event ($name)	Payload
osos/triggers/sms-generated	relatedTriggerExecutionId? , link? , recipients , body
osos/triggers/whatsapp-generated — Since 1.10	parameters: string[] , recipients: string[] , template: {name, language, category: 'authentication' | 'other'} , relatedTriggerExecutionId? , (authentication : [code] لـ)

لاحظ أنّ اسم الفئة `EmailGeneratedEvent` يقابل `$name` بالقيمة `osos/triggers/email-generated`، وأنّ `SMSGeneratedEvent` يقابل `osos/triggers/sms-generated`.

رسائل WhatsApp `Since 1.10`. يطلب `WhatsAppGeneratedEvent` (`osos/triggers/whatsapp-generated`) من notification-service إرسال قالب WhatsApp معتمد واحد. وتنشره خدمتان: reaction-service لإجراء «إرسال WhatsApp»، وusers-service لإرسال رموز التحقق لمرة واحدة عبر WhatsApp أثناء تسجيل الدخول متعدّد العوامل. وتستجيب notification-service بنشر حدث `osos/events/whatsapp-send-status` واحد لكل مستلم، وتسجّله logging-service. ولا يُضمَن وصول هذا النشر؛ إذ يُسجَّل فشل النشر ولا تُعاد محاولته أبدًا، لأنّ إعادة التسليم ستؤدي إلى إرسال الرسالة إلى جميع المستلمين مرة أخرى. أمّا استدعاءات التسليم اللاحقة من Meta فتصل إلى logging-service عبر webhook العام الخاص بها (واجهة WhatsApp)، فتُحدِّث صفّ السجل نفسه باستخدام معرّف رسالة المزوّد، و**لا** يُعاد نشرها على الناقل.

تخزين سجلّ التفاعلات `Since 1.10`. أصبح صفّ `TriggerHistory` الذي يشير إليه `trigger-executed` (صفّ لكل تشغيل تفاعل) مفهرسًا لدعم مرشّحات سجلّ التفاعلات، وأصبح حقل `result` فيه اختياريًا بقيمة افتراضية `[]` (تشغيل لم يُسجَّل فيه أيّ إجراء). كذلك أصبح صفّ سجلّ المُدخل، الذي تكتبه رسالة إنشاء / تحديث مُدخل، فريدًا لكل رسالة مصدر، بحيث لا تؤدي إعادة تسليم الرسالة إلى كتابته أكثر من مرة واحدة. التفاصيل: فهارس سجلّ التفاعلات والتنظيف [بناء التطبيقات › التفاعلات والأتمتة].

سير العمل

Event ($name)	Payload
osos/workflow/step-reached	isAdhoc? , userId , stepId , workflowId , dataEntry
osos/workflow/step-exited	isAdhoc? , action? , userId , stepId , workflowId , dataEntry
osos/workflow/action-executed	isAdhoc? , userId , action , stepId , workflowId , dataEntry
osos/workflow/ended	userId , workflowId , dataEntry
osos/workflow/workflow-step-owner-changed	additionalData? , doneAt , changes: ReassignOwnerChanges , userId , dataEntry
osos/workflow/adhoc-step-created	userId , adhocStep: WorkflowStep , dataEntry
osos/sla/time-exceeded	holidays? , workingHours , dueDate , receivedDate , isAdhoc , userId , stepId , workflowId , dataEntry

التكاملات والوظائف والملفات والمهام والمراسلة

Event ($name)	Payload
osos/events/zapier-event-triggered	eventOutput , eventType , zapId , entityId
osos/functions/function-provisioning-status-update	details: { url } , status: FunctionStatusEnum , functionModel: FunctionModel

Event ($name)	Payload
`osos/files/file-generated`	`dataEntryIds` ، `dataEntityId` ، `exportType` ، `template` ، `fileName` ، `file: IFile`
`osos/app-tokens/app-token-generated`	`email` ، `userId` ، `tokenName` ، `token`
`osos/app-tokens/function-app-token-rotated` Since dev	`functionId` ، و `userId` (المستخدم الخاص بالوظيفة)، و `token` (الرمز الخام الجديد: لا تسجّله ولا تخزّنه أبدًا). يُنشر بدلًا من أحداث الرموز المُرسلة بالبريد عند إعادة توليد رمز المستخدم الخاص بالوظيفة؛ وتخزّنه functions-service بوصفه السرّ `OSOS_API_TOKEN` للوظيفة، وتسجّل audit-service الحدث `FUNCTION_APP_TOKEN_ROTATED` دون الرمز. انظر هوية الوظيفة الخاصة.
`osos/events/job-executed`	`result{success,error?,result?}` ، `messages{startMessage,successMessage}` ، `userId` ، `instanceId` ، `messageId`
`osos/events/email-sent`	*(بلا حمولة)*
`osos/events/email-send-status`	تقرير التسليم: `from` ، `to` ، `cc` ، `bcc` ، `subject` ، `body` ، `status:'success'` `
`osos/events/sms-send-status`	`to` ، `body` ، `status` ، `strategy` ، `relatedTriggerExecutionId` ، `errorMessage?` ، `smsProviderResponse?` ، …
`osos/events/push-notification-sent`	`notificationId: UUID`
`osos/events/whatsapp-send-status` Since 1.10	`payload`: `{to, status, providerMessageId?, messageId?, spaceId, templateName, templateLanguage, category, parameters?, errorCode?, errorMessage?, relatedTriggerExecutionId?, timestamp}` تكون قيمة `status` إحدى القيم `accepted` أو `sent` أو `delivered` أو `read` أو `failed` أو `mocked` (إرسال جرى تخطّيه في وضع الاختبار). يُحذف `parameters` في قوالب `authentication` لأنه يحمل الرمز.
`osos/icons/icons-updated`	`timestamp: number`
`osos/packaging/package-exported`	`packageName` ، `resourceCount` ، `resourceTypes[]` ، `description?`
`osos/packaging/package-imported`	`sessionId` ، `created` ، `overwritten` ، `skipped` ، `failed` ، `packageFileName?`
`osos/packaging/package-changed` Since 1.10	`packageId` – تنشره packaging-service عند تغيّر حالة مراجعة الحزمة أو حالتها في المتجر. وهو إشارة لواجهة المستخدم فقط؛ انظر رسالة Socket.
`osos/dependency/rebuild-dependency-requested`	`reason` ، `resourceTypes?`

يحدّد `FileGeneratedEvent.exportType` صيغة المستند المُنتَج، وتُولَّد ملفات تصدير البيانات بصيغة **XLSX** أو **PDF**.

الوصول والمساحات والمستخدمون

Event ($name)	Payload
`osos/access-rights/access-rights-changed`	`entitlementType: EntitlementEnum` ، `entitlementId: string` `
`osos/spaces/space-created-event`	`space: SpaceDataEntry` ، `userId` ، `valueDiff?` ، `isSystemModification?`
`osos/spaces/space-updated-event`	`space` ، `userId` ، `rawDiff?` ، `isSystemModification?`
`space-deactivated-event` / `osos/spaces/space-activated-event`	`space` ، `userId`
`end-user-deactivated` / `osos/spaces/end-user-activated`	`space` ، `userId` ، `doneAt`
`osos/user/user-signed-up`	`userId` ، `value` ، `defaultGroup` ، `defaultSpace` ، …

Event ($name)	Payload
`osos/user/user-signin-attempted`	`outcome:'success'` , `timestamp` , `userId`
`osos/user/user-account-activated` / `-deactivated` / `-locked`	`userId`
`osos/user/user-external-id-linked`	`userId` , `externalId` , `syncPatches?`
`osos/user/user-password-changed` / `reset` / `-reset-by-system` / `-reset-completed`	`userId` (+ email/token إلخ بحسب الصيغة)

مفردات الأوامر

الأوامر تعليمات يوجّهها مُرسِل واحد إلى معالِج واحد، وتتولّى معالجتها الخدمة المالكة. ويُصدِر مطوّر التكامل هذه الأوامر (مباشرةً أو عبر إجراءات التفاعل) بدلًا من الاكتفاء بمراقبتها. وفيما يلي قائمة الأسماء الكاملة من `messages/commands` :

المدخلات – `create-data-entry` , `update-data-entry` , `patch-data-entry` , `upsert-data-entry` , `upsert-draft-data-entry` , `delete-data-entry` , `archive-data-entry` / `archive-data-entries` , `restore-data-entry` / `restore-data-entries` , `permanently-delete-data-entry` / `permanently-delete-data-entries` , `upgrade-data-entry` , `bulk-patch-data-entries` , `bulk-update-data-entries` , `bulk-update-entries` , `bulk-upgrade-data-entries` والصيغ الجماعية (`data-entries`).

الكيانات – `create-data-entity` , `update-data-entity` , `create-virtual-data-entity` , `update-virtual-data-entity` , `archive-data-entity` , `recalculate-references` , `migrate-history-for-entity-entries` .

سير العمل – `change-step` , `change-step-owner` , `execute-action` , `insert-adhoc-step` .

إسقاط التفويض (Authorization projection) – `Since 1.10` `osos/authz/reconcile-authz-tenant` (`ReconcileAuthzTenantCommand`): `mode` (`repair` يكتب الاختلافات، و `report` يحصيها فقط)، و `trigger` اختياري (`nightly` أو `on-demand`). يُرسَل في المساحة الجذر بصفته مستخدم النظام، ويوفّق إسقاط OpenFGA للقاطن المُرسِل؛ أمّا إرساله عند الطلب فيجري عبر مسار داخلي مخصّص للخدمات فقط. وهو جزء من البنية الداخلية، ونذكره هنا لتتمكّن من التعرّف عليه في آثار الناقل.

التفاعلات / التكاملات / المراسلة / الجدولة – `osos/reaction/create-reaction` , `osos/integrations/notify-zapier` , `osos/notifications/send-push-notification` , `osos/sms/send-sms` , `osos/data/generate-file-from-template` , `osos/shared/export-template-in-background` , `osos/scheduler/schedule-job` , `osos/scheduler/schedule-interval-job` , `osos/scheduler/remove-scheduled-job` .

أمثلة على صيغ الأوامر:

osos/data-entry/create-data-entry

```
new CreateDataEntryCommand(
  store: string,
  dataEntry: Pick<DataEntry, 'id' | 'value' | 'specificationVersion' | 'dataEntity'>,
  userId: UUID,
  isAdminModification: boolean,
  skipValidation?: boolean,
  viewId?: UUID,
  systemButtonId?: UUID,
  isSystemModification?: boolean,
);

// osos/notifications/send-push-notification
new SendPushNotificationCommand(
  tokens: string[], title: string, content: string, data: any, notificationId?: UUID,
);
```

لاحظ أنّ الأوامر تحمل عمدًا مجموعات فرعية من حقول `DataEntity` / `DataEntry` باستخدام `Pick<...>`، لا النموذج الكامل: فلا يُنقل من الحقول إلا ما يحتاج إليه المعالِج.

رسائل الـ Socket (الآنية)

تغلّف رسائل الـ socket النتائج المُعاد توجيهها إلى عملاء الواجهة المتّصلين (`messages/socket-messages`). وهي تُطبِّق `BaseSocketMessage` (`{ $name, $version }`)، وتُضمِّن مظروف الارتباط الكامل في جسم الرسالة، على خلاف أحداث الناقل:

Message ($name)	الغرض
`osos/infrastructure/command-executed`	انتهى أمر مُرسَل.
`osos/infrastructure/job-executed`	انتهت مهمة خلفية (تعكس `JobExecutedEvent`؛ تحمل `messageId`، و`instanceId`، و`userId`، و`messages`، و`result`).
`osos/infrastructure/job-progress`	تحديث تقدّم لمهمة قيد التشغيل.
`osos/infrastructure/job-scheduled` / `job-started`	إشعارات دورة حياة المهمة.
`osos/infrastructure/register-service`	مصافحة تسجيل الخدمة.
`osos/triggers/custom-event-triggered`	حدث واجهة مخصّص مُنقَل إلى العملاء.
`packages/changed` `Since 1.10`	من الخادم إلى العميل، بثّ إلى كل sockets القاطن صاحب الحدث حين تتغيّر حالة المراجعة أو المتجر لحزمة. الحمولة `{ $name, $version: 1, packageId }`. تُعيد صفحة الحزمة المفتوحة تحميل نفسها.

تختلف رسالة `packages/changed` عن بقية الرسائل في الجدول في أمرين: تقع فئتها (`PackageChangedMessage`) في `web-socket-service`، لا في `@osos/shared`؛ وهي مشتقّة من حدث على الناقل، وليست منقولة من نتيجة أمر، إذ تنشر packaging-service الحدث `PackageChangedEvent` (`osos/packaging/package-changed`) وتبثّه web-socket-service. ولا تحمل الرسالة أيّ تفاصيل عن المتجر أو المراجعة؛ لذا تعامَل معها بوصفها إشارة إلى «إعادة قراءة الحزمة»، لا بوصفها حالةً تبني عليها. تفاصيل التسليم: الوقت الحقيقي.

يغلّف `CustomEventTriggeredMessage` الحدث `CustomEventTriggeredEvent`، ويضيف إليه `messageId`، و`instanceId`، و`traceId`، و`userId`، و`spaceId`، والحقلين الاختياريين `delegateeId` / `delegationId`؛ وهو مفيد عند تشغيل حدث واجهة

مخصّص بموجب تفويض.

التسليم أثناء وضع القاطن تحت الصيانة `Since 1.10`

في بيئة نشر متعدّدة القاطنين، يمكن وضع قاطن **تحت الصيانة**: في حالة `restoring` أثناء استعادة بياناته من نسخة احتياطية، أو في حالة `moving` حين يعلّق المشغّل عمله لنقله بين الخلايا. وخلال ذلك، لا يعالج مستهلكو الناقل الذين يستخدمون بوّابة الصيانة المشتركة – ناقل رسائل خدمات LoopBack، و `indexing-service`، و `notification-service`، و `users-service`، و `web-socket-service` – رسائل ذلك القاطن ولا يُعيدونها إلى الطابور (إذ كان ذلك سيؤدّي إلى نقل حركة رسائل القاطن كلّها إلى طابور الرسائل الميتة بعد خمس محاولات). بل **يركنونها**: يُقَرّ باستلام الرسالة الأصلية، وتُرسَل نسخة منها إلى `<queue>-parked`، وهو طابور ذو مدة صلاحية ثابتة تُعاد رسائله عند انتهائها إلى الطابور الرئيسي، فتُفحص الرسالة مجدّدًا عند تسليمها التالي.

ما يمكن للمشترك ملاحظته:

- **تأخير لا فقدان، لمعظم العمل.** تُعاد الرسائل المركونة عند انتهاء الصيانة. ويسري هذا على التعليق في حالة `moving` وعلى الاستعادة التي لم تُرجِع البيانات إلى حالة سابقة.
- **فقدان مقصود بعد استعادة أرجعت البيانات إلى الوراء.** الرسالة التي رُكنت أول مرة خلال نافذة `restoring` **تُسقَط** متى أرجعت الاستعادة بيانات القاطن إلى حالة سابقة (أي كانت قيمة `lastRestoredAt` عند بداية النافذة أو بعدها): فهي تصف البيانات كما كانت بعد نقطة الاستعادة، وقد تؤدّي إعادة معالجتها، مثلًا، إلى فهرسة مُدخل حذفته الاستعادة أو تشغيل تفاعل مرتين. تُحصى حالات الإسقاط (`outcome="dropped"`) وتُسجَّل مع معرّف الاستعادة. وعلى أيّ تكامل يعتمد على حدث لإحداث أثر خارجي – webhook أو استدعاء Zapier – أن يتعامل مع الأحداث الصادرة خلال نافذة الاستعادة على أساس احتمال عدم تسليمها مطلقًا.
- **حدّ زمني.** تنتظر الرسالة ساعتين على الأكثر افتراضيًا، أو وفق الحدّ الخاص بعملية الصيانة نفسها (تخضع عملية النقل لحدّ `moving` الخاص بالخلية)؛ وبعد انقضاء الحدّ، تُنقَل الرسالة إلى طابور الرسائل الميتة الفعلي بالسبب `restore-timeout`.
- **رسائل مستوى الخلية** التي لا تحدّد قاطنًا لا تُركن مطلقًا، ولا تُطبَّق حالة الصيانة في تثبيتات القاطن الواحد.

ويقابل هذه الحالة في واجهة HTTP رفض الطلب بالاستجابة `503 TENANT_IN_MAINTENANCE` مع `Retry-After` – انظر استدعاء واجهة HTTP.

كيف يستهلك المُكامِلون الأحداث

توجد عدة طرق لاستهلاك الأحداث، وكلّها مبنية على مفردات الأحداث نفسها:

1. **التفاعلات/المشغّلات (بلا رمان)** – يضبط الباني `DataReaction` مرتبطًا بـ `EventTypeEnum`، مع مُطابِق `eventConfig`، ومرشّح `criteria` / `query`، وقائمة من `DataReactionAction`. تطابق المنصّة الأحداث المنشورة مع هذه الإعدادات وتشغّل الإجراءات. وهذا هو المسار الأساسي القائم على الإعداد.

2. **الوظائف المخصّصة** – يشغّل إجراء تفاعل من نوع `ExecuteCustomFunction` (`AutomatedActionTypeEnum = 104`) نموذج `FunctionModel` من نوع `FunctionTypeEnum.ReactionCustomAction`. ويمكن أيضًا أن تكون الوظائف من أنواع `Interceptor` أو `StandaloneValidator` أو `ValueValidator`، فتعترض دورة حياة الحفظ/التحرير/الأرشفة (`StandaloneFunctionValidatorTriggerTypeEnum`: `BEFORE_SAVE`، `BEFORE_EDIT`، `BEFORE_ARCHIVE`:

`ValueFunctionValidatorTriggerTypeEnum` : `ON_SAVE`). ويُبلَّغ عن حالة بناء/نشر الوظيفة عبر `FunctionProvisioningStatusUpdateEvent` .

3. **المعالِجات داخل الخدمة (رماز المحرّك)** – تشترك الخدمات الخلفية في الأحداث من خلال تطبيق `Handler<TEvent>` من `@node-ts/bus-core` ، فتضبط `messageType = SomeEvent` وتنفّذ `handle(event, messageAttributes)` . مثال:

```
export class AccessRightChangedHandler implements Handler<AccessRightChangedEvent> {
  handlerName = AccessRightChangedHandler.name;
  messageType = AccessRightChangedEvent;
  async handle(event: AccessRightChangedEvent, attrs: MessageAttributes<BaseMessageAttributes>) { /* ... */ }
}
```

4. **التكاملات الخارجية** – يربط الحدث `ZapierEventTriggered` (`osos/events/zapier-event-triggered`) والأمر `NotifyZapierCommand` نشاط القاطن بتدفّقات Zapier الخارجية؛ ويدعم `AppTokenGeneratedEvent` إصدار رموز API للوصول البرمجي.

في معظم التكاملات، فضّل **الأحداث** لسير العمل التفاعلي و**واجهة HTTP البرمجية** للعمليات القائمة على الطلب/الاستجابة. وإذا كنت تحتاج فقط إلى معرفة أنّ شيئًا ما قد تغيّر، فالقناة الآنية أخفّ من أيّ تكامل أعمق.

ملاحظة حول مراقِبي (observers) الـ data-service

الملفات الموجودة ضمن `backend/services/data-service/src/observers` **ليست** معالِجات لأحداث الناقل؛ بل هي مراقِبات من نوع `LifeCycleObserver` في LoopBack (مُهيِّئات لبدء التشغيل والإيقاف)، مجمّعة ضمن `ObserverGroupsEnum.Preparation` :

Observer	الدور
`ApplicationObserver`	يوقف الناقل إيقافًا سليمًا عند إيقاف التطبيق.
`DomainRedisBooter`	يُهيّئ خبء النطاقات في Redis مسبقًا عند بدء التشغيل (`DomainService.refreshDomainsInRedis`).
`SpaceRedisBooter`	يُهيّئ خبء المساحات في Redis مسبقًا (`SpaceService.refreshSpaceInRedis`).
`IconManifestRedisBooter`	يعيد توليد خبء بيان الأيقونات.
`GeojsonValidatorObserver`	يسجّل مُتحقِّقًا مخصّصًا من GeoJSON `Position` (خط الطول −180..180، خط العرض −90..90).
`LanguagesBooter`	يحمّل مُدخلات اللغات النشطة في التطبيق (ويعود إلى الإنجليزية/العربية الافتراضية إن لم توجد أيّ منها).

أمّا *معالجة* الأحداث الفعلية في data-service فتجري ضمن `data-service/src/handlers/events/**` ، وتُنشر الأحداث من المتحكّمات/الخدمات. وإذا كنت تبحث عن «مَن يتفاعل مع الحدث X»، فابحث في أشجار `handlers/events` ، لا في `observers` .

إدارة الإصدارات والتوافق

- طابِق الأحداث باستخدام القيمة النصية الثابتة لـ `$name` الخاص بها، وتحقّق من `$version` (كلّها حاليًا `1`). وإذا زيدت قيمة `$version` يومًا، فعامِل الحمولة الأقدم بوصفها عقدًا منفصلًا.
- تُضاف الحقول إلى حمولات الأحداث بوصفها معاملات اختيارية (`?`) في نهاية قائمة معاملات المُنشئ (constructor)، ويمكن تجاهلها بأمان إن لم تستخدمها (مثل `rawDiff` ، و `isSystemModification` ، و `refreshedEntities`).
- استخدم `normalizedName` (للكيان، والحقل، والتفاعل) بوصفه معرّفك الثابت في الرماز؛ فقيم `name` المُترجَمة قابلة للتغيير وتختلف باختلاف اللغة.

- يمثّل `rawDiff` في أحداث `*-updated` قائمة عمليات `just-diff` الخام؛ لذا فضّل الملخّصات الأعلى مستوى `valueChanges` / `entityChanges` / `changes` حيثما تتوفّر، واستخدم `rawDiff` فقط عندما تحتاج إلى مجموعة العمليات الدقيقة.

توفّر الميزات

- `DataEntryInvalidatedEvent` – **Since 1.6**.
- `rawDiff` في أحداث التحديث – **Since 1.6**.
- `triggerContextId` في إعداد الحدث (نُقل من خيارات الإجراء إلى مستوى أعلى) – **Since 1.8**.
- `refreshedEntities` في `DataEntryUpdatedEvent` – **Since 1.8**.
- `DataEntryRestoredEvent`، و`EventAnnouncedEvent`، و`DataEntriesPermanentlyDeletedEvent` – **Since 1.8**.
- `DataEntityDeletedEvent` (الحذف النهائي لكيان) – **Since 1.10**.
- نوعا الإجراء `SetPermission` (31) / `RemovePermission` (32) – **Since 1.10**.
- `WhatsAppGeneratedEvent`، و`WhatsAppSendStatusEvent`، و`PackageChangedEvent` (ورسالة الـ Socket `packages/changed`)، و`ReconcileAuthzTenantCommand`، ونوع الإجراء `SendWhatsApp` (107)، وسلوك الركن والإسقاط أثناء الصيانة – **Since 1.10**.
- `FunctionAppTokenRotatedEvent`، وعدم نشر حدث إنشاء/تحديث لمُدخل حُذف قبل إعادة قراءته – **Since dev**.
- `ZapierEventTriggered`، و`FunctionProvisioningStatusUpdateEvent`، و`DraftDataEntryUpsertedEvent` – **Long-standing** (متاحة قبل 1.2)؛ بلا شارة توفّر.

تشغيل المنصّة من المصدر (التطوير المحلي)

هذه الصفحة موجّهة إلى **المساهمين في تطوير المنصّة**، وإلى **مطوّري التكامل** الذين يحتاجون إلى بيئة تجريبية محلّية كاملة بدلًا من قاطن مُستضاف. وتجمع الإجراءات التي تُنفَّذ على الجهاز لتشغيل منصّة أسس **من المصدر** على محطّة عملك: توليد التهيئة الخاصّة بكلّ مطوّر، وتخصيص اسم مضيف حقيقي للنسخة، وتشغيل مجموعة الخدمات الخلفية دفعةً واحدة، واستعادة إمكان تسجيل الدخول لأوّل مرّة، وتشخيص تعثّر اكتمال التشغيل.

أُفردت هذه الصفحة عن بقيّة صفحات هذا القسم عمدًا لتكون مكتفية بذاتها؛ فبينما تصف الصفحات الأخرى البناء **على** منصّة عاملة عبر واجهتها البرمجية وأحداثها ووظائفها، تتناول هذه الصفحة تشغيل المنصّة **نفسها** على حاسوب محمول. أمّا المرجع التشغيلي لإعداد Docker Compose نفسه، بما يشمل الخدمات العاملة وأماكن تشغيلها وبيانات التهيئة الأوّلية والترحيلات، فهو التشغيل المحلي. ولا تزال بعض الخطوات تتطلّب تعديل ملفّات يجب **ألّا** تُودَع في المستودع؛ وقد وُسمت هذه الخطوات بعبارة **للتطوير المحلي فقط**.

> ⚠ **للتطوير المحلي فقط**
>
> حين تطلب منك هذه الصفحة تعديل ملفّ متعقَّب، فالغرض من التعديل هو جعل إعداد موزَّع على أجزاء في جهاز واحد يعمل كما يعمل نشر فعلي. وهذا التعديل ليس إصلاحًا ولا جزءًا من المصدر الرسمي، ويجب ألّا يصل إلى فرع مشترك أبدًا؛ احتفظ به في نسخة عملك، أو في إيداع محلّي تُزيله لاحقًا بإعادة الأساس (rebase).
>
> ولم تعد معظم الإعدادات التي كانت تستلزم هذا التعامل بحاجة إليه: فنقاط النهاية المحلّية، وخريطة مضيف الاسترجاع، وعناوين ردّ الاتصال الاسترجاعية لتسجيل الدخول، أصبحت كلّها **تهيئة** محفوظة في ملفّات يتجاهلها git.

ولّد ملفّات التهيئة المحلّية `Since 1.10`

يتطلّب التشغيل على محطّة العمل ملفّين لا يُدرجان في المستودع عمدًا، ويتولّى سكربت إنشاءهما نيابةً عنك. ويُشغّله `pnpm install` عبر `postinstall` (ويُتخطّى تشغيله حين يكون `CI` مضبوطًا)، ويمكنك إعادة تشغيله متى شئت:

```
pnpm run setup:local
```

هذا السكربت **لا يستبدل شيئًا أبدًا**، لذا فإنّ إعادة تشغيله آمنة ولن تُلغي تعديلاتك. ويُنشئ الملفّين التاليين:

الملفّ الذي يُنشئه	منقولًا عن	لماذا
`config/app-config.local.json`	`config/app-config.json`	نقاط النهاية التي تستدعيها البوّابة الإلكترونية – نسخة خاصّة بك يمكنك تعديلها بحرّية
`services/data-service/.env.migration`	`services/data-service/.env.migration.example`	هويّة مسؤول النظام الأعلى التي يُنشئها الترحيل ضمن بيانات التهيئة الأوّلية

لماذا يبقى `app-config.json` متعقَّبًا، وكيف تتجاوزه `Since 1.10`

قد تبدو إزالة `config/app-config.json` من التعقّب وترك كلّ مطوّر يحتفظ بنسخته خطوة بديهية، لكنّها غير صحيحة هنا، ومن الضروري فهم السبب لأنّ العطل يحدث دون ظهور خطأ؛ إذ إنّ Compose **يُنشئ مجلّدًا** إذا كان مسار مصدر التركيب غير موجود. وإذا جرى تجاوز خطوة النسخ، فسيُنشأ *مجلّد* في الموضع الذي تتوقّع فيه البوّابة الإلكترونية ملفّ تهيئتها، فتصل إليها تهيئة معطوبة دون ظهور أيّ خطأ في أيّ موضع، ولا يمنع ضبط `bind.create_host_path: false` حدوث ذلك.

لذلك يبقى الملفّ متعقِّبًا، ويُحدَّد مسار مصدر التركيب بمتغيّر بدلًا من ذلك:

```
- ${OSOS_APP_CONFIG:-./config/app-config.json}:/app/app-config.json
```

وحين لا يُضبط المتغيّر – كما هي الحال في جميع الإعدادات القائمة وبيئات التكامل المستمرّ – يُستخدَم المسار نفسه الذي كان يُستعمَل دائمًا. ولتوجيه البوّابة الإلكترونية إلى نقاط النهاية الخاصّة بك، اضبط المتغيّر في `backend/.env` وعدّل النسخة المحلّية بدلًا من الملفّ المتعقَّب:

```
OSOS_APP_CONFIG=./config/app-config.local.json
```

وعند ضبط هذا المتغيّر، تكون النسخة المحلّية هي الملفّ الذي تعدّله في كلّ خطوة لاحقة تطلب ضبط `API_BASE_URL` و`AUTHORITY` والإعدادات المماثلة.

> **البوّابة الإلكترونية في 1.10 تقرأ من هذا الملف أقلّ** `Since 1.10` **.** تنسخ البوّابة الإلكترونية الحالية محتويات `app-config.json` وفق قائمة سماح، وتتجاهل `API_BASE_URL` و`SOCKET_BASE_URL` و`APP_URL` و`AUTHORITY` و`CLIENT_ID`: إذ تشتقّ عنوانَي الأساس لواجهة API والمقبس من أصل الصفحة نفسها، وتأخذ إعدادات OIDC من `tenant-config` الخاص بالمضيف – انظر تسجيل الدخول إلى البوّابة الإلكترونية يأتي من tenant-config الخاص بالمضيف. ولا تنطبق الإعدادات أدناه إلّا على إصدارات البوّابة الإلكترونية السابقة للإصدار 1.10.

استخدم نطاقًا محلّيًا بدل `localhost` أو عنوان IP

أجدى تغيير يمكنك إدخاله على البيئة المحلّية هو التوقّف عن التصفّح عبر `localhost` أو عنوان IP مباشر، ومنح النسخة **اسم مضيف حقيقيًا** – `osos.lan` مثلًا – مع توجيه نطاقاته الفرعية إلى جهازك. فهذا يُغني عن جميع الحلول الالتفافية الخاصّة بالعنوان الاسترجاعي (loopback) الواردة لاحقًا في هذه الصفحة، وهو السبيل الوحيد لاختبار أجزاء المنصّة التي تعتمد على مضيف الطلب.

لماذا يُعدّ اسم المضيف ركيزة أساسية

تعتمد منصّة أسس في تعدّد القاطنين على التمييز بينهم **بحسب اسم المضيف**: ففي كلّ طلب، تحلّ بوّابة الـ API ترويسة `Host` إلى مساحة (Space) ونطاق (Domain)، وترفض كلّ ما يتعذّر عليها مطابقته. وتُشتقّ أسماء مضيفي المساحة من سجلّ النطاق الخاصّ بها:

- `domain_identifier === 'root'` ← تُقدَّم المساحة على العنوان المجرّد.
- أيّ معرّف آخر ← تُقدَّم المساحة على `<domain_identifier>.<url>`.

لذا فإنّ مساحةً ثانية بالمعرّف `hr` على النطاق `osos.lan` لا يمكن الوصول إليها إلّا عبر `hr.osos.lan`. ومع `localhost` لا يمكنك اختبار ذلك فعليًا؛ إذ يعيد المرادف الافتراضي للمضيف كتابةَ السلسلة الحرفية `localhost:4200` إلى منفذ بوّابة الـ API، لأنّ البوّابة الإلكترونية (`:4200`) وبوّابة الـ API (`:8000`) تعملان محلّيًا على منفذين منفصلين – وهو أمر لا يحدث في أيّ نشرٍ فعلي. أمّا النطاق الحقيقي فلا يوجد له مُدخل في سجلّ المرادفات، فيُحلّ إلى نفسه، ويعمل المسار العام لبناء أسماء المضيفين تمامًا كما يعمل في بيئة الإنتاج.

ويتيح لك ذلك ما يلي:

- **تصبح المساحات والنطاقات قابلة للاختبار.** أنشئ مساحة، وامنحها معرّفًا، وتصفّح نطاقها الفرعي، وراقب `resolver` و`domain-resolver` وهما يؤدّيان عملهما الفعلي.
- **يصبح سلوك تسجيل الدخول لكلّ مضيف قابلًا للاختبار** – إذ تُحدَّد خاصّية `supportLocalSignIn` والهويّة البصرية لكلّ نطاق واختيار موفّري الهويّة جميعها استنادًا إلى المضيف.

- **لا حاجة إلى حلول التفافية للعنوان الاسترجاعي.** تصبح الحلول الالتفافية لـ `127.0.0.1` في القسم التالي غير ضرورية، لأنّ المتصفّح والبوّابة الإلكترونية وبوّابة الـ API تستخدم كلّها المضيف نفسه.
- **تطابق بنية النشر بيئة الإنتاج**: اسم مضيف واحد، تُوجَّه الطلبات الواردة إليه بحسب المسار إلى البوّابة الإلكترونية وبوّابة الـ API وخدمة الهويّة.

وجِّه نطاقًا شاملًا (wildcard) إلى جهازك

استخدم اسمًا ضمن لاحقة محجوزة – `.lan` أو `.internal` أو `.test` – ولا تستخدم أبدًا نطاقًا عُلويًا حقيقيًا لا تملكه، ولا `.local` (المخصّص لـ mDNS). ويجب أن يشير **`osos.lan`** و**`*.osos.lan`** إلى عنوان محطّة العمل على الشبكة المحلّية.

الأسلوب	النطاقات الشاملة	مناسب لـ
`dnsmasq` على محطّة العمل أو على خادم منزلي	نعم – `address=/osos.lan/192.168.1.50` يغطّي النطاق وكلّ نطاقاته الفرعية	الخيار المعتاد على Linux/macOS، وهو أيضًا ما تشغّله أغلب برمجيات الموجّهات داخليًا
Pi-hole (أو أيّ جهاز DNS للشبكة المحلّية)	نعم – سجلّ شامل واحد	إعداد على مستوى الشبكة المحلّية يتطلّب وصول الهواتف والأجهزة الأخرى إلى النسخة أيضًا
مُدخَل DNS في الموجّه	يعتمد على البرمجية الثابتة	لا حاجة إلى أيّ برمجيات إضافية حين يدعم الموجّه تجاوزات المضيف الشاملة
Acrylic DNS Proxy (Windows)	نعم – `192.168.1.50 osos.lan` و`*.osos.lan` في `AcrylicHosts.txt`	محطّة عمل Windows لا تتوفّر لها خدمة DNS على الشبكة المحلّية
ملفّ `hosts`	**لا**	صالح فقط إذا أدرجت يدويًا كلّ نطاق فرعي لمساحة؛ مقبول لمساحة أو مساحتين، وعديم الجدوى لاختبار إنشاء المساحات

وهذا القيد في ملفّ `hosts` هو سبب تفضيل مُحلِّل DNS حقيقي؛ فالهدف الأساسي من هذا الإعداد أن يُحلّ اسم مساحة أُنشئت حديثًا دون الحاجة إلى تغيير أيّ شيء.

ووجِّه النطاق إلى **عنوان الجهاز على الشبكة المحلّية**، لا إلى `127.0.0.1`، إن أردت فتح النسخة من هاتف أو جهاز ثانٍ – وهذه أيضًا طريقة لاختبار تطبيق الجوّال مع خدمات خلفية محلّية.

ضع بابًا أماميًا واحدًا أمام البوّابة الإلكترونية وبوّابة الـ API

تقدّم نسخة الإنتاج كلّ شيء عبر مضيف واحد، وتوجّه الطلبات خلفه بحسب المسار. أمّا محلّيًا، فتُنشَر البوّابة الإلكترونية وبوّابة الـ API على منفذين منفصلين؛ لذا ضع وكيلًا عكسيًا صغيرًا على المنفذ 80 وطبّق فيه خريطة مسارات الإنتاج نفسها.

مسارات مُدخِل الإنتاج (ingress)، مرتّبةً بحسب درجة التخصيص:

المسار	الخدمة الخلفية
`/socket`	خدمة websocket العامّة
`/api`	بوّابة الـ API
`/sh/`	مختصِر الروابط
`/identity` و`/users-assets` و`/.well-known` و`/jwks` و`/me` و`/session` و`/ext-auth` و`/token` و`/interaction` و`/auth`	خدمة الهويّة
`/auth-callback`	البوّابة الإلكترونية
`/`	البوّابة الإلكترونية

وتطبّق حاوية Caddy هذا الإعداد محلّيًا. أضِفها إلى `backend/docker-compose.override.yml` (ملفّ التجاوز المحلّي نفسه الذي يتضمّن إعادة تعيين الصور):

```
# backend/docker-compose.override.yml — local-dev only
services:
  front-door:
    image: caddy:2-alpine
    ports:
      - "80:80"
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile:ro
    extra_hosts:
      - "host.docker.internal:host-gateway"
```

```
# backend/Caddyfile — local-dev only
http://osos.lan, http://*.osos.lan {
  handle /api/* {
    reverse_proxy osos-api-gateway:8000
  }
  handle /socket/* {
    reverse_proxy host.docker.internal:3017
  }
  handle /sh/* {
    reverse_proxy host.docker.internal:3010
  }
  @identity path /identity/* /users-assets/* /.well-known/* /jwks /me /session /ext-auth/* /token/* /interaction/* /au
th/*
  handle @identity {
    reverse_proxy host.docker.internal:3001
  }
  handle {
    reverse_proxy osos-web:5000
  }
}
```

ويُشار إلى بوّابة الـ API والبوّابة الإلكترونية باسمَي خدمتيهما في Compose لأنّ Caddy يشاركهما الشبكة، بينما تعمل خدمة الهويّة وخدمة websocket أصلًا على المضيف، ويُوصَل إليهما عبر `host.docker.internal`. ولا يطابق المسار `/auth-callback` نمط المطابقة `/auth/*`، فيُوجَّه بصورة صحيحة إلى البوّابة الإلكترونية، تمامًا كما في مُدخِل الإنتاج.

هيّئ المنصّة للنطاق

يُحدَّد المضيف في أربعة مواضع، ويجب أن تتّفق جميعها.

1. **تهيئة تشغيل البوّابة الإلكترونية** – الملفّ المُركَّب داخل حاوية البوّابة. عدّل `backend/config/app-config.local.json` واضبط `OSOS_APP_CONFIG` ليشير إليه (انظر أعلاه) بدل تعديل الملفّ المتعقَّب `backend/config/app-config.json`:

```
API_BASE_URL     = http://osos.lan/api
SOCKET_BASE_URL  = http://osos.lan/
APP_URL          = http://osos.lan
AUTHORITY        = http://osos.lan
```

2. **خدمة الهويّة** – `backend/services/users-service/config.env`. تحدّد القيمة `AUTHORITY_URL` مُصدِر OIDC ويجب أن تطابق `AUTHORITY` أعلاه تمامًا، بينما يمثّل `RP_ID` معرّف الطرف المُعتمِد في WebAuthn ويجب أن يكون النطاق القابل للتسجيل:

```
AUTHORITY_URL="http://osos.lan"
CLIENT_ORIGIN="http://osos.lan"
RP_ID="osos.lan"
BEHIND_PROXY=true
```

3. **عميل OIDC** – أضِف عناوين ردّ الاتصال للنطاق إلى العميل `localhost-angular` في `backend/services/users-service/src/clients.json` (وإلى النسخة المبنية ضمن `dist/`، أو أعِد بناء الخدمة): `http://osos.lan/auth-callback` و`http://osos.lan/silent-renew.html` ضمن `redirect_uris`، و`http://osos.lan` ضمن `post_logout_redirect_uris`.
4. **البذر** – شغّل ترحيل الإعداد الأوّلي باستخدام النطاق، بحيث يرتبط به سجلّ النطاق المبذور ومساحة الجذر منذ البداية:

```
DEFAULT_DOMAIN="osos.lan" node ./dist/migrate
```

وإن كانت النسخة قد بُذرت سلفًا باستخدام `localhost:4200`، فإمّا أن تعيد ضبطها وبذرها بقيمة `DEFAULT_DOMAIN` الجديدة، وإمّا أن تعدّل الحقل `host` في سجلّ النطاق وعنوان نطاق مساحة الجذر، ثمّ تعيد تشغيل `data-service` كي تعيد آليات المراقبة فيه كتابة خبء التوجيه عند الإقلاع.

اختبار المساحات والنطاقات على النطاق

مع توفّر DNS شامل، أنشئ مساحة ثانية لها `domain_identifier` (أحرف لاتينية صغيرة وأرقام وشُرَط سفلية فقط)، فتصبح متاحة على `<identifier>.osos.lan` دون أيّ إعداد إضافي في DNS أو الوكيل. وتحقّق من أنّ المضيف أُدرج فعلًا في شقَّي خبء التوجيه قبل أن تعزو المشكلة إلى المتصفّح: فالمساحة التي لا يقابلها نطاق مطابق تؤدّي إلى `SPACE_ERROR` أو `DOMAIN_ERROR` رغم أنّ الواجهة تُظهر أنّها أُنشئت. وتجد وصف النموذج الكامل وحقول السجلّين وسلوك المُحلِّلات في المساحات والنطاقات [الإدارة].

> **ملاحظة**
>
> يتغيّر سلوك واحد عند الانتقال من `localhost:4200`؛ إذ يرتبط تجاوز مُهيّئ المساحات بتلك السلسلة الحرفية، ولذلك ستُصدر المنصّة على نطاق حقيقي أحداث CloudEvents الخاصّة بمُهيّئ المساحات عند إنشاء المساحة أو تفعيلها أو تعطيلها. وإذا لم تُهيَّأ `SPACE_PROVISIONER_URL`، يفشل الاستدعاء ويُسجَّل هذا الفشل دون أن تفشل العملية، وهو أمر غير ضارّ محلّيًا، لكنّه يفسّر ظهور أخطاء المُهيّئ في السجلّ فور تبديل النطاق.

البديل الاسترجاعي: الحلول الالتفافية لـ `127.0.0.1`

إن لم تكن تنوي إعداد DNS، فلا بدّ من الوصول إلى النسخة عبر العنوان الاسترجاعي. ويتطلّب ذلك على Windows عدّة إعدادات متوافقة. وهذا خيار بديل لا يُوصى به؛ فقد أُدرج إعداد النطاق أعلاه خصيصًا لتفاديه.

لماذا يفشل `localhost` على Windows

يحلّ Windows الاسم `localhost` أوّلًا إلى عنوان IPv6 الاسترجاعي `::1`، لكنّ Docker Desktop لا يمرّر عنوان IPv6 الاسترجاعي. لذا لا تستجيب منافذ الحاويات المنشورة – البوّابة الإلكترونية على `:4200`، وبوّابة الـ API على `:8000`، وواجهة RabbitMQ على `:15672` – على `localhost`، بينما تستجيب على `127.0.0.1`. أمّا الخدمات التي تعمل أصلًا (خدمة الهويّة على `:3001` و`data-service` على `:3000`) فترتبط بعائلتَي العناوين كلتيهما، فتعمل في الحالتين. وهذا ما يجعل العطل مربكًا إلى هذا الحدّ: نصف المنظومة يستجيب على `localhost` والنصف الآخر لا يستجيب.

ويزيد سلوك المتصفّح الأمر تعقيدًا؛ إذ ينتقل عند تصفّح الصفحات من `::1` إلى IPv4 عند تعذّر الاتصال، أمّا `fetch` و`XHR` فلا ينتقلان إليه. لذا قد تُحمَّل البوّابة الإلكترونية على `localhost:4200` بينما يفشل كلّ استدعاء API تجريه. ولهذا، لا بدّ من توجيه البوّابة الإلكترونية إلى `127.0.0.1` صراحةً.

وثمّة بديل يجعل Windows يفضّل IPv4 على مستوى الجهاز، وذلك بتنفيذ الأمر التالي من موجّه أوامر بصلاحيات مرتفعة:

```
netsh interface ipv6 set prefixpolicy ::ffff:0:0/96 60 4
```

ويُبقي هذا الإعداد `localhost` صالحًا للاستخدام في كلّ موضع، لكنّه يتطلّب تغيير سياسة الشبكة على مستوى الجهاز كلّه. ومع ذلك، يظلّ من الضروري أن تغطّي خرائط مضيفات المساحات والنطاقات المضيفَ الذي يرسله المتصفّح فعلًا.

عناوين استدعاء OIDC الاسترجاعية مسجَّلة أصلًا `Since 1.10`

لم يعد يلزم اتخاذ أيّ إجراء هنا؛ فالعميل المخصّص للتطوير `localhost-angular` في `backend/services/users-service/src/clients.json` يتضمّن الآن صيغتَي المضيف كلتيهما جاهزتين:

```
redirect_uris:              http://localhost:4200/auth-callback
                            http://localhost:4200/silent-renew.html
                            http://127.0.0.1:4200/auth-callback
                            http://127.0.0.1:4200/silent-renew.html
post_logout_redirect_uris:  http://localhost:4200
                            http://127.0.0.1:4200
```

وكانت الصيغ المدرَجة سابقًا تقتصر على `localhost`، فكان فتح البوّابة الإلكترونية على `http://127.0.0.1:4200` – وهو العنوان الذي يفرضه Docker Desktop على Windows – يؤدّي إلى خطأ بسبب عنوان إعادة توجيه غير صالح، ويستلزم تعديل الملفّ يدويًا في البيئة المحلية. ولا يمنح هذا التغيير أيّ أذونات إضافية على ما تمنحه مُدخلات `localhost`؛ فهو العميل نفسه، بصيغتين لاسم المضيف.

أمّا أيّ مضيف **آخر**، فما زال يتطلّب إضافة مُدخلاته الخاصّة يدويًا؛ وهذا هو التعديل الخاص بالتطوير المحلي في الخطوة 3 من هيّئ المنصّة للنطاق.

وأبقِ قيمة `AUTHORITY` الخاصّة بـ OIDC على `localhost:3001`؛ فخدمة الهويّة تعمل أصلًا وترتبط بعائلتَي العناوين كلتيهما، ويجب أن يظلّ المُصدِر مطابقًا لما تعلنه.

عيّن مرادفات المضيف الاسترجاعي عبر `DEV_HOST_ALIASES` `Since 1.10`

كان هذا تعديلَ مصدر. وصار الآن تهيئة – بمتغيّر بيئة واحد، دون تعديل أيّ ملفّ أو الحاجة إلى إبقاء أيّ تغيير خارج الإيداع.

تسجّل كلّ من المساحة والنطاق مضيفًا **واحدًا**: العنوان الذي يُتوقَّع أن يستخدمه المتصفّح. وفي البيئة المحلية، يختلف هذا العنوان عن العنوان الذي تستجيب عليه بوّابة الـ API، ولذلك لا يطابق خبء التوجيه المبنيّ من تلك السجلّات قيمةَ `Host` التي يرسلها المتصفّح فعلًا، ويعيد كلّ استدعاء API الخطأ `SPACE_ERROR` ثمّ `DOMAIN_ERROR`. وأصبح جزآ الخبء يُبنيان عبر طبقة مرادفات مشتركة واحدة، تُهيَّأ باستخدام `DEV_HOST_ALIASES` في `backend/.env`:

الشكل	كائن JSON يربط مضيفًا مسجّلًا بمضيف واحد أو بقائمة مضيفات
مثال	`DEV_HOST_ALIASES={"localhost:4200":["localhost:8000","127.0.0.1:8000"]}`
الافتراضي عند عدم الضبط	`{"localhost:4200": ["localhost:8000"]}`
مضيف بلا مُدخَل	يُحلّ إلى نفسه

قيمة مشوَّهة	تُسجَّل دون إطلاق استثناء، لأنّها تُقرأ أثناء تحديث الخبء، ولا يجوز أن يؤدّي خطأ مطبعي إلى فشل التحديث كلّه. وتُستخدم القيمة الافتراضية بدلًا منها.
الإنتاج	لا يُضبط فيه شيء، فلا يتغيّر شيء في النشر الفعلي.

> ⚠ **الافتراضي لا يغطّي `127.0.0.1`**
>
> يربط الإعداد الافتراضي عند عدم الضبط `localhost:4200` بـ `localhost:8000` **وحده**. وينشر Docker Desktop على Windows منافذ الحاويات على `127.0.0.1`، ولذلك يواجه المطوّر الذي يتصفّح ذلك العنوان مضيفًا لا يتعرّف إليه أيٌّ من جزأَي الخبء، وتكون النتيجة **صفحة فارغة بلا أيّ خطأ**. فإضافات بوّابة API تستجيب بـ `SPACE_ERROR` / `DOMAIN_ERROR`، لكن لا شيء يُظهر ذلك في المتصفّح. فإن كنت تتصفّح `127.0.0.1`، فعليك ضبط `DEV_HOST_ALIASES` صراحةً وإدراج هذا العنوان فيها، كما في المثال أعلاه تمامًا.

وثمّة قاعدة واحدة ينبغي مراعاتها عند كتابة القيمة: قائمة المرادفات **تحلّ محلّ** المضيف المسجّل، ولا تُضاف إليه. ويحافظ هذا السلوك على منطق إعادة الكتابة المضمَّنة التي أصبحت طبقة المرادفات تعمّمها؛ إذ كان `localhost:4200` يُستبدَل بـ `localhost:8000`، ولا يُدرَج إلى جانبه. فإن أردت أن يبقى `localhost:8000` و`127.0.0.1:8000` صالحَين معًا، فأدرجهما معًا.

وكان التغيير الأكبر في معالجة النطاق؛ إذ كانت تستخدم مضيفه كما هو، دون أيّ إعادة كتابة، ولذلك لم تكن تغطية المساحة وحدها كافية قطّ. وأصبح الجانبان يمرّان بالدالة نفسها، فيُصلح إعداد واحد كليهما.

وبعد تغيير القيمة، **أعِد تشغيل** `data-service`؛ إذ تُعيد المراقبات كتابة خبء التوجيه عند الإقلاع، بينما تستمرّ الخدمة قيد التشغيل في تقديم قوائم المضيفات القديمة.

أضِف المضيف الاسترجاعي إلى Default Domain

يُصلح `DEV_HOST_ALIASES` ما تحدّده بوّابة الـ API. أما خدمة الهوية فلها فحصها الخاص: لا تعرض نموذج اسم المستخدم وكلمة المرور إلا إذا كان أحد **النطاقات** يُدرج المضيف الذي يعود إليه المتصفّح. ويمنح البذر **Default Domain** مضيفًا واحدًا هو `DEFAULT_DOMAIN` (افتراضيًا `localhost:4200`)، فتظهر صفحة تسجيل الدخول على `127.0.0.1:4200` **فارغة**. ابذر باستخدام `DEFAULT_DOMAIN="127.0.0.1:4200"`، أو أضِف على نسخة سبق بذرها `127.0.0.1:4200` عنوانًا آخر في **المضيف** ضمن Default Domain (**الإدارة › التطبيقات › الكيانات › النطاق**، وانظر المساحات والنطاقات [الإدارة]). وللحصول على جلسة لذلك، سجّل الدخول على `localhost:4200` بعد إجبار المتصفّح على IPv4، مثل تشغيل Chromium بالخيار `--host-resolver-rules="MAP localhost 127.0.0.1"`. ويصل التغيير المحفوظ إلى خدمة الهوية وإلى خبء النطاقات في بوّابة الـ API عبر حدث التحديث، دون إعادة تشغيل؛ أما التحرير المباشر لمجموعة `domain` في Mongo فيتخطّى هذا الحدث ويترك نسخة خدمة الهوية قديمة.

تسجيل الدخول إلى البوّابة الإلكترونية يأتي من `tenant-config` الخاص بالمضيف

`Since 1.10`

لم تعد البوّابة الإلكترونية تقرأ إعدادات OIDC من `app-config.json`؛ بل ترسل، قبل بدء تشغيل Angular، طلب `GET` إلى `/api/identity/v1/tenant-config` إلى المضيف الذي فُتحت عليه، وتستمدّ `authority` و`clientId` و`scope` من الاستجابة (العقد موثّق في المصادقة OAuthg). ويترتّب على ذلك عند التشغيل على محطة العمل ما يلي:

- **يجب ضبط `OIDC_WEB_CLIENT_ID` على خدمة الهوية.** يتضمّن `users-service/config.env` القيمة `OIDC_WEB_CLIENT_ID="localhost-angular"`؛ ودونها تستجيب `tenant-config` بـ `503`، وتعرض البوّابة الإلكترونية

صفحة «خدمة تسجيل الدخول غير متاحة / ما زالت قيد الإعداد» بدلًا من بدء التشغيل.

- **`nx serve`** يمرّر `/api` و `/socket.io` إلى بوّابة الـ API المحدّدة في **`OSOS_API_TARGET`** (افتراضيًا `http://localhost:8000`، أو بيئة بعيدة) مع `changeOrigin`، فتحدّد بوّابة الـ API القاطن استنادًا إلى مضيف *الهدف*. ويسجّل إصدار التطوير الدخول عبر العميل `localhost-angular`، لأنّ عميل الويب في البيئة الفعلية لا يُدرج `http://localhost:4200`. ولا تتضمّن حزمة الإنتاج هذا التجاوز، ويُفشل التكامل المستمر أيّ إصدار إنتاج يحتويه.

- **صورة `osos-web`** ليس فيها هذا الوكيل: يقدّم nginx فيها البوّابة الإلكترونية وحدها، فعلى الحِزمة ذات المنفذين (البوّابة الإلكترونية `:4200` وبوّابة الـ API `:8000`) يصل الطلب `/api/identity/v1/tenant-config` إلى صفحة البوّابة الإلكترونية، وتتوقف البوّابة عند **`BAD_CONFIG`**. وتقرأ الصورة (`Since 1.10`) أي ملف `*.conf` يُركَّب في `/etc/nginx/conf.d/extra/` داخل كتلة الخادم، فركّب ملفًا **للتطوير المحلي فقط** يؤدّي ما يؤدّيه `nx serve`. ولا تحتاج إليه خلف باب أمامي، إذ يوجّه الباب الأمامي `/api` و `/socket` أصلًا.

```
# same-origin-api.conf — local-dev only. Mounted into osos-web at /etc/nginx/conf.d/extra/
location ^~ /api/ {
    proxy_pass         http://osos-api-gateway:8000;
    proxy_http_version 1.1;
    proxy_set_header   Host 127.0.0.1:8000;      # a host the gateway already resolves
    proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header   X-Forwarded-Proto $scheme;
    client_max_body_size 512m;
    proxy_read_timeout 300s;
}
location ^~ /socket.io/ {
    proxy_pass         http://host.docker.internal:3017;   # the public websocket service
    proxy_http_version 1.1;
    proxy_set_header   Upgrade $http_upgrade;
    proxy_set_header   Connection "upgrade";
    proxy_set_header   Host $host;
    proxy_read_timeout 3600s;
}
```

ركّبه من `backend/docker-compose.override.yml` (واحفظ الملف خارج المستودع)، ثم نفّذ `docker compose up -d`:

```
# backend/docker-compose.override.yml — local-dev only
services:
  osos-web:
    volumes:
      # keep the file outside the repository
      - /path/to/osos-local/portal-nginx/same-origin-api.conf:/etc/nginx/conf.d/extra/same-origin-api.conf:ro
```

يرسل هذا الملف `Host: 127.0.0.1:8000`، فأبقِ هذا المضيف في `DEV_HOST_ALIASES`.

- وعند فشل التمهيد، تظهر صفحة خطأ ثابتة ثنائية اللغة تتضمّن رمز `data-reason` ومعرّف الارتباط الصادر عن بوّابة الـ API. ويعني `BOOTSTRAP_ENDPOINT_MISSING` عادةً أنّ صورة الواجهة الخلفية أو مسار `/api/identity` في بوّابة الـ API أقدم من الواجهة الأمامية.

افتراضيات أخرى للتشغيل المحلي تغيّرت `Since 1.10`

- `space-resolver` **يجيب بـ** `404 SPACE_ERROR` عند استخدام مضيف لا يطابق أيّ مساحة (كان `400`)؛ أمّا `domain-resolver` فلا يزال يستجيب بـ `400 DOMAIN_ERROR`. وتُصنّف البوّابة الإلكترونية كلتا الحالتين على أنّهما حالتا مضيفٍ تعذّر حلّه.
- `APPLY_RATE_LIMITER` – مُقيِّد معدّل `/identity` (10 طلبات من غير `GET` لكل 15 دقيقة لكل عنوان عميل) **مفعّل ما لم تكن القيمة** `false` (`Since 1.4`؛ كان منطق المفتاح معكوسًا). ويضبط `users-service/config.env` المحلي القيمة `"false"` كي لا تُقيَّد اختبارات تسجيل الدخول المتكرّرة؛ اتركه دون ضبط لتشغيل مماثل لبيئة الإنتاج.
- `OIDC_COOKIE_KEYS` إلزامي ما لم تكن قيمة `NODE_ENV` هي `development` أو `test` (يضبط `config.env` المحلي `NODE_ENV=development`، فيُولَّد مفتاح مع إصدار تحذير). ولم تعد مفاتيح التوقيع تُقرأ من ملفّ `jwks.json` متعقَّب؛ بل تُولَّد في مخزن مفاتيح القاطن عند الإقلاع الأوّل، أو تُقرأ من ملفّات PEM في `OIDC_SIGNING_KEYS_DIR`.
- `SPACE_SCOPE_ENFORCEMENT` قيمته الافتراضية في الرماز `enforce` عند عدم ضبطه – وهي الحالة المعتادة على محطة العمل – ولذلك يتلقّى البرنامج النصي الذي يستدعي خدمة مباشرةً دون ترويسة `x-space-id` الاستجابة `400 SPACE_CONTEXT_MISSING` بدلًا من استخدام المساحة الجذر بصمت. وعند المرور عبر بوّابة الـ API، تُزوَّد الترويسة دائمًا استنادًا إلى المضيف.
- `ENABLE_SPEC=false` أصبح يعطّل فعليًا `/api/openapi.json` و`/api/explorer` (`Since 1.2`، تحديث على جميع خطوط الإصدارات)؛ بينما كان يُبقي مستند JSON متاحًا.
- `SIGNIN_AUTO_REDIRECT_SINGLE_PROVIDER` وارد بوصفه مثالًا مُعلَّقًا في `config.env`؛ وعدم ضبطه يعني تفعيله، فتُتخطّى صفحة تسجيل الدخول حين يكون مزوّد خارجي واحد هو السبيل الوحيد للدخول.

أطلق أسطول الخدمات الخلفية كاملًا

لا تُشغَّل الخدمات الخلفية ضمن Compose، بل تعمل مباشرةً على المضيف. ويتولى تشغيلها سكربتان في الجذر، لكلٍّ منهما غرض مختلف:

السكربت	ما الذي يشغّله
`pnpm run start:core`	عمليّة `core` المدمجة، إضافةً إلى الخدمات التي **لا** يستضيفها `core` وحدها (`oidc`، و`storage`، و`indexing`، و`scheduler`، و`notification`، و`web-socket`، و`rtc`). وهو السكربت المستخدم في العمل اليومي.
`pnpm start`	كلّ خدمة في مساحة العمل ضمن عمليّة مستقلّة – **عدا** مُقلِع `core` المدمج.

```
cd backend
pnpm run start:core
```

يبني السكربتان المشروع أولًا (مهمة `start` في Turbo تعتمد على `build`)، ويضبط كلاهما القيم الافتراضية `TENANCY_MODE=single` و`APP_ID=local` و`ALLOW_INSECURE_FIELD_ENCRYPTION_KEY=true` لكل خدمة يشغّلها؛ وتكون الأولوية للقيمة المضبوطة مسبقًا في الصدفة. وكان هناك سكربت ثالث، `pnpm run start:main`، يشغّل مجموعة فرعية صغيرة مختارة يدويًا؛ وقد أُزيل في 1.10 `Removed 1.10` لأن `start:core` يشغّل أصلًا كل ما يحتاجه التشغيل المحلي.

يُقلع `core` عدّة تطبيقات خدمة (`audit`، و`notification-api`، و`reaction`، و`dependency-service`، و`data-service`، و`comments`، و`timeline`، و`workflow-service`، و`logging-service`، و`integrations`، و`packaging-service`، و`functions`) داخل عمليّة **واحدة**، وتشغيله إلى جانب الخدمات نفسها منفردةً يعني تشغيل كلٍّ منها مرّتين. لذلك يستثنيه `pnpm start`، ويتولّى `start:core` تشغيله، مع استثناء كلّ تطبيق يستضيفه `core` أصلًا.

ويخضع أيّ مُطلِق يدوي لقيدين: تستخدم الخدمات جميعها المنفذ الافتراضي نفسه لخادم مقاييس Prometheus، لذا تحتاج كلّ خدمة إلى قيمة مختلفة لـ `METRICS_PORT`. وهذا التعارض لا يُوقف خدمات HTTP، لكنّه يُوقف عمّال ناقل الرسائل الذين لا يستخدمون منافذ. كذلك لا تحصل الخدمة التي تُشغَّل منفردةً على القيم الافتراضية التي تضبطها سكربتات الجذر، فترفض الإقلاع دون `APP_ID` (في الوضع وحيد القاطن)، وتحتاج الخدمات التي تشفّر الحقول أيضًا إلى مفتاح رئيسي أو إلى `ALLOW_INSECURE_FIELD_ENCRYPTION_KEY=true` عند التشغيل محليًا.

ليست خدمتا altcha وshortenerg ضمن هذه القائمة، لأنهما ليستا من حِزم مساحة العمل؛ فلكلٍّ منهما مستودعها الخاص، وتعملان محليًا بوصفهما صورتين في Compose، متاحتين على المنفذين اللذين تتوقعهما الخدمات الأخرى (altcha على `3023`، shortenerg على `3010`):

```
cd backend
docker compose up -d altcha-service shortener-service
```

وإليك مُطلِقًا يوزّع المنافذ، ويضبط تلك القيم الافتراضية، ويحتفظ بسجلّ لكل خدمة:

```
#!/usr/bin/env bash
# ~/osos-start-all.sh — local-dev only
cd "<path-to>/backend"
export BASH_ENV="$HOME/.pnpm-bashenv"     # Windows-only; see the note below
mkdir -p "$HOME/osos-logs"
mp=9101
# the defaults the root start scripts would have supplied
export TENANCY_MODE="${TENANCY_MODE:-single}" APP_ID="${APP_ID:-local}" \
       ALLOW_INSECURE_FIELD_ENCRYPTION_KEY="${ALLOW_INSECURE_FIELD_ENCRYPTION_KEY:-true}"
for s in oidc data-service logging-service workflow-service reaction storage \
         notification-api indexing audit comments timeline scheduler packaging-service \
         dependency-service web-socket notification; do
  env METRICS_PORT=$mp pnpm --filter "$s" run start > "$HOME/osos-logs/$s.log" 2>&1 &
  echo "launched $s (METRICS_PORT=$mp)"
  mp=$((mp+1))
done
echo ALL_LAUNCHED
wait
```

شغّله في الخلفية واتركه قيد التشغيل، فهو العملية المالكة لعمليات الخدمات:

```
bash ~/osos-start-all.sh &
```

وتستخدم القائمة اسم **مساحة العمل (workspace)**، الذي يختلف كثيرًا عن اسم المجلّد؛ فاسم حزمة خدمة الهويّة هو `oidc`، واسم حزمة `web-socket-service` هو `web-socket`. أمّا `BASH_ENV` فلا يلزم إلّا عند التشغيل على Windows مباشرةً، إذ يوفّر إعداد `nullglob` الذي تفترضه سكربتات الحِزَم.

ويجب أن تكون الخدمات مُنصِتة **وأن** تكون طوابيرها مرتبطة بالناقل قبل تشغيل ترحيل الإعداد الأوّلي؛ إذ تُنشر التهيئة عبر الناقل، ويُسقِط مبادل الموضوعات (topic exchange) الرسائل التي لا مستهلك لها. تأكّد من وجود مستهلك لكلٍّ من الطابورين المهمّين:

```
docker exec backend-rabbitmq-1 rabbitmqctl list_queues name consumers \
  | grep -E "data-service-queue|users-service-queue"     # both should show 1
```

استعِد أوّل تسجيل دخول

تُحدَّد هويّة مسؤول النظام الأعلى في `services/data-service/.env.migration` – الملفّ الذي يُنشئه `pnpm run setup:local` – عبر `SUPER_ADMIN_EMAIL` و`SUPER_ADMIN_USERNAME`. فاضبط كليهما قبل الترحيل، واستخدم لـ `SUPER_ADMIN_EMAIL` صندوق بريد تستطيع الاطّلاع على رسائله: فالقيم الافتراضية في الرماز وفي ملفّات الأمثلة ليست عناوين تتلقّى رسائلها. ومع ملتقِط البريد (أدناه) يصلح أيّ عنوان: فكلّ الرسائل تصل إلى صندوق الوارد نفسه.

لا يحصل مسؤول النظام الأعلى على **كلمة مرور**. فحين تُنشئ خدمة الهويّة الحساب تمنحه كلمة مرور عشوائية غير صالحة للاستعمال، وتُصدر رمز إعادة تعيين كلمة المرور، وترسل رابط تعيين كلمة المرور إلى `SUPER_ADMIN_EMAIL`. ومنذ 1.10 لا يُخزَّن إلّا تجزئة SHA-256 للرمز (`resettokens.tokenHash`)، فلا يمكن إعادة بناء الرابط من قاعدة البيانات – وقراءة المجموعة لا تعطيك شيئًا تستطيع فتحه. لذا التقط الرسالة بدلًا من ذلك.

شغّل ملتقِط بريد محلّيًا ووجّه مرسِل البريد إليه. يستقبل Mailpit بريد SMTP على المنفذ `1025` ويعرض كلّ رسالة في صندوق وارد على الويب على المنفذ `8025`. أضِفه إلى `backend/docker-compose.override.yml`، وأعطِ الحاوية `email-service` إعدادات SMTP الخاصّة بها (متغيّرات `SMTP_*` نفسها التي يضبطها النشر على العنقود):

```
# backend/docker-compose.override.yml — local-dev only
services:
  mailpit:
    image: axllent/mailpit
    ports:
      - "8025:8025"
    environment:
      MP_SMTP_AUTH_ACCEPT_ANY: "1"
      MP_SMTP_AUTH_ALLOW_INSECURE: "1"
  email-service:
    environment:
      SMTP_HOST: mailpit
      SMTP_PORT: "1025"
      SMTP_SECURE: "false"
      SMTP_USERNAME: local
      SMTP_PASSWORD: local
```

أعِد إنشاء الحاويتين (`docker compose up -d mailpit email-service`)، ثمّ شغّل الترحيل. افتح `http://localhost:8025`، وابحث عن رسالة تعيين كلمة المرور، واتبع رابطها. ويُركَّب مُوجِّه إعادة التعيين تحت `/identity`، لذا يكون الرابط `http://osos.lan/identity/reset-password/<token>` – على المضيف الذي هيّأته في إعداد النطاق. عيّن كلمة مرور، ثمّ سجّل الدخول من البوّابة الإلكترونية.

وإن انتهت صلاحية الرابط (24 ساعة افتراضيًا) أو لم تصل الرسالة، فافتح `/identity/reset-password` على المضيف نفسه واطلب رابطًا جديدًا لعنوان بريد مسؤول النظام الأعلى الذي أدخلته ضمن البيانات الأولية. ويحلّ كلّ رابط جديد محلّ الرابط السابق، فلا يعمل إلّا أحدثها.

شخّص تشغيلًا يرفض أن يكتمل

تتغيّر أنماط الفشل كلّما أصلحت طبقة: رفض الاتصال، ثمّ `SPACE_ERROR`، ثمّ `DOMAIN_ERROR`، ثمّ استجابة فعلية. ويعني هذا التدرّج أنّك تتقدّم: إذ تشير كلّ حالة إلى أنّ طبقة مختلفة هي التي تستجيب.

```
# What is actually listening? Authoritative on Windows, where curl is easy to misread.
powershell -c "Get-NetTCPConnection -State Listen | ? {$_.LocalPort -ge 3000 -and $_.LocalPort -le 3025} | select Loca
lPort"

# Curl a published port without the proxy/loopback quirks getting in the way:
curl -s --noproxy '*' -o /dev/null -w "%{http_code}\n" http://127.0.0.1:8000/api

# Can the gateway container reach the native services on the host?
docker exec backend-osos-api-gateway-1 sh -c \
  'curl -s -o /dev/null -w "%{http_code}\n" http://host.docker.internal:3000/'

# Which plugin is rejecting the request? Replay the preflight and read the body:
curl -s -i --noproxy '*' -X OPTIONS http://127.0.0.1:8000/api/language \
  -H "Origin: http://127.0.0.1:4200" \
  -H "Access-Control-Request-Method: GET" \
  -H "Access-Control-Request-Headers: authorization"

# A service's boot log, if you used the launcher above:
tail -n 40 ~/osos-logs/data-service.log
```

وإعادة إرسال الطلب التمهيدي هي أسرع طريقة للتمييز بين مشكلة CORS فعلية وأخرى تتعلّق بتعدّد القاطنين. فبوّابة الـ API تسمح بجميع المصادر، لذا يرجع رفض الطلب التمهيدي في الغالب إلى رفض مُحلِّل المساحة أو النطاق للمضيف، ويحدّد متن استجابة الخطأ أيّهما رفضه.

ملاحظات حول الإصدارات

معظم ما تصفه هذه الصفحة **Long-standing**: فالبنية المحلّية الموزَّعة على منفذين، وتعدّد القاطنين المستند إلى المضيف، وسلوك البذر، جميعها تسبق الإصدار 1.2.

أمّا الآليات التي تغيّرت حديثًا فهي **وقائع بيئة المطوّر لا قدرات منتَج**: إذ لا تغيّر شيئًا يراه القاطن أو المدير أو بانٍ للتطبيقات، وليست جزءًا من أيّ إصدار *يُعلنه* المنتَج صراحةً. لكنّها موجودة في مستودع الواجهة الخلفية نفسه، فتُوزَّع مع الرماز: بدءًا من `Since 1.10`، يتضمّنها الرماز المسحوب من `develop` أو فرع الإصدار `1.10`، بينما لا يتضمّنها الرماز المسحوب من فرع `1.8` أو أقدم. ولهذا تحمل الوسم `` `Since 1.10` ``، كي يُخفي مرشِّح الإصدارات، عند ضبط هذه الصفحة على إصدار أقدم، الخطوات التي لن تعمل فيه:

- `DEV_HOST_ALIASES` وطبقة المرادفات المشتركة التي يستند إليها، على جانبَي المساحة والنطاق معًا. وقبل ذلك، كان ربط المضيف الاسترجاعي يتطلّب تعديلًا يدويًا في الرماز المصدري، وثّقته هذه الصفحة سابقًا.
- عناوين الاستدعاء للعنوان `127.0.0.1` في عميل OIDC `localhost-angular`.
- `scripts/setup-local-config.js` (`postinstall` / `pnpm run setup:local`) والتوجيه غير المباشر لمصدر التركيب عبر `OSOS_APP_CONFIG`.
- انتقال مستورِدَي الدليل لـ AD وEntra ID من هذا المستودع إلى مستودعين خاصّين بهما، وهو ما ألغى أيضًا استثناء `azure-ad-importer` الذي تضمّنه `pnpm start` و`pnpm run start:core` لفترة قصيرة.
- تهيئة البوّابة الإلكترونية انطلاقًا من `tenant-config` والقيم الافتراضية الأخرى للتشغيل المحلي المدرجة في افتراضيات أخرى للتشغيل المحلي تغيّرت – وهذه تغييرات في المنتج *فعلًا*، أُتيحت ضمن سلسلة إصدارات 1.10.

أسس كواجهة خلفية

كل ما تفعله واجهة أسس الرسومية هو في حقيقته استدعاء HTTP. فتطبيق الويب ليس إلا عميلًا واحدًا لواجهة برمجية تضم ما يقارب 600 مسار خلف بوابة واحدة (نحو 450 منها عمليات مُنمَّطة تصفها وثائق OpenAPI الخاصة بالخدمات؛ أما البقية – التحزيم والملفات وتسجيل الدخول وبعض المسارات الأخرى – فموثّقة يدويًا). ولا يملك التطبيق قناة جانبية خاصة، ولا قدرة محجوبة عن المستدعين الآخرين. وهذا ما يجعل المنصة صالحة للعمل **من دون واجهة (headless)**: واجهةً خلفية لنصوصك البرمجية وخدماتك ومنتجاتك، من غير أن تظهر واجهتها الرسومية أصلًا.

خُصّصت هذه السلسلة من الأدلة لبناء هؤلاء العملاء الآخرين. تحدّد هذه الصفحة النموذج، وتعرّف الأشكال الثلاثة التي يتخذها أي تكامل، وتبيّن ما يفرضه الخادم دائمًا مهما كان العميل المستدعي، وتوضّح الأدلة القائمة التي تتولّى كل جزء من الموضوع؛ كي تبقى بقية صفحات السلسلة مركّزة على المهام ولا تعيد سرد الأساسيات.

نموذج العمل من دون واجهة

يحتاج العميل الذي يعمل من دون واجهة إلى ثلاثة أشياء: عنوان أساس، وبيانات اعتماد، والأعراف الموثقة في استدعاء واجهة HTTP. ولا يُشترط استخدام أي SDK.

تتجه كل الطلبات إلى **مضيف البوابة**، ويحدّد المضيف نفسه القاطن: إذ تستخلص إحدى إضافات البوابة معرّف المساحة من ترويسة `Host` في الطلب، وتمرّره على هيئة `x-space-id`. فلا يرسل عميلك أي مُعامل للقاطن؛ فاستدعاء `https://acme.example.com/api/…` هو بذاته اختيار القاطن.

ويجري التحقق من بيانات الاعتماد مرة واحدة عند الحافّة. فتحوّل إضافات المصادقة في البوابة رمز الحامل أو رمز الـ API إلى مجموعة ترويسات هوية تثق بها الخدمات اللاحقة. ولهذا يجب ألّا تُتاح تلك الخدمات للوصول المباشر أبدًا؛ ويشرح دليل الحدود الأمنية هذه الحدود بالتفصيل.

وفي طريق العودة، تصل كل استجابة JSON في الغلاف نفسه `{success, result}` بصرف النظر عن الإصدار – باستثناء استجابة `GET` على `/v3`، التي عادت خامًا `Since 1.10` (التفاصيل) – ويصل كل خطأ بصيغة موحّدة ومترجَمة تحمل رمزًا آليًا ثابتًا. أما العقد نفسه – التوجيه، وإصدارات URL، والغلاف، وجسم الخطأ، والتقسيم إلى صفحات، والحصول على وثائق OpenAPI – فيحدّده دليل استدعاء واجهة HTTP، ويسري على كل ما في هذه السلسلة.

أشكال التكامل الثلاثة

يتخذ كل تكامل تقريبًا واحدًا من ثلاثة أشكال، والشكل هو الذي يحدّد بيانات اعتمادك قبل أن تكتب سطرًا واحدًا من الرماز:

الشكل	مثال نمطي	بيانات الاعتماد	تُنفَّذ الطلبات بهوية
نص برمجي أو أتمتة من الطرف الأول	مُحمِّل بيانات، مزامنة ليلية، ترحيل	رمز API (`x-app-token`)	مستخدم التطبيق المالك للرمز
خدمة من دون واجهة	منتجك الخاص الذي يتخذ أسس واجهةً خلفيةً له	رمز API لكل بيئة، أو عميل OAuth من آلة إلى آلة	هوية خدمة مخصّصة
طرف ثالث يعمل نيابةً عن مستخدم	أداة ذكاء أعمال، أو بوابة تعرض بيانات المنصة لمستخدميها	رمز OAuth الخاص بكل مستخدم	كل مستخدم نهائي على حدة

النص البرمجي من الطرف الأول هو أسرع طريق إلى استدعاء ناجح: احصل على رمز، وأرسله في ترويسة، وبذلك يكتمل الإعداد. وهذا هو المسار الذي يتّبعه البدء السريع تحديدًا. ولا يحمل الرمز نطاقات إذن خاصة به: فهو مرتبط *بمستخدم تطبيق* مخصّص،

وأذوناته هي تحديدًا أدوار ذلك المستخدم وعضويته في المساحات. لذا فإن حصر الصلاحيات في الحدّ الأدنى يعني تقييد أذونات المستخدم لا الرمز؛ ويعالج دليل اختيار بيانات الاعتماد تبعات ذلك.

أما **الخدمة التي تعمل من دون واجهة** فتستخدم الآلية نفسها بانضباط أكبر: مستخدم تطبيق (أو عميل OAuth) خاص بها لكل بيئة، بحيث يمكن إبطال بيانات اعتماد بيئة الاختبار من دون المساس بالإنتاج، وتدوير الرموز من دون توقف. ويتناول ذلك دليلا دورة حياة الرمز والبيئات والترقية.

وأما **الطرف الثالث الذي يعمل نيابةً عن مستخدم** فينبغي ألّا يحمل أي بيانات اعتماد خاصة به. يُجري كل مستخدم المصادقة عبر موفّر OIDC في المنصة، ويمرّر التكامل رمز ذلك المستخدم، فيُنفَّذ كل طلب بأذوناته تحديدًا. ترد تفاصيل التدفقات في المصادقة وOAuth، وتوفّر المنصة مثالًا حيًّا على هذا النمط.

مثال حي: مصدر بيانات Grafana

يجسّد مصدر بيانات Grafana نمط الطرف الثالث في بيئة الإنتاج، ويتضمّن خيارين في التصميم يستحقان الاقتداء بهما بدلًا من إعادة اكتشافهما:

- **لا يحمل أي بيانات اعتماد.** يُلحِق وكيل Grafana على جانب الخادم رمز OAuth الخاص بالمُشاهد المسجَّل دخوله بكل طلب (`oauthPassThru`). لذا قد يرى مُشاهدان للوحة المعلومات نفسها بيانات مختلفة من اللوحة ذاتها، لأن كل طلب يُنفَّذ بهوية مُشاهده.
- **يقرأ عبر مسار المعرض في البوابة.** يتجه كل استدعاء إلى `/api/v1/v/{viewId}/…`، حيث تتحقق إضافة Kong المسماة `osos-access` من بقية المسار وفق خطة تنفيذ ذلك المعرض، وتتحقّق من أن للمستدعي حق الوصول إلى المعرض، وتحقن ترويسة `x-view-id` بنفسها. فلا يصل التكامل إلا إلى ما يتيحه المعرض عمدًا؛ فهذه قائمة سماح يفرضها الخادم، وليست وعدًا يقطعه العميل.

إن كنت تبني أي شيء يعرض بيانات المنصة على أشخاص آخرين، فابدأ من هذا النمط: مرّر رمز المُشاهد، واحصر القراءة في معرض.

ما يفرضه الخادم مهما كان العميل

لا تثق المنصة بواجهتها الرسومية، ولذلك لا تحتاج إلى الثقة بواجهتك. وتسري أربعة ضمانات على كل مستدعٍ:

- **سلّم الصلاحيات.** تُقيَّم الأدوار وعضوية المساحة وحقوق الوصول إلى كل معرض على جانب الخادم مع كل طلب. ومن المفيد معرفة السلوك التالي مبكرًا: طلب سرد المُدخلات العادي (غير المرتبط بمعرض) على كيان لا يُسمح للهوية بقراءته يُعيد نتيجة *ناجحة فارغة*، لا خطأً، بينما يجيب المسار المرتبط بمعرض بـ `403` حين يمنع المعرض العملية.
- **قواعد العمل.** تمرّ عمليات الكتابة من عميلك بسلسلة المعالجة نفسها التي تمرّ بها عمليات الكتابة من الواجهة: تظل انتقالات سير العمل، وتفاعلات الأتمتة، وتحدّي التحقق الذي قد يتطلبه إنشاء مُدخل، كلّها سارية.
- **التحقق من الصحة.** يجري التحقق من الحقول والنماذج على جانب الخادم، ويؤدي فشله إلى إرجاع رموز خطأ مُنمَّطة وثابتة. ففحوص عميلك المحلية تُحسِّن تجربة الاستخدام، لكنها لا توفّر أمانًا.
- **نطاق القاطن.** تُحدَّد المساحة من المضيف عند البوابة، وتُنفَّذ كل عملية داخلها. ولا يستطيع العميل مخاطبة قاطن آخر بإضافة ترويسة؛ ويتناول دليل القاطنون والهوية ما تستطيع الهوية التعبير عنه وما لا تستطيعه.

من يملك ماذا

صُمّمت هذه السلسلة عمدًا لتكمّل الأدلة السابقة. فالأدلة القائمة في المنصة تظل مسؤولة عن الموضوعات التي تغطيها أصلًا، وتحيل إليها صفحات هذه السلسلة من دون إعادة سردها. وإذا اختلفت الصياغة في أي وقت، فالمرجع هو الصفحة المسؤولة عن الموضوع:

الشأن	الدليل المالك
التوجيه، ومخطط URL وإصداراته، وغلاف الاستجابة، وجسم الخطأ، وأعراف التقسيم إلى صفحات والترشيح، والحصول على مواصفة OpenAPI	استدعاء واجهة HTTP
موفّر OIDC، وأنواع المنح، وإضافات المصادقة في البوابة، وتمرير ترويسات الهوية	المصادقة OAuthg
إنشاء مستخدم تطبيق ورمز الـ API الخاص به	إضافة مستخدم تطبيق [الإدارة › أنشئ رمز API لعملية تكامل]
إعادة توليد رموز الـ API وإبطالها	رموز الـ API · دورة حياة الرمز
المقابس (sockets) من المتصفح، وRTC	الزمن الحقيقي (المقابس وRTC)
الويب هوك الصادر والتكاملات الواردة	الويب هوك والتكاملات
قراءة بيانات المنصة من Grafana	لوحات معلومات Grafana

ما الذي تغيّر للمُكامِلين `Since 1.10`

تؤثر هذه التغييرات ضمن سلسلة إصدارات 1.10 مباشرةً في العميل الذي يعمل من دون واجهة. ويُوثَّق كلٌّ منها في موضعه؛ والقائمة التالية للإحالة إليها فحسب:

- **صارت المدخلات تحمل رمز تزامن تفاؤلي.** أعِد إرسال `updateIdentifier` الذي قرأتَه، فتفشل عملية الكتابة التي كانت ستطمس تغيير شخص آخر بالحالة `409`، بدلًا من أن تمرّ بصمت. وهو اختياري لكل طلب، وفي تغطيته ثغرات معروفة؛ ترد طريقة الاستخدام في كتابة البيانات، والعقد في استدعاء واجهة HTTP.
- **يستطيع المعرض أن ينشر مستند OpenAPI خاصًّا به**، يضمّ خمس عمليات، ويوفّر أساسًا أضيق نطاقًا وأكثر ثباتًا لتوليد عميل من مواصفة خدمة كاملة – توليد عميل.
- **يقدّم `GET /v1/collections/{id}` تجميعاتك الخاصة والمشاركة معك**، فلم تعد قائمة أنشأتها لتوّك تُرجع `404` – قراءة البيانات.
- **يحمل حدث الـ socket `entries/value-updated` رمزَ المدخل**، مما يتيح للعميل المتّصل ملاحظة التعارض قبل محاولة الحفظ – الزمن الحقيقي.
- **صارت حالات رفض كانت تُجاب بـ `500` تُجاب بحالة 4xx مع رمز ثابت** – ومنها أرشفة كيان مؤرشف أصلًا، وإجراء سير عمل مخفي، ومؤشّر ترشيح غير معروف، وإنشاء مُدخل بلا جسم، وملف مرفوع انتهت صلاحيته؛ وتحتفظ حالة 4xx المنقولة من خدمة أخرى بحالتها (وحالة `422` للإجراء المخفي والحالة المنقولة هما أيضًا `Since 1.8`) – استدعاء واجهة HTTP، مع إرشادات إعادة المحاولة في الأخطاء وإعادة المحاولة.
- **قد ترفض حدود الخطة عمليات الإنشاء** بـ `403` `ENTRY_CAP_EXCEEDED` أو `ENTITY_CAP_EXCEEDED` أو `SPACE_CAP_EXCEEDED` أو `STORAGE_CAP_EXCEEDED` – ولا ترفض القراءة أو التحرير أو التصدير أبدًا – حدود الخطة.
- **يجيب القاطن قيد الصيانة بـ `503 TENANT_IN_MAINTENANCE` مع `Retry-After`** في النشر متعدّد القاطنين، أثناء استعادة بياناته أو نقله؛ فانتظر ثم أعد المحاولة – الرفض المتعلق بالقاطن والمساحة.
- **يستطيع مسؤول النظام لدى القاطن أخذ نسخة احتياطية من بيانات القاطن واستعادتها وتصديرها** عبر واجهة API (في النشر متعدّد القاطنين فقط؛ ولا توجد شاشة ويب في 1.10) – النسخ الاحتياطي للقاطن واستعادته وتصدير بياناته والعمل طويل الأمد.

- **تنزع البوابة x-service التي يرسلها العميل وتمسح `x-view-id` الخاصة به** – ما تنزعه البوابة.
- **عائلات REST جديدة** – قراءة إعدادات النظام العلنية، وقائمة قوالب WhatsApp وخطّافه، ومسارات التحزيم، التي لا تتضمّنها أي مواصفة مُولَّدة – عائلات REST أخرى أُضيفت في 1.10.

وقد أُعيد توليد مرجع نقاط النهاية من لقطات OpenAPI الحالية، وصار يسرد العمليات الجديدة في هذه الدورة إلى جانب سائر العمليات. ارجع إليه للاطلاع على العمليات، لا إلى قائمة في نصّ سرديّ: فهو مُولَّد، ولذلك لا يتقادم.

صفحات هذه السلسلة

ابدأ من هنا. يرشدك البدء السريع إلى الحصول على رمز، ثم يرافقك خطوةً خطوة في أول قراءة مُصادَق عليها، والتعرّف على الغلاف، وخطأ حقيقي واحد، وقراءة مقسّمة إلى صفحات، وعملية إنشاء واحدة – في أقل من خمس عشرة دقيقة.

بيانات الاعتماد والهوية. يقدّم اختيار بيانات الاعتماد مصفوفة لاتخاذ القرار بين الأشكال الثلاثة؛ ويتناول دورة حياة الرمز التجديد والانتهاء والتدوير من دون توقف وتأخّر ظهور أثر الإبطال؛ ويشرح القاطنون والهوية نطاق المساحة والتفويض؛ ويحدّد الحدود الأمنية ما يجوز أن يحمله العميل وما يجب أن يبقى على جانب الخادم.

العمل مع البيانات. يتناول قراءة البيانات بالتفصيل صيغتَي الترشيح والإسقاط والتقسيم الآمن إلى صفحات؛ ويشرح كتابة البيانات الإنشاء والتحديث والعمليات بالجملة ونتائجها لكل صف؛ ويتناول الملفات الرفع والتنزيل ودورة حياة المرفقات؛ ويوضّح الاشتراك في التغييرات كيفية توصيل عميل من خارج المتصفح بالمقبس العام؛ ويتناول الأعمال طويلة الأمد مقابض المهام والاستطلاع والحالات النهائية، والاستجابات التي تأتي في صورة تنزيلات بدلًا من JSON.

إطلاق عميل ناضج. يقدّم الأخطاء وإعادة المحاولة فهرسًا لرموز الخطأ مع سياسة مكتوبة لإعادة المحاولة؛ ويحوّل توليد عميل وثائق OpenAPI لكل خدمة إلى رماز مُنمَّط؛ ويرشدك البيئات والترقية إلى نقل الناتج من التطوير إلى الإنتاج.

البدء السريع مع الواجهة الخلفية

خلال الدقائق الخمس عشرة القادمة، ستنفّذ أول استدعاء مُصادَق عليه إلى قاطن في أسس، وتتعلّم الممارسات الأربع التي يحتاج إليها كل عميل: إرسال بيانات الاعتماد، وفكّ غلاف الاستجابة، والتفرّع بناءً على خطأ فعلي، وتقسيم نتائج السرد إلى صفحات بأمان. يمكنك تنفيذ كل ما يلي باستخدام `curl` وحده؛ وكل جسم استجابة معروض يطابق الصيغة المُغلَّفة التي تُنقَل فعليًا عبر الشبكة.

ست خطوات: احصل على رمز ← نفّذ قراءة مُصادَقًا عليها ← فكّ الغلاف ← عالج خطأً واحدًا ← قسّم إلى صفحات ← أنشئ مُدخلًا.

ما تحتاج إليه

- **قاطن يمكنك الوصول إليه**، مثل `https://acme.example.com`. فالمضيف نفسه يحدّد القاطن، إذ تستنتج البوابة المساحة من ترويسة `Host` في الطلب؛ ولذلك لن تجد أي مُعامل للقاطن فيما يلي.
- **مستخدم تطبيق ورمز API الخاص به**، يُنشئهما لك مسؤول القاطن (الخطوة 1).
- **كيان فيه بضعة مُدخلات** يُسمح للرمز بقراءتها (وبإنشاء مُدخلات فيه، للخطوة 6). تستخدم الأمثلة `employees`؛ فاستبدله بمعرّف أي كيان في مساحتك أو باسمه النموذجي (normalized name)، إذ يقبل المسار كليهما.
- `curl`، واختياريًا `jq`.

اضبط متغيّرين، ثم يمكنك نسخ بقية أوامر الصفحة ولصقها:

```
export BASE="https://acme.example.com"
export TOKEN="<your API token>"
```

الخطوة 1 – احصل على رمز

ينتمي رمز API إلى **مستخدم تطبيق**، أي مستخدم من النوع *تطبيق*. يُنشئ مسؤول القاطن هذا المستخدم من **الإدارة › المنظمة › المستخدمون** (راجع إضافة مستخدم تطبيق [الإدارة › أنشئ رمز API لعملية تكامل])، ثم ترسل المنصة **الرمز الجديد بالبريد الإلكتروني إلى المدير الذي أنشأه**. ولا يُخزَّن الرمز إلا بصيغة مُجزَّأة (hashed)، ولذلك لا يُعرض على الشاشة أبدًا ولا يمكن استرجاعه؛ فإذا فُقد، أعاد مسؤول النظام توليده (**إعادة إنشاء الرمز**). أمّا التدوير والإبطال، فتجد شرحهما في دورة حياة الرمز.

والرمز الذي أُنشئ للتو ليس سرًّا مستقلًّا، بل يرتبط بـ**مستخدم تطبيق** مخصّص، ويُنفَّذ كل طلب يُرسَل به *بهوية ذلك المستخدم*. لذا تتحدّد أذونات الوصول بالضبط بأدوار ذلك المستخدم وعضويته في المساحة، لأن الرمز نفسه لا يحمل نطاقات إذن. قبل المتابعة، تأكد من أن مستخدم التطبيق يملك إذن سرد مُدخلات الكيان الذي اخترته، لأن هذا الدليل يقرؤها من الكيان مباشرةً. ويتطلّب ذلك مستوى **تعديل** على الكيان أو على وحدته: فمستوى **عرض** يُظهر تصميم الكيان لا مُدخلاته، وتعود القائمة حينها فارغة. ويشمل «تعديل» أيضًا إنشاء المُدخلات للخطوة 6. ويمنح المدير هذا الوصول بالطريقة نفسها التي يمنح بها أي وصول آخر: راجع منح مجموعة حق الوصول إلى وحدة أو كيان بيانات [الإدارة › الأذونات والوصول]. وفي عملية تكامل حقيقية تقتصر على القراءة، فضّل معرضًا تجري مشاركته مع مستخدم التطبيق، واقرأ البيانات عبره (`/api/v1/v/{viewId}/…`)، انظر استدعاء واجهة HTTP): فذلك لا يتطلّب إلا الوصول إلى المعرض، لا «تعديل» على الكيان. وإذا كانت الهوية الثابتة لرمز التطبيق لا تلائم تكاملك، كأن تعمل نيابةً عن مستخدمين تفاعليين، فراجع اختيار بيانات الاعتماد.

الخطوة 2 – نفّذ قراءة مُصادَقًا عليها

اسرد مُدخلين. يُرسَل الرمز في الترويسة `x-app-token`، ويُرسَل المرشِّح على هيئة كائن JSON في مُعامل الاستعلام `filterCriteria`؛ ولهذا استُخدم `-G --data-urlencode`:

```
curl -sG "$BASE/api/v2/entities/employees/entries" \
  -H "x-app-token: $TOKEN" \
  --data-urlencode 'filterCriteria={"limit":2}'
```

هذه هي عملية سرد المُدخلات في الإصدار v2. وتأتي الاستجابة بالرمز `200`، وترد المُدخلات ضمن `result`:

```
{
  "success": true,
  "result": [
    {
      "id": "665f1c0d2e9a4b3a9c1d0e2f",
      "dataEntity": "664ade91b2f7a2418c0c11aa",
      "specificationVersion": 1,
      "entityVersion": 4,
      "state": 1,
      "value": {
        "full_name": "Layla Hassan",
        "department": "HR"
      },
      "creationInfo": {
        "doneAt": "2026-05-02T09:14:33.201Z",
        "doneBy": { "dataEntity": "…", "dataEntry": "…" }
      }
      // … further entry metadata trimmed
    },
    { /* second entry */ }
  ]
}
```

يمثّل كل عنصر مُدخل بيانات: ترد حقول العمل ضمن `value`، مفهرسةً بالاسم النموذجي لكل حقل، وتحيط بها البيانات الوصفية للمنصة (مرجع الكيان، والإصدارات، والحالة، ومعلومات التدقيق).

وهناك سلوك ينبغي معرفته منذ البداية، لأن الاستجابة تبدو ناجحة: إذا كانت قيمة `result` هي `[]`، فإما أن الكيان لا يحتوي على مُدخلات **وإما أن مستخدم التطبيق المالك للرمز لا يُسمح له بسردها** – إذ يعيد الخادم في القراءة العادية (غير المقيّدة بمعرض) قائمة فارغة عند غياب إذن السرد، ولا يعيد خطأً، والسرد يتطلّب مستوى **تعديل**، إذ لا يكفي «عرض». لذا، إذا كانت نتيجة القراءة الأولى فارغة، فتحقّق من أذونات مستخدم التطبيق قبل الاستنتاج بأن البيانات غير موجودة.

الخطوة 3 – فكّ الغلاف

تستخدم كل استجابة JSON من واجهة برمجة التطبيقات الغلاف نفسه، `{"success": …, "result": …}`، أيًّا كان الإصدار، مع استثناء واحد `Since 1.10`: أصبح طلب `GET` على مسار `/v3/` يعيد الاستجابة الخام مجددًا (التفاصيل). يستخدم هذا الدليل `v2` للقراءة، لذا ترد الحمولة دائمًا ضمن `result`، وتكون قيمة `result` هي `null` حين لا تعيد العملية أي محتوى. أمّا الاستجابات الوحيدة التي تصل من دون غلاف، فهي التي لا تمثّل نتائج JSON أصلًا: تنزيلات الملفات، والتدفّقات التي تكتب مباشرةً في الاستجابة، ووثيقة OpenAPI. وتجد مواصفات الاستجابة كاملةً في استدعاء واجهة HTTP.

لذلك، يفكّ العميل الغلاف مرة واحدة في طبقة النقل، لتتعامل جميع الطبقات الأعلى منها مع الحمولات مباشرةً:

```
curl -sG "$BASE/api/v2/entities/employees/entries" \
  -H "x-app-token: $TOKEN" \
  --data-urlencode 'filterCriteria={"limit":2}' \
| jq -r '.result[].value.full_name'

Layla Hassan
Omar Haddad
```

وتنطبق القاعدة نفسها إذا ولّدت عميلًا مُحدَّد الأنواع من وثائق OpenAPI؛ إذ تصف مخططات كل عملية حمولة `result`، ويبيّن توليد عميل الموضع المناسب لفكّ الغلاف.

الخطوة 4 – عالج خطأً واحدًا

اطلب كيانًا غير موجود (يطبع الخيار `-i` سطر الحالة):

```
curl -siG "$BASE/api/v2/entities/no-such-entity/entries" \
  -H "x-app-token: $TOKEN"
```

تكون الحالة `404`، ويأتي الجسم بصيغة الخطأ الموحّدة في المنصة:

```
{
  "success": false,
  "result": {
    // remnants of the server-side exception — do not build on this field
    "errorCode": "ENTITY_NOT_EXIST",
    "errorMessage": "The data entity 'no-such-entity' does not exist",
    "service": "data-service",
    "retryable": false,
    "requestCode": 404
  },
  "error": {
    "code": "ENTITY_NOT_EXIST",
    "message": {
      "en": "The entity does not exist",
      "ar": "الكيان غير موجود"
    },
    "traceId": "8c2f5e0d-…"
  }
}
```

أربع قواعد، مرتّبة بحسب أهميتها:

1. **تفرَّع على `error.code`.** فقيمته سلسلة نصية ثابتة مخصّصة للمعالجة الآلية، نتجت هنا عن استثناء «غير موجود» مُحدَّد النوع. أمّا `error.message`، فهو نص مُترجَم مخصّص للعرض وقد يتغيّر، فلا تعتمد عليه في المطابقة أبدًا.
2. **اعتمد على حالة HTTP في المعالجة العامة.** فاستثناءات المنصة المُحدَّدة النوع تتضمّن حالة HTTP الخاصة بها: التحقق `400`، وعدم المصادقة `401`، والمنع `403`، وعدم الوجود `404`، والتعارض `409`. وتجد القائمة الكاملة للرموز، مع بيان قابلية إعادة المحاولة لكل رمز، في الأخطاء وإعادة المحاولة.
3. **سجّل `error.traceId`.** فهو معرّف الربط الخاص بالطلب، ويتيح تسجيله تتبّع المشكلة على امتداد مسارها في سجلّات المنصة.

4. **تجاهل `result` عند الفشل.** فقيمتها *ليست* `null`؛ إذ يُبقي مصنع الاستجابات حاليًا بقايا كائن الاستثناء الداخلي فيها. لذا اعتبر `"success": false` مع `error` علامة الفشل، ولا تقرأ سواهما.

الخطوة 5 – قسّم إلى صفحات

ابدأ بالعدّ، فهو عملية مستقلة عن السرد:

```
curl -sG "$BASE/api/v2/entities/employees/entries/count" \
  -H "x-app-token: $TOKEN"
```

```
{ "success": true, "result": { "count": 137 } }
```

ثم قسّم النتائج إلى صفحات باستخدام `limit` و`skip` داخل كائن `filterCriteria` نفسه، مع تحديد `order` صراحةً دائمًا:

```
total=$(curl -sG "$BASE/api/v2/entities/employees/entries/count" \
  -H "x-app-token: $TOKEN" | jq '.result.count')

for ((skip = 0; skip < total; skip += 50)); do
  curl -sG "$BASE/api/v2/entities/employees/entries" \
    -H "x-app-token: $TOKEN" \
    --data-urlencode "filterCriteria={\"limit\":50,\"skip\":$skip,\"order\":[\"creationInfo.doneAt ASC\"]}" \
  | jq '.result[].value'
done
```

ينبغي أن تراعي في تصميمك ثلاث حقائق، يحدّدها جميعًا مخطط المرشِّح:

- التقسيم **قائم على الإزاحة**: باستخدام `limit` مع `skip` (أو مرادفه `offset`)؛ ولا مؤشرات تسلسل (cursors). وقد تؤدي التغييرات في مجموعة البيانات أثناء التقسيم إلى انتقال الصفوف بين الصفحات؛ لذا رتّب دائمًا بحسب مفتاح ثابت، كالطابع الزمني للإنشاء في المثال أعلاه.
- يقبل `order` الصيغة `"<field> ASC|DESC"`، في صورة نص مفرد أو مصفوفة، تمامًا كما في الأمثلة أعلاه.
- قد يتغيّر العدد الذي جلبته أثناء التقسيم؛ لذا تعامَل معه بوصفه تقديرًا للتقدّم، لا عددًا مضمون الثبات.

الخطوة 6 – أنشئ مُدخلًا

تستقبل عمليات الكتابة جسمًا بصيغة JSON. أرسل حقول العمل للمُدخل الجديد ضمن `value`، مفهرسةً بالاسم النموذجي لكل حقل:

```
curl -s -X POST "$BASE/api/v1/entities/employees/entries" \
  -H "x-app-token: $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"value": {"full_name": "Sara Nasser", "department": "Finance"}}'
```

يرد المُدخل المُنشأ ضمن `result`، ومعه `id` الجديد والبيانات الوصفية نفسها التي رأيتها في الخطوة 2:

```
{
  "success": true,
  "result": {
    "id": "6660a1b2c3d4e5f6a7b8c9d0",
    "dataEntity": "664ade91b2f7a2418c0c11aa",
    "state": 1,
    "value": { "full_name": "Sara Nasser", "department": "Finance" }
    // … further entry metadata trimmed
  }
}
```

وإذا لم يستوفِ حقلٌ قواعد الكيان، كأن يُغفَل حقل إلزامي، فتكون حالة الاستجابة `400`، وقيمة `error.code` هي `ENTRY_VALIDATION_ERROR` (أو `ALL_ENTRY_VALIDATION_ERROR`، مع عنصر لكل حقل أخفق في التحقق)، وتحدّد تفاصيل الخطأ الحقل المعنيّ؛ لذا تفرّع بناءً على رمز الخطأ تمامًا كما في الخطوة 4. أمّا التحديثات JSON Patchg والإنشاء على دفعات ونتائج الكتابة بالجملة لكل صف، فتجدها في كتابة البيانات.

إلى أين بعد ذلك

- قراءة البيانات – شجرة التعبير `criteria` داخل `filterCriteria`، والإسقاط باستخدام `fields`، والتحقق من مرشِّح قبل حفظه، والمرشِّحات السريعة ومرشِّحات الوسوم.
- كتابة البيانات – الإنشاء والتحديث والعمليات بالجملة، مع نتائج كل صف.
- الأخطاء وإعادة المحاولة – القائمة الكاملة لقيم `error.code`، وسياسة مكتوبة لإعادة المحاولة والتراجع.
- اختيار بيانات الاعتماد ودورة حياة الرمز – متى لا يلائم رمزُ التطبيق تكاملك، وكيف تدوّر الرمز دون توقف الخدمة.
- استدعاء واجهة HTTP – الأعراف التي استندت إليها هذه الصفحة: التوجيه، والإصدارات، والغلاف، ولهجتا الترشيح.

اختيار بيانات الاعتماد

قبل أن يستطيع عميل خارجي استدعاء واجهة أسس البرمجية، يحتاج إلى بيانات اعتماد تقبلها البوّابة، واختيار النوع الخاطئ هو أكثر الطرق شيوعًا لبدء التكامل بداية متعثّرة. تفهم البوّابة شكلين من بيانات الاعتماد: **رمز OAuth 2.0** في ترويسة `Authorization` يُتحقّق منه عبر الاستبطان (introspection)، و**رمز تطبيق** طويل العمر في ترويسة `x-app-token`. وخلف الشكل الأول تقف عدة أنواع من المنح، لكلٍّ منها نوع مُستدعٍ مختلف صُمّمت له.

تملك هذه الصفحة قرارًا واحدًا: **أي بيانات اعتماد تناسب أي مُستدعٍ**. وهي لا تشرح كيفية إصدار أيٍّ منها — فصفحة المصادقة OAuthg تغطّي مزوّد OAuth ومنحه ونقاط نهايته بعمق، وصفحة رموز الـ API تغطّي إنشاء رموز التطبيق وإدارتها. ومتى حسمت قرارك، تأخذك البداية السريعة من رمز جديد إلى استدعاء ناجح.

القرار في لمحة

أنت تبني…	استخدم	لأن
تطبيقًا تفاعليًا يسجّل فيه شخص دخوله (ويب، جوال)	OAuth **authorization code + PKCE**	المنصة تستضيف عنك تسجيل الدخول والموافقة والتحقّق المتعدد ومفاتيح المرور؛ وهوية المستخدم نفسه هي التي تسري في الطلبات.
نصًا برمجيًا موثوقًا من الطرف الأول يحمل بيانات اعتماد مستخدم	منحة **password** في OAuth	مبادلة مباشرة لاسم المستخدم/كلمة المرور بالرموز — مقبولة فقط عندما يستطيع النص حمل كلمة مرور خام بأمان.
خدمة بلا واجهة، أو تكاملًا، أو مهمة مجدولة	**رمز التطبيق** (`x-app-token`)	لا مبادلة رموز ولا حلقة تحديث؛ هوية آلية مستقرة تحمل أذونات مستخدم واحد بعينه.
منتجًا من طرف ثالث يتصرّف نيابة عن مستخدميك	**authorization code** في OAuth مع الموافقة	كل مستخدم يفوّض عميلك بنفسه؛ فلا ترى بيانات اعتماد أحد، وإبطالُ مستخدمٍ لا يمسّ الآخرين.

بقية هذه الصفحة تمشي صفًّا صفًّا في هذا الجدول، ثم تغطّي حدود رمز التطبيق — بيانات الاعتماد التي ينتهي إليها معظم العملاء غير التفاعليين.

مستخدم تفاعلي

استخدم **تدفّق authorization code مع PKCE**. عملاء SPA المشحونون (`osos-<env>` و`osos-<env>`) هم النمط الذي يُحتذى: `token_endpoint_auth_method: "none"` (عميل عام)، ومنحتا `authorization_code` و`refresh_token`، وعناوين إعادة توجيه بنطاقات فرعية ذات حرف بدل بحيث يخدم تسجيلُ عميلٍ واحد كلَّ نطاق فرعي لقاطن. يُجهَّز العملاء تجهيزًا ساكنًا — فلا توجد نقطة تسجيل ديناميكي — أي أن تسجيل عميلك يعني إضافته إلى ملف العملاء (أو ملف التجاوز `additional-clients.json`) وإعادة تشغيل خدمة الهوية. ومنذ `Since 1.10` صار هذان الملفان زرعًا يُكتب في مجموعة عملاء القاطن عند الإقلاع، وفي نشرٍ متعدّد القاطنين تُولَّد العملاء القياسيون لكل قاطن بدلًا من ذلك (انظر مزوّدو OpenID لكل قاطن). والصفحة في المتصفح التي لا تحتاج إلا وصولًا مجهولًا تستخدم `POST /api/identity/v1/anonymous/token` — لا سرّ عميل في المتصفح أبدًا.

سلوكان يمنحهما المزوّد للعملاء التفاعليين دون كلفة:

- العميل العام من نوع الويب يتلقّى **رمز تحديث حتى دون طلب** `offline_access` (الدالة `issueRefreshToken`)، فتبقى الجلسات حيّة بعد انتهاء صلاحية رمز الوصول.
- واجهة تسجيل الدخول المستضافة تعرض العوامل التي فعّلها القاطن — كلمة المرور، رمز لمرة واحدة، مفاتيح المرور، مزوّدي هوية خارجيين — دون أن ينفّذ عميلك أيًّا منها.

نص برمجي موثوق من الطرف الأول

يسجّل المزوّد **منحة password** مخصّصة: يرسل النص البرمجي اسم مستخدم وكلمة مرور إلى `/token` ويتلقّى الرموز مباشرة. هذا هو الشكل الصحيح فقط عندما يصدق الوصفان *طرف أول* و*موثوق* معًا فعلًا – فالنص يرى كلمة المرور الخام، فيجب أن يكون نصَّك أنت، والحساب يخضع لسياسة الحسابات المعتادة (المحاولات الخاطئة تُحتسب نحو القفل).

وازِنها مع رمز التطبيق قبل اختيارها. منحة password تربط أتمتتك بحساب إنسان: أذوناته تتغيّر حين تتغيّر أذونات صاحبه، وكلمة مروره تُدوَّر على جدول صاحبه، وقفلُ الحساب يوقف مهمتك. فإن كان النص البرمجي في حقيقته خدمةً متنكّرة، فمستخدمُ تطبيقٍ مخصّص برمزه الخاص هو الهوية الأثبت.

خدمة بلا واجهة

استخدم **رمز التطبيق**. يُسَكّ الرمز لصالح **مستخدم تطبيق** مخصّص – مُدخل `USER` نوعه `APPLICATION_USER` – ويُرسَل في `x-app-token` مع كل طلب. تتحقّق منه إضافة البوّابة `app-token-auth` لدى خدمة الهوية وتحقن معرّف المستخدم المالك في `x-user-id`؛ ومن هناك لا يتميّز الطلب عن ذلك المستخدم وهو يتصرّف بنفسه. لا مبادلة OAuth، ولا حلقة تحديث، ولا جلسة تُبقيها حيّة – أما دورة الحياة التشغيلية التي يملكها فعلًا (الانتهاء والتدوير والإبطال) فتغطّيها صفحة دورة حياة الرموز.

ولماذا ليس `client_credentials`؟ منحة الآلة-إلى-آلة موجودة ومفعّلة، لكن رمز client credentials يعرّف *عميلًا* لا *مستخدمًا*: نتيجة استبطانه لا تحمل subject، فلا تمرّر البوّابة أي `x-user-id`، وتحلّ الخدمات اللاحقة المُستدعي إلى ملف تعريف **بأذونات فارغة**. الطلب يُصادَق – ثم لا يكاد يقرأ شيئًا، لأن مستوى البيانات يفوّض المستخدمين لا العملاء. لا تصبح المنحة مفيدة إلا مقترنة بإعداد الانتحال `trustedClientId` في البوّابة، وهو قرار ثقة على مستوى النشر خاص بالبنية الداخلية، لا نمط تكامل للتطبيقات. فللمُستدعي بلا واجهة الذي يحتاج رؤية بيانات القاطن، رمز التطبيق هو الجواب المدعوم. والسطح الوحيد الذي لا يفتحه هو مقبس الوقت الفعلي، إذ يحتاج إلى رمز OAuth لمستخدم؛ وللخدمة يعني ذلك منحة password لمستخدم عادي مخصّص – فمستخدم التطبيق لا يستطيع تسجيل الدخول بكلمة مرور (الاشتراك في الأحداث).

طرف ثالث يتصرّف نيابة عن مستخدم

شغّل **تدفّق authorization code** ودَع كل مستخدم يمنح موافقته. العملاء المشحونون `zapier` و`wordpress` و`n8n` في `localhost` أمثلة عاملة على هذا النمط بعينه: عنوان إعادة توجيه ثابت لدى الطرف الثالث، ومنحتا `authorization_code` و`refresh_token`، ونطاقات قياسية.

طريقان مختصران مغريان، وسببُ خطئهما هنا:

- **ليست منحة password.** لا يجوز أبدًا أن يلمس طرف ثالث كلمات مرور المستخدمين؛ فالمنحة وُجدت لثقة الطرف الأول حصرًا.
- **ولا رمز تطبيق مشترك.** رمز التطبيق هوية واحدة ثابتة، فتُنسَب أفعال جميع المستخدمين إلى مستخدم التطبيق نفسه، وتنهار الأذونات الفردية في ملف تعريف واحد، ويعني إبطال وصول عميلٍ واحد إبطال وصول الجميع.

وليست آلية **التفويض** في المنصة (ترويسة `X-Delegation-Id`) هذا الشيء أيضًا: فهي تتيح *لشخص* أن يتصرّف عن شخص آخر ضمن نافذة مصرّح بها، وليست سطحًا لتكامل الآلات. تغطّيها صفحة القاطنون والهوية.

ما لا يستطيع رمز التطبيق التعبير عنه

رمز التطبيق **لا يحمل نطاقات (scopes)**. فمستند الرمز المخزّن يضم هوية وبيانات دورة حياة – تجزئة argon2 والاسم ورايتي التفعيل والانتهاء – ولا شيء غيرها؛ والبوّابة لا تستهلك من التحقّق الناجح إلا معرّف المستخدم المالك. لا يوجد في هذا المسار

كله حقلٌ يمكن أن يضيّق رمزًا إلى "قراءة فقط" أو "هذا الكيان دون سواه".

كلُّ ما يستطيع الرمز فعله يقرّره إذن **مستخدم التطبيق المالك**: أدواره، وعضويات مساحاته، وأذونات معارضه. ولهذا ثلاث نتائج عملية:

- **الحد الأدنى من الامتيازات يُنمذَج على المستخدم لا على الرمز.** أنشئ مستخدم تطبيق مخصّصًا لكل تكامل وامنحه ما يحتاجه ذلك التكامل تحديدًا.
- **تكامل واحد، رمز واحد.** لمّا كان الرمز لا يُضيَّق، فالعزل بين التكاملات يأتي من مستخدمين مالكين منفصلين – وهو ما يجعل الإبطال جراحيًا أيضًا.
- **التدقيق يتبع المستخدم.** تظهر الأفعال باسم مستخدم التطبيق، فرمزٌ مشترك بين عدة أنظمة يدمّر نسبة الأفعال إلى أصحابها.

الوراثة نفسها تحدّ الرمز مكانيًا أيضًا: فلا يعمل إلا في المساحات التي ينتمي إليها مستخدمه المالك، ويُفرَض ذلك مع كل طلب عند البوّابة – انظر القاطنون والهوية.

القرار السريع

إن كان ثمة شخص أمام شاشة، فـ authorization code + PKCE. وإن لم يكن ثمة شخص والمُستدعي مُستدعيك أنت، فرمز تطبيق يملكه مستخدم تطبيق مبني لغرضه. وإن لم يكن ثمة شخص والمُستدعي ملك غيرك، فليمنح مستخدموه موافقتهم عبر تدفّق authorization code. واحفظ منحة password للحالة الضيّقة: نص برمجي من الطرف الأول لا بدّ له فعلًا من حمل كلمة مرور، وعامل `client_credentials` معاملة سباكة البنية التحتية لا بيانات اعتماد تكامل.

دورة حياة الرموز

كل بيانات اعتماد تشيخ: رموز الوصول تنتهي في دقائق، ورموز التحديث تُدوَّر، ورموز التطبيق تبلغ تاريخ انتهائها أو يدوّرها مدير، وأيٌّ منها قد يُبطَل. هذه الصفحة هي الجانب التشغيلي من تلك القصة لمؤلّف العميل – ما الذي ينتهي ومتى، وكيف تحدّث وتدوّر دون انقطاع، والخاصية الوحيدة التي تفاجئ معظم المتكاملين: **الإبطال ليس فوريًا**، لأن البوّابة تخبّئ نتائج التحقّق.

صفحة اختيار بيانات الاعتماد تقرّر أي بيانات اعتماد تحمل؛ وصفحة المصادقة OAuthg توثّق جدول مُدد الحياة الكامل لدى المزوّد وآليات المنح؛ وصفحة رموز الـ API توثّق إنشاء رموز التطبيق وإدارتها. تفترض هذه الصفحة أن لديك بيانات اعتماد وتحتاج إلى إبقائها عاملة.

نوعان من الاعتماد، وساعتان مختلفتان

يشيخ شكلا بيانات الاعتماد على مقياسين مختلفين تمامًا، ويُصمَّم العميل حول الساعة التي يعيش عليها:

	رمز OAuth	رمز التطبيق
العمر المعتاد	رمز الوصول: ساعة واحدة افتراضيًا، مع تجاوز لكل عميل (`ttl.AccessToken`)	تاريخ انتهاء تقويمي، أو بلا حد
ما يبقيه حيًا	دورة التحديث (رمز التحديث: 14 يومًا، مع تدوير)	لا شيء – هو صالح ببساطة حتى الانتهاء أو التعطيل أو التدوير
مَن يجدّده	العميل نفسه، برمجيًا	مديرٌ، عن عمد
عند الاختراق	أبطِله لدى المزوّد	عطّل رمز مستخدم التطبيق المالك أو دوّره

لذا يحمل عميل OAuth آلية تحديث الرموز مكوّنًا جوهريًا فيه؛ أما عميل رمز التطبيق فلا يحمل شيئًا منها، ويحتاج بدلها إلى جواب تشغيلي عن سؤال "ماذا يحدث يوم يُدوَّر الرمز".

تحديث جلسة OAuth

رموز الوصول قصيرة العمر بالتصميم؛ ورمز التحديث هو ما يدوم. المنح التي تُصدر رموز تحديث، وسياسة التدوير، ومُدد الحياة كلها محدّدة وموثّقة في المصادقة OAuthg؛ وما يهم العميل وهو يعمل هو الانضباط:

- **خزّن دائمًا رمز التحديث من آخر استجابة.** التدوير يعني أن استجابة تحديثٍ قد تحمل رمز تحديث *جديدًا* يحلّ بصمت محل الذي استعملته – العملاء العامّون من نوع الويب يدوّرون مع كل استعمال، وكل عميل يدوّر متى تجاوز الرمز 70% من عمره (`rotateRefreshToken`). وإعادة استعمال رمزٍ دُوِّر بعيدًا تفشل.
- **التدوير ليس خلودًا.** العمر الكلي بالتدوير محدود بسنة واحدة، والرمز المُدوَّر لعميلٍ عام يرث العمر *المتبقي* من سلفه بدل أن يبدأ عدّاد الأربعة عشر يومًا من جديد.
- **حدِّث استباقًا لا ارتكاسًا.** حقل `expires_in` في استجابة الرمز يخبرك متى يموت رمز الوصول؛ والتحديث قُبيل ذلك خيرٌ من سباق `401` في الإنتاج. وإذا تلقّيت `401` في منتصف العمل، فتحديثٌ واحد وإعادة محاولة هما التعافي الصحيح؛ أما `401` ثانية فتعني أن الجلسة انتهت فعلًا وعلى العميل إعادة المصادقة.

انتهاء صلاحية رمز التطبيق

`Since 1.8` – تاريخ انتهاء رمز التطبيق **يُفرَض عند التحقّق**. قيمة `expiresAt` المخزّنة تاريخٌ تقويمي (`yyyy-MM-dd`)، ويبقى الرمز صالحًا **حتى نهاية ذلك اليوم (بتوقيت UTC)**؛ والرمز الموسوم `unlimitedExpiry` ، أو الذي لا تاريخ له، لا ينتهي أبدًا (`isTokenExpired`). وقيمة `expiresAt` التي لا تسمّي تاريخًا تقويميًا حقيقيًا تُعامَل على أنها **منتهية** – الفحص يفشل مُغلقًا ويسجّل القيمة الفاسدة بدل أن يمنح رمزًا خالدًا بالصدفة.

النتائج العملية:

- الانتهاء **حدُّ يومٍ** لا طابعٌ زمني. خطّط للتدوير قبل اليوم بيوم، لا قبل الساعة بساعة.
- الرمز الذي توقّف في موعده ليس حادثة؛ وإعادة توليده (أدناه) تُصدر سرًّا جديدًا وتعيد في الوقت نفسه مزامنة حقول الانتهاء من مُدخل مستخدم التطبيق.
- **صار بالإمكان ضبط تاريخ انتهاء فعلًا** `Since 1.10` . قبل 1.10 كان نموذج مستخدم التطبيق يُبقي حقل *تاريخ الانتهاء* مخفيًا، فكان كل رمز يُنشأ بلا تاريخ انتهاء ولا ينتهي أبدًا. صار الحقل الآن تاريخًا (`yyyy-MM-dd`) يظهر ما لم تكن *صلاحية غير محدودة* مفعّلة. والمُدخل الذي خُزّن تاريخ انتهائه طابعًا زمنيًا كاملًا يُعامل رمزه على أنه منتهٍ إلى أن يُعاد إدخال التاريخ – انظر رموز الـ API.

التدوير دون انقطاع

`Since 1.8` – يستطيع المدير **إعادة توليد** رمز تطبيقٍ من منطقة الإدارة. تُصدر إعادةُ التوليد سرًّا جديدًا، وترسله بالبريد إلى المدير الطالب، و– هذا هو الجزء الذي يجعل التبديل دون انقطاع ممكنًا – تُبقي الرمز **السابق** صالحًا طوال نافذة سماح (`regenerateApiToken`). تُضبط النافذة على مستوى النشر عبر `APP_TOKEN_GRACE_PERIOD_MS` ، وقيمتها الافتراضية **24 ساعة**؛ وضبطها على `0` يُبطل الرمز المستبدَل فورًا.

إجراء التبديل الذي تمليه الآلية:

1. أعد التوليد. الرمز القديم يواصل المصادقة؛ لا شيء ينكسر بعد.
2. انشر السر الجديد في التكامل ضمن نافذة السماح.
3. تحقّق من سريان الحركة على الرمز الجديد قبل انغلاق النافذة – سجل الرمز يتتبّع `lastUsedAt` ، فالتكامل الذي ما يزال على السر القديم قابلٌ للاكتشاف.

وحافة حادّة واحدة، من الرماز مباشرة: السجل يتذكّر تجزئة سابقة **واحدة** بالضبط. فإعادة التوليد مرتين متتاليتين تكتب فوق مدخل السماح الأول – الرمز الذي ما يزال تكاملك يعمل به – وتبطله في الحال. دوِّر، ثم رحِّل، ثم تحقّق، ولا تدوِّر ثانيةً إلا بعدها. (ومنذ `Since 1.10` صارت عمليتا إعادة التوليد *المتزامنتان* متسلسلتين لكل تطبيق عبر الحاويات، فلم يعد بإمكانهما أن تدوّرا كلتاهما من الرمز نفسه؛ أما الحالة المتتالية أعلاه فلا تزال قائمة.)

الإبطال

لبيانات اعتماد OAuth يعرض المزوّد **نقطة الإبطال** القياسية (`/token/revocation` ، ضمن الخصائص المفعّلة)؛ فإبطال رمز التحديث ينهي قدرة الجلسة على التجدّد، ويمكن إبطال رمز وصولٍ بمفرده.

أما رمز التطبيق فيُبطَل عبر مُدخل مستخدم التطبيق المالك، بإحدى درجتين:

- **تعطيل** مستخدم التطبيق – يطفئ معالجُ التحديث راية `isActive` على الرمز؛ وإعادة التفعيل تعيدها.
- **أرشفة** المُدخل – يُحذف مستند الرمز حذفًا ناعمًا ويتوقف عن اجتياز التحقّق نهائيًا؛ وتُحفظ السجلات للتدقيق ولا تُحذف حذفًا صلبًا أبدًا.

كلاهما نافذٌ لدى *خدمة الهوية* فورًا. أما هل تواصل *البوّابة* قبول الرمز مدةً بعدها، فذلك موضوع القسم التالي.

فجوة ظهور الإبطال

تخبّئ البوّابة نتائج التحقّق الناجحة في ريديس كي لا تكلّف الحركةُ المستقرة أي جولة إلى خدمة الهوية. والخبء نفسه يعني أن بيانات اعتماد أُبطلت لدى المزوّد قد تواصل المصادقة عند الحافة حتى يموت مدخلها المخبوء. المساران يخبّئان على نحوين مختلفين – وكلا المدتين في مخططي الإضافتين وقابلتان للضبط في النشر؛ والإعداد المشحون (`api-gateway/config/kong.yml`) لا يتجاوز أيًّا منهما فتبقى القيم الافتراضية:

المسار	النتيجة المخبوءة	المدة	هل يُحترم الانتهاء من الخبء؟
رمز OAuth (إضافة `oauth2-introspection`)	استجابة الاستبطان	`cache_ttl`، افتراضيًا **300 ثانية**، ومقيّدة إضافيًا بالعمر المتبقي للرمز ناقص 10 ثوانٍ؛ والرموز التي يفصلها عن الانتهاء أقل من 30 ثانية لا تُخبَّأ أصلًا	**نعم** – المدخل المخبوء الذي تجاوز `exp` رمزه يُطرد (بهامش انحراف ساعةٍ خمسُ ثوانٍ) ويُعاد استبطانه
رمز التطبيق (إضافة `app-token-auth`)	نتيجة التحقّق	`cache_ttl`، افتراضيًا **300 ثانية**	**لا** – إصابة الخبء تُصدَّق كما هي طوال المدة كاملة؛ لا يُعاد فحص شيء حتى تنقضي

اقرأ الجدول بعين مؤلّف العميل وبعين المشغّل:

- **الانتهاء سريع؛ والإبطال متّسقٌ في نهاية المطاف.** الرمز المنتهي يتوقف عند الحافة خلال ثوانٍ، أما رمز OAuth *المُبطَل غير المنتهي* – ورمز التطبيق المعطَّل أو المُدوَّر عنه حديثًا – فقد يواصل المصادقة حتى `cache_ttl` المضبوطة بعد الإبطال.
- **حدُّ الفجوة خيارُ نشرٍ.** فهو يساوي `cache_ttl` في الإضافتين (300 ثانية كما شُحنت)، ويُقيَّد على مسار OAuth إضافيًا بالعمر المتبقي للرمز نفسه. والنشر ذو متطلبات الإبطال الأشد يضبط `cache_ttl` أصغر ويدفع الثمن حركةَ تحقّق.
- المصادقة ليست الشيء الوحيد المخبوء: فما *يُسمح* للمستخدم بفعله يُخبَّأ على حدة وعلى ساعات أطول – انظر جدول الخبء في القاطنون والهوية.

تصميم العميل حول هذا كله

تنضغط حقائق دورة الحياة أعلاه في مجموعة صغيرة من قواعد العميل. حدّث جلسات OAuth قبل الانتهاء وعامل `401` الواحدة مشغّلَ تحديثٍ لا فشلًا. خزّن رمز التحديث الذي وصل آخرًا أيًّا كان. ولرموز التطبيق، جدوِل التدوير قبل يوم الانتهاء، وأتمّ التبديل ضمن نافذة السماح، ولا تفترض أبدًا أن بيانات اعتماد مُبطَلة ماتت في كل مكان لحظيًا – ابنِ مسارات سحب الصلاحيات على مصدر الهوية (عطّل المستخدم، أبطل الجلسة) ودع خبء الحافة ينضب في وقته المحدود.

القاطنون والهوية

منصة أسس متعددة القاطنين، ويمتد نموذج القاطنين إلى كل استدعاء للواجهة البرمجية: فالعملية نفسها، ببيانات الاعتماد نفسها، تعيد بيانات مختلفة بحسب **المساحة** التي يُوجَّه إليها الطلب. تشرح هذه الصفحة كيف يُحدَّد ذلك النطاق، وكيف تنتقل الهوية والتفويض مع الطلب، ثم – وهو الأهم عند تصميم عميل خارجي – ما الذي *لا تستطيع* بيانات الاعتماد فعله عبر المساحات، مهما كانت طريقة تقديمها.

ترد ترويسات الهوية نفسها (`x-user-id` و `x-shared-key` وأخواتهما) في استدعاء الواجهة البرمجية HTTP؛ أما سبب ثقة الخدمات اللاحقة بها أصلًا فتتناوله صفحة الحدود الأمنية. وهنا نتتبّع مسار القاطن والتفويض من أوله إلى آخره.

المساحة تأتي من المضيف لا من الرمز

لا يرسل العميل معرّف قاطن أبدًا؛ فإضافة البوّابة `space-resolver` تعمل في بداية سلسلة الوصول قبل جميع إضافات المصادقة: تقرأ ترويسة `Host` في الطلب، وتبحث عن قيمتها في مفتاح ريديس اسمه `:spaces` ، وتضبط معرّف المساحة المطابقة في ترويسة الطلب `x-space-id` (مع `x-space-maintenance` حين تكون المساحة مغلقة أمام المستخدمين النهائيين). ويُرفض الطلب الوارد إلى مضيف لا يطابق أي مساحة فورًا بـ `404 SPACE_ERROR` (`Since 1.10` ؛ وكان `400`) – حتى قبل أي محاولة للمصادقة.

أما خبء `:spaces` فتملكه خدمة البيانات، وتعيد كتابته كلما تغيّرت المساحات (`refreshSpaceInRedis`): وهو مصفوفة مُدخلات `{id, hosts, name, is_end_user_deactivated}` تشمل كل مساحة نشطة غير معطَّلة، ويُشتق مضيف كل مساحة من نطاقها ومعرّف نطاقها الفرعي. وتستخدم إضافة شقيقة، `domain-resolver` ، ترويسة `Host` نفسها للبحث عبر مفتاح `:domains` وتحديد قيمة ترويسة `X-DOMAIN-ID` التي سيتطلبها فحص الوصول إلى المعارض لاحقًا.

وتترتب على ذلك نتيجتان تهمّان مطوّر العميل:

- **أنت تخاطب القاطن بعنوان URL.** توجيه العميل نفسه إلى `hr.example.com` ثم إلى `sales.example.com` يتيح الوصول إلى بيانات مختلفة كليًا دون أي تغيير آخر.
- **أيّ `x-space-id` ترسله لا قيمة له.** تضبط البوّابة الترويسة بنفسها بناءً على نتيجة البحث عن المضيف؛ وبدءًا من `Since 1.10` تزيل أولًا أي `x-space-id` / `x-space-maintenance` قدّمه العميل. (وفي تدفّقات OAuth يُربط القاطن أيضًا *داخل الرمز الموقّع* عبر مطالبة `extra.domain` – انظر المصادقة OAuthg.)

العضوية تُفحص مع كل طلب

لكن تحديد المساحة من المضيف لا يثبت أنك *أنت* تنتمي إليها؛ فتلك مهمة إضافة `user-space-checker` التي تتحقّق مع كل طلب مُصادَق من أن المساحة المحدّدة في `x-space-id` ضمن مساحات المستخدم المُستدعي، وتجيب بـ `403` `SPACE_FORBIDDEN` إن لم تكن كذلك – وتُرفق قائمة `available_spaces` (المعرّف والاسم والأيقونة والعنوان) كي يعرض العميل على المستخدم مساحاته الفعلية بدلًا من تركه أمام طريق مسدود.

تُجلب العضوية من خدمة البيانات (`GET /api/v1/spaces/user/{userId}`) وتُخزَّن مؤقتًا في ريديس لمدة قابلة للضبط (افتراضيًا **300 ثانية**)، بمفتاح لكل مستخدم. وفترة بقاء البيانات المتقادمة أقصر مما تبدو: فحين تتغيّر عضوية مستخدمٍ ما في المساحات، تحذف خدمة البيانات مدخل الخبء الخاص به لدى البوّابة مباشرةً (`invalidateKongUserSpacesCache`)، فتغدو المدة حدًّا احتياطيًا لا مدةً لانتشار التغيير.

وتتخطّى الفحصَ قائمةُ سماح صغيرة تضم نقاط نهاية التهيئة الأولية – الملف الشخصي، ومساحات المستخدم، واللغة، والسمات وما شابهها (`مهيّأة عند البوابة`) – مما يتيح للعميل أن يسأل «ما المساحات المتاحة لي؟» قبل أن يكون *داخل* أي مساحة.

كيف تبلغ المساحة مستوى البيانات

بعد البوّابة، ينتقل معرّف المساحة في ترويسة `x-space-id` وحدها. وتحدّده خدمات LoopBack عبر مزوّد يعمل ضمن نطاق الطلب (`SpaceIdProvider`) – وكان قبل 1.10 يرجع إلى معرّف المساحة الجذر حين تغيب الترويسة (وصار الآن يرفض الطلب عند غيابها) – وتحقن المتحكّمات ذلك المعرّف المحدّد في كل أمر واستعلام تصدره؛ كما تمرّره متحكّمات مستوى البيانات في عمليات القراءة والكتابة والعدّ والتصدير على حدّ سواء.

وتفرض الخدمات التي تملك موارد حسّاسة لسياق القاطن نطاقَ المساحة مجددًا عند حدودها، وخدمة التغليف أوضح مثال على ذلك: فكل عملية تشترط الترويسة صراحةً (`400` عند غيابها)، ويُرفض استطلاع جلسة استيراد تخص مساحة أخرى أو العبث بها بـ `403`، استنادًا إلى فحص صريح للملكية. فالعملية طويلة الأمد تخصّ القاطن، ومقبضها ليس متاحًا للجميع.

الخدمات ترفض العمل دون مساحة `Since 1.10`

أُلغي الرجوع إلى المساحة الجذر الموصوف أعلاه (`SpaceIdProvider`):

- **ترتيب الحلّ**: ترويسة `x-space-id` الوحيدة غير الفارغة، ثم نطاق المساحة المحيط (استدعاء داخل العملية، أو مهمّة خلفية تعود إلى المساحة التي تحملها حمولتها). وإن لم يوجد أيّ منهما، يُردّ على الاستدعاء بـ `400` `SPACE_CONTEXT_MISSING` بدلًا من العمل في المساحة الجذر. وتُعدّ الترويسة المرسلة مرتين غائبة.
- تُحدَّد المساحة **نطاقًا للطلب كله**، فترثها الاستدعاءات داخل العملية. وإذا حدّدت الترويسة مساحةً تختلف عن المساحة الموجودة أصلًا في النطاق، أو كانت غير صالحة ضمن نطاق قائم، يُردّ على الطلب بـ `400 SPACE_NOT_RESOLVED`. وتعمل مسارات البنية التحتية (`/health`، `/ready`، `/live`، `/ping`، `/metrics`، `/explorer`، `/openapi.json`) في المساحة الجذر.
- تحمل الاستدعاءات الصادرة بين الخدمات `x-space-id` من النطاق.
- **قراءات التعريفات بنطاق المساحة**: تعيد قوائم تعريفات الكيانات والمعارض التعريفاتِ `Public` إضافةً إلى تعريفات `Private` / `Shared` التي تُدرج المساحة الحالية؛ ولم تعد تعيد التعريفات الخاصة بمساحة أخرى.

ولا يغيّر أيّ من ذلك ما يفعله العميل عبر البوّابة – فالمضيف لا يزال يحدّد المساحة. وإنما يهمّ هذا التغيير البرنامجَ النصي الذي يستدعي خدمةً مباشرةً على شبكة العنقود، إذ يتلقّى الآن `400` بدلًا من القراءة من المساحة الجذر بصمت.

في نشرٍ متعدّد القاطنين `Since 1.10`

حين يخدم نشرٌ واحد عدة قاطنين، يحدّد المضيف **القاطن** أيضًا. ويُردّ على الطلب الوارد إلى مضيف لا يقابل أي قاطن نشط بـ `404` (`APP_ERROR` "Unknown host" عند البوّابة، و `TENANT_NOT_RESOLVED` من الخدمة). ولكل قاطن مُصدِر OIDC وJWKS وعملاء خاصّون به، فتحقّق من الرموز مقابل مُصدِر القاطن الذي تستدعيه – انظر مزوّدو OpenID لكل قاطن. ويبدأ عميل المتصفح تهيئته من نقطة النهاية العامة `GET /api/identity/v1/tenant-config`، ويحصل على الرموز المجهولة من `POST /api/identity/v1/anonymous/token` (التفاصيل). وقد تفرض البوّابة أيضًا حصة طلبات لكل قاطن (`429 RATE_LIMITED` مع `Retry-After`) – انظر استدعاء واجهة HTTP. وقد يوضع القاطن **قيد الصيانة** – أثناء استعادة بياناته من نسخة احتياطية، أو أثناء احتجاز المشغّل له لنقله إلى خلية أخرى – فتجيب كل خدمة عن طلباته بـ `503 TENANT_IN_MAINTENANCE` مع `Retry-After` (`Since 1.10`؛ الرابط ذاته). ويبقى القاطن قابلًا للتوجيه وتبقى بيانات اعتمادك صالحة: انتظر ثم أعد المحاولة.

الطلبات بنطاق المعرض وكائن الوصول

تمرّ الطلبات تحت `/api/v1/v/{viewId}/…` بفحص إضافي: إذ تحمّل إضافة `osos-access` **كائن الوصول** الخاص بالمعرض وتفرض بالترتيب ما يلي:

1. **النطاق** – يجب أن ينتمي المعرض إلى النطاق المحدّد للطلب (وإلا فالاستجابة `403`؛ وعند غياب معرّف النطاق تكون `400 MISSING_DOMAIN_ID`).
2. **المساحة** – ما لم يكن المعرض موسومًا بأنه عامّ لكل المساحات، يجب أن ترد قيمة `x-space-id` الخاصة بالطلب في قائمة المساحات المضبوطة للمعرض (`403`).
3. **خطة التنفيذ** – يجب أن يطابق عنوانُ الطلب وفعلُه الفعليّان قائمةَ السماح بالمسارات لدى المعرض؛ والمعرض بلا خطة يقبل القراءات ويرفض الكتابات.
4. **نمط المصادقة** – يحدّد كائن الوصول مَن يُسمح له بالدخول: المجهولون فقط (`0`)، أو أي مستخدم مُصادَق (`1`)، أو الجميع (`2`)، أو المصرَّح لهم صراحةً فقط (`3`)، ويُفحص ذلك بالرجوع إلى ملف أذونات المُستدعي المخزَّن مؤقتًا.

عندها فقط تُعاد كتابة المسار، ويُضبط معرّف المعرض في `x-view-id`، ويُمرَّر الطلب إلى الخدمة المالكة. وبدءًا من `Since 1.10` تزيل الإضافة أولًا أي ترويسة `x-view-id` أرسلها العميل، فلا يُحدَّد المعرض إلا عبر المسار (التفاصيل). ولهذا كثيرًا ما تعني الاستجابة `403` على مسارٍ بنطاق معرضٍ «هذا المعرض لا يجيز ذلك» لا «رمزك غير صالح» – فقد تكون بيانات الاعتماد سليمة تمامًا، ومع ذلك يظلّ المعرض خارج النطاق المسموح به في هذه المساحة.

يخفّف خبآن كلفة هذا الفحص، ولكلٍّ منهما أثر في توقيت ظهور التغييرات: كائن الوصول نفسه (افتراضيًا **600 ثانية**) وملف أذونات المستخدم (**3600 ثانية**، بمفتاح لكل مستخدم – ولكل تفويض حين يكون نشطًا). لذا قد يستغرق انعكاس منح إذن أو إبطاله على الاستدعاءات بنطاق المعرض ما يصل إلى ساعة؛ قارن ذلك بخبء *المصادقة* في دورة حياة الرموز، وهو أضيق.

التفويض: التصرّف نيابةً عن غيرك علنًا

يتيح التفويض لشخص أن يتصرّف نيابةً عن آخر خلال فترة زمنية مصرّح بها، وهو السبيل المشروع الوحيد لتنفيذ طلب بهوية غير الهوية المرتبطة ببيانات اعتماده. ولا يخصّ العميلَ منه سوى ترويسة واحدة: `X-Delegation-Id`.

وتتولّى إضافة البوّابة `user-delegation` الباقي: فهي تشترط أن يكون المُستدعي مُصادَقًا، وتحمّل سجل التفويض (المخزَّن مؤقتًا، افتراضيًا **600 ثانية**)، وتتحقّق من الفترة الزمنية – وخارجها يُرفض الطلب بـ `403 DELEGATION_EXPIRED`. وخلالها **تعيد كتابة الهوية**: تصبح قيمة `x-user-id` معرّف *المفوِّض* (الشخص الذي يُتصرَّف نيابةً عنه)، وينتقل معرّف المُستدعي الأصلي إلى `x-delegatee-id`، ويُوسَم الطلب بـ `x-is-delegation: true`. فتصل الهويتان كلتاهما إلى كل خدمة لاحقة، مما يحافظ على دقة السجل التاريخي وسجل التدقيق، بينما تُحدَّد الأذونات استنادًا إلى هوية المفوِّض.

ولا يمنح التفويض في الخدمات اللاحقة حرية مطلقة؛ فاستراتيجية المصادقة تُظهر التفويض في ملف تعريف المستخدم، وتحصر خدمة البيانات معارضَ الطلبات ضمن نطاق التفويض المضبوط بإعادة كتابة مرشّحاتها، وتشترط بوّابة المعارض إذن معرضٍ *صريحًا* لمعظم المعارض ما دام التفويض نشطًا. فالتفويض أداةٌ بين شخص وآخر – أما في تكاملات الأنظمة الآلية، فمثّل الهوية تمثيلًا سليمًا بمستخدم تطبيق مخصّص (اختيار بيانات الاعتماد).

ما لا يستطيع الرمز فعله

تتكامل الآليات السابقة لتفرض حدودًا صارمة تسري على *أي* بيانات اعتماد – رمز OAuth أو رمز تطبيق – لأن الحافة والخدمات هي التي تفرضها، لا محتوى الرمز:

- **لا يستطيع اختيار قاطن.** يُشتق النطاق والمساحة من مضيف الطلب؛ وتُستبدل قيمة `x-space-id` المقدَّمة من العميل، ولا يمكن تزوير ترويسات الهوية عبر البوّابة – إذ تزيل الحافة `x-shared-key` وتعيد إصدارها، ولا تضبط `x-user-id` إلا استنادًا إلى بيانات اعتماد تحقّقت منها.
- **لا يستطيع دخول مساحةٍ لا ينتمي إليها مستخدمه.** تُفحص العضوية مع كل طلب؛ وكل ما يراه العميل عبر المساحات هو تلميح `available_spaces` في `403`.
- **لا يستطيع العبور إلى قاطن آخر من باب خلفي.** يخضع الوصول إلى المعارض لفحص المساحة، وتخضع مقابض العمليات طويلة الأمد (التصديرات وجلسات الاستيراد) لفحص الملكية في خدماتها.
- **لا يستطيع التصعيد عبر معرض.** يحدّد نمط المصادقة وخطة التنفيذ في كائن الوصول ما يستطيع أي مُستدعٍ فعله على المسارات بنطاق المعرض، كائنًا من كان.
- **لا يستطيع الانتحال.** لا يمكن التصرّف باسم مستخدم آخر إلا عبر تفويض سارٍ محدود المدة – يُبقي الهويتين كلتيهما في سجل التدقيق دائمًا – أو عبر إعداد داخلي للثقة في النشر لا تستطيع أي بيانات اعتماد خارجية تفعيله.

وما *يستطيعه* الرمز عبر المساحات هو أمر واحد تحديدًا: إعادة استخدامه. فبيانات الاعتماد نفسها تعمل على كل مضيف لمساحةٍ ينتمي إليها المستخدم المالك لتلك البيانات؛ إذ يتغيّر النطاق مع المضيف، وتبقى الهوية ثابتة. وينبغي للعميل الذي يخدم عدة مساحات أن يعامل المضيف بوصفه جزءًا من إعداداته، لا أن يحاول تعداد القاطنين عبر الترويسات.

قراءة البيانات

يوضّح هذا الدليل كيفية قراءة البيانات من عميل خارجي. وتشرح صفحة استدعاء واجهة HTTP الأعراف المشتركة: مخطّط عناوين URL، وغلاف الاستجابة، ومرجع نموذج الترشيح، وأعراف التقسيم إلى صفحات ومعاملات القوائم؛ أمّا هذه الصفحة فتشرح كيفية تطبيقها على نحو سليم: اختيار لهجة الترشيح المناسبة، والتحقّق من صحّة المعايير قبل اعتمادها، وجلب جزء من المُدخل بدلًا من جلبه كاملًا، والتقسيم إلى صفحات بأمان في ظلّ خادم لا يقيّد أحجام الصفحات المطلوبة. وتقرأ الأمثلة كيانًا واحدًا هو `employees` باستخدام رمز تطبيق، وتَرِد كل استجابة JSON ناجحة بالشكل `{"success": true, "result": …}` أيًّا كان إصدار العملية، باستثناء `GET` على `/v3`، الذي عاد إلى إرجاع استجابة خام `Since 1.10` (التفاصيل).

اختيار لهجة الترشيح

تدعم المنصّة لهجتَي ترشيح، ويحدّد إصدارُ العملية أيّهما تُستخدم، لذا فالاختيار محسوم سلفًا: عملية القائمة في v1 موسومة بـ `@deprecated()` في وحدة التحكّم، ويظهر ذلك بوصفه `deprecated: true` في وثيقة OpenAPI. ولا تزال هذه العملية تعمل، إذ تعتمد المنصّة وسم العمليات بالإهمال دون إيقافها فجأة، لكن لا ينبغي لعميل جديد أن يعتمد عليها.

العملية	المسار	مُدخل الترشيح
قائمة *(مهمَلة)*	`GET /api/v1/entities/{entity}/entries`	`filter` (LoopBack)
قائمة	`GET /api/v2/entities/{entity}/entries`	`search` + `filterCriteria`
قائمة مع الوسوم	`GET /api/v3/entities/{entity}/entries`	`appliedTagsFilters` + `search` + `filterCriteria`
عدّ *(v1)*	`GET /api/v1/entities/{entity}/entries/count`	`where` (LoopBack)
عدّ	`GET /api/v2/entities/{entity}/entries/count`	`search` + `whereCriteria`
مُدخل واحد	`GET /api/v1/entities/{entity}/entries/{id}`	`fields`، `excludeResolved`

القاعدة: **اكتب مسارات القراءة الجديدة على v2 أو v3 ومرّر `filterCriteria`**. واختر v3 حين تحتاج إلى معلومات الوسوم مع الصفوف، إذ تكون نتيجتها كائنًا يحتوي على `entries` إضافةً إلى مصفوفة `tags`، واختر v2 حين تحتاج إلى المصفوفة وحدها. أمّا العدّ فهو دائمًا عملية منفصلة عن القائمة.

يشمل `filterCriteria` جميع عناصر مُرشِّح LoopBack: فهو يحتفظ بـ `limit` و`offset` / `skip` و`order` و`fields` و`include` وشرط `where` الموروث، ويضيف شجرة التعبير التكرارية `criteria`. أمّا نموذج التعبير نفسه، بما يشمله من عوامل وعوامل حقول وقيم محدَّدة النوع، فموثَّق في دليل الأعراف، وتفترض هذه الصفحة معرفتك به.

```
curl -G "https://{host}/api/v2/entities/employees/entries" \
  -H "x-app-token: $TOKEN" \
  --data-urlencode 'filterCriteria={
    "limit": 100, "skip": 0, "order": ["value.hire_date DESC"],
    "criteria": {
      "operator": "And",
      "args": [
        { "field": "value.department", "operator": "Equal",
          "args": [ { "type": 0, "value": "HR" } ] }
      ]
    }
  }'
```

```
{ "success": true, "result": [ /* entries */ ] }
```

تؤدّي المعايير غير السليمة إلى استجابة بالرمز `400` ورمز خطأ ثابت من محرّك الاستعلام: فالعامل غير المعروف لدى المحرّك يُعيد `UNKNOWN_OPERATOR_ERROR`، وقيمة `fieldOperator` الخارجة عن المجموعة المدعومة تُعيد `UNSUPPORTED_FIELD_EXPRESSION`.

Since 1.8 – المعايير التي تشير إلى حقل حسّاس [بناء التطبيقات ‹ حماية حقل] لا يملك المستخدم المستدعي إذن الوصول إليه **تُرفَض رفضًا صريحًا** بالرمز `400` ورمز الخطأ `SENSITIVE_FIELD_QUERY_NOT_ALLOWED`، بدلًا من إسقاطها بصمت، وفق قرار تصميمي مقصود مسجَّل في المحرّك. ويشمل هذا الفحص المسارات الفرعية أيضًا: فالترشيح على `salary.en` أو `manager.dataEntry` يُعامَل معاملة الترشيح على `salary` أو `manager`. وقد يختلف ظهور هذا الخطأ باختلاف بيانات الاعتماد؛ فالمُرشِّح الذي يعمل لمستخدم قد يُعيد 400 لمستخدم آخر.

التحقّق من المعايير عبر المحلّل

قبل حفظ تعبير معقّد بصورة دائمة في استعلام محفوظ، أو أداة لبناء المُرشِّحات لدى العميل، أو ملف تهيئة، تحقّق منه باستخدام المحرّك نفسه الذي تستخدمه نقاط النهاية الفعلية:

```
curl -X POST "https://{host}/api/v1/queries/parser" \
  -H "content-type: application/json" -H "x-app-token: $TOKEN" \
  -d '{
    "operator": "And",
    "args": [
      { "field": "value.department", "operator": "Equal",
        "args": [ { "type": 0, "value": "HR" } ] }
    ]
  }'
```

جسم الطلب هو `FilterExpression` مجرّد، أي عقدة `criteria` وحدها، وليس كائن `filterCriteria` كاملًا. وعند النجاح، تحتوي الاستجابة على الشرط المُحلَّل في صورة سلسلة نصية داخل الغلاف؛ فتعامَل مع تلك السلسلة بوصفها قيمة مبهمة لا تُفسَّر، لأنّ ما يُعتدّ به هو الرمز `200` في مقابل `400` المصحوب بأحد رمزَي الخطأ `UNKNOWN_OPERATOR_ERROR` / `UNSUPPORTED_FIELD_EXPRESSION` نفسيهما المستخدمَين في الاستعلام الفعلي.

وهنا تنبيهان: الأول أنّ المحلّل يعمل بلا سياق كيان، فلا يتحقّق من مطابقة أسماء الحقول لحقول أي كيان؛ لذا قد لا يطابق تعبيرٌ سليم البنية شيئًا عند التشغيل. والثاني أنّه لا يُجري فحص الحقول الحسّاسة الخاص بكل مستخدم والمذكور أعلاه، إذ لا يُجرى ذلك الفحص إلا عند تنفيذ الاستعلام فعليًا؛ ولذلك قد تُرفَض شجرة المعايير وقت القراءة رغم تحليلها دون أخطاء.

تُفحَص قيم المؤشّرات بصرامة `Since 1.10`. يجب أن تشير قيمة المؤشِّر في الترشيح، مثل `@me` و`@today('-1d')`، إلى دالة مؤشِّر معروفة بإحدى الصيغ `@name` أو `@name()` أو `@name('<amount>')`؛ وأي صيغة أخرى (`@Today`، `@Me`، `@none`) تُرفَض بالرمز `400 INVALID_POINTER`. وأصبح المُرشِّح المحفوظ الذي يعتمد على تهجئة غير دقيقة يفشل بعد أن كانت تلك التهجئة مقبولة، لذا صحّحها.

الإسقاط

في **القوائم**، يُحدَّد الإسقاط ببند `fields` المعتاد داخل `filter` أو `filterCriteria`، لتضمين الخصائص أو استبعادها وفق دلالات LoopBack.

وفي **القراءة المفردة**، يقبل `GET /api/v1/entities/{entity}/entries/{id}` معاملين مخصّصين:

- `?fields=value.name,value.email` – إسقاط مفصول بفواصل. وتظلّ القراءة تمرّ عبر سلسلة الوصول الخاصة بالمعرض، وهذا مقصود: فالجلب الخام مع الإسقاط كان سيسمح لـ `?fields=` بتجاوز التقييد على مستوى الصفوف، لذلك تبقى مراحل التقييد فعّالة، بينما تُعطَّل مراحل الحلّ المكلفة (الحقول الافتراضية، وحلّ العلاقات والتصنيفات). وبذلك تكون القراءة مع الإسقاط سريعة، لكن قيم العلاقات والتصنيفات تُعاد **غير محلولة**: مراجع خام لا كائنات جاهزة للعرض.
- `?excludeResolved=true` – يُعيد القيمة كاملة، لكنه يتخطّى مرحلتي حلّ العلاقات والتصنيفات فقط. استخدمه حين تحتاج إلى المُدخل كاملًا بكلفة منخفضة وتستطيع حلّ المراجع بنفسك، أو لا تحتاج إلى حلّها أصلًا.

التقسيم إلى صفحات بأمان

لا يفرض الخادم حدًّا أقصى لحجم الصفحة. فمخطّط الترشيح لا يشترط سوى `limit >= 1`، بلا حدّ أعلى في أيٍّ من اللهجتين، والطلب الذي لا يتضمّن `limit` أصلًا يُعيد كل صف مطابق. ولا تمنعك المنصّة من قراءة المجموعة بأكملها، لذا تقع مسؤولية ضبط الطلبات على العميل:

- أرسل `limit` صريحًا في **كل** طلب قائمة، حتى الطلبات الاستكشافية. واختر حجم صفحة ثابتًا في حدود بضع مئات والتزم به.
- قسّم النتائج باستخدام `skip` (أو `offset`) بزيادات تساوي `limit`؛ والصفحة التي يقلّ عدد نتائجها عن `limit` هي الأخيرة.
- رتّب النتائج بحسب مفتاح ثابت وفريد: ألحِق معرّف المُدخل بأيّ ترتيب تقتضيه متطلبات عملك (`"order":["value.hire_date DESC", "id ASC"]`). فالتقسيم بالإزاحة ضمن مجموعة متغيّرة قد يتخطّى صفوفًا أو يكرّرها عندما لا يكون مفتاح الترتيب ثابتًا؛ ولا تتوفّر مؤشّرات تسلسل (cursors) لتجنّب ذلك.
- تعامَل مع نتيجة العدّ بوصفها قيمة استرشادية: فقد تُنشأ صفوف أو تُؤرشف بين إجراء العدّ وجلب الصفحات اللاحقة.

المُرشّحات السريعة ومُرشّحات الوسوم

تتيح الميزتان للطلب إعادة استخدام مُرشّحات عُرِّفت مرة واحدة في جهة الخادم، بدل إعادة ترميزها في كل عميل.

`appliedQuickFilters` مصفوفة من **معرّفات استعلامات محفوظة**. تُدمَج معايير كل استعلام مُشار إليه مع `filterCriteria` الخاص بك باستخدام العامل AND، وتُستخلص من الطلب قيم معاملات الاستعلام إذا كان يقبل معاملات. والمعرّف الذي لا يشير إلى استعلام ذي معايير **يُتخطّى بصمت**؛ فالخطأ المطبعي لا يضيّق النتائج ولا يؤدّي إلى خطأ، لذا أعد فحص المعرّفات التي أرسلتها إذا جاءت النتائج أكثر مما تتوقّع.

يقبل `appliedTagsFilters` (في عمليات القائمة v3 والشجرة ولوحة المهام Kanban) **معرّفات وسوم أو معرّفات مجموعات وسوم**؛ ومعرّف المجموعة يُطبِّق كل وسم مفعَّل فيها. ثم تبيّن مصفوفة `tags` في استجابة v3 المُدخلات المُعادة التي تحمل كل وسم، في صورة كائنات `{tagId, entriesId, groupId}`. وتعني مصفوفة `tags` الفارغة أنّ أيًّا من الوسوم المطبَّقة لم يطابق الصفوف المُعادة، ولا تعني أنّ الوسوم غير موجودة.

قراءة تجميعة بمعرّفها

`Since 1.10` – أصبح `GET /api/v1/collections/{id}` يتيح للمستدعي المسجَّل دخوله قراءة أربعة أصناف من التجميعات: تجميعاته **الخاصة به**؛ والتجميعات **المفتوحة** المشتركة مع كل مستخدم مسجَّل؛ والتجميعات المشتركة معه **مباشرةً**؛ والتجميعات المشتركة مع **مجموعة ينتمي إليها**. وكان نطاقه قبل ذلك أضيق، فكان العميل الذي ينشئ تجميعة ثم

يفتحها بمعرّفها يتلقّى `404` عند محاولة قراءة قائمة أنشأها للتوّ، وهو إخفاق محيّر، إذ لا خطأ في بيانات الاعتماد ولا في المعرّف.

ولم تتغيّر الأذونات. فلا تزال قاعدة الظهور تُطبَّق بوصفها شرط `where` وقت الاستعلام، وليس فحصًا بعد جلب المستند، فلا يمكن استخدام إسقاط `fields` لتسريب تجميعة لا يحقّ لك الاطلاع عليها؛ ولا يزال المستدعي المقيَّد الذي يملك إذنًا على الكيان عاجزًا عن جلب تجميعة مستخدم آخر بتخمين معرّفها. وما يخرج عن هذه الحالات الأربع لا يزال يُعيد `404` بالجسم نفسه. ولم يتغيّر أي مسار أو فعل أو معامل أو شكل استجابة؛ وإنّما تغيّرت التجميعات التي يطابقها الشرط فحسب.

قدرات كيان الوكيل `Since 1.10`

أصبح `GET /v1/entities/{id}`، أي عملية قراءة كيان مفرد وليس نقاط نهاية القوائم، يُعيد خاصية إضافية هي `proxyCapabilities?: ProxyCapabilitiesDto` على كيانات `VirtualProxy`: وهي سقف للقدرات يُشتقّ لكل طلب على حدة ويقتصر على الاستجابة (الحقول والعمليات والواجهات التي يسمح بها معرض الوكيل المرتبط بالكيان حاليًا). ولا يُقبَل هذا الحقل أبدًا عند الكتابة. ويعني غياب الخاصية «ليس كيان وكيل»، في حين يعني وجودها بقيمة `bound: false` «كيان وكيل لم تُحَلّ سلسلة قدراته»؛ لذا اقرأ الحقل لمعرفة ما يسمح به الكيان حاليًا، بدلًا من استنتاج ذلك من `type` وحده. وقد أُعيد توليد لقطة OpenAPI لخدمة `data-service` منذ ذلك الحين، وأصبحت تتضمّن هذه الخاصية على `DataEntityDto`. وللاطلاع على البنية الكاملة وقواعد الاشتقاق، راجع الكيانات الوهمية § حوكمة معرض الوكيل [بناء التطبيقات › الكيانات الوهمية].

قراءة السجل التاريخي لتشغيل تفاعل `Since 1.10`

يُعيد `GET /v1/reaction/{entity}/{id}/history` (خدمة reaction-service) صفحةً من عمليات التشغيل المسجَّلة لتفاعل واحد، باستخدام التقسيم بمفتاح تسلسلي، مع عرض الأحدث أولًا. ومعاملات الاستعلام هي `outcome` (`all` / `succeeded` / `failed` / `refused`؛ و`refused` تشغيلٌ أوقفه الحدّ الشهري لتشغيلات القاطن – راجع حدود الخطة)، و`from` / `to`، و`before` (مؤشِّر مفتاح تسلسلي)، و`limit` (الافتراضي 10، والسقف 100)؛ وتكون الاستجابة `{items, hasMore, nextBefore}`. دون عدد إجمالي، وهذا مقصود. كما أصبح `GET /v1/reaction/{entity}` يُعيد أيضًا لكل تفاعل `executedCount` / `succeededCount` / `failedSinceLastChangeCount` / `statsSince`، و`refusedCount` (ولا تُحتسب التشغيلات المرفوضة في `executedCount`). وكلا المسارين نسبيّ إلى المسار الأساسي لـ `reaction-service`، أي `/api/reaction`. وأصبحت لقطة `reaction-service` المُعاد توليدها تتضمّن كليهما – راجع مرجع DataReaction. أمّا العقد الكامل، بما في ذلك سبب احتفاظ السجل التاريخي للتشغيل عمدًا بمُدخلات كل إجراء ومخرجاته دون تنقيح، فموثّق في التفاعلات والأتمتة § أعداد التنفيذ والفشل [بناء التطبيقات › التفاعلات والأتمتة].

حين لا تكون النتيجة الفارغة ما تبدو عليه

خارج سياق المعرض، لا يتلقّى المستدعي الذي يفتقر إلى منحة سرد المُدخلات خطأ `403` من عمليتَي القائمة والعدّ، بل تُعيدان عمدًا **قائمة فارغة وعدًّا صفريًّا**. وهذا هو الإخفاق الصامت المعهود لدى العميل الذي يعمل بلا واجهة: فعند استخدام رمز تطبيق لا يملك مستخدمُ التطبيق المالك له إذن القراءة، تكون النتيجة `[]` في كل مكان دون أي خطأ.

ويمكن تشخيص الحالة بالقراءة المفردة، التي لا تُخفي الإخفاق: فقراءة مُدخل واحد معروف المعرّف دون المنحة تُطلق `ForbiddenToReadEntryException`. لذا، حين تُعاد قائمة فارغة رغم توقّع احتوائها على نتائج، اجلب مُدخلًا واحدًا معروفًا: فالرمز `403` الصريح يدلّ على أنّ المشكلة في أذونات بيانات الاعتماد، وإرجاع المُدخل بنجاح يدلّ على أنّها في مُرشِّحك. وتصميم أذونات مستخدمي التطبيقات مشروح في اختيار بيانات الاعتماد؛ ومعنى `403` على المسارات المقيّدة بمعرض مشروح في دليل الأعراف.

كتابة البيانات

يتناول هذا الدليل عمليات الكتابة التي يجريها العميل الخارجي: العمليات التي تُنشئ المُدخلات وتعدّلها، وصيغ النتائج التي تُعيدها العمليات المجمَّعة لكل صف، وكيفية وصول إخفاقات التحقّق فعليًا عبر الشبكة، وكيفية ظهور التعارضات وتصادمات القيم الفريدة، وبوّابة الكتابة الوحيدة التي لا ترتبط ببيانات اعتمادك. ترد الأعراف المشتركة – الغلاف، ونموذج الأخطاء، وتحديد رمز الحالة – في استدعاء واجهة HTTP، ويرد الفهرس الكامل لرموز الأخطاء في الأخطاء وإعادة المحاولة. يُعرض كل جسم خطأ في هذه الصفحة كما يصل: مُغلَّفًا، مع `success: false`، ورمز الخطأ المقروء آليًا في `error.code`.

سطح الكتابة

عمليات كتابة المُدخلات الأساسية:

العملية	المسار	ملاحظات
إنشاء	`POST /api/v1/entities/{entity}/entries`	الجسم من النوع `CreateDataEntryDto`: قيمة `value` (مطلوبة)، و`specificationVersion` (الافتراضي `1`)، واختياريًا `saveAsInvalid` و`systemButtonId` وحقول المسودّة. تُعيد العملية المُدخل المُنشأ.
تحديث بالقيمة الكاملة	`PATCH /api/v1/entities/{entity}/entries/{id}`	يحمل الجسم قيمة `value` الكاملة للمُدخل. تُعيد العملية المُدخل المحدَّث.
تحديث بـ JSON Patch	`PATCH /api/v2/entities/{entity}/entries/{id}`	الجسم مصفوفة عمليات JSON Patch تُطبَّق على الخادم (`fast-json-patch`). فضِّله عند تعديل عدد قليل من الحقول؛ فهو يقلّل حجم الحمولة ويحدّ نطاق الأثر معًا.
إنشاء دفعي	`POST /api/v1/entities/{entity}/entries/batch`	مُدخل رئيسي واحد مع مُدخلات مرتبطة لكل علاقة، تُنشأ في استدعاء واحد.
أرشفة	`DELETE /api/v1/entities/{entity}/entries/{id}`	حذف قابل للاسترجاع؛ يُلغى باستخدام `PATCH …/{id}/restore`.
حذف نهائي	`DELETE /api/v2/entities/{entity}/entries/{id}/permanent`	لا رجعة فيه.

يُعيد الإنشاء الدفعي نتيجةً لكل صف؛ فالنجاح أو الفشل لا يشمل بالضرورة الصفوف كلها معًا: تأتي `mainResult` بالصيغة `{id, success, error}`، ولكل علاقة مصفوفة `entries` من عناصر بالصيغة `{id, index, success, error}`. وعند الكتابة عبر معرض، يجب أن تكون كل علاقة تكتب فيها من علاقات المعرض المُهيَّأة، وإلا فشل الاستدعاء كله بـ `403` قبل إنشاء أي شيء.

العمليات المجمَّعة والنتائج لكل صف

يحدّد التحديث المجمَّع والترقية المجمَّعة المُدخلات المستهدفة بالطريقة نفسها: إمّا بمصفوفة صريحة `dataEntriesIds` أو باستخدام `selectAllModel`، وهو واصف «تحديد كل المطابق» الذي يحمل `sourceMode` (كل المُدخلات، أو مجموعة، أو علاقة)، وشجرة معايير `query` اختيارية، و`search`، و`appliedQuickFilters`. وإذا لم تقدّم أيًّا منهما، أُعيد `400`. وقبل أي تنفيذ واسع النطاق، تتيح لك العملية `POST /api/v1/entities/{entity}/entries/versions/count` معاينة توزيع المُدخلات المحدَّدة على إصدارات مخطّط الكيان، في صورة أزواج `{version, count}`.

التحديث المجمَّع يطبّق قائمة `updateDefinitions`، ويتّخذ كل عنصر فيها الصيغة `{field, type, value}`، حيث يحدّد `type` مصدر القيمة: `0` قيمة ثابتة، `1` تعبير، `2` القيمة الافتراضية للحقل، `3` قيمة فارغة (`BulkUpdateFieldValueSourceEnum`). وله صيغتان:

- `PATCH /api/v1/entities/{entity}/bulk-update` تعمل تزامنيًّا وتُعيد حصيلة معالجة الصفوف مباشرةً:

```
{
  "success": true,
  "result": {
    "successfulUpdates": 8,
    "savedAsInvalid": 1,
    "failures": [
      { "entryId": "3f2c…", "errors": ["ENTRY_VALIDATION_ERROR"] }
    ]
  }
}
```

النجاح الجزئي هو الحالة المعتادة؛ لذا استعرض عناصر `failures` وعالج كل صف على حدة. وتحمل مصفوفة `errors` رموز الأخطاء في صورة سلاسل نصية (أو رسائل خامًا للإخفاقات غير المُنمَّطة)، وهي الرموز نفسها التي تُعيدها عملية الصف المفرد، لكن دون `details`. وعند الحاجة إلى تفاصيل صفٍّ فاشل، أعِد تنفيذ العملية على ذلك الصف وحده عبر نقطة نهاية المُدخل المفرد.

- `PATCH /api/v2/entities/{entity}/bulk-update` تعمل لاتزامنيًّا: تُدرج العمل في الطابور وتُعيد معرّف رسالة داخل الغلاف (`{"success": true, "result": "<messageId>"}`). استعلم دوريًّا باستخدام `GET /api/v1/job/{messageId}` لمعرفة النتيجة؛ وهذا النمط موصوف في العمل طويل الأمد.

الترقية المجمَّعة تعيد كتابة المُدخلات وفق إصدار أحدث من مخطّط الكيان، باستخدام محدِّد المُدخلات نفسه وصيغة الاستجابة نفسها لكل صف (`successfulUpgrades` + `failures`). صيغة `v1` مهمَلة، و`v2` تزامنية، أمّا `PATCH /api/v3/entities/{entity}/bulk-upgrade` فهي الصيغة اللاتزامنية التي تُعيد معرّف رسالة. وثمّة حالة تستدعي الانتباه عند احتساب النتائج في مسار v1 التزامني: الصف المُحدَّد الذي لا تطابق قيمة `entityVersion` المخزَّنة فيه قيمة `entryVersion` في الطلب **يُتخطّى بصمت**، فلا يظهر في `successfulUpgrades` ولا في `failures`؛ لذلك قد لا يساوي مجموع الأعداد عدد الصفوف التي حدّدتها.

يسمح `saveInvalidEntries` (و`saveAsInvalid`) في عمليات الكتابة المفردة) بتخزين الصف رغم إخفاقه في التحقّق من القيمة، مع وسمه بأنه غير صالح، بدلًا من إفشال العملية؛ وتُحسَب هذه الصفوف في `savedAsInvalid`. وفي الاستدعاءات المرتبطة بمعرض، لا يُعتدّ بهذا الخيار إلا إذا كانت تهيئة التحرير في المعرض تسمح بحفظ المُدخلات غير الصالحة؛ وإلا عُطِّلت الراية بصمت.

إخفاقات التحقّق وتفاصيلها

قد ترفض الكتابةَ طبقتان من التحقّق، لكل منهما رمز خطأ وصيغة مختلفة لـ `details`.

تحقّق جسم الطلب – فحص مخطّط كائن النقل نفسه – يفشل بـ `422` وبرمز الخطأ `VALIDATION_FAILED`. وترد تفاصيله في كائن لكل مخالفة بالصيغة `{path, code, message, info}`؛ فمخالفة القيم المعدودة مثلًا تُعيد `code: "enum"`، والمسار المخالف `path`، و`info.allowedValues` (انظر `InvalidEnumValue` الذي يحاكي هذه الصيغة بعينها).

تحقّق قيمة المُدخل – تحقّق المنصّة من `value` وفق حقول الكيان – يفشل بـ `400` . ويحمل كل إخفاق مفرد الرمز `ENTRY_VALIDATION_ERROR` ، وتأتي تفاصيله بالصيغة `{fieldName, reason}` ، حيث `reason` جملة مقروءة مثل `"Field is required!"` . وعندما تكون *جميع* إخفاقات الكتابة ناتجة عن التحقّق من الحقول، تجمعها آلية الكتابة تحت رمز واحد هو `ALL_ENTRY_VALIDATION_ERROR` ، وتكون تفاصيله **مصفوفة** من كائنات تفاصيل الحقول تلك؛ أمّا إذا اختلطت أنواع الإخفاقات، فلا يُعاد إلا الخطأ الأول. يصل الإخفاق المُجمَّع هكذا:

```
// HTTP 400
{
  "success": false,
  "result": {
    "errorCode": "ALL_ENTRY_VALIDATION_ERROR",
    "errorMessage": "Entry validation errors",
    "service": "data-service",
    "details": [
      { "fieldName": "email", "reason": "Field is required!" },
      { "fieldName": "skills", "reason": "the value of skills should be an array" }
    ],
    "retryable": false
  },
  "error": {
    "code": "ALL_ENTRY_VALIDATION_ERROR",
    "message": { "en": "Entry validation errors", "ar": "أخطاء في تحقق من المدخل" },
    "traceId": "…"
  }
}
```

لاحظ موضع التفاصيل: لا يحمل `error` نفسه سوى `code` و `message` و `traceId` . أمّا تفاصيل الحقول `details` فترد ضمن `result` ، الذي يحمل عند الفشل استثناء المنصّة بعد تسلسله، أي إن له قيمة فعلية وليست `null` . ويعود هذا الموضع إلى مصنع أجسام الاستجابة الذي يصفه الرماز المصدري نفسه بأنه ينتظر إعادة الهيكلة؛ لذا ابنِ منطق التفرّع على `error.code` ورمز حالة HTTP، واقرأ `result.details` بحذر: فقد تحمل الأخطاء المُمرَّرة من خدمة أخرى تفاصيل متبقية مختلفة.

Since 1.8 – تُرفَض بـ `400` وبرمز الخطأ `SENSITIVE_FIELD_WRITE_NOT_ALLOWED` أي عملية كتابة من شأنها إضافة صفوف إلى كتلة تكرارية تحوي حقلًا حسّاسًا [بناء التطبيقات › حماية حقل] لا يستطيع المستدعي قراءته، أو حذف صفوفها أو إعادة ترتيبها؛ إذ تُعاد مطابقة القيم الحسّاسة المخزَّنة بالصفوف بحسب مواضعها فقط، فلا يمكن التوفيق بينها بأمان إذا تغيّر عدد الصفوف.

صفوف الكتلة التكرارية. تُكتب صفوف الكتلة التكرارية في مصفوفة واحدة تحلّ محلّ المصفوفة المخزَّنة، ويحمل كل صف معرّف `__id` يتحكّم فيه الخادم (`Since 1.10`). أعِد إرسال المعرّفات التي قرأتها؛ إذ يفشل إرسال صفّين بالمعرّف `__id` نفسه في مصفوفة واحدة بـ `400 ENTRY_VALIDATION_ERROR` (والسبب `duplicate __id in <field>: <ids>`). وعند الكتابة عبر **معرض** لا يتيح إلا بعض أعمدة الكتلة، تُستعاد القيم المخزَّنة للأعمدة الأخرى لكل صف (`Since 1.4`)، لكن ابتداءً من الإصدار 1.10.1 لا يحدث ذلك إلا للصفوف التي يطابق فيها `__id` معرّف صف مخزَّن؛ أمّا الصف المرسل بلا معرّف فيُعدّ صفًّا جديدًا ولا يُستعاد له شيء (`Since 1.10`). وترد المواصفات الكاملة في الكتلة: حفظ الصفوف عبر معرض [بناء التطبيقات › حقل الكتلة].

تعارضات التحديث

`Since 1.10` – **صارت المُدخلات تحمل رمز تزامن متفائل خاصًّا بها.** كانت نسخة سابقة من هذه الصفحة تقول إنّها لا تحمله، ولم يعد ذلك صحيحًا. والعميل المبني على الإرشاد السابق يفقد تحديثات كان بوسعه رصدها.

و `updateIdentifier` رمزٌ يحفظه الخادم مع كل مُدخل و**يُدوِّره مع كل عملية كتابة**. لا تُنشئ هذا الرمز ولا تستطيع ضبطه؛ فهو يصلك مع المُدخل في كل قراءة، ويستبدله الخادم كلّما تغيّر المُدخل.

والفحص اختياري لكل طلب على حدة. لا يوجد مفتاح على مستوى القاطن، ولا إعداد عام، ولا راية ميزة تُفعَّل؛ بل تختار تفعيله لكل عملية كتابة تريد حمايتها، بإعادة إرسال الرمز الذي قرأته مع النسخة التي تعدّلها. وإذا أغفلته، جرت الكتابة كما كانت تمامًا: يُعتمد آخر تحديث مكتوب.

وثمّة ثلاث طرق فقط لإرساله، ولا يمكن إرساله عبر ترويسة، سواء باستخدام `If-Match` أو ETag:

العملية	أين ينتقل الرمز
`PATCH /api/v1/entities/{entity}/entries/{id}`	الخاصّية `updateIdentifier` في الجسم إلى جانب `value`
`PATCH /api/v2/entities/{entity}/entries/{id}`	**معامل استعلام** `updateIdentifier`؛ لأن جسم هذه العملية مصفوفة JSON Patch ولا يمكنه حمل الرمز
`PUT /api/v1/views/{id}/entries/{entryId}`	الخاصّية `updateIdentifier` في الجسم

تتألّف الآلية إذًا من أربع خطوات تنطبق على العمليات الثلاث جميعًا:

1. **اقرأ المُدخل** بأي استدعاء تُجريه أصلًا؛ فالرمز يأتي في الاستجابة دون طلب إضافي.
2. **احتفظ بالرمز** مع النسخة التي تحرّرها، لا مع حمولة النموذج التي توشك أن ترسلها؛ فهو يصف النسخة التي قرأتَها.
3. **أعِد إرساله مع تغييرك**، في الموضع الذي يحدّده الجدول أعلاه للعملية التي تستخدمها.
4. **وعند `409` ، أعِد القراءة، وأعِد التطبيق، وأعِد الإرسال.** فالاستجابة تخبرك بأن طرفًا آخر عدّل المُدخل بعد قراءتك له. أعِد القراءة لتحصل على النسخة الحالية *وعلى* الرمز الحالي، ثم أعِد تطبيق التغيير المقصود عليها وأرسِل من جديد. ولا تُعِد إرسال الطلب نفسه دون تعديل أبدًا؛ فهو يحمل الرمز المتقادم ويُرفض للسبب نفسه في كل مرة.

أمّا المُدخل الذي **لا رمز مخزَّنًا له بعد فلا يُرفض** لهذا السبب أبدًا: تُتخطّى المقارنة، ويُثبَت الرمز في عملية الكتابة تلك، ويصبح المُدخل مشمولًا بالحماية من عملية الكتابة التالية فصاعدًا. ولا يتطلّب ذلك أي ترحيل أو أي استكشاف مسبق من عميلك.

⚠ ثغرتان في التغطية – معروفتان ولم تُغلقا بعد

تمثّل الحالتان التاليتان ثغرتين، ولم تنتجا عن قرارين تصميميّين، ولا يستطيع عميل خارجي الالتفاف على أيٍّ منهما:

- تطبيق الويب المرفق بالمنصّة **لا** يرسل الرمز على مسار التحرير المباشر باستخدام JSON Patch الإصدار `v2` ، مع أنّ معامل الاستعلام متاح على ذلك المسار. لذا لا تشمل هذه الحماية عمليات التحرير المباشر في واجهة المنصّة نفسها.
- تصل ثلاثة مسارات كتابة داخلية في `data-service` إلى تجميعة المُدخلات عبر برنامج تشغيل MongoDB مباشرةً، دون المرور بالمستودع الذي يُثبِت الرمز؛ لذا فإن هذه الكتابات **لا** تُدوِّره. ويحتفظ المُدخل الذي لم يُعدَّل إلا عبر أحد هذه المسارات برمز لم يعد يصف محتواه.

لذا ابنِ منطق عميلك على أنّ الحماية تغطّي العمليات الثلاث أعلاه فقط.

أمّا كائنات التعريف، فتعمل بهذه الآلية منذ زمن أطول. فالكيانات والمعارض والاستعلامات والعلاقات والتجميعات تحمل كلها `updateIdentifier` تقرؤه مع الكائن ويجب أن تُعيده كما هو؛ فإن لم يعد مطابقًا للقيمة المخزَّنة – لأن طرفًا آخر حفظ الكائن

في الأثناء – فشلت الكتابة بـ `409` وبرمز الخطأ `UPDATE_CONFLICT` (`UpdateIdentifierConflictException`). وتُعيد تعارضات المُدخلات الجسم نفسه:

```
// HTTP 409
{
  "success": false,
  "result": {
    "errorCode": "UPDATE_CONFLICT",
    "errorMessage": "Update conflict updateIdentifier value not match with the one in the database",
    "service": "data-service",
    "retryable": false
  },
  "error": {
    "code": "UPDATE_CONFLICT",
    "message": { "en": "UPDATE_CONFLICT", "ar": "" },
    "traceId": "…"
  }
}
```

وطريقة التعافي واحدة دائمًا: أعِد قراءة الكائن، وأعِد تطبيق التغيير على النسخة الحديثة، ثم أعِد الإرسال بقيمة `updateIdentifier` الجديدة. ولا تُعِد إرسال الطلب المتقادم؛ فسيفشل بالنتيجة نفسها في كل مرة. وقد تفشل كتابة كائن تعريف أيضًا بخطأ يفيد بأن الكائن مقفل، عندما يحتفظ مستخدم آخر بقفل تحرير عليه؛ ويزول هذا الخطأ عند تحرير القفل.

ويستطيع العميل الذي يحتفظ أيضًا باتصال socket معرفة التغيير قبل محاولة الحفظ: فقد صار الحدث `entries/value-updated` يحمل قيمة `updateIdentifier` الحالية للمُدخل، مما يتيح للصفحة مقارنتها بالرمز الذي تحتفظ به وتنبيه المستخدم بدلًا من انتظار `409` . وهذا إشعار فوري لا يُضمَن وصوله، ولا يُغني عن الفحص؛ راجع الزمن الحقيقي.

تصادمات القيم الفريدة

عندما تنتهك عملية كتابة قيدَ التفرّد لأحد الحقول، تُصدر طبقة التخزين خطأ مفتاح مكرّر من MongoDB (الرمز `11000`). وتوحّد طبقة البيانات معالجة هذا الخطأ في مسارات الإنشاء والتحديث والترقية لدى الكاتب، مع خط دفاع أخير في مزوّد الرفض المشترك، فتحوّله إلى استجابة `400` تحمل رمز الخطأ `UNIQUE_VALUE` ، ويصف فيها `details.field` التصادم:

```
// HTTP 400
{
  "success": false,
  "result": {
    "errorCode": "UNIQUE_VALUE",
    "errorMessage": "email is unique",
    "service": "",
    "details": {
      "field": {
        "path": "email",
        "originalPath": "value.email",
        "value": "jane@example.com"
      }
    },
    "retryable": false
  },
  "error": {
    "code": "UNIQUE_VALUE",
    "message": { "en": "UNIQUE_VALUE", "ar": "" },
    "traceId": "…"
  }
}
```

يشير `path` إلى الحقل كما تعرفه أنت؛ إذ تُنزَع بادئة التخزين من مفتاح الفهرس، فيصبح `value.email` هو `email`، بينما يحتفظ `originalPath` بالمفتاح الخام، وتمثّل `value` القيمة المتصادمة (`UniqueValueException`). تعامَل مع `UNIQUE_VALUE` بوصفه خطأً نهائيًّا لتلك الحمولة: فإعادة المحاولة دون تغيير القيمة تؤدّي إلى الإخفاق نفسه. وفي العمليات المجمَّعة، يقتصر الإخفاق الناتج عن التصادم على الصفّ المعنيّ، ولا تفشل الدفعة كلها.

بوّابة التحقّق البشري على الكتابات مجهولة الهوية

Since 1.8 – تشترط عملية إنشاء المُدخل وعمليّتا التحديث كلتاهما اجتياز التحقّق البشري قبل الكتابة. ونطاق البوّابة أضيق مما يوحي به اسمها؛ إذ لا تعترض معظم العملاء الخارجيين أبدًا، ويجب لتطبيقها أن تجتمع الشروط الثلاثة:

- الطلب **مرتبط بمعرض**، ويستهدف معرض بيانات تتضمّن تهيئة الوصول المجهول فيه `humanVerification.enabled: true`. **و**
- المستدعي **مجهول الهوية** – لا يرتبط بالطلب مستخدم مصادَق عليه.

لا يُطلب أبدًا من المستخدم المسجَّل دخوله، أو رمز التطبيق، أو أي عميل مصادَق عليه عبر OAuth حلّ التحدّي؛ فالبوّابة مخصّصة للحالة المتبقية وحدها: تشغيل **نموذج عام مجهول الهوية** برمجيًّا. وفي هذه الحالة، يجب على العميل حلّ تحدّي إثبات العمل من Altcha: اجلبه من `GET /api/human/challenge`، واحسب الحلّ، ثم أرسله في ترويسة `x-altcha-token` مع طلب الكتابة. تتحقّق بوّابة الـ API من الحلّ وتُضيف `x-human-verification: true` إلى الطلب المُمرَّر إلى الخدمات الخلفية، بعد أن تنزع أولًا أي نسخة من تلك الترويسة قدّمها العميل منعًا لتزويرها. وتفشل الكتابة مجهولة الهوية دون حلّ صالح باستجابة `403` والرسالة `Human verification required to perform this action`. أمّا تهيئة الميزة فهي من مهام الإدارة؛ انظر التحقق البشري [الإدارة › التحقق البشري (الحماية من الروبوتات)].

حالات رفض الكتابة المضافة في 1.10 `Since 1.10`

قد يواجه العميل الذي يكتب البيانات خمس حالات رفض جديدة عليه في سلسلة إصدارات 1.10. ولكلٍّ منها رمز ثابت، ولا تُجدي إعادة المحاولة دون تغيير إلا في حالتَي `503` :

الحالة	`error.code`	متى	ما العمل
`400`	`MISSING_REQUIRED_PARAMETER`	أُرسل طلب إنشاء مُدخل أو مسودّة (`POST /api/v1/entities/{entity}/entries` ، و`…/entries/draft` ، وصيغتا `/api/v1/views/{id}/entries`) **بلا جسم إطلاقًا**. أصبح جسم طلب الإنشاء إلزاميًّا؛ أمّا الطلب الذي يتضمّن جسمًا فيُتحقَّق منه كما كان.	أرسل `CreateDataEntryDto` .
`403`	`ENTRY_CAP_EXCEEDED`	إنشاء مُدخل (مفردًا أو ضمن دفعة أو مسودّةً أو عبر وكيل) حين يبلغ القاطن حدّ خطته دون السماح له بتجاوزه. أمّا التحديث والأرشفة والقراءة فلا تُرفض أبدًا. `details` : `{cap, hardLimit, current}`.	توقّف عن الإنشاء؛ وعلى مسؤول القاطن ترقية الخطة أو أرشفة مُدخلات.
`400`	`FILE_EXPIRED`	تشير عملية الحفظ إلى ملف مؤقّت مرفوع انتهت صلاحيته قبل أن تُرقّيه عملية الحفظ. ويسرد `details` القيم `{fileId, fieldName?, path?, reason}` لكل ملف.	ارفع الملف من جديد وأعد الحفظ بالمعرّف الجديد.
`503`	`STORAGE_SERVICE_UNAVAILABLE`	يتعذّر الوصول إلى التخزين حين تضيف عملية الحفظ ملفًّا غير موجود أصلًا في المُدخل. أمّا الحفظ الذي لا يضيف ملفًّا جديدًا فيستمرّ.	أعِد المحاولة مع زيادة الفاصل الزمني بين المحاولات.
`503`	`TENANT_IN_MAINTENANCE`	في النشر متعدّد القاطنين: تجري استعادة بيانات القاطن أو نقله. وتتضمّن الاستجابة `Retry-After` .	انتظر `Retry-After` ثانية، ثم أعد المحاولة.

وأصبح تنفيذ إجراء سير عمل مخفيّ على المُدخل يُرفض باستجابة `422 WORKFLOW_ACTION_EXECUTION_NOT_ALLOWED` ، كما أصبح تولّي ملكية خطوة لها مالك أصلًا يُرفض باستجابة `422 CURRENT_STEP_ALREADY_OWNED` ؛ وكانت الاستجابة في كلتا الحالتين `500` . وترد القائمة الكاملة في حالات الرفض المُنمَّطة وحدود الخطة.

التعامل مع الملفات

الملفات هي الجزء الوحيد من مستوى البيانات الذي لا يتبع أعراف المنصّة المعتادة؛ فهي تتبع خدمة `storage-service`، وهي تطبيق Express بسيط لا يتّبع أعراف LoopBack، ولا تتوفّر له وثيقة OpenAPI مولَّدة. ويختلف غلاف استجابته عن بقية الواجهة البرمجية في جوانب ينبغي أن يراعيها رماز معالجة الأخطاء لديك. وتغطي هذه الصفحة ما يفعله العميل الخارجي بالملفات فعليًا: رفعها، وإرفاقها بالمُدخلات، وتنزيلها، وفهم دورة حياتها.

أما تفاصيل الخدمة الداخلية – مزوّدو التخزين، والتحقّق من الرفع، ونموذج البيانات الوصفية، وتوجيه الملفات المؤقتة – فتتناولها صفحة تخزين الملفات والمرفقات. وتقتصر هذه الصفحة على جانب العميل في الاتصال، مع الإحالة إلى تلك الصفحة للتعمّق.

سطح التخزين

يعتمد العميل على عمليتين اثنتين فقط، إذ يعطّل الموجّه عمدًا كل ما عداهما من واجهة الملفات التقليدية؛ فمسارات السرد والجلب بالمعرّف والحذف والتحديث معلَّقة في الرماز لأنها لم تكن مستخدمة ولم تتضمّن أي تحقّق من التفويض:

الطريقة	المسار	الغرض
`POST`	`/api/storage/v2/files`	رفع ملف واحد أو أكثر
`GET`	`/api/storage/v2/files/download/{id}`	تنزيل بايتات ملف بمعرّفه

تمرّ كلتاهما عبر البوابة ببيانات اعتمادك المعتادة – أي `x-app-token` نفسه أو الرمز الحامل نفسه المستخدم في أي استدعاء آخر. ولن تتعامل أبدًا مع ترويسات الهوية الداخلية للخدمة؛ فالبوابة تحذف أي `x-shared-key` يرسله العميل وتحقن مفتاحها الخاص (`request-transformer` في `api-gateway/config/kong.yml`). وتُصنَّف الطلبات المُمرَّرة عبر البوابة على أنها *خارجية*، لذا تخضع الملفات التي ترفعها للتحقّق الكامل.

وللخدمة مسارات أخرى لا يُوجَّه أيّ منها إلى العميل؛ فمسارا الروابط الموقّعة (أدناه) يخدمان المتصفحات. أما `GET /api/storage/v2/exports/{id}/content` (`Since 1.10`) فلا يتيح تنزيل تصدير القاطن إلا عبر رابط صالح لمرّة واحدة تصدره المنصّة، ولا تنشئه أنت أبدًا. وأمّا المسارات الداخلية فلا توجّه البوابة الطلبات إليها، ولا تقبل إلا خدمات محدّدة بالاسم من خدمات المنصّة. التفاصيل في تخزين الملفات والمرفقات.

ولأن هذه الخدمة لا تتوفّر لها وثيقة OpenAPI، فلا يغطيها أي عميل مولَّد (راجع توليد عميل). لذا اكتب رماز التكامل مع نقطتي النهاية هاتين يدويًا؛ فالواجهة صغيرة، ولا يتطلّب ذلك إلا بضعة أسطر.

رفع ملف

أرسل طلبًا متعدد الأجزاء (multipart) يكون اسم حقل النموذج فيه `file`. ويقبل المسار عدة أجزاء تحت هذا الاسم الواحد (`multer.array('file')`)، مما يتيح إرسال عدة ملفات في طلب واحد:

```
curl -s -X POST "$BASE/api/storage/v2/files" \
  -H "x-app-token: $TOKEN" \
  -F "file=@invoice.pdf"
```

تُعاد عند النجاح الحالة `201` مع الغلاف، وتكون `result` دائمًا **مصفوفة** تحتوي على واصفة لكل جزء مرفوع، حتى لو أرسلت ملفًا واحدًا:

```
{
  "success": true,
  "result": [
    { "id": "5f0e9b2a-…-uuid", "name": "invoice.pdf", "mimeType": "application/pdf", "size": 20481 }
  ]
}
```

المعرّف `id` هو المقبض الوحيد الذي ستحصل عليه، فاحفظه؛ إذ يشير إليه حقل المُدخل، وتستقبله نقطة نهاية التنزيل. أما تفاصيل التخزين الداخلية (موضع حفظ البايتات فعليًا) فلا تُكشف أبدًا.

ويهمّ العميلَ معاملا استعلام:

- `temp=true` يخزّن الملف **مؤقتًا**، ويُحذف تلقائيًا بعد يوم واحد (فهرس TTL على `createdAt`) ما لم تُرقِّه عملية حفظ قبل ذلك. و`Since 1.10` أصبح هذا الأسلوب موصى به لرفع ملف توشك على إرفاقه بمُدخل (راجع رفع مؤقت ثم حفظ)؛ وهو مناسب أيضًا لتجهيز الملفات المرحلية، مثل حزمة ZIP توشك على استيرادها.
- `ownerType` و`ownerId`، حين يُرسلان **معًا**، يسجّلان كائن المنصّة الذي يتبعه الملف، ويضعان الكائن المخزَّن تحت مالكه. استخدمهما حين يكون الملف مُعدًّا للإرفاق بمُدخل.

تخضع الملفات المرفوعة من الخارج للتحقّق باستخدام قائمة بالامتدادات المسموحة، وفحص المحتوى عبر البايتات السحرية للأنواع الثنائية، وفحص ملفات PDF بحثًا عن JavaScript مضمَّن. وتُرفض الملفات بالحالة `400` مع إحدى القيم `INVALID_FILE_TYPE` أو `INVALID_FILE_CONTENT` أو `PDF_JAVASCRIPT_NOT_ALLOWED` في `error.code`. وتجد القواعد كاملة، بما فيها الامتدادات التي تُفحص وتلك التي يُكتفى فيها بالقائمة المسموحة، في تخزين الملفات والمرفقات. ولاحظ أن الخدمة نفسها **لا تضبط حدًا أقصى لحجم الملف الواحد**؛ فأي سقف لحجم الطلب تفرضه الوكلاء (proxies) التي تسبق النشر. أمّا ما تفرضه الخدمة `Since 1.10` فهو **حدّ تخزين** القاطن؛ إذ يُرفض أي رفع يؤدّي إلى تجاوز القاطن هذا الحدّ بالاستجابة `403` `STORAGE_CAP_EXCEEDED` (راجع الأخطاء عند حفظ الملفات).

إرفاق ملف بمُدخل

حقل المرفقات في الكيان حقلٌ نوع بياناته `blob` (`isBlobField`). وقيمته ليست بايتات، بل الواصفة التي أعادتها عملية الرفع. والصيغة القياسية هي `FileModel` المشترك:

```
{
  "id": "5f0e9b2a-…",        // required — the storage handle
  "name": "invoice.pdf",
  "mimeType": "application/pdf",
  "size": 20481,
  "thumbnail": { /* optional, same shape */ }
}
```

إذًا، يكون تسلسل العمل في العميل الخارجي: الرفع أولًا ثم الكتابة. ارفع الملف (مع `ownerId` / `ownerType` مشيرَين إلى الكائن المالك إن كنت تعرفه)، وخذ `result[0]` من الاستجابة، واجعله – أو مصفوفةً من هذه الواصفات في حقول الملفات المتعددة – قيمةً للحقل في عملية إنشاء أو تحديث عادية للمُدخل. أما آليات الكتابة فتتناولها صفحة كتابة البيانات.

وعند القراءة، قد تتضمن استجابة تفاصيل المُدخل كتلة `attachments`؛ إذ يمرّ الخادم على جميع حقول `blob` في الكيان، ويجمع الواصفات من قيم المُدخل (حتى داخل الكتل القابلة للتكرار)، ثم يجمّعها بحسب الحقل (`getEntryAttachments`). لكن وجود الكتلة يتوقّف على إعداد *إظهار المعلومات* في المعرض، فلا تعدّ غيابها دليلًا على «عدم وجود ملفات»؛ فقيم الحقول نفسها هي المرجع الموثوق.

رفع مؤقت ثم حفظ `Since 1.10`

`Since 1.10` يحتفظ التخزين بسجلّ تتبّع يبيّن الملفات التي يشير إليها كل مُدخل محفوظ. فعندما تتضمّن عملية حفظ مُدخل معرّفَ ملف، تسجّله data-service قبل الكتابة؛ وإن كان المعرّف لملف مرفوع رفعًا **مؤقتًا**، **يرقّيه** التخزين إلى التخزين الدائم بالمعرّف نفسه. لذا يكون النمط الموثوق للعميل هو ما تعتمده بوابة الويب الآن:

1. ارفع مع `?temp=true` .
2. احفظ المُدخل بالواصفة المُعادة خلال اليوم.
3. يرقّي الحفظُ الملفَ، فيصبح منذئذٍ دائمًا ويشير إليه ذلك المُدخل.

ويترتّب على ذلك لعميل الواجهة البرمجية ما يلي:

- **كل معرّف ملف جديد في الحفظ يجب أن يكون موجودًا.** إذ تُفحص الواصفات في حقول `blob` ، أي الكائنات التي تتضمّن `id` و `name` و `mimeType` و `size` ذا قيمة رقمية. وإذا لم يجد التخزين معرّف الواصفة – بسبب رفع مؤقت مضى عليه أكثر من يوم، أو معرّف لم يوجد قطّ – يفشل الحفظ بالاستجابة `400 FILE_EXPIRED` مع ذكر اسم الحقل. ولا يُعاد فحص المعرّفات التي كان المُدخل المخزَّن يحملها أصلًا، فلا تفشل إعادة حفظ مُدخل قديم بسببها أبدًا.
- **قد يتوقّف الحفظ على التخزين.** فإن تعذّر الوصول إلى التخزين أثناء عملية حفظ تضيف ملفًا جديدًا، يفشل الحفظ بالاستجابة `503 STORAGE_SERVICE_UNAVAILABLE` القابلة لإعادة المحاولة، بدلًا من كتابة مرجع لا تتتبّعه أي جهة.
- **الرفع الدائم لا يزال يعمل** كما كان تمامًا، غير أن سجلّ التتبّع لا يشمله، فلا يُنظَّف تلقائيًا أبدًا.

التنزيل

التنزيلات هي المثال النموذجي على استجابة **تتجاوز الغلاف بنيويًا**؛ إذ يكتب المعالج مباشرةً في استجابة HTTP – نوع المحتوى، ثم ترويسة التصرّف (disposition)، ثم البايتات – فلا يوجد غلاف `{success, result}` ولا محتوى يُحلَّل بوصفه JSON. ولذلك سيفشل هنا تحديدًا أي عميل يمرّر كل استجابة إلى محلّل JSON.

```
curl -s "$BASE/api/storage/v2/files/download/5f0e9b2a-…" \
  -H "x-app-token: $TOKEN" \
  -o invoice.pdf
```

ما تُعيده نقطة النهاية (`downloadFile`):

- تُشتق `Content-Type` من **امتداد اسم** الملف المخزَّن (باستخدام `mime-types` للبحث، مع اعتماد `application/octet-stream` قيمةً احتياطية)، لا من `mimeType` المسجَّل عند الرفع.
- تُضبط `Content-Disposition: attachment` باسم الملف الأصلي دائمًا، فلا يُقدَّم أي شيء أبدًا بوصفه محتوى مضمّنًا قابلًا للتنفيذ.
- الجسم هو البايتات الخام.

وتخدم نقطة النهاية نفسها الملفات المؤقتة؛ إذ يفحص المعالج المجموعة الدائمة أولًا ثم المؤقتة، ولذلك يمكن تنزيل الملف المرفوع باستخدام `temp=true` بمعرّفه وحده كأي ملف آخر. ويُعاد `404` للمعرّف غير الموجود، وكذلك `Since 1.6` للمعرّف الذي توجد بياناته الوصفية لكن بايتاته المخزّنة مفقودة (وكان يُعاد له سابقًا `500` غير معالَج).

روابط موقّعة للمتصفحات `Since 1.6` **.** إذا احتاج المتصفح إلى عرض ملف عبر `<img src>` عادي (دون رمز حامل)، فاطلب من التخزين عنوانًا موقّعًا قصير العمر عبر `POST /api/storage/v2/files/links` ، واستخدم `url` المُعاد. والرابط **صلاحية**

لحامله: إذ يستطيع كل من يحمل الرابط جلب الملف إلى أن تنتهي صلاحيته (من ساعة إلى ساعتين افتراضيًا). أبقِ هذه الروابط خارج السجلات والعناوين القابلة للمشاركة. التفاصيل: روابط محتوى موقّعة قابلة للتخزين.

وثمة اختلاف ينبغي الانتباه إليه: **مسار النجاح وحده ثنائي.** أما التنزيل الفاشل – بسبب معرّف مجهول أو خلل في الخلفية – فيظل جسم استجابته جسم خطأ بصيغة JSON مع الحالة المناسبة. لذا افحص رمز الحالة (أو `Content-Type`) قبل أن تقرّر كيفية قراءة الجسم.

دورة الحياة: ما الذي يوجد، وإلى متى

تنشئ الواجهة البرمجية العامة الملفات وتقرؤها، لكنها لا تحذفها أبدًا. أمّا التنظيف، في الحالات التي يحدث فيها أصلًا، فهو مهمّة المنصّة:

- **المرفوعات الدائمة تبقى إلى أجل غير مسمّى.** فلا توجد نقطة نهاية للحذف في الواجهة العامة، ولذلك يبقى الملف المرفوع مباشرةً إلى التخزين الدائم بعد زوال أي مُدخل أشار إليه؛ كما أن إزالة مرفق من مُدخل تعدّل قيمة المُدخل ولا تعدّل الملف المخزَّن. وتُحدَّث `lastAccessedAt` عند كل تنزيل لملف دائم، مما يوفّر للمشغّلين مؤشّرًا يعتمدون عليه في التنظيف خارج الخط؛ لكن التنظيف مهمة تشغيلية، وليس استدعاءً للواجهة البرمجية.
- **الملفات المُرقّاة قد تُكنس** `Since 1.10` **.** فالملف الذي رقّته عملية حفظ من التخزين المؤقت يصبح مُتتبَّعًا. وإذا تخلّت عنه كل المُدخلات التي أشارت إليه (كأن يُحذف المُدخل نهائيًا) وانقضت مهلة السماح، يحذفه كانسٌ يعمل في الخلفية، لكن ليس قبل أن يؤكّد كل مالك سابق أنه لم يعد يُدرج الملف. ويحتفظ المُدخل في تاريخه بكل ملف حمله يومًا، فلا تُكنس ملفاته ما دام المُدخل موجودًا. ويأتي مخطط Helm الخاص بالمنصّة مع تفعيل الكانس في وضع تجريبي يكتفي بالتسجيل، إلى أن يأذن له مشغّل بالحذف؛ راجع كانس الملفات اليتيمة.
- **المرفوعات المؤقتة تُحذف تلقائيًا بعد يوم واحد** عبر فهرس TTL. وهكذا تتعامل المنصّة مع ملفاتها العابرة أيضًا: فحزمة الحلول المصدَّرة، مثلًا، تُرفع بوصفها ملفًا مؤقتًا، لذا يبقى المعرّف `fileId` الذي تعيده مهمة التصدير صالحًا نحو يوم واحد، فبادر إلى تنزيله (راجع العمل طويل الأمد). أمّا ملف ZIP لإصدار حزمة منشور فيُخزَّن تخزينًا دائمًا، وحذف الإصدار أو التصدير يزيل سجلّه فقط، فيبقى ملف ZIP في التخزين.
- `createdBy` **يُسجَّل من هويتك المصادَق عليها** في المرفوعات الدائمة، وتربط البيانات الوصفية `ownerId` / `ownerType` الملفَ بالكائن المالك له إن كنت قد أرسلتها.

الأخطاء من خدمة التخزين

تشترك أخطاء التخزين مع المنصّة في الهيكل العام `{success, result, error}`، لكن **ليس** في شكله الدقيق: فخدمات Express تبني الغلاف يدويًا، وهذا الاختلاف مقصود، وقد أُبقي عليه حفاظًا على توافق البيانات المتبادلة عبر الشبكة:

```
// storage-service failure body (middleware.error.ts)
{
  "success": false,
  "result": {
    "code": "NOT_FOUND",
    "message": "File is not available!",
    "serviceName": "storage",
    "traceId": "…",
    "details": { "error": "File not found." }
  },
  "error": {
    "code": "NOT_FOUND",
    "message": "File is not available!"   // a plain string — NOT {en, ar}
  }
}
```

وثمة ثلاثة فروق مقارنةً بجسم خطأ LoopBack الذي يتعامل معه معالجك أصلًا، والموضّح في نداء واجهة HTTP البرمجية:

1. `error.message` **سلسلة نصية بلغة واحدة**، وليست كائن `{en, ar}` مترجمًا.
2. `error` لا يحمل **أي** `traceId`؛ فمعرّف التتبّع هنا موجود في `result.traceId`.
3. `result` كائن تشخيصي ذو محتوى عند الفشل، وليس `null`.

كذلك تصدر الاستجابة `401` الناتجة عن فحص المصادقة الخاص بالخدمة قبل تشغيل وسيط الأخطاء، وهي **غير مغلّفة إطلاقًا**: إذ يُرسل كائن الخطأ الخام كما هو. لذا خصّص فرعًا صغيرًا مستقلًا لمعالجة أخطاء التخزين: استند في المطابقة إلى حالة HTTP و`error.code`، واقرأ معرّف التتبّع من موضعه أيًّا كان. أما سياسة إعادة المحاولة نفسها فهي المتّبعة في سائر المواضع؛ راجع الأخطاء وإعادة المحاولة.

الأخطاء عند حفظ الملفات `Since 1.10`

تخصّ الملفات ثلاثة رموز. انتبه إلى الخدمة التي تُعيد الاستجابة، فهي التي تحدّد شكل الجسم: إما شكل أخطاء التخزين الخاص الموصوف أعلاه، وإما الشكل المعتاد للمنصّة الموضّح في الأخطاء وإعادة المحاولة:

الحالة / الرمز	يُطلقه	المعنى	ما الذي تفعله
`403` `STORAGE_CAP_EXCEEDED`	storage-service، عند الرفع	بلغ القاطن حدّ تخزينه (`maxStorageGb` في خطته). تأتي الاستجابة بصيغة أخطاء التخزين، ويحمل `result.details` القيمتين `capBytes` و`usedBytes`. والملف المرفوع الذي يتجاوز الحدّ أثناء البثّ يُزال ويُرفض بالطريقة نفسها.	غير قابل لإعادة المحاولة؛ يجب رفع حدّ القاطن قبل أن ينجح الرفع مجددًا.
`400` `FILE_EXPIRED`	data-service (reaction-serviceg)، عند الحفظ	تعذّر العثور على معرّف ملف جديد في القيمة المحفوظة. تأتي الاستجابة بصيغة أخطاء المنصّة مع رسالة مترجمة، ويسرد `result.details` القيمة `{fileId, fieldName}` (أو `{fileId, path}`) لكل ملف مفقود، ولكلٍّ منها `reason` مترجم.	ارفع الملف من جديد وأعد الحفظ بالواصفة الجديدة.
`503` `STORAGE_SERVICE_UNAVAILABLE`	data-service (reaction-serviceg)، عند الحفظ	تعذّر الوصول إلى التخزين أثناء عملية حفظ تضيف ملفًا جديدًا.	قابل لإعادة المحاولة: انتظر بفواصل زمنية متزايدة، ثم كرّر الحفظ نفسه.

الاشتراك في الأحداث

الاستطلاع أداة خاطئة لسؤال «أخبرني حين يتغيّر شيء». قناة الدفع في المنصّة هي **socket العام** – خدمة `web-socket-service`، وهي خادم Socket.IO يحوّل أحداث المجال إلى عمليات دفع محصورة بالغرف. صفحة الزمن الحقيقي تملك قصة تلك الخدمة كاملة: فهرس الأحداث، وجسر RabbitMQ خلفها، والتوسّع، وتكامل المتصفح. أما هذه الصفحة فتغطي ما لا تغطيه تلك: الاتصال من **عميل غير متصفح** – عامل خلفي، أو عفريت نظام، أو خادمك أنت – حيث يبدو سؤال بيان الاعتماد وانضباط المطابقة مختلفَين.

قاعدة تأطيرية قبل كل شيء: القناة **للدفع فقط وبأفضل جهد**. تشترك وتتلقّى؛ ولا تكتب عبرها أبدًا، ولا تعاملها يومًا سجلًا كاملًا. كل كتابة، وكل قراءة موثوقة، نداء HTTP.

القناة والبروتوكول

تتحدث نقطة النهاية **Socket.IO v4** (`socket.io ^4.8.1`)، وهو بروتوكول قائم بذاته فوق WebSocket/الاستطلاع الطويل – فعميل WebSocket مجرد لا يستطيع الاتصال. من عملية غير متصفحية، استخدم عميل Socket.IO متوافقًا مع v4؛ وتوجد تنفيذات رسمية ومجتمعية لـ JavaScript وPython وJava وGo وNET. وغيرها، وكل ما في هذه الصفحة محايد تجاه مكتبة العميل: حمولة مصادقة عند المصافحة، وأحداث مسماة في الاتجاهين.

الخدمة عمليةُ Socket.IO + Express مستقلة تصغي افتراضيًا على المنفذ `3017`. وهي منشورة ومكشوفة **بمعزل عن مسار البوابة** `/api` – فجدول مسارات البوابة (`api-gateway/config/kong.yml`) لا يحمل أي مسار socket – لذا فمضيف socket خاصية من خصائص نشرك. احصل عليه ممن يشغّل بيئتك، إلى جانب عنوانك الأساسي. أما CORS على الخادم فهو `origin: '*'`، وهو غير ذي صلة بعميل غير متصفحي على أي حال.

مصادقة المصافحة

تجري المصادقة مرة واحدة، في وسيط المصافحة، ويوضع بيان الاعتماد في حمولة `auth` الخاصة بـ Socket.IO تحت المفتاح `token` (الدالة `setupSocketAuthMiddleware`):

```
import { io } from "socket.io-client";

const socket = io("https://<socket-host>", {
  auth: { token: accessToken },
});
```

وهنا الفخ للعميل بلا واجهة: **يجب أن يكون هذا رمز وصول OAuth 2.0، لا رمز API.** فالخادم يتحقّق من البيان بإرساله – مع `CLIENT_ID` / `CLIENT_SECRET` الخاصّين به – إلى نقطة نهاية introspection لدى مزوّد الهوية ويشترط جوابًا بـ `active: true`. أما رمز API فبيانُ اعتماد مختلف يتحقّق منه طرف آخر (ملحق `app-token-auth` في البوابة)؛ وهو لا يظهر رمزَ OAuth نشطًا في introspection، فلا يستطيع فتح socket. فالخدمة التي تصادق نداءاتها HTTP بـ `x-app-token` تحتاج إذًا بيان اعتماد ثانيًا لأجل socket: رمز OAuth **لمستخدم**، وهو للخدمة منحة كلمة المرور لحساب خدمة مخصّص. ويجب أن يكون هذا الحساب مستخدمًا عاديًا (`NORMAL_USER`): فمستخدم `APPLICATION_USER` لا يُحفظ إلا رمزَ API، ولا سجلّ دخول له، وتجيب منحة كلمة المرور عنه بـ `invalid_grant`. ولا يغني عنه رمز `client_credentials`: فهو يظهر نشطًا في introspection فيُفتح socket، لكنه بلا `sub`، فيبقى `socket.userId` غير معيَّن، ويجيب `subscribeToNotifications` بـ `error {message: 'User ID is required'}` ثم يقطع الاتصال. وتغطّي صفحة اختيار بيان الاعتماد المفاضلة.

عند النجاح يربط الخادم هويتك: تُضبط `socket.userId` من `sub` الخاص بالرمز، وهذه الهوية – لا أي شيء ترسله لاحقًا – تقرر أي غرفة مستخدم يحق لك الانضمام إليها. والرمز غير الصالح أو غير النشط يرفض الاتصال بـ `Invalid token`، ويصل عميلك بوصفه `connect_error`.

نتيجتان لدورة الحياة من مبدأ «تحقّق مرة واحدة عند المصافحة»:

- **كل إعادة اتصال مصافحةٌ جديدة.** فالعميل المعمّر الذي انتهت صلاحية رمزه في الأثناء سيفشل في إعادة الاتصال. جدّد الرمز قبل المحاولة – مع `socket.io-client`، عيّن `socket.auth = { token: freshToken }` داخل معالج `reconnect_attempt` (أو `connect_error`).

الغرف: اشترك قبل أن يصلك شيء

الـ socket المتصل حديثًا لا يتلقى شيئًا. التسليم محصور بـ**الغرف**، وأنت تنضم طوعًا بإصدار حدث اشتراك؛ ويُجاب كل اشتراك بحدث إشعارِ استلامٍ يؤكّد الانضمام (`setupSocketHandlers`):

الغرفة	الانضمام بإصدار	الحمولة	إشعار الاستلام
`user:{userId}` – إشعاراتك	`subscribeToNotifications`	*(لا شيء – الهوية من المصافحة)*	`subscribedToNotifications`
`entries/{entityId}_{entryId}` – مُدخل واحد	`entries/subscribe-to-entry`	`{entityId, entryId}`	`subscribedToEntry`
`job:{messageId}` – مهمة غير متزامنة واحدة	`subscribeToJobExecution`	`{messageId}`	`subscribedToJobExecution`

توجد أحداث `unsubscribe` مقابلة لغرفتي المُدخل والمهمة؛ ونقص حقول الحمولة ينتج حدث `error` بدل الانضمام. **انتظر إشعار الاستلام قبل افتراض أنك في الغرفة** – فبعض إخفاقات الانضمام تُبتلع على جانب الخادم ولا تنتج أي `error` أصلًا، وهو تنبيه توثّقه صفحة الزمن الحقيقي. أما ما يصل في كل غرفة – تحديثات قيم المُدخلات، وأحداث سير العمل والتعليقات، والإشعارات، واكتمالات المهام – فهو فهرس الأحداث الذي تملكه تلك الصفحة؛ فلا تُرسّخ افتراضات تتجاوز أسماء الأحداث التي تستهلكها فعلًا.

وثمّة أمران يصلان **دون** انضمام إلى غرفة، وعلى العميل العامل من دون واجهة أن يتوقّعهما كليهما. فبعض الأحداث تُبثّ إلى كل socket للقاطن – إشعارا مجموعة الأيقونات ومدخلات الطلبات، و`packages/changed` `Since 1.10` حين تتغيّر حالة مراجعة الحزمة أو حالتها في المتجر (إشارةٌ لإعادة القراءة لا تحمل إلا `packageId`). كما أنّ الإشعارات **غير مُرشَّحة بحسب المساحة** على الـ socket: فغرفة المستخدم الخاصّة بك تتلقّى الإشعارات الصادرة في كل مساحة أنت عضو فيها. و`Since 1.10` يخفي عميل الويب المشحون ما يأتي من المساحات الأخرى بنفسه، لكنّ ذلك الترشيح على جانب العميل فقط – فرشّح على `value.space` في مدخل الإشعار إن كان عميلك يعمل في مساحة واحدة (الزمن الحقيقي).

متابعة مهمة بدأتها عبر HTTP

الاشتراك الأنفع لعميل بلا واجهة هو غرفة المهمة، لأنه يستبدل حلقة الاستطلاع. النمط من طرف إلى طرف:

```
import { io } from "socket.io-client";

const socket = io(SOCKET_HOST, { auth: { token: oauthToken } });

socket.on("connect", () => {
  // messageId came back from an async HTTP operation, e.g. a bulk update
  socket.emit("subscribeToJobExecution", { messageId });
});

socket.on("subscribedToJobExecution", ({ messageId }) => {
  console.log("watching job", messageId);
});

socket.on("osos/infrastructure/job-executed", (evt) => {
  // evt: { messageId, userId, messages, result: { success, error?, result? } }
  if (evt.messageId !== messageId) return;
  finish(evt.result);
});
```

يُبثّ حدث الاكتمال إلى غرفة المهمة **وغرفة مستخدمك معًا** (الدالة `sendJobExecutionFinished`)، وحمولته تحمل كائن نتيجة المهمة. عامله نداءَ إيقاظ لا سجلًا: أكّد الحالة النهائية عبر HTTP بنقطة نهاية حالة المهمة، فهي المصدر الموثوق – وتقسيم العمل هذا مشروح في العمل طويل الأمد.

اشترك **قبل** إطلاق العمل أو بُعيده مباشرة، لا حين يضجرك الانتظار: فلو انتهت المهمة قبل اكتمال انضمامك، يكون بثّ الغرفة قد جرى ولن يُعاد شيء.

ضمانات التسليم والمطابقة

صمّم عميلك حول ما يعِد به النقل فعلًا، وهو قليل:

- **أفضل جهد، لا إعادة بث.** تُبثّ الأحداث لمن في الغرفة لحظتها. بثّ غرفة المُدخل يُتخطى كليًا حين لا مشتركين في الغرفة (`sendDataToEntry` تعود مبكرًا)، ولا يُصفّ لك شيء وأنت منقطع.
- **إعادة الاتصال فجوة.** بعد أي انقطاع افترض أنك فوّتَّ أحداثًا: أعد الاشتراك في غرفك (الانضمامات لا تنجو من اتصال جديد) وطابق الحالة بقراءة HTTP – أعد جلب المُدخل، أعد استطلاع حالة المهمة.
- **معالجة بمرة واحدة على الأقل.** قد يصلك اكتمال مهمة مرتين – عبر غرفة المهمة ثم غرفة مستخدمك. اربط معالجتك بـ `messageId` واجعلها عديمة الأثر عند التكرار.
- **الترتيب ليس تعاقديًا.** عامل كل حدث على أنه «تغيّر شيء، تحقّق عبر HTTP» لا فرقًا تطبّقه على عماها.

هذا هو الموقف نفسه الذي تتخذه واجهة المنصّة: socket يجعل الأمر سريعًا؛ وHTTP يجعله صحيحًا.

ما ليست هذه القناة إياه

- **ليست مسار كتابة.** لا حدث من عميل إلى خادم يعدّل البيانات؛ وأي شيء يشبه أمرًا عبر socket تتجاهله هذه الخدمة. الكتابة عبر واجهة HTTP البرمجية.
- **ليست قناة بين الخدمات.** `web-socket-service` هي الـ socket الوحيدة في المنصة. وكانت الإصدارات الأقدم تشغّل أيضًا مرحّلًا داخليًا بين خدمات الواجهة الخلفية (`socket-service`)؛ وقد أُزيل في 1.10 `Removed 1.10`، وصارت خدمات الواجهة الخلفية تتابع المهام عبر ناقل الرسائل. وقد تظل الصفحات المكتوبة للإصدار 1.8 تصفه.

- **ليست صفّ رسائل.** إن احتجت أحداث تكامل معمّرة قابلة لإعادة القراءة، فاستطلع سطح HTTP أو ابنِ على الويب هوك والتكاملات – ولا تحاول أن تجعل من socket واحدًا منها.

العمل طويل الأمد

بعض العمليات أثقل من أن تُنجَز ضمن الطلب نفسه: كتابات جماعية على آلاف المُدخلات، وتصدير حزم الحلول واستيرادها، وإعادة بناء كاملة لمخطط الاعتماديات، وتصدير ملف في الخلفية. وفي جميع هذه الحالات، تستجيب المنصّة فورًا بـ**مقبض**، فتتحوّل مهمة عميلك من «قراءة الاستجابة» إلى «متابعة المقبض حتى يبلغ حالة نهائية». وتغطي هذه الصفحة عائلات المقابض الأربع التي يتعامل معها العميل الخارجي، ونقاط نهاية استطلاعها وحالاتها النهائية، وما يمكن إلغاؤه وما لا يمكن إلغاؤه، والاستجابات التي تَرِد في صورة **تنزيلات ملفات لا JSON**.

تشرح صفحة نداء واجهة HTTP البرمجية النمط العام – إطلاق، فاستطلاع، فحالة نهائية. أما هذه الصفحة فهي الدليل العملي المبني عليه، وتعرض العقود الدقيقة بعد التحقّق منها في الخدمات التي تنفّذها.

النمط

تتبع كل عائلة الحلقة نفسها، ولا تختلف العائلات إلا في المقبض ومفردات الحالة:

1. **أطلق.** تعيد العملية مقبضًا – `messageId` أو `sessionId` / `jobId` – بدلًا من النتيجة.
2. **استطلع** نقطة نهاية الحالة الخاصة بالعائلة باستخدام ذلك المقبض، على فترات معتدلة تزيدها تدريجيًا.
3. **توقّف عند حالة نهائية**، ثم اقرأ النتيجة من حمولة الحالة – وقد تكون قيمة، أو تقريرًا لكل صف، أو `fileId` تستخدمه لتنزيل الملف.

وتسري قاعدتان على العائلات كلها:

- **المقابض مقيّدة بالقاطن.** يعيد استطلاع مهمة تخص مساحة أخرى `403`؛ إذ تتحقّق نقاط نهاية الحالة من الملكية قبل الاستجابة.
- **المقابض تنقضي.** تحتفظ كل عائلة بالحالة مدة محدودة (مفضّلة أدناه). ابدأ الاستطلاع فورًا وخزّن النتائج التي تعنيك – فالمقبض ليس أرشيفًا.

وللاستعاضة عن الخطوة 2 ببديل قائم على الدفع، راجع التقدّم دون استطلاع.

مهام مستوى البيانات: مقبض `messageId`

ترسل العمليات الثقيلة في `data-service` أمرًا عبر ناقل الرسائل، وتعيد معرّف UUID جديدًا – هو `messageId` – في نتيجة مغلّفة. وفيما يلي أكثر هذه العمليات شيوعًا لدى العملاء الخارجيين:

العملية	المسار
تحديث جماعي	`PATCH /api/v1/entities/{entity}/bulk-update`
ترقية جماعية	`PATCH /api/v3/entities/{entity}/bulk-upgrade`
استعادة مُدخلات	`PATCH /api/v1/entities/{entity}/restore/entries`
تصدير قالب في الخلفية	`GET /api/v1/entities/{entity}/entries/export/{templateId}` (للقوالب من نوع الخلفية)
تنفيذ زر إجراء مخصّص `Since 1.8`	`POST /api/v1/system-buttons/{id}`

استطلع نقطة نهاية الحالة المشتركة الوحيدة:

```
curl -s "$BASE/api/v1/job/$MESSAGE_ID" -H "x-app-token: $TOKEN"
```

```
{
  "success": true,
  "result": {
    "messageId": "1c9a…",
    "status": "InProgress",            // Scheduled | InProgress | Success | Failed
    "message": { "en": "…", "ar": "…" },
    "progress": 40,                     // 0–100
    "result": null                      // populated on completion, e.g. a per-row report
  }
}
```

العقد (`JobStatusResponseDto`):

- تتدرّج قيمة `status` وفق التسلسل `Scheduled → InProgress → Success | Failed` (`JobStatusEnum`). والحالتان `Success` و `Failed` نهائيتان.
- عند الاكتمال، يحمل `result` تقرير العملية – وهو، في التحديث الجماعي، كائن DTO الخاص بالإجماليات والإخفاقات الذي تشرحه كتابة البيانات: أما النجاح الجزئي فيأخذ الحالة `Success` مع تقرير يسرد الإخفاقات، لذا اقرأ التقرير دائمًا.
- **تُحفظ الحالة في الخبء وتنقضي بعد 24 ساعة** (`JOBS_CACHE_EXPIRY`). ويُجاب عن المقبض المجهول أو المنقضي بـ `404` مع `error.code: JOB_STATUS_NOT_EXIST` – ولا يمكن تمييزه عن معرّف خاطئ، لذا احفظ ما تحتاجه قبل انقضاء صلاحيته.
- **لا إلغاء** لهذه العائلة: لا توجد نقطة نهاية للإلغاء في متحكّم المهام. فمتى أُرسل الأمر، استمرّ حتى نهايته.

مهام الحزم: مقابض BullMQ والإلغاء

تشغّل `packaging-service` أعمال التصدير والاستيراد على طوابير BullMQ بعمّال مخصّصين، وتعتمد مفردات الحالة الخاصة بـ BullMQ بعد توحيدها عند حدود الواجهة البرمجية.

التصدير. يضع `POST /api/packaging/v1/export` المهمة في الطابور ويعيد `{jobId}` ؛ استطلع `GET` `/api/packaging/v1/export/{jobId}/status`:

```
{
  "success": true,
  "result": {
    "status": "completed",   // waiting | waiting-children | active | prioritized |
                             // delayed | completed | failed | unknown
    "result": {              // present when completed
      "fileId": "…",
      "name": "package-….zip",
      "resolvedResources": [ /* what actually went into the package */ ]
    },
    "failedReason": { "message": { "en": "…", "ar": "…" }, "code": "…" }  // when failed
  }
}
```

الحالتان `completed` و `failed` نهائيتان. أما `unknown` فتعني أن الحالة لا تطابق أيًّا من الحالات المعرَّفة – **واصل الاستطلاع** ولا تعدّها فشلًا (`normalizeExportJobStatus`). وعند الاكتمال، نزّل الحزمة فورًا: إذ يُرفع ملف ZIP إلى التخزين بوصفه **ملفًا مؤقتًا** ويُحذف تلقائيًا بعد نحو يوم – راجع التعامل مع الملفات. و `Since 1.10` ينشئ التصدير أيضًا **سجلَّ تصدير** يمكنك سرده

وقراءته لاحقًا (`GET /api/packaging/v1/exports` ، `GET /api/packaging/v1/exports/{id}`)؛ وحذفه (`DELETE /exports/{id}/…`) يزيل السجلّ فقط، ولا يزيل ملف ZIP من التخزين (سجلّات التصدير).

الاستيراد. يعتمد الاستيراد على الجلسات: ارفع حزمة ZIP إلى التخزين، ثم تحقّق من صحتها لإنشاء جلسة (`POST /api/packaging/v1/import/validate` مع `fileId`)، ثم نفّذ الاستيراد – يعيد `POST /api/packaging/v1/import/{sessionId}/execute` القيمة `{jobId, sessionId}` ، ويكون هدف الاستطلاع هو **الجلسة**: `GET /api/packaging/v1/import/{sessionId}/status` ، وتعيد نقطة النهاية `{status, importResult, errorMessage, resourceMappings}`. ويمكن إعادة تشغيل الجلسة الفاشلة أو الفاشلة جزئيًا باستخدام `POST …/{sessionId}/retry` . أما نموذج الجلسة وإعداداتها فتتناولها صفحات الحزم – راجع الحزم.

الإلغاء متاح هنا، بدلالات واضحة (`POST /api/packaging/v1/import/{sessionId}/cancel`):

- لا تُلغى إلا جلسة حالتها الحالية `importing` أو `validated` ؛ وما عدا ذلك يعيد `400` .
- تُنقل الجلسة ذريًا إلى `failed` مع العلامة `Cancelled:` ؛ وأي تغيير متزامن للحالة يعيد `409` .
- تبيّن الاستجابة ما حدث فعلًا للمهمة الأساسية: `removed_from_queue` (لم تبدأ قط)، أو `cancel_requested` (كانت قيد التنفيذ – لا يُوقَف العامل الجاري قسرًا؛ بل يتوقّف عند فحصه الدوري التالي لحالة الجلسة)، أو `not_found` (انتهت بالفعل أو أُزيلت أثناء التنظيف).

لذا تعامل مع الإلغاء على أنه *مطلوب لا فوري*: بعد `cancel_requested` ، واصل استطلاع الجلسة حتى تستقرّ حالتها.

إعادة بناء الاعتماديات

إعادة البناء الكاملة لمخطط الاعتماديات هي أبسط أفراد العائلة، ولها سلوك مميّز واحد – **تفرّد التشغيل**:

- يعيد `POST /api/dependency/v1/rebuild` الرمز `202` مع `{jobId, status: "queued"}` .
- وإن كانت إعادة بناء جارية بالفعل، فستحصل على `409` – وتحدّد رسالة الخطأ `jobId` الجاري. والتصرّف الصحيح هو أن **تستطلع تلك المهمة** بدلًا من تكرار الإطلاق.
- يعيد `GET /api/dependency/v1/rebuild/status/{jobId}` القيمة `{jobId, state, result?, failedReason?}` ؛ ويُجاب عن المعرّف المجهول أو المنقضي بـ `404` .

النسخ الاحتياطي للقاطن واستعادته وتصدير بياناته `Since 1.10`

في النشر متعدّد القاطنين، يستطيع مسؤول النظام لدى القاطن نفسه أخذ نسخة احتياطية من بيانات القاطن واستعادتها وتصديرها عبر `data-service` . وهذه **واجهة API فقط** – لا شاشة ويب لها في 1.10 – وفي النشر وحيد القاطن، يعيد كل مسار `501 BACKUP_NOT_SUPPORTED_IN_SINGLE_MODE` . أمّا التفاصيل الخاصة بالمشغّل (المحرّك ومدة الاحتفاظ والحصص اليومية) فترد في صفحة النسخ الاحتياطي والاستعادة ضمن قسم الإعداد؛ ويرد عقد HTTP الكامل ورموز الرفض في النسخ الاحتياطي للقاطن واستعادته وتصدير بياناته.

الإطلاق	الإجابة	الاستطلاع
`POST /api/v1/admin/backups` (بلا جسم)	`202` مع `backupId`	`GET /api/v1/admin/backups/{backupId}`
`POST /api/v1/admin/restores` (`{backupId}` أو `{pointInTime}` ، مع `confirmationToken`)	`202` مع `restoreId`	`GET /api/v1/admin/restores/{restoreId}`
`POST /api/v1/admin/exports`	`202` مع `exportId`	`GET /api/v1/admin/exports/{exportId}`

ما الذي يختلف في هذه العائلة:

- **الاستعادة خطوتان.** أولًا، أرسل `POST /api/v1/admin/restores/preview` بالهدف ذاته: فتوضّح الاستجابة ما ستفعله الاستعادة، مع التحذيرات، وتتضمّن `confirmationToken` لاستخدام واحد تنتهي صلاحيته. ثم ابدأ الاستعادة بذلك الرمز. والرمز غير الصالح أو المنتهي أو المُستخدَم يعيد `400 CONFIRMATION_INVALID` — اطلب معاينة جديدة.
- **الاستعادة تضع القاطن قيد الصيانة.** منذ بدئها حتى انتهائها، يُجاب عن كل طلب للقاطن بـ `503 TENANT_IN_MAINTENANCE` مع `Retry-After` — **عدا** `GET` على `/v1/admin/restores…` و`/v1/admin/backups…` و`/v1/admin/exports…` . استطلع الاستعادة عبر هذه المسارات؛ ولا تتوقّع أن يعمل غيرها، ولا تبدأ استعادة ثانية (`409 RESTORE_IN_PROGRESS`).
- **يُنزَّل التصدير عبر رابط لاستخدام واحد.** حين يكتمل التصدير، يعيد `POST /api/v1/admin/exports/{exportId}/link` القيمة `{url, expiresAt}` : رابط على مضيف القاطن نفسه، يُستخدم مرّة واحدة، ولا تتجاوز مدة صلاحيته 10 دقائق. وتُتاح خمسة روابط بحدّ أقصى لكل تصدير (`409 EXPORT_LINKS_EXHAUSTED`). ويضمّ الأرشيف قواعد بيانات القاطن **دون حسابات المستخدمين**.
- **حصص لا حِمل.** تُحتسب عمليات البدء ضمن حصص يومية (`backupsPerDay` و`restoresPerDay` و`dumpDownloadsPerDay`)، ويبلّغ عنها `GET /api/v1/admin/backups` مع موعد إعادة ضبطها. ويؤدي تجاوز الحصّة إلى `429` مع `Retry-After` يمتدّ حتى إعادة الضبط.
- **لا توجد نقطة نهاية للإلغاء** في هذه العائلة.

نتائج هي تنزيلات لا JSON

تَرِد النتيجة في صورة ملف على شكلين مختلفين، وعلى العميل معالجة كليهما:

التنزيلات المباشرة. تكتب بعض العمليات الملف مباشرة في استجابة HTTP — وتُعدّ نقطة نهاية تصدير القوالب الحالة القياسية. فحين يكون القالب معدًّا للتصدير المباشر، يضبط `GET /api/v1/entities/{entity}/entries/export/{templateId}` نوع محتوى الملف وترويسة الإرفاق، ويرسل البايتات تدفّقيًا. وهذه الاستجابات **تتجاوز غلاف الاستجابة بنيويًا** — فلا تتضمّن غلاف `{success, result}` ولا أي محتوى يُحلَّل بوصفه JSON. وتعيد نقطة النهاية نفسها `messageId` ضمن استجابة JSON مغلّفة حين يكون القالب من نوع الخلفية، ويكون شكل الطلب متطابقًا في الحالتين — لذا **ابنِ قرارك على ترويسة `Content-Type` في الاستجابة**، لا على ما توقّعته: تشير `application/json` إلى غلاف؛ وأي قيمة أخرى تعني أن الاستجابة هي الملف ذاته.

مقبض ثم تنزيل. يعيد العمل الخلفي الذي ينتج ملفًا `fileId` بدلًا من الملف، وتُجلَب البايتات من نقطة نهاية تنزيل التخزين في طلب ثانٍ. ويُعدّ تصدير الحزم أعلاه إحدى هذه الحالات، أما تصدير القالب في الخلفية فحالة أخرى ذات خطوة وسيطة واحدة: فحين يجهز الملف، يُنشأ لك مُدخل إشعار تتضمّن قيمته واصفة الملف (`value.attachment`، مع `reference` يشير إلى معرّف الملف)، ويُدفع حدث `notification` إلى غرفة المستخدم الخاصّة بك على socket. وفي الحالتين، تكون الخطوة الأخيرة هي `GET /api/storage/v2/files/download/{fileId}` — وهي استجابة بايتات خام تنطبق عليها المحاذير التي تفصّلها التعامل مع الملفات، ومنها مدة بقاء الملفات المؤقتة البالغة يومًا واحدًا.

التقدّم دون استطلاع

تتيح كل مهمة ذات `messageId` الاشتراك عبر socket أيضًا: انضم إلى غرفة `job:{messageId}` لتتلقّى حدث الاكتمال بدلًا من اكتشافه في الاستطلاع التالي، مع تحديثات التقدّم حين تبلّغ عنها العملية. وترد تفاصيل المصافحة وآليات الغرف والتنبيه المتعلّق ببيانات الاعتماد (إذ يستخدم socket رمز OAuth لا رمز API) في الاشتراك في الأحداث.

استخدم القناتين وفق دور كل منهما: socket *للإيقاظ*، ونقطة نهاية الحالة *للحسم*. فالحالة التي تُقرَأ عبر HTTP هي السجل الموثوق؛ وحدث socket إشارة إلى أن الوقت مناسب لقراءتها.

أين تعمل المهام فعلًا

تساعد معرفة الآلية الداخلية على تفسير ما تلاحظه:

- **أوامر مستوى البيانات** تنتقل عبر ناقل الرسائل (RabbitMQ) وتُنفَّذ على عمّال BullMQ داخل `data-service` (التحديث والترقية والاستعادة الجماعية، وينقسم كلٌّ منها إلى مهمة أم ومهام مُجزّأة). وتُكتب معلومات الاكتمال في حالة المهمة المستندة إلى Redis التي تستطلعها، ثم تُنشر على الناقل في صورة حدث اكتمال المهمة، فيمرّره socket العام إلى المتصفح. (وحتى 1.10 كانت تمرّ عبر مرحّل socket داخلي أُزيل لاحقًا `Removed 1.10`.)
- **أعمال الحزم والاعتماديات** تعمل على طوابير BullMQ وعمّال كل خدمة.
- **النسخ الاحتياطي للقاطن واستعادته وتصدير بياناته** `Since 1.10` تتولّاها خدمة التحكّم الخاصّة بالخلية متعدّدة القاطنين؛ وتقتصر مهمة `data-service` على تحديد القاطن المستدعي وتمرير الطلب والحالة.
- **العمل المجدول والمتكرر** تتولّاه `scheduler-service`، وهي خدمة داخلية قائمة على Agenda تشغّلها أوامر الناقل حصرًا – ولا تتيح أي مسارات HTTP للعملاء إطلاقًا (مسار HTTP الوحيد فيها هو فحص الصحة). لذا لا يستطيع العميل جدولة عمل عبرها مباشرة؛ بل يُعِدّ السلوك المتكرر عبر ميزات المنصّة، كالتفاعلات والتقارير المجدولة، وهي التي تستخدمها داخليًا.

والنتيجة العملية لهذا كله: تنفيذ المهام غير متزامن وموزّع، ويجري مرة واحدة على الأقل بحكم التصميم. لذا صمّم عميلك ليستطلع بفواصل متزايدة، ويتعامل مع المقابض على أنها قصيرة الأجل، ويجعل معالجة النتيجة متساوية الأثر عند التكرار.

الأخطاء وإعادة المحاولة

يندرج كلّ إخفاق يتلقّاه عميلك من أسس ضمن أحد ثلاثة أنواع: رفض عند البوابة، أو استثناء مُنمَّط من إحدى خدمات المنصّة، أو خطأ من إحدى خدمات Express الطرفية. وتُعدّ هذه الصفحة مرجعًا للأنواع الثلاثة؛ إذ تستعرض شكل كلّ منها عند وصوله، وفهرس `error.code` الكامل مجمَّعًا بحسب صنف الاستثناء الذي يُطلقه، وسياسةً مكتوبة لإعادة المحاولة، وكيفية استخدام `traceId` عند التصعيد. أمّا الآليات المشتركة بين جميع أجزاء واجهة API – غلاف الاستجابة والتوجيه والإصدارات والتقسيم إلى صفحات – فيتناولها دليل استدعاء واجهة HTTP البرمجية ولا تُكرَّر هنا؛ في حين يعرض دليل البدء السريع معالجة خطأ واحد من البداية إلى النهاية.

تُورَد الرموز وأسماء الأصناف أدناه كما تُصدرها المنصّة تمامًا، بما فيها أخطاؤها الإملائية؛ فطابِقها حرفيًّا.

من أين تنشأ الأخطاء

قد يعبر الطلب ثلاث طبقات قادرة على إنتاج الأخطاء، ولكلٍّ منها شكل خاص لجسم الاستجابة. وأوّل ما ينبغي لمعالج الأخطاء فعله هو تحديد الطبقة التي يتعامل معها.

البوابة. تتولّى مكوّنات البوابة الإضافية المصادقة وتحديد القاطن وفحص الوصول قبل بلوغ أيّ خدمة، وتُصدر الرفض عبر مساعد مشترك يشبه غلافَ المنصّة في شكله، لكنّه يختلف عنه بعلامتين فارقتين: `error.message` نصّ عادي وليس كائنًا مُوطَّنًا، وقيمة `error.serviceName` دائمًا `"API Gateway"`:

```
{
  "success": false,
  "error": {
    "code": "SPACE_FORBIDDEN",
    "message": "User does not have access to the specified space",
    "serviceName": "API Gateway",
    "traceId": "5b2f…",
    "details": null
  }
}
```

حالات الرفض بسبب *حدّ المعدّل* عند البوابة هي الاستثناء الوحيد؛ إذ تصدر من مكوّن Kong القياسي `rate-limiting` (`api-gateway/config/kong.yml`) بحالة `429` وجسم صغير خاص بـ Kong يتضمّن `message`، ولا تتّبع هذا الشكل.

خدمات المنصّة. تمرّر كلّ خدمة LoopBack – أي مستوى البيانات بأكمله – أخطاءها عبر مزوِّد رفض واحد ومصنع واحد لجسم الاستجابة، ويكون جسم الإخفاق المُغلَّف على النحو الآتي:

```
{
  "success": false,
  "result": {
    "errorCode": "UNIQUE_VALUE",
    "errorMessage": "email is unique",
    "service": "",
    "details": {
      "field": {
        "path": "email",
        "originalPath": "value.email",
        "value": "a@example.com"
      }
    },
    "retryable": false
  },
  "error": {
    "code": "UNIQUE_VALUE",
    "message": { "en": "…", "ar": "…" },
    "traceId": "5b2f…"
  }
}
```

يضمن الرماز أربع حقائق عن هذا الشكل، وعلى العميل معرفتها:

- **يحمل `error` بالضبط `code` و`message` و`traceId`.** حدّد مسار المعالجة بحسب `code`؛ إذ تُحدَّد قيمته بالترتيب `errorCode ?? code ?? name`، ولذلك يُظهر الخطأ الذي لم يُمنَح رمز منصّة اسمَ صنفه في JavaScript بدلًا من الرمز – `UnauthorizedError` و`ForbiddenError` و`NotFoundError`، وحتى `TypeError` عند وجود علّة فعلية في الخادم.
- **`error.message` نتيجة بحث في قاموس خاص بكل خدمة.** يوطّن قاموس خدمة البيانات 262 رمزًا؛ ولأيّ رمز غير وارد فيه، يُعيد المصنع `{ "en": "<the code itself>", "ar": "" }`. لا تفترض أبدًا أنّ `message.ar` غير فارغ، ولا تحلّل `message` إطلاقًا.
- **`result` ليس `null` عند الإخفاق.** يُبقي المصنع فيه ما تبقّى من الاستثناء الداخلي: الكائنَ المُسلسَل أعلاه للاستثناءات المُنمَّطة – وهو الموضع الفعلي لحمولة `details` البنيوية، مثل حقل `field` المخالف عند انتهاك شرط فرادة القيمة أو مصفوفة المخالفات عند إخفاق التحقّق من جسم الطلب – ونصَّ الرسالة فقط للأخطاء غير المُنمَّطة. وتحمل هذه الدالة في المصنع ملاحظة `@fixme` لم تُعالَج؛ لذا عامِل `result` عند الإخفاق بوصفه سلوكًا مُلاحَظًا يفيد في التشخيص، لا عقدًا يُبنى عليه.
- **الأخطاء المُمرَّرة تعبر كما هي.** حين تمرّر خدمةٌ إخفاقًا واردًا من خدمة أخرى، يُعاد استخدام كائن `error` الوارد كما هو، وقد يحمل حقولًا إضافية (`details` و`serviceName`). تقبَّل الخصائص غير المعروفة. ومن `Since 1.8` صارت **الحالة** تتبعه أيضًا؛ فإذا وردت حالة 4xx من الخدمة الأخرى، تُعاد الحالة نفسها في الاستجابة، ولا تُحوَّل إلى `500` (`relayedClientError`). أمّا الردّ غير JSON من الخدمة الأخرى، فما زال يُفسَّر على أنّه إخفاق في النقل.

خدمات Express الطرفية. لا تعتمد خدمتا `users-service` (المصادقة) و`storage-service` على LoopBack، ولا تُغلِّفان الاستجابات. تُعيد نقاط نهاية الرموز وOIDC عند الإخفاق كائنات خطأ OAuth 2.0 القياسية (`error` و`error_description`). أمّا نقاط نهاية كشف الحقول الحسّاسة عبر التحقّق الإضافي (step-up) *(Since 1.8)* فتُعيد جسمًا مسطّحًا `{ "error": "<CODE>" }` – ورموزها مفهرَسة أدناه.

كيف تُحدَّد حالة HTTP

يحدّد مزوِّد الرفض الحالة بترتيب ثابت – `statusCode` ثم `status` ثم `requestCode` الخاص بالاستثناء المُنمَّط، ثم `404` إن كان الرمز `ENTITY_NOT_FOUND`، وإلّا `500` – كما هو موثَّق في استدعاء واجهة HTTP البرمجية. ولا تفصّل تلك الصفحة مصدر `requestCode` وما يحدث عند غيابه، وهو أهمّ ما تتناوله هذه الصفحة لمنطق إعادة المحاولة.

تعرّف كلّ خدمة مجموعة صغيرة من عائلات الاستثناءات المجرّدة، والعائلة وحدها هي التي تضبط الحالة (لاحظ اسم الملف):

العائلة	`requestCode`	إعادة المحاولة؟
`ValidationException`	400	أبدًا – الطلب خاطئ؛ أصلحه.
`UnauthorizedException`	401	جدّد بيانات الاعتماد وأعد المحاولة مرّة واحدة.
`ForbiddenException`	403	أبدًا – قرار يتعلّق بالأذونات أو بالوصول عبر معرض.
`NotFoundException`	404	أبدًا – راجع المعرّف.
`ConflictException`	409	أعد قراءة الحالة ثم أعد المحاولة بصورة مقصودة.
`UnprocessableEntityException`	422	أبدًا – الحمولة غير صالحة دلاليًّا.
`DataException` المجرّدة	**500** (بلا قيمة)	**أبدًا – إنه حتميّ لا عابر.**

الصف الأخير هو موضع الالتباس: فأيّ صنف استثناء ملموس يرث من `DataException` مباشرةً – دون المرور بعائلة تضبط `requestCode` – يظهر بحالة HTTP `500`، وتضمّ خدمة البيانات وحدها **101** من هذه الأصناف لإخفاقات دلالية محضة: `ACTION_NOT_ALLOWED` ومعظم أخطاء إعداد سير العمل وانتهاكات تكرار الأسماء. لذا فإنّ حالة `500` من أسس **لا** تعني عطلًا عابرًا في الخادم. افحص `error.code` في الفهرس قبل اتخاذ قرار إعادة المحاولة؛ فالاستجابة بالحالة `500` التي تحمل رمزًا مفهرَسًا ستُخفق بالطريقة نفسها في كلّ محاولة. وقد انتقل عدد من حالات الرفض هذه إلى العائلة المناسبة – فتنفيذ إجراء مخفيّ في سير العمل صار `422` (`Since 1.8`)، وأرشفة كيان مؤرشف أصلًا صارت `422` وحفظ معرض بلا عملية تحرير مدعومة صار `400` (`Since 1.10`) – وأُضيفت حالات رفض جديدة إلى العائلة المناسبة منذ البداية؛ وهذه الصفوف مُعلَّمة في الجدول أدناه وملخّصة في استدعاء واجهة HTTP. وعلى الرماز الذي كان يعالج أحد هذه الرموز معالجةً خاصّة ضمن الحالة `500` أن يبحث عنه الآن ضمن حالته الجديدة.

يُجري مزوِّد الرفض عمليّتَي تطبيع قبل التسلسل:

- يتحوّل خطأ المفتاح المكرَّر في MongoDB (رمز برنامج التشغيل `11000`) إلى `UniqueValueException`: الحالة `400` والرمز `UNIQUE_VALUE`، مع مسار الحقل المخالف وقيمته تحت `result.details.field`.
- يتحوّل إخفاق المصادقة داخل خطّ المعالجة (`USER_PROFILE_NOT_FOUND`) إلى استجابة `401` عادية تكون فيها قيمة `error.code` هي `UnauthorizedError`.

ويكشف الصنف الأساس أمرًا أخيرًا: يعرّف `DataException` حقلًا باسم `retryable` يظهر فعلًا في البيانات المنقولة ضمن ما تبقّى في `result`. ولا تضبطه على `true` إلّا حالات قليلة من الرفض بالحالة `503` – تعذّر الوصول إلى التخزين (`STORAGE_SERVICE_UNAVAILABLE`)، وفحص الاعتماديات عند الحذف النهائي لكيان، وتعذّر إتاحة خدمة الوظائف نفسها، ومن `Since 1.10` حالات `503` في واجهة النسخ الاحتياطي؛ أمّا سائر الحالات فتُبقيه `false`. لا تحدّد مسار المعالجة بناءً عليه؛ بل اشتقّ قابلية إعادة المحاولة من الحالة والرمز كما تصف هذه الصفحة.

فهرس رموز الأخطاء

يجمع الفهرس أدناه كلّ صنف استثناء مُنمَّط ملموس تعرّفه الخدمات والبوابة ضمن العائلة التي تحدّد حالته، وبذلك يشمل جميع الأخطاء *المُنمَّطة*. وتلزم مراعاة ثلاثة تحفّظات صريحة:

- تعريف الصنف لا يعني أنّ كلّ عملية قادرة على إطلاقه؛ فبعضها لا يحدث إلّا في المهامّ الخلفية، حيث يظهر الإخفاق عبر استطلاع حالة المهمّة لا عبر خطأ HTTP (انظر العمل طويل الأمد).
- قد يقترن نصّ الرمز الواحد *بحالة مختلفة في خدمة مختلفة* – فالرمز `ENTITY_FIELD_NOT_EXIST` يقترن بالحالة `404` في خدمة البيانات، لكنّه يقترن بالحالة `500` في خدمة الأتمتة، لأنّ صنفَي الخدمتين ينتميان إلى عائلتين مختلفتين. حدّد مسار المعالجة بحسب `code`، وعامل الحالة بوصفها مؤشرًا عامًّا.
- تُصدَّر ثلاثة ثوابت بأسماء معيّنة، لكنّها تُرسل *نصًّا مختلفًا في البيانات المنقولة*. تسرد الجداول القيم المنقولة؛ وابحث في رماز الخادم الخلفي عن اسم الثابت إن احتجت إلى المصدر:

الثابت (المصدر)	قيمة السلك (`error.code`)
`FILTER_FIRST_FIELD_CAN_NOT_BE_OF_THIS_TYPE`	`FILTER_FIRST_FIELD_CAN_NOT_BE_OF_THIS_TYPE_EXISTS`
`ROLE_MUST_BE_JOB_POSITION`	`ROLE_MUST_BE_JOP_POSITION`
`INVALID_MANAGER_STEP_OWNER_SELECTION_CRITERIA`	`INVALID__MANAGER_STEP_OWNER_SELECTION_CRITERIA` (شرطة سفلية مزدوجة)

أخطاء البوابة

تُطلقها مكوّنات البوابة قبل بلوغ أيّ خدمة، وفق شكل البوابة المعروض أعلاه. وحالات `500` في هذا الجدول هي إخفاقات في البنية التحتية (تعذّر على البوابة الوصول إلى اعتمادياتها)، وهي *الوحيدة* بين جميع حالات `500` في هذا الفهرس التي تستحقّ إعادة محاولة تلقائية.

الحالة	`error.code`	المكوّن	متى
400	`DOMAIN_ERROR`	`domain-resolver`	لا يمكن تحديد قاطن من ترويسة `Host` في الطلب.
404	`SPACE_ERROR`	`space-resolver`	لا توجد مساحة تطابق المضيف (`Since 1.10` ؛ وكانت الحالة `400`).
404	`APP_ERROR`	`app-resolver`	في نشرٍ متعدّد القاطنين: لا يقابل المضيف أيّ قاطن (`Since 1.10`).
400	`SPACE_ID_MISSING`	`user-space-checker`	ترويسة `x-space-id` غائبة حيث تكون مطلوبة.
400	`MISSING_DOMAIN_ID`	`osos-access`	معرّف النطاق غائب من السياق بعد حلّه.
400	`MISSING_USER_ID`	`user-delegation`	طُلب تفويض دون معرّف مستخدم.
400	`INVALID_APP_TOKEN_API_ERROR`	`app-token-auth`	رمز API في `x-app-token` غير معروف أو منتهي الصلاحية أو معطَّل أو مؤرشف. وتمرّر الإضافة الرمز `400` الذي تعيده نقطة التحقق. لا تُعِد المحاولة.
401	`AUTHENTICATION_ERROR`	`oauth2-introspection`	أخفق استبطان رمز الحامل.
403	`INVALID_ACCESS`	`osos-access`	يمنع كائن الوصول هذا المستدعي من الوصول إلى هذا المسار/المعرض.
403	`SPACE_FORBIDDEN`	`user-space-checker`	المستخدم ليس عضوًا في المساحة المطلوبة.
403	`DELEGATION_EXPIRED`	`user-delegation`	انقضت نافذة التفويض.
429	*(جسم Kong)*	`rate-limiting`	تجاوز 100 طلب/ثانية أو 10,000/ساعة إلى خدمة واحدة.
429	`RATE_LIMITED`	`tenant-rate-limit`	في نشرٍ متعدّد القاطنين: تجاوز القاطن حصته في الثانية أو في الدقيقة؛ التزم بـ `Retry-After` (`Since 1.10`).

الحالة	error.code	المكوّن	متى
503	RATE_LIMIT_UNAVAILABLE	tenant-rate-limit	أخفق مخزن مُقيِّد المعدّل واختار المشغّل الرفض؛ Retry-After: 1 (Since 1.10).
500	REDIS_ERROR ، REDIS_CONNECTION_ERROR	عدّة مكوّنات	تعذّر على البوابة الوصول إلى Redis الخاص بها أو القراءة منه.
500	API_REQUEST_ERROR ، API_DECODE_ERROR ، API_LOGIC_ERROR	user-space-osos-checker ، access	أخفق استدعاء بحث داخلي أو أعاد بيانات فاسدة.
500	AUTHENTICATION_ERROR	oauth2-introspection	أخفقت *بنية* الاستبطان التحتية (وهذا يختلف عن حالة 401).
500	APP_TOKEN_VALIDATION_REQUEST_FAILED	app-token-auth	تعذّر إتمام استدعاء التحقّق من الرمز.
500	USER_PROFILE_FETCH_ERROR ، ACCESS_OBJECT_DECODE_ERROR	osos-access	أخفق جلب الملف الشخصي أو كائن الوصول.
500	DELEGATION_API_REQUEST_FAILED ، DELEGATION_API_DECODE_ERROR ، DELEGATION_API_LOGIC_ERROR ، DELEGATION_DATE_PARSE_ERROR	user-delegation	أخفقت عملية البحث عن التفويض.
500	INTERCEPTOR_DECODE_ERROR	interceptor	تعذّر على المُعترِض فكّ ترميز استجابة داخلية.
500	KNATIVE_REQUEST_FAILED	knative-functions	تعذّر إرسال استدعاء دالّة.

أخطاء إطار العمل (أي خدمة LoopBack)

يُطلقها خطّ معالجة الطلب نفسه في أيّ خدمة LoopBack، قبل تنفيذ عمليتك أو في المعالجة المحيطة بها. وتأتي رموز التحقّق من الطلب من `@loopback/rest`؛ أمّا البقية فتُظهر اسم الصنف لأنّه لم يُسنَد إليها رمز منصّة.

الحالة	error.code	متى
400	INVALID_PARAMETER_VALUE	أخفق تحويل معامل استعلام/مسار أو التحقّق منه – مثل JSON مشوَّه في filter .
400	MISSING_REQUIRED_PARAMETER	أُغفل معامل مطلوب – ويشمل ذلك من Since 1.10 أيضًا طلب إنشاء مُدخل أو مسودّة أُرسل بلا جسم إطلاقًا، ومن Since dev طلب حفظ معرض (PATCH /api/v1/entities/{entity}/views/{id}) أُرسل بلا جسم.
400	SPACE_CONTEXT_MISSING	وصل طلب إلى خدمة دون x-space-id صالح ودون نطاق مساحة (Since 1.10 ؛ انظر القاطنون والهوية). لا تُعِد محاولة الطلب كما هو.
400	SPACE_NOT_RESOLVED	تتعارض ترويسة x-space-id مع المساحة الموجودة أصلًا في النطاق (Since 1.10).
400	UNIQUE_VALUE	انتهاك مفتاح مكرَّر من Mongo بعد تطبيعه؛ الحقل المخالف تحت result.details.field .
401	UnauthorizedError	بيانات اعتماد مفقودة أو غير صالحة – بما فيها حالة USER_PROFILE_NOT_FOUND بعد تطبيعها.
403	ForbiddenError	رفض غير مُنمَّط – وأبرز حالاته بوّابة التحقّق البشري عند إنشاء مُدخل مجهول الهوية.
404	NotFoundError	لا يوجد مسار مطابق، أو حالة غياب غير مُنمَّطة كمعرّف مهمّة مجهول.
404	TENANT_NOT_RESOLVED	في نشرٍ متعدّد القاطنين: لا يقابل المضيف أيّ قاطن نشط (Since 1.10).
415	UNSUPPORTED_MEDIA_TYPE	content-type غير مقبول.
422	VALIDATION_FAILED	أخفق التحقّق من جسم الطلب وفق المخطّط؛ تفاصيل المخالفات (مسار JSON والكلمة المفتاحية والقيم المسموحة) تحت result.details .

الحالة	error.code	متى
503	TENANT_IN_MAINTENANCE	في نشرٍ متعدّد القاطنين: تُستعاد بيانات القاطن من نسخة احتياطية، أو يُحتجَز القاطن لنقله إلى خلية أخرى (Since 1.10). تأتي الاستجابة مع Retry-After (من 1 إلى 3600 ثانية، والافتراضي 30)؛ وتُعيدها أيضًا users-service وstorage-service. تُكتب مباشرةً، فتكون قيمة traceId هي NO-TRACE-ID وتكون message.ar فارغة. انتظر ثم أعد المحاولة.
500	TypeError وأسماء أصناف JS أخرى	علّة غير معالَجة في الخادم. أبلغ عنها؛ ولا تدخل أبدًا في حلقة لإعادة المحاولة عند حدوثها.

مستوى البيانات – data-service

هذا المستوى هو الأوسع نطاقًا؛ إذ تعرّف خدمة البيانات نحو **520 صنف استثناء ملموسًا**، وتتوزّع بحسب عائلاتها على نحو 311 حالة فشل في التحقّق (400)، و2 لعدم التصريح (401)، و34 للمنع (403)، و46 لعدم الوجود (404)، و10 تعارضات (409)، و21 حالة غير قابلة للمعالجة (422)، و95 صنفًا مشتقًّا من DataException المجرّدة – يظهر معظمها بالحالة 500 ، ويحدّد عدد قليل منها حالته الخاصّة (حالات 503 وواجهة النسخ الاحتياطي، التي تعيد حالات الرفض بالحالة التي يحدّدها محرّك النسخ الاحتياطي). والجدول أدناه غير شامل (إضافات 1.10 وإعادات تصنيفها مُعلَّمة بـ Since 1.10)؛ لذا فإنّ غياب رمز عنه لا يدلّ على عدم وجوده.

نقاط تستحقّ الإبراز قبل استعراض الجدول:

- **مجموعة الحقول الحسّاسة** *(Since 1.8)*: SENSITIVE_FIELD_NOT_ALLOWED و SENSITIVE_FIELD_CANNOT_BE_MAIN_TITLE و SENSITIVE_FIELD_QUERY_NOT_ALLOWED و SENSITIVE_FIELD_WRITE_NOT_ALLOWED كلّها حالات رفض للتحقّق بالحالة 400 ، تُطلَق حين يتناول تعريف أو استعلام أو عملية كتابة حقلًا حسّاسًا على نحو تمنعه المنصّة.
- **عائلة الأقفال 409** : LOCKED_OBJECT و LOCKED_OBJECT_ALREADY_OWNED و NOT_ALLOWED_TO_LOCK_OBJECT – جهة فاعلة أخرى تحتفظ بقفل الكائن. والاستجابة الصحيحة هي الانتظار وإعادة القراءة، وتجنّب إعادة المحاولة العمياء.
- **UPDATE_CONFLICT (409)** يعني أنّ updateIdentifier لديك لم يعد يطابق القيمة المخزَّنة – فقد تغيّر المُدخل أثناء ذلك. أعد جلبه، ثم أعد تطبيق تغييرك وإرساله. انظر كتابة البيانات.
- **أخطاء إعداد الأذونات** (بهذا الإملاء بالضبط) هي أخطاء 400 تتعلّق بإعداد الاستحقاقات – مثل ENTITLED_ENTRY_RESTRICTED_PERMISSION و RESTRICTED_LEVEL_SYSTEM_ONLY – وليست حالات رفض للأذونات أثناء التشغيل؛ فتلك تنتمي إلى عائلة 403 .
- **حالتا 404 مُستثناتان**: يرث DataSourceEntityNotExist و EntitiesNotFound من الصنف الأساسي المجرّد، ولا يعيدان الحالة 404 إلّا لأنّ مزوِّد الرفض يربط الرمز ENTITY_NOT_FOUND بها صراحةً.
- **حالتا 401 غير مألوفتين.** DashboardConfigurationNotEnabledException حالة تتعلّق بإعداد لوحة معلومات، وليست مشكلة في بيانات الاعتماد، رغم رمز حالتها؛ أمّا UserNotAuthenticatedException (USER_NOT_AUTHENTICATED) فتخصّ نقطة نهاية خاصّة بالمستخدم استُدعيت برمز مجهول.
- **حدود الخطة حالات 403** – Since 1.10 ENTRY_CAP_EXCEEDED و ENTITY_CAP_EXCEEDED و SPACE_CAP_EXCEEDED و STORAGE_CAP_EXCEEDED) من storage-service). تقتصر حالات الرفض هذه على عمليات الإنشاء، ولن تفيد إعادة المحاولة حتى تتغيّر خطة القاطن أو استهلاكه – انظر حدود الخطة.
- **تنقل واجهة النسخ الاحتياطي رموز محرّك النسخ الاحتياطي بحالاتها الخاصّة** Since 1.10 – 400 و 404 و 409 و 429 (حصّة، مع Retry-After) و 501 (نشر وحيد القاطن) و 502 و 503 (مع Retry-After: 60). وترد القائمة الكاملة في النسخ الاحتياطي للقاطن واستعادته وتصدير بياناته.

الحالة	error.code	صنف الاستثناء المُنشئ
400	ALL_ENTRY_VALIDATION_ERROR	AllEntryValidationExceptions
400	ARCHIVE_ENTRIES_MUST_PROVIDE_IDS_OR_SELECT_ALL	ArchiveEntriesMustProvideIdsOrSelectAllException
400	ASSIGNMENT_LEVEL	OrgUnitAcceptJustMainLevelException
400	ATTACHMENT_EXTENSIONS_NOT_SET	ViewDiscussionsAttachmentExtensionsException
400	BULK_UPDATE_ENTRIES_MUST_PROVIDE_IDS_OR_SELECT_ALL	BulkUpdateEntriesMustProvideIdsOrSelectAllException
400	BULK_UPGRADE_ENTRIES_MUST_PROVIDE_IDS_OR_SELECT_ALL	BulkUpgradeEntriesMustProvideIdsOrSelectAllException
400	CAN_NOT_ARCHIVE_DOMAIN_IN_USE	CanNotArchiveDomainInUse
400	CAN_NOT_ARCHIVE_TEMPLATE_IN_USE	CanNotArchiveTemplateInUse
400	CAN_NOT_CHANGE_CONDITIONAL_INCREMENT_CONFIGURATIONS	CanNotChangeConditionalIncrementConfigurations
400	CASCADED_FIELD	CascadedFieldException
400	CIRCULAR_RELATION_NOT_ALLOWED	CircularRelationNotAllowedException
400	COLLECTION_IS_IMPLICIT	ImplicitDeleteException
400	COLLECTION_ITEM_IS_IMPLICIT	ImplicitItemUpdateException
400	COMPONENT_GROUP_ID_NOT_EXIST	ComponentGroupIdNotExistException
400	COMPONENT_GROUP_ID_NOT_UNIQUE	ComponentGroupIdMustBeUniqueException
400	COMPONENT_ID_NOT_UNIQUE	ComponentIdMustBeUniqueException
400	COMPONENT_INCOMPATIBLE_RESTRICTED_PERMISSION	IncompatibleRestrictedPermissionException
400	COMPONENT_VISIBILITY	ComponentVisibilityForAnonymousException
400	COUNTER_VIEW_MISSED_MINICARD_VIEW_TYPE	CounterViewMissedMiniCardException
400	CURRENT_USER_FILTER_CAN_NOT_BE_ENABLED	CurrentUserFilterCanNotBeEnabledException
400	CURRENT_USER_GROUP_FILTER_CAN_NOT_BE_ENABLED	CurrentUserGroupsFilterCanNotBeEnabledException
400	CUSTOM_EVENT_INVALID_CONFIGS	CustomEventInvalidConfigsException
400	CUSTOM_VALIDATION_EXPRESSION_FAILED	CustomValidationExpressionFailed
400	DASHBOARD_ELEMENT_DUPLICATE	DashboardElementDuplicateException
400	DASHBOARD_ELEMENT_USED_MORE_THAN_ONCE	DashboardElementUsedMoreThanOneException
400	DATA_QUERY_UPDATE	DataQueryUpdateException
400	DATA_VIEW_INVALID_ALLOWED_OPERATIONS	Since 1.10 (500 was) DataViewAllowedOperationNotFoundException
400	DATA_VIEW_INVALID_ARCHIVED_OPERATION_FIELDS_VALUE	DataViewArchivedOperationFieldsExistException
400	DATA_VIEW_MISSED_REQUIRED_FIELDS	DataViewMissedRequiredFieldsException
400	DATA_VIEW_OPERATION_FIELDS_REQUIRED	DataViewOperationFieldsRequiredException
400	DATA_VIEW_SOURCE_ENTITIES_MAPPING_INVALID_SOURCE	DataViewSourceEntitiesMappingInvalidSourceException
400	DATA_VIEW_SOURCE_ENTITIES_MAPPING_NOT_SUPPORTED_FOR_NON_AGGREGATOR_ENTITY	DataViewSourceEntitiesMappingNotSupportedForNonAggregatorEntityException
400	DELETE_IMPLICIT_RESOURCE_RESTRICTION	DeleteImplicitResourcesException

الحالة	error.code	صنف الاستثناء المُنشئ
400	DENY_CIRCULAR_RELATION_WITH_MULTI_VALUE_OR_REPEATABLE	DenyCircularRelationWithMultiValueOrRepeatableException
400	DETAILS_META_DATA_FILTER_CAN_NOT_BE_ENABLED	DetailsMetaDataFilterCanNotBeEnabledException
400	DRAFT_ENTRY_NOT_BELONG_TO_USER	DraftEntryNotBelongToUserException
400	DUPLICATE_FIELD_UNIQUE_ERROR	DuplicatedFieldCanNotBeUniqueError
400	DUPLICATE_PERMISSIONS	ViewAnonymousAndRestrictedException
400	DUPLICATED_ASSIGNMENT	AnotherAssignmentExists
400	DUPLICATED_SHOW_AS_FIELD_OPTION	DuplicatedShowAsException
400	EMBEDDED_PAGE_INVALID_URL	EmbeddedPageInvalidURLException
400	EMBEDDED_PAGE_MUST_HAVE_URL	EmbeddedPageMustHaveURLException
400	EMPTY_ARRAY	ArrayCanNotBeEmptyException
400	EMPTY_LOCALIZED_FIELD	EmptyLocalizedException
400	END_USER_SPACE_ALREADY_ACTIVATED	EndUserSpaceAlreadyActivatedException
400	END_USER_SPACE_ALREADY_DEACTIVATED	EndUserSpaceAlreadyDeactivatedException
400	ENTITLED_ENTITY_NOT_ACTIVE	EntitledEntityNotActiveException
400	ENTITLED_ENTITY_NOT_EXIST	NotExistEntitledException
400	ENTITLED_ENTRY_RESTRICTED_PERMISSION	RestrictedPermissionEntitledEntriesEmptyExcecption
400	ENTITLEMENT_TYPE_REQUIRED	EntitlementTypeRequiredException
400	ENTITY_TYPE_NOT_REQUEST	EntityNotOfTypeRequestException
400	ENTRIES_CONFIGURATION_NOT_ENABLED_FOR_VIEW	ViewEntriesNotAccessibleException
400	ENTRIES_CONFIGURATIONS_MUST_BE_ENABLED	EntriesConfigurationMustBeEnabled
400	ENTRY_COMPLETED	SpecifiedEntryCompletedException
400	ENTRY_NOT_IN_WORKFLOW	EntryNotInWorkflowException
400	ENTRY_VALIDATION_ERROR	EntryValidationException
400	ENTRY_VALIDATION_ERROR	ValidatorFunctionEntryValidationException
400	ENTRY_VERSION_REQUIRED	EntryVersionRequiredException
400	EXTERNAL_STEP_CONFIG_REQUIRED	ExternalStepConfigRequiredException
400	EXTERNAL_STEP_CUSTOM_PAYLOAD_REQUIRES_POST	ExternalStepCustomPayloadRequiresPostException
400	EXTERNAL_STEP_MANUAL_EXECUTION_NOT_ALLOWED	ExternalStepManualExecutionNotAllowedException
400	EXTERNAL_STEP_OWNER_MUST_BE_USER	ExternalStepOwnerMustBeUserException
400	FAIL_TO_ADD_FIELD	FailToAddFieldException
400	FAILURE_ACTION_REQUIRED	FailureActionsRequiredException
400	FIELD_MUST_ENABLE_CIRCULAR_RELATION	FieldMustEnableDenyCircularRelationException
400	FILE_EXPIRED	Since 1.10 FileExpiredException
400	FILTER_FIRST_CAN_ENABLE_AUTOFILL_WITHOUT_ENABLING_ADD_ENTRIES	FilterFirstCanNotEnableAutofillWithoutEnablingAddEntries
400	FILTER_FIRST_CAN_NOT_BE_ENABLED_WITH_DASHBOARD	FilterFirstCanNotBeEnabledWithDashboard
400	FILTER_FIRST_CUSTOM_QUERY_NOT_PROVIDED	FilterFirstCustomQueryNotProvidedException

الحالة	error.code	صنف الاستثناء المُنشئ
400	FILTER_FIRST_FIELD_CAN_NOT_BE_OF_THIS_TYPE_EXISTS	FilterFirstFieldCanNotBeAppliedToFieldType
400	FILTER_FIRST_FIELD_NOT_EXISTS	FilterFirstFieldNotExists
400	FILTER_FIRST_FIELDS_MISMATCH	FilterFirstFieldsMismatch
400	FILTER_FIRST_MAX_FIELDS_LENGTH	FilterFirstMaxFieldsLength
400	FILTER_FIRST_MIN_FIELDS_LENGTH	FilterFirstMinFieldsLength
400	FILTER_FIRST_SHOULD_BE_ENABLED_WITH_QUERY_PARAMETER	FilterFirstShouldBeEnabledWithQueryParameter
400	FONT_FORMAT_NOT_SUPPORTED	FontFormatNotSupportedException
400	GENERATING_COUNTER_ERROR	GeneratingCounterException
400	GENERATING_COUNTER_SEQUENCE_ERROR	GeneratingCounterSequenceException
400	GENERIC_VIEW_NORMALIZED_NAME_DUPLICATE	GenericViewNormalizedNameDuplicateException
400	GENERIC_VIEW_RELATED_TO_DATA_ENTITY	GenericViewRelatedToDataEntityException
400	GET_SYSTEM_BUTTONS_STATUS_FOR_ENTRIES_MUST_PROVIDE_IDS_OR_SELECT_ALL	GetSystemButtonsStatusForEntriesMustProvideIdsOrSelectAllException
400	IMPLICIT_DELETE_RESTRICTION	ImplicitEntityDeleteException
400	IMPLICIT_DELETE_RESTRICTION	ImplicitRelationDeleteException
400	IMPLICIT_ENTRY_UPDATE	ImplicitEntryException
400	IMPLICIT_FIELD_RESTRICTION	ImplicitFieldException
400	IMPLICIT_UPDATE_RESTRICTION	ImplicitEntityUpdateException
400	IMPLICIT_UPDATE_RESTRICTION	ImplicitRelationUpdateException
400	IMPLICIT_UPDATE_RESTRICTION	ImplicitResourceUpdateException
400	INLINE_EDIT_NOT_ALLOWED	ViewDoesNotSupportInlineEdit
400	INTERNAL_ENTITY_NOT_FOUND	InternalEntityNotFoundException
400	INVALID__MANAGER_STEP_OWNER_SELECTION_CRITERIA	InvalidManagerStepOwnerSelectionCriteria
400	INVALID_CATEGORY_OBJECT	InvalidCategoryObject
400	INVALID_DATA_VIEW	InvalidDataViewException
400	INVALID_EXPORT_FORMAT	InvalidExportFormatException
400	INVALID_FIELD_MAPPING	InvalidFieldMappingException
400	INVALID_FIELD_SOURCE_FOR_DENY_CIRCULAR_RELATION	InvalidFieldSourceForDenyCircularRelationException
400	INVALID_GEOJSON_OBJECT	InvalidGeojsonObject
400	INVALID_LOCALIZATION	InvalidLocalizationException
400	INVALID_LOCK_END_DATE	InvalidLockEndDate
400	INVALID_OR_DISABLED_TRIGGER	InvalidOrDisabledTriggerException
400	INVALID_POINTER	Since 1.10 InvalidPointerException
400	INVALID_QUERY_EXPRESSION_TYPE	DataQueryExpressionTypeValidationException
400	INVALID_QUERY_EXPRESSION_VALUE	DataQueryExpressionValueNotStringException
400	INVALID_RELATED_FIELD	InvalidRelatedFieldException

الحالة	error.code	صنف الاستثناء المُنشئ
400	INVALID_STATISTIC_QUERY_SOURCE	StatisticQuerySourceNotSameAsViewException
400	INVALID_STATISTIC_QUERY_VALIDATION	StatisticQueryValidationExcpetion
400	INVALID_STEP_HAS_MULTIPLE_MAIN_ACTIONS	InvalidStepHasMultipeMainActionsException
400	INVALID_STEP_ID	WorkflowStepIdNotExistException
400	INVALID_STRING_DATE	InvalidStringDateException
400	INVALID_SYSTEM_LINK_TOKEN	InvalidViewTokenExption
400	INVALID_THRESHOLD_VALUE	InvalidStepOwnerThresholdException
400	INVALID_TOKENS_MAP	InvalidTokensMapException
400	JOINER_PROXY_MAX_FIELDS_LIMIT_EXCEEDED	JoinerProxyMaxFieldsLimitExceeded
400	JOINER_PROXY_MAX_SOURCES_LEVELS_LIMIT_EXCEEDED	JoinerProxyMaxSourcesLevelsLimitExceeded
400	JOINER_PROXY_MAX_SOURCES_LIMIT_EXCEEDED	JoinerProxyMaxSourcesLimitExceeded
400	LANGUAGE_MAX_LENGTH	LanguageMaxLengthException
400	LANGUAGE_MIN_LENGTH	LanguageMinLengthException
400	LANGUAGE_NOT_SUPPORTED	LanguageNotSupportedException
400	LOCALIZATION_DUPLICATE	DuplicatedLocalization
400	MAIN_TITLE_CANNOT_BE_REPEATABLE_OR_MULTIPLE	MainTitleCannotBeRepeatableOrMultipleException
400	MAX_LESS_THAN_MIN	MinMaxValidationException
400	MAX_VALUE_SHOULD_BE_NUMBER	MaxValueShouldBeNumberException
400	MAX_VALUE_SHOULD_NOT_BE_SMALLER_THAN_MIN_VALUE	MaxValueShouldNotBeSmallerThanMinValueException
400	META_DATA_FILTER_CAN_NOT_BE_ENABLED	MetaDataFilterCanNotBeEnabledException
400	MIN_VALUE_SHOULD_BE_NUMBER	MinValueShouldBeNumberException
400	MISSING_FIELDS_FOR_FILTER_FIRST	MissingFieldsForFilterFirstException
400	MISSING_FILTER_FIRST_FIELD_NAME	MissingFilterFirstFieldName
400	MISSING_REQUIRED_STEP_ACTIONS	MissingStepActionsException
400	MODULE_HAS_ENTITIES	ModuleHasEntities
400	MULTIPLE_MANAGER_ROLE_NOT_ALLOWED	MultipleManagerRoleNotAllowed
400	MUST_AUTOMATE_AT_LEAST_ONE_STEP	MustAutomateAtLeastOneStepException
400	MUST_ENABLE_AUTHENTICATED	AuthenticatedMustEnabledExcecption
400	MUST_ENABLE_DISCUSSIONS_TO_ENABLE_DISCUSSIONS_ATTACHMENTS_INSIDE_ENTRY_INFO	MustEnableDiscussionsToEnableDiscussionsAttachmentsInsideEntryInfo
400	MUST_ENABLE_ENTRIES_LIST_OR_ENTRY_DETAILS_TO_ENABLE_SYSTEM_BUTTONS_ON_VIEW	MustEnableEntriesListOrEntryDetailsToEnableSystemButtonsOnView
400	MUST_ENABLE_RESTRICTED	RestrictedMustEnabledExcecption
400	MUST_PROVIDE_ITEMS_TO_EXPORT_PROPERTY	MustProvideITemsToExportProperty
400	NESTED_REFERENCE_MAIN_TITLE_REFERENCED_BY_OTHER_ENTITY	NestedReferenceMainTitleReferencedByOtherEntity
400	NESTED_REFERENCE_MAIN_TITLE_SOURCE_IS_REFERENCE	NestedReferenceMainTitleSourceIsReference
400	NEW_FIELDS_UNIQUE_ERROR	NewFieldCantBeUniqueError

الحالة	error.code	صنف الاستثناء المُنشئ
400	NORMAL_QUERY_TO_PARAMETERIZED_QUERY_RESTRICTION	DataQueryParameterizationChangeException
400	NOT_ALLOWED_TO_PATCH_UPDATE_FIELDS	NotAllowedToPatchUpdateFields
400	NOT_ENOUGH_STORE_SPACES_SELECTED	NotEnoughStoreSpacesSelectedException
400	OBSOLETED_SAME_VERSION	EntrySameVersionException
400	OPERATION_NOT_SUPPORTED_FOR_THIS_STORE_TYPE	OperationNotSupportedForThisType
400	ORIGINAL_OWNER_NOT_FOUND	OriginalOwnerNotFoundException
400	PARALLEL_STEP_OWNER_ALREADY_EXECUTED_STEP	ParallelStepOwnerAlreadyExecutedStepException
400	PATCH_UPDATE_DUPLICATE_REFERENCE_ENTRY_ID	PatchUpdateDuplicateReferenceEntryIdException
400	PATCH_UPDATE_SELF_MODE	PatchUpdateSelfModeException
400	PERMANENT_DELETE_ENTRIES_MUST_PROVIDE_IDS_OR_SELECT_ALL	PermanentDeleteEntriesMustProvideIdsOrSelectAllException
400	PERMISSIONS_ANONYMOUS_OR_RESTRICTED_NOT_ENABLED	AnonymousOrRestrictedNotEanabledException
400	PERMISSIONS_ANONYMOUS_RESTRICTED_ENABLED	AnonymousAndRestrictedEnabledException
400	PERMISSIONS_NOT_SET	ViewPublishedWithoutPermissionsException
400	PERSON_CAN_HAVE_ONE_MAIN_ASSIGNMENT	UserCanHaveOnlyOneAssignment
400	PLATFORM_NOT_SUPPORTED	PlatformNotSupportedException
400	POST_TEMPLATE_NOT_EXIST	PostTemplateNotExistException
400	QUERY_ALREADY_IN_USE	QueryInUsedException
400	QUERY_ENGINE_EXPRESSION_ERROR	ExpressionParsingException
400	READONLY_ENTITY_OPERATION	ReadOnlyEntityOperationException
400	RELATED_FIELDS_REQUIRED	RelatedFieldsRequiredException
400	RELATION_NOT_FOUND_INSIDE_VIEW	RelationNotFoundInsideViewException
400	RELATIVE_VALUE_NOT_SUPPORTED	RelativeValueNotSupportedException
400	REPORT_DATA_SOURCE_NOT_SUPPORTED	ReportDataSourceSupportedException
400	REPORT_EXISTS	ReportExistException
400	REPORT_FIELDS_LENGTH	ReportFieldsLengthException
400	REPORT_FIELDS_NOT_EXIST	ReportFieldsNotExistException
400	REPORT_SENDING_TIME	ReportSendingTimeCanNotBeInThePastException
400	REQUEST_CANNOT_EDITED	CanNotEditEntryException
400	REQUEST_CANNOT_PUBLISHED	ServiceHasNoWorkflowException
400	RESTORE_ENTRIES_MUST_PROVIDE_IDS_OR_SELECT_ALL	RestoreEntriesMustProvideIdsOrSelectAllException
400	RESTRICTED_AUTHENTICATED_VIEW_OPTIONS_REQUIRE_PERMISSIONS	RestrictedAuthenticatedViewOptionsRequirePermissionsException
400	RESTRICTED_LEVEL_SYSTEM_ONLY	RestrictedLevelSystemOnlyException
400	RESTRICTED_NORMALIZED_NAME	RestrictedNormalizedNameError
400	RESTRICTED_VIEW_CAN_NOT_BE_DEFAULT	RestrictedViewCanNotBeDefaultException
400	ROLE_MUST_BE_JOP_POSITION	RoleMustBeOfTypeJobPosition

الحالة	error.code	صنف الاستثناء المُنشئ
400	SAVE_QUERY_NOT_ALLOWED	SaveQueryNotAllowedException
400	SELECTED_SPACES_SHOULD_NOT_HAVE_DUPLICATION	StoreSelectedSpacesDuplicationNotAllowedException
400	SENSITIVE_FIELD_CANNOT_BE_MAIN_TITLE	SensitiveFieldCannotBeMainTitleException
400	SENSITIVE_FIELD_NOT_ALLOWED	SensitiveFieldNotAllowedException
400	SENSITIVE_FIELD_QUERY_NOT_ALLOWED	SensitiveFieldQueryNotAllowedException
400	SENSITIVE_FIELD_WRITE_NOT_ALLOWED	SensitiveFieldWriteNotAllowedException
400	SERVICE_AUDIENCE_REQUIRED	ServiceAudienceIsRequired
400	SERVICE_CATEGORY_NOT_EXIST	ServiceCategoryNotExistException
400	SERVICE_CATEGORY_REQUIRED	ServiceCategoryRequiredException
400	SERVICE_ENTITY_NOT_ACTIVE	CanNotRequestServiceException
400	SINGLE_ROLE_ASSIGNED_MULTIPLE_TIMES	SingleRoleAssignedMultipleTimes
400	STEP_NOT_EXECUTED_BEFORE	StepNotExecutedBeforeException
400	STORE_WITH_SAME_WEIGHT_ALREADY_EXIST	StoreWithSameWeightExistsException
400	SUB_WORKFLOW_STEP_MUST_NOT_HAVE_ACTIONS	SubWorkflowStepActionsException
400	SYSTEM_BUTTON_CONFIRMATION_DIALOG_VARIABLES_NOT_ALLOWED_FOR_MULTI_SCOPE	ConfirmationDialogVariablesNotAllowedForMultiScopeException
400	TABLE_VIEW_CONFIG_NOT_FOUND	TableViewConfigNotFoundException
400	THIS_ROLE_CAN_ONLY_BE_ASSIGNED_ONE_ORG_UNIT	ThisRoleCanOnlyBeAssignedToOneOrgUnit
400	TIMELINE_ATTRIBUTE_MAPPING_NOT_MATCHING_ANY_CRITERIA	TimelineAttributeMappingNotMatchingAnyCriteriaException
400	TIMELINE_ATTRIBUTE_NOT_MATCH_POST_CATEGORY_ATTRIBUTES	TimelineAtrributeNotMatchPostCategoryAttributes
400	TIMELINE_CORE_ENTITY	TimelineCoreEntityExcecption
400	TIMELINE_DESCRIPTION_FIELD_INVALID	TimelineDescriptionFieldInvalidFieldTypeExcecption
400	TIMELINE_FIELD_NAME_NOT_EXIST	TimelineFieldNameNotExistException
400	TIMELINE_FIELD_VALUE_MUST_BE_DEFINED	TimelineFieldValueMustBeDefined
400	TIMELINE_FIELD_VALUE_MUST_BE_HIDDEN	TimelineFieldValueMustBeHidden
400	TIMELINE_MISSING_REQUIRED_FIELDS	TimelineMissingRequiredFieldsException
400	TIMELINE_PUBLISH_CONFIG_CONFLICT	TimelinePublishConfigConflictException
400	TIMELINE_PUBLISH_WORKFLOW_EMPTY_STEPS	TimelinePublishWorkflowEmptyStepsException
400	TIMELINE_PUBLISH_WORKFLOW_INIT_STEP	TimelinePublishWorkflowInitStepException
400	TIMELINE_PUBLISH_WORKFLOW_LAST_STEP	TimelinePublishWorkflowlastStepException
400	TIMELINE_PUBLISH_WORKFLOW_STEP_NOT_EXIST	TimelinePublishWorkflowStepNotExistException
400	TIMELINE_TITLE_FIELD_INVALID	TimelineTitleFieldInvalidFieldTypeExcecption
400	TO_ENABLE_CLICK_TO_BROWSE_MUST_ENABLE_VIEW_CONFIGURATIONS	ToEnableClickToBrowseMustEnableViewConfiguration
400	TRANSFORMATION_REQUIRED	TransformationModelRequiredException
400	TRIGGER_TIMER_OPTIONS_CONFLICT	TriggerTimerOptionsConflict
400	TRIGGERS_CAN_NOT_AUTOMATE_LAST_STEP	CanNotAutomateLastStepException

الحالة	error.code	صنف الاستثناء المُنشئ
400	TRIGGERS_LINKED_TO_NON_EXISTING_WORKFLOW	TriggersLinkedToNonExistingWorkflow
400	TRIGGERS_LINKED_TO_NON_EXISTING_WORKFLOW_STEP	TriggerLinkedToNonExistingWorkflowStep
400	TRIGGERS_LINKED_TO_NON_EXISTING_WORKFLOW_STEP_ACTION	TriggerLinkedToNonExistingWorkflowStepAction
400	UNDEFINED_STEP_OWNER_THRESHOLD	UndefinedStepOwnerThresholdException
400	UNKNOWN_OPERATOR_ERROR	UnknownOperatorExpression
400	UNSUPPORTED_CONDITIONAL_INCREMENT_CALCULATION_METHOD	UnsupportedConditionalIncrementCalculationMethod
400	UNSUPPORTED_FIELD_EXPRESSION	UnsupportedFieldExpression
400	UPDATE_COLLECTION_NO_ITEMS	UpdateCollectionNoItemsException
400	UPDATE_DRAFT_ENTRY_NOT_ALLOWED	UpdateDraftEntryNotAllowedException
400	UPDATE_IMPLICIT_RESOURCE_RESTRICTION	UpdateImplicitResourcesException
400	UPDATED_ENTRY_EVENT_CONFIG_INVALID_FIELD	UpdatedEntryEventConfigInvalidFieldException
400	USER_ACCOUNT_TYPE	UserIsNotOrganizationAccount
400	USER_ALREADY_OWN_STEP	UserAlreadyOwnException
400	USER_COLLECTION_DELETE	UserCollectionsDeleteException
400	USER_COLLECTION_REPLACE	UserCollectionReplaceException
400	USER_COLLECTION_SAVE_NOT_ALLOWED	SaveUserCollectionNotAllowedException
400	USER_COLLECTION_UPDATE	UserCollectionsUpdateException
400	USER_GROUP_ALREADY_HAS_HIGHER_PERMISSION_LEVEL	UserGroupAlreadyHasHigherPermissionLevelException
400	USER_MUST_HAVE_VIEW_ACCESS_FIRST	UserMustHaveViewAccessFirstException
400	USER_TYPE_CHANGED	UserTypeCannotBeUpdated
400	VIEW_ALLOW_SAVED_QUERIES_CAN_NOT_BE_ENABLED	ViewAllowSavedQueriesCanNotBeEnabledException
400	VIEW_ALLOW_USER_COLLECTIONS_CAN_NOT_BE_ENABLED	ViewAllowUserCollectionsCanNotBeEnabledException
400	VIEW_CONFIGURATIONS_MUST_BE_ENABLED_WITH_FILTER_FIRST	ViewConfigurationMustBeEnabledWithFilterFirst
400	VIEW_CREATE_DRAFT_DATA_ENTRY_NOT_ALLOWED	ViewCreateDraftDataEntryNotAllowedException
400	VIEW_INVALID_PERMISSIONS	ViewInvalidPermissionsException
400	VIEW_MUST_BE_BASED_ON_QUERY_TO_ENABLE_OVERRIDE_SOURCE_FOR_STATISTIC_QUERY	ViewMustBeBasedOnQueryToEnableOverrideSourceForStatisticQuery
400	VIEW_NOT_PUBLISHED	ViewNotPublishedException
400	VIEW_URL_TOKEN_LENGTH	ViewUrlTokenLengthException
400	VIEW_URL_TOKEN_REQUIRED	ViewUrlTokenRequiredException
400	VIEW_URL_TOKEN_USED_BEFORE	ViewUrlTokenUsedBeforeException
400	VIEW_URLS_ITEMS_EMPTY	ViewUrlsItemsEmptyException
400	VIEW_USER_COLLECTIONS_CAN_NOT_BE_ENABLED	ViewUserCollectionsCanNotBeEnabledException
400	VIRTUAL_ENTITY_OPERATION	VirtualEntityNotAllowedOperation
400	VISIBILITY_CUSTOM_SETTINGS_ENABLED	VisibilityCustomSettingsEnabledException
400	WORKFLOW_CAN_NOT_MOVE	CanNotMoveFromLastStepException

الحالة	error.code	صنف الاستثناء المُنشئ
400	WORKFLOW_CAN_NOT_RESET	CanNotResetStepException
400	WORKFLOW_CAN_NOT_RETURN	CanNotReturnStepException
400	WORKFLOW_NORMALIZED_NAME_DUPLICATE	WorkflowNormalizedNameDuplicateException
400	WORKFLOW_OWNER_COLLECTION_NOT_EXIST	WorkFlowOwnerCollectionNotFound
400	WORKFLOW_OWNER_GROUP_NOT_EXIST	WorkFlowOwnerGroupNotFound
400	WORKFLOW_RESET_REQUIRED	WorkflowResetRequired
400	WORKFLOW_STEP_NORMALIZED_NAME_DUPLICATE	WorkflowStepNormalizedNameDuplicateException
400	WRONG_DATA_TYPE	ValidationValueWrongTypeException
401	DASHBOARD_CONFIGURATION_NOT_ENABLED	DashboardConfigurationNotEnabledException
401	USER_NOT_AUTHENTICATED	Since 1.8 UserNotAuthenticatedException
403	ACCESS_OBJECT_NO_AUDIENCE	Since 1.10 AccessObjectNoAudienceException
403	ACCESS_REQUEST_FORBIDDEN	ForbiddenToAccessRequestException
403	ARCHIVED_ENTRY_NOT_ACCESSIBLE_THROUGH_VIEW	ArchivedEntryNotAccessibleThroughViewException
403	BATCH_CREATE_ENTRIES_QUICK_SAVE_NOT_ENABLED	BatchCreateEntiresQuickSaveNotEnabledException
403	BATCH_CREATE_ENTRIES_RELATION_NOT_ALLOWED	BatchCreateEntriesRelationNotAllowedException
403	CANNOT_EXPORT_ENTITY	ForbiddenToExportEntity
403	CANNOT_EXPORT_MODULE	ForbiddenToExportModule
403	COUNTER_VIEW_NOT_MATCHED	CounterViewNotMatchedWithMiddlewareException
403	EDIT_MODULE_DASHBOARD_NOT_ALLOWED	EditModuleDashboardNotAllowedException
403	ENTITY_CAP_EXCEEDED	Since 1.10 EntityCapExceededException
403	ENTITY_DASHBOARD_EDIT_NOT_ALLOWED	EntityDashboardEditNotAllowedException
403	ENTRY_CAP_EXCEEDED	Since 1.10 EntryCapExceededException
403	ENTRY_NOT_ACCESSIBLE_THROUGH_VIEW	EntryNotAccessibleThroughViewException
403	ENTRY_OPERATION_NOT_ALLOWED	EntryOperationNotAllowedException
403	ENTRY_READ_FORBIDDEN	ForbiddenToReadEntryException
403	INVALID_ENTRY_NOT_ACCESSIBLE_THROUGH_VIEW	InvalidEntryNotAccessibleThroughViewException
403	INVALID_REQUIREMENTS_ENTRY_NOT_ACCESSIBLE_THROUGH_VIEW	InvalidRequirementsEntryNotAccessibleThroughViewException
403	LICENSE_ADMIN_FEATURE_REQUIRED	LicenseAdminFeatureRequiredException
403	OBSOLETE_ENTRY_NOT_ACCESSIBLE_THROUGH_VIEW	ObsoleteEntryNotAccessibleThroughViewException
403	PERMISSION_GRANT_EXCEEDS_CALLER_ACCESS	Since 1.8 PermissionGrantExceedsCallerAccessException
403	RESOLVE_VALUE_PATH_NOT_STORED	Since 1.8 ResolveValuePathNotStoredException
403	SPACE_CAP_EXCEEDED	Since 1.10 SpaceCapExceededException
403	SYSTEM_BUTTON_IS_DISABLED	SystemButtonIsDisabled
403	SYSTEM_BUTTON_NOT_UI_CUSTOM_ACTION	SystemButtonNotUiCustomAction
403	SYSTEM_DASHBOARD_EDIT_NOT_ALLOWED	SystemDashboardEditNotAllowedException
403	VIEW_ACCESS_DENIED	ViewAccessNotAuthorizedException
403	VIEW_DASHBOARD_CONFIGURATION_NOT_ENABLED	ViewDashboardConfigurationNotEnabledException

الحالة	error.code	صنف الاستثناء المُنشئ
403	WORKFOLOW_ADD_ADHOC	UserCannotAddAdhocException
404	ACCESS_OBJECT_NOT_EXIST	AccessObjectNotExistException
404	ACTION_NOT_EXIST	ActionNotExistException
404	BLOCK_NOT_EXIST	BlockNotExistException
404	CATEGORY_NOT_EXIST	CategoryNotExistException
404	COLLECTION_NOT_EXIST	CollectionNotExistValidationException
404	COUNTER_NOT_FOUND	CounterNotFoundException
404	DATA_COLLECTION_NOT_EXIST	DataCollectionNotFoundException
404	DATA_QUERY_NOT_EXIST	DataQueryNotFound
404	DATA_VIEW_NOT_EXIST	DataViewNotExistException
404	ENTITIES_NOT_EXIST	EntitiesNotExistException
404	ENTITY_DATA_SOURCE_NOT_EXIST	EntityDataSourcesNotExistsException
404	ENTITY_FIELD_NOT_EXIST	EntityFieldNotExistException
404	ENTITY_NOT_EXIST	EntityNotExistException
404	ENTITY_NOT_EXIST_VALIDATION	EntityNotExistValidationException
404	ENTITY_NOT_FOUND	DataSourceEntityNotExist
404	ENTITY_NOT_FOUND	EntitiesNotFound
404	ENTRIES_SOURCE_NOT_FOUND	EntriesSourceNotFoundException
404	ENTRY_NOT_EXIST	EntryNotExistException
404	FALLBACK_OWNER_REQUIRED	FallbackOwnerRequiredException
404	FIELD_SOURCE_NOT_FOUND	FieldSourceNotFoundException
404	FILTER_FIRST_CUSTOM_QUERY_NOT_FOUND	FilterFirstCustomQueryNotFoundException
404	GROUP_NOT_EXIST	GroupNotExistValidationException
404	INVALID_DASHBOARD_ID	InvalidDashboardId
404	INVALID_MERGE_STEP	MergeStepValidationException
404	INVALID_MODULE_ID	InvalidModuleId
404	ITEM_NOT_EXIST	DataNotExistException
404	JOB_STATUS_NOT_EXIST	JobStatusNotExistException
404	LOCKED_OBJECT_NOT_FOUND	LockedObjectNotFound
404	MERGE_STEP_ID_REQUIRED	MergeStepRequiredException
404	MODULE_NOT_EXIST	ModuleNotExistException
404	ORG_UNIT_NOT_EXIST	OrgUnitNotExistException
404	OWNER_NOT_EXIST	OwnerNotExistValidationException
404	QUERY_NOT_FOUND	DataSourceQueryNotExistValidationException Since 1.10 (unknown query saves naming an)
404	RELATION_NOT_EXIST	RelationNotExistException
404	STATISTIC_QUERY_NOT_EXIST	StatisticQueryNotFoundException
404	SUB_WORKFLOW_ID_NOT_FOUND	SubWorkflowIdNotFoundException

الحالة	error.code	صنف الاستثناء المُنشئ
404	SUB_WORKFLOW_ID_REQUIRED	SubWorkflowIdRequiredException
404	SYSTEM_BUTTON_NOT_EXIST	SystemButtonNotFound
404	TAG_GROUP_NOT_FOUND	TagGroupNotFoundException
404	TEMPLATE_NOT_EXIST	TemplateNotExists
404	TRIGGER_NOT_EXIST	TriggerNotExistException
404	USER_NOT_EXIST	UserNotExistException
404	WORKFLOW_ACTION_NOT_EXIST	WorkFlowActionNotExistException
404	WORKFLOW_NOT_EXIST	WorkFlowNotExistException
404	WORKFLOW_STEP_NOT_EXIST	WorkflowStepNotExistException
409	LOCKED_OBJECT	LockedObjectException
409	LOCKED_OBJECT_ALREADY_OWNED	LockedObjectAlreadyOwned
409	NOT_ALLOWED_TO_LOCK_OBJECT	NotAllowedToLockObject
409	UPDATE_CONFLICT	UpdateIdentifierConflictException
422	CURRENT_STEP_ALREADY_OWNED	Since 1.8 CurrentStepAlreadyOwnedException
422	ENTITY_ARCHIVE	Since 1.10 (500 was) EntityArchiveException
422	ENTRY_HAS_INCOMING_REFERENCES	EntryHasIncomingReferencesException
422	ENTRY_NOT_ARCHIVED	EntryNotArchivedException
422	ENTRY_NOT_ARCHIVED_FOR_DELETION	EntryNotArchivedForDeletionException
422	ENTRY_NOT_OBSOLETE	EntryNotObsoleteException
422	OBSOLETED_ENTRY	ObsoleteEntryException
422	PERMANENT_DELETE_BATCH_LIMIT_EXCEEDED	PermanentDeleteBatchLimitExceededException
422	REPORT_PERMISSION_NOT_SELECTED	ReportPermissionsNotSelectedException
422	SYSTEM_SETTINGS_DUPLICATE_PHONE_NUMBER	Since 1.10 SystemSettingsDuplicatePhoneNumberException
422	SYSTEM_SETTINGS_FIELD_NOT_SECRET	Since 1.10 SystemSettingsFieldNotSecretException
422	SYSTEM_SETTINGS_SPACE_MISMATCH	Since 1.10 SystemSettingsSpaceMismatchException
422	SYSTEM_SETTINGS_TENANT_DEFAULT_ARCHIVE	Since 1.10 SystemSettingsTenantDefaultArchiveException
422	VALIDATION_FAILED	InvalidEnumValue
422	WORKFLOW_ACTION_EXECUTION_NOT_ALLOWED	Since 1.8 (500 was) WorkFlowActionExecutionNotAllowed
500	ACTION_DUPLICATE	ActionDuplicateException
500	ACTION_NOT_ALLOWED	ActionNotAllowed
500	CHANGE_MODE_NOT_IMPLEMENTED	ChangeModeNotImplementedException
500	CIRCULAR_DEPENDENCY_DETECTED	CircularDependencyException
500	DASHBOARD_ACCESS_NOT_ALLOWED	DashboardAccessNotAllowedException
500	DASHBOARD_STATISTIC_QUERY_NOT_IN_MODULE	DashboardStatisticQueryNotInModule
500	DATA_COLLECTION_DUPLICATE	DataCollectionDuplicateException
500	DATA_COLLECTION_NORMALIZED_NAME_DUPLICATE	DataCollectionNormalizedNameDuplicateException
500	DATA_MODULE_ICON_REQUIRED	DataModuleIconIsRequired
500	DATA_QUERY_NORMALIZED_NAME_DUPLICATE	DataQueryNormalizedNameDuplicateException

الحالة	error.code	صنف الاستثناء المُنشئ
500	DATA_QUERY_NOT_BELONG_TO_ENTITY	DataQueryNotBelongToEntity
500	DATA_RELATION_NORMALIZED_NAME_DUPLICATE	DataRelationNormalizedNameDuplicateException
500	DATA_SOURCE_STORE_REQUIRED	DataSourceStoreRequireException
500	DATA_TYPE_NOT_SUPPORTED	DataTypeNotSupportedException
500	DATA_VIEW_AFTER_SAVE_NOT_FOUND	DataViewAfterSaveNotFoundException
500	DATA_VIEW_AFTER_SAVE_URL_EMPTY	DataViewAfterSaveUrlEmptyException
500	DATA_VIEW_AFTER_SAVE_URL_HTTP_METHOD_REQUIRED	DataViewAfterSaveUrlHttpMethodRequiredException
500	DATA_VIEW_EDIT_CONFIGURATION_NOT_SUPPORTED_FOR_VIRTUAL_AGGREGATOR_ENTITY	DataViewEditConfigurationNotSupportedForVirtualAggregatorEntity
500	DATA_VIEW_EDIT_CONFIGURATION_NOT_SUPPORTED_FOR_VIRTUAL_PROXY_ENTITY	DataViewEditConfigurationNotSupportedForVirtualProxyEntity
500	DATA_VIEW_INVALID_ENTITY_ID	ViewNotBelongToEntity
500	DATA_VIEW_NORMALIZED_NAME_DUPLICATE	DataViewNormalizedNameDuplicateException
500	DATA_VIEW_SAVE_MESSAGE	DataViewAfterSaveMessageEmptyExcpetion
500	DUPLICATE_ITEM	DuplicateItemException
500	ENTITY_ARCHIVED	EntityArchivedException
500	ENTITY_BELONG_TO_MODULE	EntityBelongToModule
500	ENTITY_BLOCK_OPTIONS_NOT_SUPPORTED	EntityBlockOptionsException
500	ENTITY_DUPLICATE	EntityDuplicateException
500	ENTITY_FIELD_DUPLICATE	EntityFieldDuplicateException
500	ENTITY_FIELD_NOT_CATEGORY	EntityFieldNotCategoryFieldException
500	ENTITY_FIELD_NOT_REFERENCE	EntityFieldNotReferenceFieldException
500	ENTITY_FIELD_OPTIONS_ERROR	EntityFieldOptionsException
500	ENTRIES_SOURCE_NOT_BELONG_TO_SOURCE	EntriesSourceNotBelongToSourceException
500	ENTRY_ARCHIVED	EntryArchivedException
500	ENTRY_NOT_ACTIVATE	EntryNotActiveException
500	ENTRY_NOT_IN_WORKFLOW	EntryNotInWorkflow
500	EXPORT_ENTRIES_ERROR	DataExportException
500	EXPORT_ENTRIES_ERROR	ExportEntriesErrorException
500	EXPRESSION_INVALID_RETURN_TYPE	ExpressionExecutionReturnedInvalidTypeException
500	EXTRA_ENTRY_VALUE	ExtraEntryValueException
500	FALLBACK_OWNER_LAST_ITEM_OWNER_TYPE	FallbackOwnerLastItemOwnerTypeException
500	FIELD_SOURCE_REQUIRED	FieldSourceRequiredException
500	FILE_KIND_NOT_SUPPORTED	UnsupportedImportFileKindException
500	FILE_NOT_FOUND	FileNotFoundException
500	IMPORT_FILE_IS_INVALID	InvalidImportFileException
500	INVALID_CURRENT_WF_COLLECTION_OWNER	InvalidWorkflowOwnerCollection
500	INVALID_CURRENT_WF_GROUP_OWNER	InvalidWorkflowOwnerGroup
500	INVALID_CURRENT_WF_ORG_UNIT_OWNER	InvalidWorkflowOwnerOrgunit

الحالة	error.code	صنف الاستثناء المُنشئ
500	INVALID_CURRENT_WF_OWNER	InvalidWorkflowOwner
500	INVALID_DASHBOARD	DashboardValidationException
500	INVALID_DASHBOARD_GROUP_ID	InvalidDashboardGroup
500	INVALID_DATA_SOURCE_ACTION	DataSourceActionNotSupported
500	INVALID_ELEMENT_ID	InvalidElementID
500	INVALID_ENTRY_STEP_TYPE	InvalidEntryStepTypeException
500	INVALID_EXIT_STEP_TYPE	InvalidExitStepException
500	INVALID_ID	InvalidIdException
500	INVALID_ITEM	InvalidItemException
500	INVALID_ITEM_VALUE	InvalidItemValueException
500	INVALID_PROPERTY_VALUE	InvalidPropertyValueException
500	INVALID_QUERY_CRITERIA	QueryCriteriaNotSupported
500	INVALID_STEP_OWNER_TYPE_DATA_COLLECTION	InvalidStepOwnerTypeDataCollectionException
500	INVALID_STEP_OWNER_TYPE_GROUP	InvalidStepOwnerTypeGroupException
500	INVALID_STEP_OWNER_TYPE_USER	InvalidStepOwnerTypeUserException
500	INVALID_STRING_ITEMS	InvalidStringItemsException
500	INVALID_TREE	InvalidTreeException
500	INVALID_TREE_NODE	InvalidTreeNodeException
500	INVALID_TREE_NODE_PARENT	InvalidTreeNodeParentException
500	INVALID_TREE_NODE_VALUE	InvalidTreeNodeValueException
500	INVALID_VIEW_DEFINITION	DataViewExcpetion
500	INVALID_WORKFLOW	InvalidWorkFlowException
500	INVALID_WORKFLOW_OWNER_SOURCE	InvalidWorkflowOwnerSourceExption
500	IS_DEFAULT_DUPLICATE	DataViewIsDefaultException
500	MISSING_PROPERTY	MissingPropertyException
500	MISSING_REQUIRED_STEPS	MissingStepsException
500	MODULE_DUPLICATE	ModuleDuplicateException
500	MODULE_IS_IMPLICIT	ImplicitModuleException
500	NOT_SUPPORTED	DataNotSupportedException
500	PERMISSION_SCOPE_NOT_SUPPORTED	PermissionScopeNotSupportedException
500	PREVIOUS_OWNER_STEP_IS_EXIT_STEP	PreviousOwnerStepIsExitStepException
500	PREVIOUS_STEP_NOT_FOUND	PreviousStepNotFoundException
500	QUERY_EXPRESSION_ERROR	ExpressionExecutionException
500	QUERY_NOT_FOUND	DataSourceQueryNotExist (reads of a stored reference)
500	REDIRECT_TO_ENTRY_ACTION_MUST_ENABLE_SHOW_ENTRY	DataViewAfterSaveRedirectToEntryViewActionMustEnableEntriesConfiguration
500	RELATIVE_OPTIONS_INVALID_STEP_OWNER_TYPE	RelativeOptionsOwnerTypeException
500	RELATIVE_STEP_NOT_EXECUTED	RelativeStepNotExecuted

الحالة	error.code	صنف الاستثناء المُنشئ
500	REQUIRED_FIELD	DataSourceQueryRequireException
500	STORE_NOT_FOUND	DataSourceStoreNotExist
500	TRIGGER_DUPLICATE	TriggerDuplicateException
500	UNDEFINED_ITEM	UndefinedObject
500	UNDEFINED_NEXT_WORKFLOW_STEP_ACTION	UndefinedNextWorkFlowStepActionException
500	UNDEFINED_WORKFLOW	UndefinedWorkFlowException
500	UNDEFINED_WORKFLOW_ACTION	UndefinedWorkFlowActionException
500	UNDEFINED_WORKFLOW_ACTION_TARGET_STEPS	UndefinedWorkFlowActionTargetStepsException
500	UNDEFINED_WORKFLOW_STEP_ACTIONS	UndefinedWorkFlowStepActionsException
500	UNDEFINED_WORKFLOW_STEPS	UndefinedWorkFlowStepsException
500	VIEW_MUST_BE_DRAFT_FIRST_CREATION	DataViewFirstPublishException
500	WF_CAN_NOT_CHANGE_OWNER	CanNotChangeOwnerException
500	WORKFLOW_ACTION_COMMENT_REQUIRED	WorkFlowActionCommentRequired
500	WORKFLOW_STEP_CAN_NOT_MOVE	CanNotMoveStepException
501	BACKUP_NOT_SUPPORTED_IN_SINGLE_MODE	Since 1.10 BackupNotSupportedException
503	BACKUP_SERVICE_UNAVAILABLE	retryable; the facade also relays the backup engine's) BackupApiException Since 1.10 (own codes with their status
503	STORAGE_SERVICE_UNAVAILABLE	retryable; also on saves that add a) StorageServiceUnavailableException (Since 1.10 file

الأتمتة — reaction-service

تضمّ هذه الخدمة 51 صنفًا مُنمَّطًا. ويخصّ رمزا التحقّق EXECUTE_CUSTOM_ACTION_INVALID_CONFIGS و EXECUTE_CUSTOM_ACTION_INCOMPATIBLE_TARGET تفاعل تنفيذ الإجراء المخصّص *(Since 1.8)*. أمّا الرموز المشتركة مع خدمة البيانات، فتحمل عدّةٌ منها حالة مختلفة هنا (ENTITY_FIELD_NOT_EXIST و ORG_UNIT_NOT_EXIST و STORE_NOT_FOUND حالتها 500 في هذه الخدمة).

الحالة	error.code	صنف الاستثناء المُنشئ
400	BLOCK_FIELD_NOT_EXIST	BlockFieldNotFoundException
400	COLLECTION_NOT_EXIST	CollectionNotExistValidationException
400	CUSTOM_EVENT_INVALID_CONFIGS	CustomEventInvalidConfigsException
400	ENTRIES_CONFIGURATIONS_MUST_BE_ENABLED	EntriesConfigurationMustBeEnabled
400	ENTRY_NOT_IN_WORKFLOW	EntryNotInWorkflowException
400	EXECUTE_CUSTOM_ACTION_INCOMPATIBLE_TARGET	ExecuteCustomActionIncompatibleTargetException
400	EXECUTE_CUSTOM_ACTION_INVALID_CONFIGS	ExecuteCustomActionInvalidConfigsException
400	FIELD_NOT_REPEATABLE	BlockFieldNotRepeatableException
400	FILE_EXPIRED	Since 1.10 FileExpiredException
400	GROUP_NOT_EXIST	GroupNotExistValidationException
400	IMPLICIT_CREATE_RESTRICTION	ImplicitReactionCreateException
400	IMPLICIT_DELETE_RESTRICTION	ImplicitReactionDeleteException

الحالة	error.code	صنف الاستثناء المُنشئ
400	IMPLICIT_UPDATE_RESTRICTION	ImplicitReactionUpdateException
400	INVALID_FIELDS_SOURCE	FieldSourceWrongTypeException
400	INVALID_QUERY_EXPRESSION	DataQueryExpressionValueNotStringException
400	INVALID_QUERY_EXPRESSION_TYPE	DataQueryExpressionTypeValidationException
400	INVALID_TOKENS_MAP	InvalidTokensMapException
400	MUST_AUTOMATE_AT_LEAST_ONE_STEP	MustAutomateAtLeastOneStepException
400	QUERY_ENGINE_EXPRESSION_ERROR	ExpressionParsingException
400	RESERVED_HEADER_NAME	Since 1.10 ReservedHeaderNameException
400	TRIGGER_TIMER_OPTIONS_CONFLICT	TriggerTimerOptionsConflict
400	TRIGGERS_CAN_NOT_AUTOMATE_LAST_STEP	CanNotAutomateLastStepException
400	TRIGGERS_LINKED_TO_NON_EXISTING_WORKFLOW	TriggersLinkedToNonExistingWorkflow
400	TRIGGERS_LINKED_TO_NON_EXISTING_WORKFLOW_STEP	TriggerLinkedToNonExistingWorkflowStep
400	TRIGGERS_LINKED_TO_NON_EXISTING_WORKFLOW_STEP_ACTION	TriggerLinkedToNonExistingWorkflowStepAction
400	UNKNOWN_OPERATOR_ERROR	UnknownOperatorExpression
400	UNSUPPORTED_FIELD_ERROR	UnsupportedFieldExpression
400	UPDATE_BLOCK_ITEM_ACTION_ON_FAILURE	UpdateBlockItemActionOnFailureException
400	UPDATE_ENTRY_MODE_NOT_SUPPORTED	InvalidUpdateEntryModeException
400	UPDATED_ENTRY_EVENT_CONFIG_INVALID_FIELD	UpdatedEntryEventConfigInvalidFieldException
400	VIEW_NOT_PUBLISHED	ViewNotPublishedException
403	REACTION_AUTHORING_FORBIDDEN	Since 1.8 ReactionAuthoringForbiddenException
404	ENTITY_NOT_EXIST	EntityNotExistException
404	EVENT_SCHEMA_NOT_FOUND	EventSchemaNotFoundException
404	INTERNAL_ENTITY_NOT_FOUND	InternalEntityNotFoundException
404	REACTION_NOT_FOUND	ReactionNotFoundException
404	REACTION_SCHEMA_NOT_FOUND	ReactionSchemaFoundException
404	REFERENCE_FIELD_NOT_EXIST	ReferenceFieldNotFoundException
404	TRIGGER_CONTEXT_ENTITY_NOT_FOUND	TriggerContextEntityNotFoundException
404	WORKFLOW_ACTION_NOT_EXIST	WorkFlowActionNotExistException
404	WORKFLOW_NOT_EXIST	WorkFlowNotExistException
404	WORKFLOW_STEP_NOT_EXIST	WorkflowStepNotExistException
409	ANNOUNCEMENT_NAME_ALREADY_EXISTS	AnnouncementNameAlreadyExistsException
409	REACTION_NORMALIZED_NAME_DUPLICATE	ReactionNormalizedNameDuplicateException
409	UPDATE_CONFLICT	UpdateIdentifierConflictException
409	whatsapp_not_configured	Since 1.10 WhatsAppNotConfiguredException
422	VALIDATION_FAILED	InvalidEnumValue
500	DATA_NOT_SUPPORTED	DataNotSupportedException
500	ENTITY_ARCHIVED	EntityArchivedException
500	ENTITY_FIELD_NOT_EXIST	EntityFieldNotExistException

الحالة	error.code	صنف الاستثناء المُنشئ
500	EXPRESSION_INVALID_RETURN_TYPE	ExpressionExecutionReturnedInvalidTypeException
500	FIELD_SOURCE_NOT_FOUND	FieldSourceNotFoundException
500	ORG_UNIT_NOT_EXIST	OrgUnitNotExistException
500	QUERY_EXPRESSION_ERROR	ExpressionExecutionException
500	STORE_NOT_FOUND	DataSourceStoreNotExist
502	whatsapp_provider_error	Since 1.10 WhatsAppProviderException
503	STORAGE_SERVICE_UNAVAILABLE	Since 1.10 (retryable) StorageServiceUnavailableException

الخدمات الأصغر

تعرّف خدمات سير العمل والتعليقات والشريط الزمني والإشعارات والتدقيق عددًا قليلًا من الأصناف المُنمَّطة وفق نمط العائلات نفسه. أمّا خدمات الوظائف والسجلّات والتكاملات فلا تعرّف إلّا العائلات العامّة؛ لذا تَرِد أخطاؤها بصفتها أخطاء إطار عمل أو برموز مضمَّنة. ولا توجّه البوابة الطلبات إلى خدمة التدقيق (لا يسرد `api-gateway/config/kong.yml` مسارًا للتدقيق)؛ فلا تظهر أخطاؤها إلّا عند تمريرها عبر خدمات أخرى.

`workflow-service`:

الحالة	error.code	صنف الاستثناء المُنشئ
400	WORKFLOW_BULK_EXECUTION_NOT_ALLOWED	WorkflowStepsNotSameException
400	WORKFLOW_STEP_OWNER_CHANGE_TYPE_NOT_SUPPORTED	WorkflowStepOwnerChangeTypeNotSupported
403	FORBIDDEN_WORKFLOW_ADMIN_ACTION	ForbiddenWorkflowAdminActionException
404	DATA_ENTITY_NOT_EXIST	DataEntityNotExistException
404	DATA_VIEW_NOT_EXIST	DataViewNotExistException
404	EVENT_SCHEMA_NOT_FOUND	EventSchemaNotFoundException
500	EXPRESSION_INVALID_RETURN_TYPE	ExpressionExecutionReturnedInvalidTypeException
500	ITEM_ARCHIVED	EntityArchivedException
500	QUERY_EXPRESSION_ERROR	ExpressionExecutionException
500	REACTION_SCHEMA_NOT_FOUND	ReactionSchemaFoundException
500	UNDEFINED_ITEM	UndefinedWorkFlowException

`comments-service`:

الحالة	error.code	صنف الاستثناء المُنشئ
400	COMMENT_ATTACHMENTS_EXTENSION_NOT_ALLOWED	CommentAttachmentExtensionNotAllowed
400	COMMENT_ATTACHMENTS_NOT_ALLOWED	CommentAttachmentNotAllowed
400	COMMENT_ATTACHMENTS_SIZE_NOT_ALLOWED	CommentAttachmentSizeIsNotAllowed
400	COMMENT_TEXT_NOT_ALLOWED	CommentTextNotAllowed
404	DATA_ENTITY_NOT_EXIST	DataEntityNotExistException
404	DATA_VIEW_NOT_EXIST	DataViewNotExistException
500	COMMENT_NOT_FOUND	CommentNotFound
500	COMMENT_UPDATE_NOT_ALLOWED	CommentUpdateNotAllowed

الحالة	error.code	صنف الاستثناء المُنشئ
500	INVALID_COMMENT_AUDIENCE	InvalidCommentAudience

timeline-service:

الحالة	error.code	صنف الاستثناء المُنشئ
400	INVALID_PUBLISH_DATE	InvalidPublishDateException
400	INVALID_UNPUBLISH_DATE	InvalidUnpublishDateException
400	POST_MANUAL_PIN_SAME_PIN_STATE	DuplicatePinnedStatusPostException
400	POST_MANUAL_PUBLISH_IS_NOT_ALLOWED	PublishNotAllowedException
400	POST_MANUAL_UNPUBLISH_IS_NOT_ALLOWED	UnpublishNotAllowedException
400	TIMELINE_PUBLISH_DATE_CONFLICT	TimelinePublishDateConflictException
400	UNABLE_PINNED_MANUALLY	UnablePinnedPostManuallyException
400	UNABLE_SCHEDULING_ACTION	UnableScheduleManualPublishUnpublishPostException
400	UNABLE_UNPINNED_MANUALLY	UnableUnPinnedManuallyPostException
404	DATA_VIEW_NOT_EXIST	DataViewNotExistException

notification-api (ملاحظة: تُرسَل قيمة الثابت NOTIFICATION_MARK_AS_READ_EXCEPTION عبر الشبكة بصيغة NOTIFICATION_MARK_AS_READ_ERROR):

الحالة	error.code	صنف الاستثناء المُنشئ
500	NOTIFICATION_MARK_AS_READ_ERROR	MarkAsReadRequestEmptyException
500	NOTIFICATION_TEMPLATE_NOT_EXIST	NotificationTemplateNotExist

audit-service:

الحالة	error.code	صنف الاستثناء المُنشئ
404	DATA_ENTITY_NOT_EXIST	DataEntityNotExistException
404	DATA_VIEW_NOT_EXIST	DataViewNotExistException
500	ITEM_ARCHIVED	EntityArchivedException

التحزيم والاعتماديات

لا تعرّف packaging-service و dependency-service **أيّ استثناء منصّة مُنمَّط** (فقاموسا الأخطاء فيهما فارغان)، بل تُطلقان أخطاء إطار العمل HttpErrors مباشرةً من وحدات التحكّم، ولذلك تكون قيمة error.code في الغالب اسم صنف من الإطار (BadRequestError و ForbiddenError و NotFoundError و ConflictError و PayloadTooLargeError)، ويَرِد الشرح المقروء في نصّ الرسالة داخل result. وتُستثنى من ذلك مسارات السوق والمتجر؛ إذ تُلحق رموزها الخاصّة بخطأ الإطار (مثل 422 PUBLISHER_NAME_INVALID و 502 MARKETPLACE_REFUSED)، وهي موثّقة في صفحات التحزيم. ومن Since 1.10 يُرفض مسبقًا تثبيتُ الحزمة الذي سيتجاوز حدّ مُدخلات القاطن بالرمز 403 ENTRY_CAP_EXCEEDED. عامِل ذلك النصّ بوصفه نصًّا تشخيصيًّا للعرض؛ وحدّد فرع المعالجة بحسب الحالة ونقطة النهاية التي استدعيتها:

الحالة	الخدمة	الموقف
400	كلتاهما	معاملات غير صالحة: نصّ بحث فارغ، أو limit خارج النطاق (يقيّده بحث الاعتماديات عند 1000)، أو حمولة تجاوز مشوَّهة، أو دمج غير صالح لمرجع قيم.
403	التحزيم	استُطلعت جلسة تصدير/استيراد من مساحة غير المساحة المالكة لها.

الحالة	الخدمة	الموقف
404	كلتاهما	جلسة أو ملفّ تعيين أو معرّف مهمّة إعادة بناء غير معروف.
409	التحزيم	آلة حالات جلسة الاستيراد: تنفيذ جلسة ليست في حالة `validated`، أو إعادة محاولة جلسة ليست في حالة `failed` / `completed`، أو جلسة عُدِّلت بالتزامن.
409	الاعتماديات	تجري إعادة بناء كاملة بالفعل – تذكر الرسالة معرّف المهمّة الجارية لاستطلاع حالتها بدلًا من بدء مهمّة أخرى.
413	التحزيم	حمولة تجاوز في الاستيراد تتعدّى الحدّ الأقصى المُعدّ للبايتات.

حالات `409` في التحزيم والاعتماديات هي الحالات الوحيدة التي صُمِّمت إعادة المحاولة فيها لتكون *بعد اتّباع تعليمات الرسالة نفسها*: أعد قراءة حالة الجلسة أو المهمّة، ثم تصرّف بناءً على الحالة التي تجدها.

المصادقة والتحقّق الإضافي – `users-service`

عند الإخفاق، تُعيد نقاط نهاية الرموز وOIDC والحسابات كائنات خطأ OAuth 2.0 القياسية – `invalid_grant` و`invalid_client` وأشباهها، وفق مكتبة `oidc-provider` – فعالجها بمكتبة عميل OAuth لا بمنطق الغلاف. انظر دورة حياة الرمز.

وتُعيد نقاط نهاية كشف الحقول الحسّاسة عبر التحقّق الإضافي (step-up) *(Since 1.8)* الجسمَ `{ "error": "<CODE>" }` مع تعيين صريح لحالة كلّ رمز:

الحالة	الرمز	متى	إعادة المحاولة؟
400	`INVALID_DURATION`	مدّة الكشف المطلوبة ليست من القيم المسموحة.	لا
400	`MFA_NO_METHOD`	لا توجد وسيلة صالحة للتحقّق متعدّد العوامل للحساب.	لا
400	`USER_NOT_FOUND`	تعذّر تحديد الحساب.	لا
400	`NO_PENDING_CHALLENGE`	استُدعي التحقّق دون تحدٍّ معلَّق.	لا
403	`EXTERNAL_USER_FORBIDDEN`	لا يجوز للمستخدمين الخارجيين فتح جلسات تحقّق إضافي.	لا
429	`MFA_RATE_LIMITED`	عدد محاولات التحدّي أكبر من المسموح.	تراجَع ثم نعم
503	`STEP_UP_UNAVAILABLE`	يتعذّر الوصول إلى مخزن جلسات الكشف – مشكلة توافر وليست خطأ عميل.	تراجَع ثم نعم
500	`INTERNAL`	إخفاق غير متوقَّع.	أبلغ

سياسة إعادة المحاولة

نادرًا ما تشير أسس في الاستجابات المرسلة عبر الشبكة إلى قابلية إعادة المحاولة – فحقل `retryable` لا يُضبط إلّا في عدد قليل من حالات `503` – وكما سبق بيانه، فإنّ معظم حالات `500` إخفاقات حتمية في منطق المجال. لذا لا بدّ من كتابة السياسة لدى العميل، استنادًا إلى الحالة والرمز. وفيما يلي السياسة التي يدعمها الفهرس.

ما الذي يُعاد، بحسب الحالة

الحالة	السياسة
`400`، `403`، `404`، `415`، `422`	**لا تُعِد المحاولة دون تغيير أبدًا.** يوجد خطأ في الطلب أو أذونات بيانات الاعتماد أو الهدف؛ وإعادة المحاولة تكرّر الإخفاق نفسه. وتُعالَج حالتان من حالات `400` بإعادة خطوة سابقة لا الطلب نفسه: `FILE_EXPIRED` (ارفع الملف من جديد) و`CONFIRMATION_INVALID` عند الاستعادة (اطلب معاينة جديدة).
`401`	**جدّد مرّة، وأعد مرّة.** انتهاء صلاحية الرمز هو السبب المعتاد: نفّذ مسار التجديد ثم أعد المحاولة مرّة واحدة. أمّا تكرار `401` فيعني أنّ بيانات الاعتماد أُلغيت أو أنّ رمز API غير صالح – توقّف وصعّد.

الحالة	السياسة
`409`	**أعد المزامنة ثم أعد المحاولة عن قصد.** `UPDATE_CONFLICT` : أعد جلب المُدخل وأعد تطبيق تغييرك. الأقفال: انتظر مالكها. جلسات التحزيم وإعادة بناء الاعتماديات: استطلع الحالة أو المهمّة التي تحدّدها الرسالة، ثم تصرّف بناءً على النتيجة. لا تُعِد محاولة `409` باستخدام مؤقّت أبدًا.
`429`	**تراجَع وخفّف.** تسمح البوابة بـ100 طلب/ثانية و10,000 طلب/ساعة *لكلّ خدمة*، محسوبةً عبر **جميع** مستدعي تلك الخدمة (`limit_by:` `service` في `api-gateway/config/kong.yml`) – فقد يكون سبب `429` الحملَ الكلّي، لا عميلك وحده. التزم بترويسة `Retry-After` عند وجودها. وينبغي لمهمّة تحميل مستمرّة يُراد لها البقاء ضمن نافذة الساعة أن تضبط وتيرتها عند نحو 2-3 طلبات في الثانية لكلّ خدمة. أمّا حالات `429` في واجهة النسخ الاحتياطي فسببها حصصٌ يومية لا الحمل: إذ يمتدّ `Retry-After` فيها حتى إعادة ضبط الحصّة، فلا تُعِد المحاولة قبل ذلك `Since 1.10` .
`500`	**أعد المحاولة للرموز ذات طابع البنية التحتية فقط.** حالات `500` عند البوابة في الفهرس (`REDIS_ERROR` ومجموعة `*_REQUEST_FAILED` و…) تستحقّ محاولتين متباعدتين على الأكثر. أمّا استجابة `500` من المنصّة تحمل رمزًا مفهرَسًا فتشير إلى إخفاق حتميّ – لا تُعِد المحاولة؛ بل أبلغ. وتشير استجابة `500` برمز هو اسم صنف JS (`TypeError`) إلى علّة في الخادم – أبلغ عنها مع `traceId` الخاص بها.
`502` ، `503` ، `504`	**عابرة بطبيعتها** – خدمة قيد إعادة التشغيل أو يتعذّر الوصول إليها. أعد المحاولة بتراجُع كامل. ويندرج `STEP_UP_UNAVAILABLE` و `STORAGE_SERVICE_UNAVAILABLE` (`503`) ضمن هذه الفئة صراحةً. ويندرج `Since 1.10` كذلك `TENANT_IN_MAINTENANCE` (`503`)، لكن انتظر على الأقل المدّة المحدّدة في `Retry-After` الخاصّة به – فقد تطول الاستعادة، ويتكرّر الرفض ذاته حتى تنتهي.

ينبغي تصنيف الإخفاقات على مستوى الشبكة بالطريقة نفسها التي تطبّقها المنصّة داخليًّا – فهذا التصنيف وارد في منطق إعادة المحاولة الخاص بناقل رسائلها: الاتصال الذي أخفق *قبل إرسال الطلب* – بسبب رفض الاتصال أو انقطاعه أو DNS – لم يصل إلى الخادم قطّ، وإعادة المحاولة آمنة دائمًا. أمّا **المهلة المنقضية بعد الإرسال فلا**: فقد تكون العملية قد اكتملت. أعد محاولة القراءات بحرّية. أمّا الكتابات، فتحقّق أولًا من اتّساق الحالة – استعلم عن المُدخل الذي حاولت إنشاءه قبل إعادة إنشائه، أو أعد جلب `updateIdentifier` قبل إعادة التحديث – وللكتابات المجمَّعة، استخدم كائنات النتائج لكلّ صف لإعادة إرسال الصفوف التي أخفقت فعلًا فقط (كتابة البيانات).

شكل التراجع

تراجُع أسّي مع عشوائية كاملة (jitter)، وهو الشكل القياسي:

```
delay = random(0, min(cap, base × 2^attempt))
base  = 1s     cap = 60s     max attempts = 5
```

- طبّقه على `429` و `502` / `503` / `504` وإخفاقات الشبكة قبل الإرسال وحالة `500` المتعلّقة بالبنية التحتية ذات المحاولتين.
- عند وجود ترويسة `Retry-After` (حدّ المعدّل، وصيانة القاطن، وواجهة النسخ الاحتياطي)، عامِلها بوصفها حدًّا أدنى للتأخير المحسوب.
- تُعاد محاولة `401` مرّة واحدة بالضبط بعد التجديد – دون حلقة تراجُع.
- `409` تحكمه الأحداث لا المؤقّت: «التأخير» هو الوقت الذي تستغرقه إعادة القراءة أو تحرير القفل.
- قيّد التزامن كما تقيّد المعدّل: فحدود كلّ خدمة مشتركة، وقد يتجاوز عشرة عمّال يتراجعون بالتزامن الحدّ معًا من جديد. والعشوائية هي ما يمنع ذلك.

متى تتوقّف وتصعّد

توقّف عن إعادة المحاولة وافتح بلاغًا عند تحقّق أيٍّ ممّا يلي:

- استُنفدت ميزانية المحاولات واستمرّت الحالة نفسها.
- حمل `500` رمزًا مفهرَسًا (حتميًّا)، أو أيّ رمز هو اسم صنف JS.
- استمرّ `401` بعد نجاح تجديد الرمز.
- ظهر `UNIQUE_VALUE` لقيمة يعتقد نظامك أنها فريدة – فالتعارض في البيانات حقيقي وليس عارضًا.

- ظهر `403` في عملية كانت تنجح سابقًا ببيانات الاعتماد نفسها – تغيّر إعداد أذونات أو معرض لدى الخادم؛ انظر الحدّ الأمني لمعرفة ما يُفرض وأين.

استخدام `traceId` مع الدعم

يحمل كلّ طلب يمرّ عبر البوابة معرّف ارتباط في الترويسة `X-CORRELATION-ID`. مكوّن البوابة `osos-correlation-id` **يحتفظ بالقيمة التي يورّدها العميل** ولا يولّد UUID إلّا عند غياب الترويسة – ثم يعيد المعرّف نفسه في الاستجابة (`echo_downstream: true` في `api-gateway/config/kong.yml`). وداخل المنصّة، تقرأ كلّ خدمة الترويسة دون تمييز بين الأحرف الكبيرة والصغيرة، وتُدرج قيمتها في `error.traceId`.

استخدم هذا السلوك بصورة مقصودة:

- **أرسل معرّفك أنت.** ولّد UUID لكلّ عملية منطقية وأرسله في `X-CORRELATION-ID`؛ لتشترك سجلّاتك وسجلّات المنصّة في مفتاح ربط اخترته أنت – حتى للطلبات التي لم تُنتج أيّ جسم خطأ.
- **سجّل `error.traceId` عند كلّ إخفاق**، ومعه جسم الخطأ كاملًا وحالة HTTP والفعل والمسار (دون بيانات الاعتماد مطلقًا) ومضيف القاطن وطابعًا زمنيًّا بالتوقيت العالمي. هذه الحزمة هي ما يحتاجه الدعم للعثور على الطلب في سجلّات المنصّة.
- عندما تساوي قيمة `traceId` القيمة `NO-TRACE-ID`، فهي القيمة الاحتياطية التي تستخدمها الخدمات لطلب وصل دون الترويسة – فإن رأيتها، فالطلب لم يسلك مسار البوابة المعتاد. والاستثناء الوحيد هو `503 TENANT_IN_MAINTENANCE`، الذي تكتب بوّابة القاطن جسمه مباشرةً، فيحمل دائمًا `NO-TRACE-ID`.
- إن تضمّن الخطأ `serviceName`، فهو يحدّد الجهة التي أنتجته: `"API Gateway"` لحالات الرفض عند الحافّة، واسم خدمة للأخطاء المُمرَّرة من المنصّة. أمّا غيابه فيعني أنّ الخطأ أنتجته الخدمة التي استجابت لطلبك.

بالاستناد إلى الفهرس والسياسة أعلاه، لا يحتاج معالج الأخطاء إلّا إلى ثلاثة مدخلات – الحالة و`error.code` و`traceId` – وكلّها ثابتة وقابلة للقراءة آليًّا وللتتبّع من طرف إلى طرف.

توليد عميل

تقدّم منصّة أسس مساحة HTTP كبيرة ومُنمَّطة، وكتابة نماذج الطلبات والاستجابات يدويًا في مواجهتها معركة خاسرة – فأنماط المجال عميقة والعمليات بالمئات. المسار المدعوم هو **توليد عميل من مواصفة OpenAPI لكل خدمة**. تتناول هذه الصفحة كيفية القيام بذلك على النحو الصحيح: من أين تأتي المواصفات، ولماذا يكون العميل المولَّد من مواصفةٍ *حالية* صحيحًا بينما تكون النسخة الأقدم خاطئة على نحوٍ خفيّ، والحفنة من الاستجابات التي لن ينمذجها العميل المولَّد من تلقاء نفسه أبدًا.

تفترض الصفحة أنّك قرأت استدعاء واجهة HTTP، وهي المرجع لآليات التوجيه والإصدارات والغلاف. لا نعيد هنا سردها؛ بل نشرح ما تعنيه لمولِّد الرماز.

من أين تأتي المواصفات

تُنتج كلّ خدمة LoopBack مستند OpenAPI 3 **الخاصّ بها** من وحدات التحكّم فيها. لا يوجد مستند موحّد عابر للخدمات – فأنت تولّد عميلًا واحدًا لكلّ خدمة وتؤلّف بينها في رمازك أنت. الطريقتان للحصول على المواصفة مشروحتان في قسم «مواصفة OpenAPI» من استدعاء واجهة HTTP: نقطة النهاية أثناء التشغيل (`{basePath}/openapi.json`)، تُخدَم حين يكون `ENABLE_SPEC` بقيمة `'true'`)، ونقطة التصدير دون اتصال (التي تُقلع التطبيق وتكتب المستند إلى ملفّ).

فضّل **لقطة مُودَعة** على استدعاء نقطة نهاية حيّة. فالتوليد من ملفٍّ تتحكّم به يعني أنّ بناءك لا يعتمد على عنقود قيد التشغيل، ويجعل – وهذا مقصد هذه الصفحة – *أصل* المواصفة أمرًا يمكنك فحصه وتثبيته. أمّا المواصفة المسحوبة حيًّا من أيّ عنقودٍ صادف أن ردّ فهي مواصفة لا يمكنك أن ترى عمرها.

المواصفة الحالية تقول الحقيقة

كلّ استجابة JSON مُغلَّفة على هيئة `{success, result}`، دون أيّ شرط إصدار – عدا قراءات `/v3`، الخام من جديد `Since` `1.10` (انظر أدناه). ولا يستطيع مولِّد الرماز أن يستنتج ذلك من مخطّط العملية، لذا توثّق المنصّة الغلاف **داخل المواصفة نفسها**، عبر مُحسِّن مواصفة. والعميل المولَّد من مواصفةٍ تحمل مُخرَج المُحسِّن مولَّدٌ في مواجهة العقد الحقيقي.

ما الذي ينشره مُحسِّن المواصفة

يعزّز المُحسِّن كلّ مستند مولَّد بثلاث طرائق قابلة للقراءة آليًا:

- مفتاح امتدادٍ في جذر المستند باسم `x-osos-response-envelope`، حتى تكتشف الأدوات العقد دون تحليل النثر. وهو يسمّي مخطّط الغلاف واسمَي الخاصّيتين `result` و`error`.
- مخطّطا مكوّناتٍ مُسمّيان، `OsosResponseEnvelope` و`OsosResponseEnvelopeError`**. والأسماء مُنمّطة بفضاء اسمٍ عمدًا بدلًا من إعادة استخدام اسم النموذج المجرّد `ResponseBody`، لأنّ ثلاثة أصنافٍ متمايزة بهذا الاسم موجودة في قاعدة الرماز، ولولا ذلك لتصادمت مساهمة وحدة تحكّم تُسهم باسمها الخاصّ.
- قسم `## Response envelope` يُلحَق بوصف المستند `info.description`، يذكر الثابت نثريًّا لمن يقرأ Swagger UI.

لماذا تُنمَّط العمليات الحمولة لا الغلاف

هنا الدقيقة التي تحسم صحّة عميلك المولَّد. المُحسِّن **لا** يعيد كتابة كلّ عملية لتغليف مخطّط استجابتها بالغلاف. فمخطّط استجابة العملية الناجحة (2xx) لا يزال يصف **حمولة** `result` **– لا الجسم بأكمله**. والمخطّط المنشور `OsosResponseEnvelope` لا تشير إليه أيّ عملية عمدًا؛ فهو موجود ليُوصَف، لا ليُشار إليه.

وهذا التصميم مقصود، وهو ما يبقي التوليد نظيفًا: فالإشارة إلى الغلاف من كلّ عملية كانت ستغيّر نوع النتيجة المولَّد لكلّ دالّة. ولأنّ تهيئات توليد الرماز في المنصّة تتجاهل النماذج غير المُشار إليها، فإنّ ترك مخطّط الغلاف دون إشارة يعني أنّ المُحسِّن يُنتج **صفرًا من التغيير** في مُخرَج العميل المولَّد – إذ تُنمَّط تواقيع الدوالّ المولَّدة الحمولة مباشرةً.

والنتيجة بالنسبة إليك: العميل المولَّد من مواصفةٍ حالية **مُنمَّط بصورة صحيحة على مستوى الحمولة**، لكن على طبقة النقل فيه أن تفكّ غلاف `result` للاستجابات الناجحة (وأن تقرأ `error` عند الإخفاق). وتكون `result` بقيمة `null` حين لا تُعيد العملية حمولة. وسواء كتبتَ فكّ الغلاف يدويًا أم هيّأت مولِّدك ليقوم به، فالعقد الذي تبرمج في مواجهته هو الغلاف الموحّد الواحد – نفسه لكلّ إصدار من كلّ عملية، باستثناء واحد `Since 1.10` هو قراءات `/v3` الموصوفة أدناه.

كيف تعرف أنّ اللقطة تسبق مُحسِّن المواصفة

أصبح الغلاف موحّدًا في إعادة هيكلةٍ أضافت المُحسِّن أيضًا. قبل ذلك كان الغلاف مشروطًا بالإصدار ولم تقل عنه المواصفة شيئًا البتّة. لذا فالعميل المبنيّ من **لقطةٍ تسبق المُحسِّن** غير آمنٍ من وجهين: فهو لا يوثّق أيّ غلاف، وربّما يكون قد وُصِل في مواجهة سلوك «`v3` خام» القديم. (ومنذ `Since 1.10` عاد ذلك السلوك لعمليات **القراءة** على `/v3` وحدها – انظر استثناء قراءات `/v3` – فتحقّق من الشكل الذي تجيب به كل عملية فعلًا بدلًا من الوثوق بأيّ من اللقطتين.)

يمكنك معرفة أصل اللقطة بفحصها بحثًا عن بصمات المُحسِّن. فالمواصفة **الحالية** تحمل الثلاث جميعًا؛ واللقطة **السابقة للمُحسِّن** لا تحمل أيًّا منها:

- المفتاح `x-osos-response-envelope` في جذر المستند؛
- المخطّطان `OsosResponseEnvelope` و`OsosResponseEnvelopeError` تحت `components.schemas`؛
- العنوان `## Response envelope` داخل `info.description`.

فإن غابت عن اللقطة، فلا تولّد في مواجهتها – بل حدّث اللقطة من خلفيّةٍ حالية أوّلًا. فالعميل المولَّد صادقٌ بقدر صدق المواصفة التي جاء منها، واللقطة القديمة تصف أشكال استجابةٍ لم تعد المنصّة تُنتجها.

استثناء قراءات `/v3` `Since 1.10`

على خط 1.10 صار `GET` على مسار `/v3/` يُرسَل **خامًا** من جديد – الحمولة المجرّدة دون {success, result}؛ أما عمليات الكتابة على `/v3` وكل مسارات `v1` / `v2` فتحتفظ بالغلاف (استدعاء واجهة HTTP). ويُدرج نثر المُحسِّن `## Response envelope` "`GET` على `/v3` " بين الحالات غير المُغلَّفة، ومخطّط 2xx الموثَّق لتلك العمليات **هو** الجسم بأكمله. لذا لا يجوز للعميل المولَّد أن يفكّ `result` دون شرط: فإمّا أن يعامل `GET` على `/v3` معاملةً خاصّة، وإمّا – وهو الخيار المتين، وما تفعله البوابة – ألّا يفكّ الغلاف إلا حين يحمل الجسم `success: true`. والصيغة المتينة تغطّي أيضًا نشرات 1.8، التي لا تزال تُغلّف قراءات `/v3` حتى 1.8.21.

لماذا يُستثنى مستند OpenAPI من الغلاف

للغلاف الموحّد استثناءٌ واحدٌ غير بنيويّ: مستند OpenAPI نفسه يُخدَم **خامًا**، دون تغليفٍ أبدًا (مُطابَقٌ على مقطع مسارٍ خاتمٍ `/openapi.json` أو `/openapi.yaml`). وهذا يهمّك مباشرةً، لأنّ مولِّدك يقرأ ذلك المستند.

والسبب آليّ. فتغليف المواصفة في `{success, result}` سينتج مستندًا يُدفَن فيه كائن `paths` العلويّ مستوًى واحدًا تحت `result` – وهو لم يعد مستند OpenAPI صالحًا. عندئذٍ لن يعرض Swagger UI شيئًا، وسيخفق كلّ مولِّدٍ مُهيَّأ لقراءة رابط المواصفة (تستخدم عملاء المنصّة أنفسهم `ng-openapi-gen`) في إيجاد أيّ عملية. والاستثناء مُطابَقٌ على مقطع مسارٍ *خاتم*، حتى إنّ مسار API عاديًّا يحوي `openapi.json` في وسطه يظلّ مُغلَّفًا كأيّ استجابةٍ أخرى.

عمليًا: وجّه مولّدك إلى رابط المواصفة فيتلقّى مستند OpenAPI عاديًّا. ووجّه استدعاء API عاديًّا إلى نقطة نهاية بيانات فيتلقّى الغلاف. ولا يختلط الاثنان أبدًا.

مصدر أضيق: مستند المعرض الواحد `Since 1.10`

إن كان تكاملك يعمل في مواجهة **معرض واحد ثابت** لا في مواجهة مستوى البيانات كلّه، فثمّة الآن مستند ثانٍ أصغر بكثير تولّد منه. وتفعيل **Swagger document** في خيارات عرض المعرض ينشره لذلك المعرض:

```
GET /api/v1/views/{viewId}/openapi.json
```

وهو مستند OpenAPI 3.0 عاديّ – يستهلكه `ng-openapi-gen` أو `openapi-generator` أو أيّ مولِّد آخر يفهم OpenAPI 3 تمامًا كما يستهلك مستند خدمةٍ كاملة. وما يجعله مدخلًا أفضل لتكامل محصور هو ما يُسقطه. فهو يصف **خمس عمليات** بالضبط – قراءة مخطّط الكيان، وسرد المدخلات، وقراءة مدخل واحد، وإنشاء مدخل، وترقيع مدخل – ومعرّف المعرض ومعرّف الكيان **مخبوزان في كل مسار** بدل تركهما موضعين في قالب. والنتيجة عميلٌ سطحه هو سطح التكامل الفعليّ، ولا تحتاج مساراته إلى طبقة استبدال قوالب تكتبها بنفسك. وهو كذلك أثبت: فالعمليات الخمس مأخوذة من وحدات تحكّم حقيقية، فيتغيّر المستند حين تتغيّر تلك العمليات، لا كلّما نبت مسار جديد في ركن غير ذي صلة من `data-service`.

> ⚠ **الاستثناء يطال التسليم لا العقد الذي يصفه المستند**
>
> القسم أعلاه يتناول *هذه* النقطة أيضًا – وهنا يشتدّ أثر التمييز، لأنّ شطريه معًا يقعان في عميلك المُولَّد:
>
> - **الاستجابة غير مُغلَّفة.** فـ `GET /v1/views/{viewId}/openapi.json` يطابق قاعدة اللاحقة الخاتمة نفسها، فيتلقّى مولِّدك مستند OpenAPI مجرّدًا. لا توجّهه إلى `result`؛ فلا `result` هناك.
> - **والمستند الذي يسلّمه إليك يصف الغلاف.** فهو يمرّ عبر المُحسِّن نفسه الذي تمرّ به مواصفة أيّ خدمة، فيحمل `OsosResponseEnvelope` و`OsosResponseEnvelopeError` ومفتاح الجذر `x-osos-response-envelope` وفقرة `## Response envelope` النصّية. وكلّ عملية من العمليات الخمس التي يصفها تردّ في وقت التشغيل داخل `{success, result}`، وعلى طبقة النقل عندك أن تفكّ غلاف `result` تمامًا كما تفعل مع عميل خدمةٍ كاملة.
>
> وعكس هذا الفهم يُنتج محلِّلًا خاطئًا مرّتين: يخفق في قراءة المواصفة، ثم يخفق في قراءة كلّ استجابة مولَّدة عنها.

وقيدان آخران ينبغي التصميم حولهما:

- **تقديم المستند محكوم لكل معرض على حدة.** فمع إطفاء الخيار يتلقّى المستدعي غير المُميَّز `403` – فخطوة البناء التي تجلب المستند تحتاج إلى تفعيل الخيار على ذلك المعرض، وإلى إبقائه مفعَّلًا.
- **عملية مخطّط الكيان تشترط رمزًا حاملًا دائمًا**، حتى على معرض يسمح بالقراءة مجهولة الهوية فيما عداها. فالعميل المُولَّد الذي يفترض سياسة اعتماد واحدة عبر العمليات الخمس سيخفق في تلك الواحدة.

أمّا عقد نقطة النهاية كاملًا – رمازات الحالة، وبادئة البوابة `/v1/v/{viewId}` التي تحملها المسارات الموثَّقة، وممّ يُبنى مخطّط `value` للمدخل – ففي مستند OpenAPI للمعرض الواحد.

أيّ الاستجابات تتجاوز الغلاف

بعض الاستجابات ليست `{success, result}` على الإطلاق، ولن ينمذجها العميل المولَّد – الذي يفترض أجسام JSON. فهذه تتجاوز الغلاف **بنيويًّا**: إذ يكتب المعالِج إلى الاستجابة بنفسه، فتعود مسيرة الإرسال باكرًا (`result === response ||` `response.headersSent`) بدل تغليف أيّ شيء.

- **تنزيلات الملفّات والبيانات الثنائية والدفوق.** تُعيد عمليات تصدير البيانات (XLSX، PDF)، وCSS السمات، وملفّات الترجمة، والخطوط محتوًى ثنائيًّا أو مدفوقًا. عالِجها كتنزيلاتٍ؛ ولا تحاول تحليلها بوصفها JSON. وهي الاستجابات نفسها المشار إليها في العمليات طويلة الأمد.
- **مسارات الصحّة والجاهزية والحياة** (`/health` و `/ready` و `/live`).

في هذه، اخرج من عميلك المولَّد واقرأ استجابة HTTP الخام. فالعميل الذي يفترض أنّ كلّ استجابةٍ هي JSON مُغلَّف سيخفق تحديدًا في العمليات – كتنزيل تصدير – التي يقصدها التكامل أوّلًا.

خدمات بلا مواصفة

لا يشمل التوليد إلّا خدمات LoopBack التي تنشر مواصفة. فثلاث خدماتٍ طرفية تطبيقات Express عادية ولا تُصدِر مستند OpenAPI البتّة: `users-service` و `storage-service` و `scheduler-service` . (ولم تعد `shortener-service` جزءًا من الخلفية – إذ تقدّم مسارَ الروابط المختصرة دالّةٌ مشتركة منفصلة.) وهي تمتلك فيما بينها المصادقة وتخزين الملفّات – وهما أوّل ما يحتاجه عميلٌ خارجيّ – فلا يستطيع العميل المولَّد تغطيتها، وينبغي تكاملها يدويًّا من أدلّتها الخاصّة. حافِظ على خطٍّ واضح في قاعدة رمازك بين العملاء المولَّدين وهذه التكاملات المكتوبة يدويًّا، حتى لا يطمس توليدٌ جديد رمازًا مصونًا يدويًّا.

وأربع خدماتٍ أخرى مبنيّة على LoopBack لكن **بلا تصدير مواصفة دون اتّصال**: `packaging-service` و `logging-service` و `dependency-service` و `integrations-service` . فلا تحوي أيّ لقطة مُودَعة مساراتها، ولا يحويها كذلك مرجع نقاط النهاية – ولا سيّما مسارات التحزيم والمتجر ومسارات سجلّ WhatsApp وخطّافه. وجرودها في استدعاء واجهة HTTP. وتذكّر أيضًا أنّ كلّ خدمة غير `data-service` تقع تحت مسارها الأساسي الخاصّ (`/api/reaction` ، `/api/packaging` ، …)، وأنّ المسارات في مواصفتها نسبيّة إليه – فاضبط عنوان URL الأساسي لكلّ عميل مولَّد وفقًا لذلك (التفاصيل).

قائمة تحقّق

قبل أن تثق بعميلٍ مولَّد:

- عميل واحد لكلّ خدمة LoopBack، مولَّد من مواصفة تلك الخدمة نفسها – أو، للتكامل المحصور بمعرض، عميل واحد من مستند المعرض الواحد الخاصّ به `Since 1.10` .
- ولِّد من **لقطة مُودَعة** لا من نقطة نهاية حيّة، حتى يكون الأصل قابلًا للفحص.
- تأكّد من أنّ اللقطة تحمل بصمات المُحسِّن الثلاث قبل التوليد؛ وحدّثها إن لم تكن كذلك.
- فُكّ غلاف `result` (واقرأ `error`) على طبقة النقل – فالأنماط المولَّدة تصف الحمولة لا الغلاف.
- مرّر التنزيلات والدفوق ومسارات الصحّة حول العميل المولَّد؛ واقرأ استجاباتها الخام.
- أعطِ كلّ عميل مولَّد المسار الأساسي لخدمته – `/api` لـ `data-service` وحدها.
- كامِل خدمات Express الثلاث، ومسارات خدمات LoopBack الأربع التي لا تصدير لها، يدويًّا وأبقِها خارج مسار إعادة التوليد.

البيئات والترقية

يُكتب العميل الخارجي مرّةً واحدة ويُشغَّل في مواجهة أكثر من نشرٍ واحد – مساحة تطوير، ومساحة تجهيز، وإنتاج. تتناول هذه الصفحة ما الذي يختلف فعلًا بين تلك النشرات من وجهة نظر العميل، حتى تكون ترقية عميلٍ من واحدةٍ إلى التالية تغييرًا في التهيئة لا تغييرًا في الرماز أبدًا. وتلتزم بما تكشفه منصّة أسس حقًّا؛ أمّا الأجزاء العامّة من نظافة التكامل المستمرّ فتبقى موجزةً عمدًا.

ما الذي يتغيّر فعلًا بين البيئات

أقلّ ممّا قد تظنّ. فعقد الـ API – التوجيه والإصدارات وغلاف الاستجابة ونموذج الأخطاء – واحدٌ عبر النشرات؛ وذلك هو كامل مقصد توثيقه. أمّا ما يتغيّر لكلّ بيئة فضيّق:

- **الرابط الأساسي** الذي يحادثه العميل؛
- **بيانات الاعتماد** التي يقدّمها العميل.

ينبغي أن يختزل كلّ ما يحمله العميل إلى هاتين القيمتين. فإن لزم تغيير أيّ شيءٍ آخر للانتقال بين البيئات، فعُدّ ذلك عيبًا في حدّ تهيئة العميل، لا حقيقةً من حقائق المنصّة.

الرابط الأساسي هو القاطن

أنت لا تخاطب الخدمات فرادى. فكلّ استدعاءٍ يذهب إلى البوّابة، والبوّابة تحدّد *أيّ* قاطنٍ تحادثه من **مضيف الطلب** – يبحث مُلحقا `domain-resolver` و`space-resolver` عن `Host` الوارد في النطاقات المسجّلة ويختمان `x-space-id` الخاصّ بالقاطن على الطلب قبل أن تراه أيّ خدمة.

والنتيجة العمليّة: الرابط الأساسي ليس مجرّد عنوان، بل *هو* محدِّد القاطن. فمساحة التطوير ومساحة الإنتاج يُوصَل إليهما عند مضيفين مختلفين، وتوجيه العميل إلى مضيفٍ مختلف يغيّر بصمتٍ أيّ مساحةٍ يقرأ بياناتها ويكتب فيها. لذا:

- ينتمي الرابط الأساسي إلى التهيئة، لا يُثبَّت في الرماز أبدًا.
- ليس هناك `x-space-id` ليضبطه العميل – فالبوّابة تشتقّه من المضيف، والعميل الذي يحاول ضبط ترويسات القاطن بنفسه غيرُ فعّالٍ وعلامةٌ حمراء معًا (انظر الحدّ الأمني).
- تحقّق من المضيف قبل أيّ كتابة. فترقية مهمّةٍ كانت مُوجَّهةً إلى التجهيز لتعمل في مواجهة مضيف إنتاجٍ حادثةُ بياناتٍ لا خطأٌ مطبعيٌّ في تهيئة.

بيانات الاعتماد لكلّ بيئة

بيانات اعتمادٍ مُصدَرة في مساحةٍ لا تعمل – ولا يجب أن تعمل – في أخرى. فبيانات الاعتماد هي الشيء الثاني والأخير الذي يتغيّر عند الترقية.

رموز التطبيق

أبسط بيانات اعتمادٍ للعمل بلا واجهة هو **رمز تطبيق**، يُصدَر لمستخدم تطبيقٍ مخصَّص ويُقدَّم عبر ترويسة `x-app-token`. ويُصدَر رمزٌ لكلّ مستخدم تطبيقٍ في كلّ مساحة، فلكلّ بيئةٍ رمزُها الخاصّ. ولا يحمل الرمز أذوناتٍ خاصّةً به: فوصوله هو بالضبط

أدوار مستخدم التطبيق المالك وعضويّته وأذونات معارضه في *تلك* المساحة (انظر اختيار بيانات الاعتماد). جهّز مستخدم تطبيقٍ منفصلًا، بأقلّ امتياز، في كلّ بيئة.

بيانات اعتماد عميل OAuth

حين يتصرّف العميل نيابةً عن مستخدمٍ تفاعليّ، يستخدم مسار OAuth 2.0 / OIDC الذي تخدمه `users-service`، بمعرّف عميلٍ وسرٍّ مسجَّلين لذلك النشر. وتلك أيضًا قيمُ لكلّ بيئة: فسرّ عميل التجهيز ليس سرّ عميل الإنتاج. وهي تنتمي إلى حدّ التهيئة نفسه الذي ينتمي إليه الرابط الأساسي، وتخضع للقاعدة نفسها أدناه حول عدم إدماجها في أثرٍ مبنيّ أبدًا.

ما الذي يبقى في جانب الخادم

بعض الأسرار تنتمي إلى ثقة المنصّة *الداخليّة* ويجب ألّا تصل إلى عميلٍ البتّة. فـ**المفتاح المشترك** (`SHARED_KEY`) هو السرّ المُسبَق الذي تستخدمه البوّابة لتصادق نفسها لدى الخدمات الخلفية؛ وهو مطلوبٌ في أيّ بيئةٍ غير تطويريّة وترفض العمليّة الإقلاع دونه (`getSharedKey`). وثمّة قيمةٌ احتياطيّة للتطوير للتشغيل المحلّي فقط.

لا يحمل العميل المفتاح المشترك قطّ، ولا يضبط ترويسات هويةٍ مثل `x-user-id` أو `x-space-id`، ولا يحادث منفذ خدمةٍ مباشرةً – فكلّ ذلك من عمل البوّابة، والعميل الذي في وسعه فعل أيٍّ منه حدٌّ أخفق سلفًا. يغطّي الحدّ الأمني السببَ كاملًا. وحين ترقّي عميلًا، فالمادّة الوحيدة التي يحملها بين البيئات رابطٌ أساسيّ وبيانات اعتماده الخاصّة؛ أمّا مادّة الثقة في جانب الخادم فتُهيَّأ على المنصّة وتبقى هناك.

لا تُدمِج بيانات اعتماد في أثرٍ مبنيّ أبدًا

رمز التطبيق مادّةٌ حاملة: فمن يحمله يتصرّف بوصفه مستخدم تطبيقه. عامِله تمامًا كما تعامل كلمة مرور.

- لا تُودِع رمزًا أو سرّ عميلٍ في التحكّم بالمصدر أبدًا، ولا تُدمِج واحدًا في أثرٍ مبنيّ (artifact) – صورة حاوية، أو حزمة مُصرَّفة، أو رزمةٍ منشورة. فالأثر يُنسَخ ويُخبَّأ ويُشارَك؛ وبيانات الاعتماد داخله تتسرّب معه.
- احقن بيانات الاعتماد وقت النشر من البيئة أو من مدير أسرار، مُفهرَسةً بالبيئة. ويسافر الرابط الأساسي بالطريقة نفسها.
- ناوِب دوريًّا وعند أيّ اشتباهٍ بانكشاف. تدعم رموز التطبيق المناوبة بنافذة سماحٍ حتى يظلّ الرمز الجاري نافذًا بينما يلتقط المستدعون الجديد – انظر دورة حياة الرمز: صمّم نشرك بحيث تكون المناوبة دفعةَ تهيئةٍ لا إعادةَ نشر.

نمط ترقية

يبقي الإعداد النظيف الرمازَ متطابقًا عبر البيئات ولا يغيّر إلّا التهيئة:

- أثرُ بناءٍ واحد، يُرقَّى دون تغيير من التجهيز إلى الإنتاج. فإن اختلف الأثر باختلاف البيئة، فقد تسرّبت بيانات اعتمادٍ أو رابطٌ داخله.
- الرابط الأساسي وبيانات الاعتماد يُوفَّران بوصفهما تهيئة بيئةٍ أو أسرارًا مُثبَّتة، تُختار حسب البيئة الهدف وقت النشر.
- فحصٌ سابقٌ للترقية يتأكّد أنّ المضيف المُحلَّل هو المقصود، حتى لا يعمل العميل في مواجهة القاطن الخطأ.
- مستخدمو تطبيقٍ متمايزون بأقلّ امتياز لكلّ بيئة، حتى لا يمسّ إبطال أحدهم أو مناوبته أخرى.

قائمة تحقّق

- يتغيّر شيئان اثنان فقط بين البيئات: الرابط الأساسي وبيانات الاعتماد.

- الرابط الأساسي يختار القاطن – تحقّق من المضيف قبل أيّ كتابة.
- بيانات اعتمادٍ منفصلة بأقلّ امتياز لكلّ بيئة؛ ولا إعادة استخدامٍ عبر المساحات.
- المفتاح المشترك وكلّ ترويسات الهوية تبقى في جانب الخادم؛ ولا يحملها العميل ولا يضبطها.
- لا بيانات اعتمادٍ في المصدر، ولا في صورة، ولا في أيّ أثرٍ مبنيّ – احقنها وقت النشر.
- ناوِب بيانات الاعتماد بوصفها تهيئة، مستخدمًا نافذة السماح لتجنّب التعطّل.

الحدّ الأمني

تصف هذه الصفحة حدَّ ثقةٍ حقيقيًّا في معماريّة منصّة أسس، لا ثغرة. وفهمه على نحوٍ صحيح هو ما يتيح لمشغِّلٍ نشر المنصّة بأمان، ولمؤلّف عميلٍ التفكيرَ فيما يجوز لرمازه حمله وما لا يجوز. الخلاصة الموجزة: **تحدث المصادقة مرّةً واحدة، عند البوّابة. وكلّ ما خلف البوّابة يثق بترويسات الهوية.** أصِب الحدّ فالتصميم سليم؛ واكشِف خدمةً وراء البوّابة يُصبح التصميمُ نفسُه بابًا مفتوحًا.

أين تُفحَص الرموز، وأين لا تُفحَص

يقدّم المستدعي الخارجي رمز وصولٍ حامل (OAuth 2.0 / OIDC) أو رمز تطبيق. وتُتحقَّق تلك البيانات **عند الحافّة** – يقوم مُلحقا البوّابة `oauth2-introspection` و `app-token-auth` بالتحقّق (`api-gateway/kong/plugins/`).

أمّا الخدمات الخلفية فلا **تعيد** التحقّق من الرمز. بل لا تراه أصلًا. تحلّ استراتيجيّة مصادقة خدمة LoopBack هويةَ المستدعي كلّيًّا من **ترويسات الطلب** التي تضبطها البوّابة بعد أن تصادق المستدعي، ولا تُرخِّص الطلب إلّا حين يرافق تلك الترويسات مفتاحٌ مشتركٌ صالح. ليس هناك فحصٌ ثانٍ مستقلّ للرمز الأصلي خلف البوّابة. وهذا خيارٌ متعمَّد في الأداء وفصل الاهتمامات – الحافّة تصادق، والخدمات تُرخِّص – وهو صحيحٌ **ما دام لا شيء يستطيع بلوغ خدمةٍ إلّا عبر الحافّة.**

ترويسات الهوية التي تثق بها الخدمات

بعد أن تصادق البوّابة مستدعيًا، تمرّر الهوية بوصفها مجموعة ترويسات. وتقرؤها استراتيجيّة الخدمة مباشرةً:

الترويسة	المعنى
`x-shared-key`	مفتاح الثقة المُسبَق بين البوّابة والخدمة. لا يُحلّ الطلب إلى ملفّ مستخدمٍ **إلّا** إذا طابق هذا مفتاح الخدمة المُهيَّأ.
`x-user-id`	الذات المُصادَقة. الملفّ المُحمَّل والأدوار والأذونات كلّها مشتقّة من هذا المعرّف.
`x-space-id`	القاطن / المساحة. العملية نفسها تُعيد بياناتٍ مختلفة لكلّ مساحة.
`x-client-id`	معرّف عميل OAuth للمستدعي.
`x-is-delegation` ، `x-delegatee-id` ، `x-delegation-id`	سياق التصرّف نيابةً (التفويض).
`x-service`	اسم الخدمة المستدعية. وقد تعدّ الخدمة القيمة `API Gateway` هنا دليلًا على أنّ الطلب مرّ عبر البوّابة.
`x-view-id`	المعرض الذي يُرسَل الطلب عبره. وتُخوِّل الخدمة الطلبَ الحاملَ له بوصفه مُرسَلًا عبر ذلك المعرض.

والنتيجة هي كامل هذه الصفحة: بالنسبة إلى خدمةٍ خلفية، **الهوية هي ما تقوله هذه الترويسات.** فالطلب الوارد حاملًا `x-shared-key` مضبوطًا على مفتاح الخدمة و `x-user-id` مضبوطًا على معرّف أيّ مستخدمٍ هو، بحسب تلك الخدمة، ذلك المستخدم – بكامل أذونات ذلك المستخدم، في المساحة المسمّاة في `x-space-id` . ولا يلزم رمز. لأنّ الرموز شأن حافّة. ولهذا بالضبط يجب ألّا يُكشَف الحدّ أبدًا: فزوج الترويستين `x-shared-key` + `x-user-id` كافٍ للتصرّف بوصفك أيَّ أحد.

كيف تحمي البوّابة الترويسات

التصميم آمنٌ لأنّ البوّابة تجعل تلك الترويسات غير قابلةٍ للانتحال من الخارج. آليّتان تفعلان ذلك، وفهمها هو ما يجعل قاعدة النشر أدناه غيرَ قابلةٍ للتفاوض.

يُنزَع المفتاح المشترك ثمّ يُحقَن من جديد

تُزيل البوّابة **أيّ** `x-shared-key` **مُوفَّرٍ من العميل** من كلّ طلبٍ وارد ثمّ تُعيد إضافة القيمة الصحيحة بنفسها (مُلحق `request-transformer` في `config/kong.yml`). والحاسم أنّها لا تحقن المفتاح *الحقيقي* إلّا حين يكون الطلب قد صُودِق: فالطلب غير المُصادَق تُمحى ترويسات هويته وتُضبَط علامة، فتكون القيمة المُعاد حقنها بوصفها `x-shared-key` نائبةً لا المفتاح الحقيقي، والخدمة الخلفية – إذ ترى مفتاحًا لا يطابق المُهيَّأ لديها – تُحلُّ إلى **لا مستخدم** وترفض الاستدعاء (`checkSharedKey`). لذا لا يستطيع مستدعٍ تهريب مفتاحٍ مشتركٍ صالح، ولا الحصول على واحدٍ بمجرّد إسقاط الرمز.

تُمحى ترويسات الهوية ثمّ تُضبَط من الرمز المُتحقَّق منه

يقوم مُلحق `oauth2-introspection` بـ**محو** `x-user-id` و`x-client-id` و`x-session-id` في كلّ طلبٍ ويعيد ضبطها من مطالبات الرمز المُستبطَن نفسه. فـ`x-user-id` المُوفَّر من العميل يُطرَح قبل أن يُوثَق به؛ والمعرّف الذي يبلغ الخدمة هو الذي صادق الرمز عليه فعلًا. ويتصرّف مُلحق رمز التطبيق بالمثل، ضابطًا `x-user-id` من مستخدم تطبيق الرمز المُتحقَّق منه. فالهوية التي تتلقّاها الخدمة هي دائمًا كلمة البوّابة، مشتقّةً من بيانات اعتمادٍ فحصتها – لا ادّعاء العميل.

قاعدة النشر

كلّ ما سبق يصمد بشرطٍ واحد، وهو أهمّ حقيقةٍ تشغيليّةٍ مفردة في هذه الصفحة:

> **يجب ألّا يكون منفذ خدمةٍ خلفية قابلًا للبلوغ إلّا عبر البوّابة.**

اضبط جدار حماية لكلّ خدمة بحيث لا تتّصل بها إلّا البوّابة. فالخدمة تثق ضمنًا بالترويسات التي تحمل الهوية – وفي مقدّمتها `x-shared-key` و`x-user-id` و`x-space-id` و`x-service` و`x-view-id`، وكلّ ترويسة داخلية أخرى تضبطها البوّابة. فإن أتاح أيّ مسار شبكيّ لمستدعٍ الوصول إلى خدمة مباشرةً، استطاع ذلك المستدعي ضبط تلك الترويسات بنفسه وانتحال هوية أيّ مستخدم في أيّ مساحة، متجاوزًا المصادقة كلّيًّا. فإزالة البوّابة للترويسات وإعادة حقنها تحميان الطلبات التي تمرّ *عبرها*؛ ولا توفّران أيّ حماية لطلب يمرّ *من حولها*. حدِّد هذه الترويسات في سياسة شبكتك واحجبها عند أيّ مدخل سوى البوّابة.

ما الذي يحمله العميل، وما الذي يبقى في جانب الخادم

يحدّد الحدّ فصلًا نظيفًا لما ينتمي إلى أين.

يحمل العميل بمشروعيّةٍ فقط:

- **رابطه الأساسي** (مضيف البوّابة)؛
- **بيانات اعتماده** – رمز وصولٍ حامل، أو رمز تطبيقٍ يُرسَل بوصفه `x-app-token`.

ويجب على العميل **ألّا** يحمل أو يضبط قطّ:

- **المفتاح المشترك** (`x-shared-key`) – فهو سرّ الثقة بين البوّابة والخدمة ويعيش في جانب الخادم فقط (`SHARED_KEY`؛ انظر البيئات والترقية)؛
- أيّ **ترويسة هوية** – `x-user-id` أو `x-space-id` أو `x-client-id` أو `x-service` أو `x-view-id` أو أيّ ترويسة داخلية أخرى. فالبوّابة هي المسؤولة عن ضبط هذه الترويسات بناءً على بيانات الاعتماد المُتحقَّق منها ومسار الطلب. وإذا ضبطها العميل، فإمّا أن تُتجاهَل قيمها (عبر البوّابة، حيث تُستبدَل)، وإمّا أن يكون متّصلًا بخدمة مكشوفة (متجاوزًا البوّابة، وهو خرق أمني).

فإن وجدت رماز عميلٍ يضبط `x-shared-key` أو `x-user-id`، فتلك إشارةٌ للتوقّف: فإمّا أنّه رمازٌ ميّت، أو دليلٌ على أنّ خدمةً قابلةٌ للبلوغ من خارج البوّابة.

التفويض

يتيح التفويض للطلب التصرّف بهوية شخص غير المستدعي المباشر، ولا يتمّ إلا بوساطة البوّابة، للسبب نفسه الذي ينطبق على بقية الحالات.

التفويض (`x-is-delegation`، `x-delegatee-id`، `x-delegation-id`) يتيح لمستدعٍ التصرّف نيابةً عن مستخدمٍ آخر، بأثرٍ تدقيقيّ. يسمّي العميل التفويضَ بـ `x-delegation-id`، ويتحقّق مُلحق البوّابة `user-delegation` من سجلّ التفويض – من وجوده، ومن أنّ المستدعي هو المُفوَّض إليه فيه، ومن وقوع الوقت الحالي ضمن نافذته النشطة – وعلى تفويضٍ مُتحقَّقٍ منه يضبط `x-delegatee-id` على المستدعي، ويعيد كتابة `x-user-id` إلى المُفوِّض، ويضبط `x-is-delegation: true`. ويُرفَض عند الحافّة بـ `403` تفويضٌ يخصّ مستخدمًا آخر أو منتهٍ أو لم يبدأ بعدُ.

الخلاصة

- البوّابة تصادق؛ والخدمات تُرخِّص من ترويسات موثوقة. ولا يُعاد فحص الرموز خلفيًّا.
- `x-shared-key` + `x-user-id` كافٍ لانتحال هوية أيّ مستخدم – فالهوية لدى الخدمة هي ما تقوله الترويسات.
- تنزع البوّابة `x-shared-key` وترويسات الهوية المُوفَّرة من العميل وتُعيد حقنها من بيانات الاعتماد المُتحقَّق منها، وهذا ما يجعل التصميم آمنًا.
- تعتمد تلك السلامة كلّيًّا على تعذّر بلوغ الخدمات إلّا عبر البوّابة. احجب ترويسات الهوية عند كلّ مدخلٍ ليس البوّابة.
- يحمل العميل رابطًا أساسيًّا وبيانات اعتماد – لا غير. المفتاح المشترك وكلّ ترويسات الهوية تبقى في جانب الخادم.

نموذج سير العمل ومرجع واجهته البرمجية

تخاطب هذه الصفحة **المُكامِلين والمطوّرين** الذين يقرؤون تعريفات سير العمل أو يكتبونها عبر واجهة API، أو يتحكّمون في انتقالات سير العمل من نظام آخر، أو يبنون نظامًا خارجيًا تُسلِّم إليه خطوةٌ من سير العمل المدخلات؛ إذ توثّق النموذج المخزَّن، وتعداداته العددية، وعقود وقت التشغيل، ونقاط النهاية.

> **تصمّم سير العمل من الواجهة؟**
>
> إن كنت تصمّم الموافقات في مصمّم الكيان، فابدأ من سير العمل والموافقات [بناء التطبيقات › سير العمل والاعتمادات]؛ إذ يغطّي الدليل الميزات نفسها بالتسميات التي يعرضها المصمّم. أما التنفيذ وقت التشغيل من منظور المستخدمين النهائيين (مَن يحقّ له التصرّف، والنقل، والإرجاع، وأخذ الملكية) فتجده في تنفيذ إجراءات سير العمل [استخدام أسس › تنفيذ إجراءات خطوات سير العمل].

أين يُخزَّن سير العمل

عندما يتقدّم المدخل، يقرأ محرّك سير العمل في `data-service` وفي `workflow-service` ذلك التعريف المُضمَّن ليحدّد الخطوة التالية ومالكها، ثم يسجّل التقدّم في حالة خاصة بكل مدخل (`_metaCurrentSteps` ، `_metaWorkflowHistory`).

> **التعدادات عددية.** فكل مُميِّز تقريبًا في سير العمل (نوع الخطوة، ونوع المالك، ونوع المصدر، ونوع الخطوة الهدف، …) تعدادٌ (enum) عددي يُحفَظ بوصفه عددًا صحيحًا؛ لذا تسرد هذه الصفحة القيمة العددية بجانب كل اسم. وحين تبني تعريفًا عبر واجهة API، فإنك ترسل **العدد** لا التسمية.

تُحرَّر **تعريفات** سير العمل بوصفها جزءًا من كيان البيانات (ضمن مصفوفته `workflows`)؛ فلا توجد نقطة نهاية مستقلة لـ«إنشاء سير عمل». راجع سطح API للمطوّرين للاطّلاع على مسارات الاستعلام ووقت التشغيل.

تعريف سير العمل (`DataEntityWorkflow`)

يمثّل سير العمل تسلسلًا مرتّبًا من الخطوات يحدّد المراحل التي يمرّ بها المدخل. وفي المصمّم، تضيفه من تبويب **سير العمل** في الكيان؛ فتُنشأ خطوتان افتراضيًا (باسمَي *البداية* و*النهاية*)، ثم تُدرِج بينهما ما تحتاجه من خطوات قبل الحفظ.

الحقل	النوع	مطلوب	ملاحظات
`id`	UUID	نعم	موروث من `Identified<UUID>` .
`name`	`LocalizedValue`	نعم	الاسم المعروض، متعدّد اللغات.
`normalizedName`	string	نعم	الاسم الآلي؛ يُتحقَّق منه عبر `NormalizedNameValidation` (الحدّان الأدنى والأقصى للطول + النمط).
`isDefault`	boolean	نعم	يميّز سير العمل الافتراضي للكيان.
`steps`	`WorkflowStep[]`	نعم	المجموعة المرتّبة من الخطوات (انظر أدناه).

يمكن أن يضمّ الكيان أكثر من سير عمل؛ ويتتبّع المدخل نسخة سير العمل النشطة عبر العلَم `active` على مدخلات `_metaCurrentSteps` الخاصة به (انظر `getActiveWorkflowInstance`). أما المدخلات القديمة التي تسبق اعتماد العلَم

`active`، فيُرجَع فيها إلى أول مدخل في الخطوات الحالية.

الخطوات (`WorkflowStep`)

تُعدّ الخطوة اللبنة الأساسية لسير العمل: مرحلة يُسنَد فيها مالك وتُنفَّذ فيها الإجراءات. وتشكّل الخطوات **مصفوفة مرتّبة**، ويحدّد ترتيبها التنقّل النسبي (الأولى، والسابقة، والحالية، والتالية، والأخيرة).

الحقل	النوع	ملاحظات
`id`	UUID	معرّف الخطوة. تشير إليه الخطوات الهدف للإجراءات ومصادر المالك وغيرها.
`name`	`LocalizedValue`	الاسم المعروض متعدّد اللغات.
`normalizedName`	string	الاسم الآلي، ويخضع للتحقق نفسه المطبّق على سير العمل.
`description`	nullable localized	اختياري.
`type`	(number) `StepType`	انظر جدول أنواع الخطوات أدناه.
`actions`	`WorkflowAction[]`	الانتقالات الخارجة من هذه الخطوة. القيمة الافتراضية `[]` .
`owner`	`WorkflowStepOwner`	إعداد إسناد المالك (اختياري).
`allowAddAdHoc`	boolean	هل يجوز إدراج خطوات مخصّصة (Ad hoc) بعد هذه الخطوة. القيمة الافتراضية `false` .
`inMainFlow`	boolean	هل الخطوة جزء من المسار الرئيسي (المسار المثالي). القيمة الافتراضية `true` .
`mainFlowIndex`	number	موضع الخطوة في المسار الرئيسي (تستخدمه معارض التقدّم «حالة المسار الرئيسي»).
`fallbackOwner`	`WorkflowStepOwner[]`	بدائل مرتّبة تُستخدَم حين يتعذّر تحديد المالك الأساسي (كأن لا يوجد مدير مباشر، أو لم تُنفَّذ الخطوة السابقة). القيمة الافتراضية `[]` .
`timeMonitoring`	`WorkflowStepTimeMonitoring`	إعداد SLA / زمن التنفيذ لكل خطوة (اختياري).
`subWorkflowId`	string	معرّف سير عمل (في الكيان **نفسه**) يُشغَّل بوصفه تدفقًا فرعيًا.
`mergeStep`	`NullableActionTargetStep`	الخطوة في سير العمل **الأصل** التي يُستأنف عندها التنفيذ حين يبلغ التدفق الفرعي خطوة النهاية (Exit).

تُنشأ خطوتا البداية (Entry) والنهاية (Exit) افتراضيًا ولا يمكن حذفهما، غير أنه يمكنك تعديل اسمَيهما ووصفَيهما ومتطلبات حقولهما.

يخضع اسم الخطوة المعروض (`name`)، وهو قيمة متعدّدة اللغات، لحدّ أدنى قدره **3 أحرف** وحدّ أقصى قدره **128 حرفًا**؛ أما `normalizedName` فيخضع للتحقق من الاسم الآلي المذكور لسير العمل أعلاه.

المسار الرئيسي والمسارات الفرعية

تشكّل خطوات سير العمل **مسارًا رئيسيًا** – وهو التسلسل الأساسي من الخطوات الذي يُتّبَع لبلوغ الهدف – وقد تشكّل كذلك **مسارًا فرعيًا** واحدًا أو أكثر. والمسار الفرعي فرع بديل يُسلَك حين **لا** تتحقق شروط المسار الرئيسي (كالرفض، أو التوجيه الشرطي إلى خطوة أخرى). ويسجّل الحقلان `inMainFlow` و `mainFlowIndex` في كل خطوة ما إذا كانت تنتمي إلى المسار الرئيسي وموضعها فيه، وتستند إليهما معارض تقدّم المسار الرئيسي.

`Since 1.8` – حين يكون المدخل حاليًا في خطوة **خارج** المسار الرئيسي، يتنبّأ معرض التقدّم بالخطوات المتبقّية باتّباع الإجراء الأساسي لتلك الخطوة. ويُحَلّ الهدف **النسبي** (التالية، والسابقة، والأولى، والأخيرة) في ذلك الإجراء بالطريقة نفسها التي يُحَلّ

بها معرّف الخطوة المطلق، فيستأنف التنبّؤ من الموضع الصحيح. وإذا تعذّر حلّ الهدف إلى خطوة على المسار الرئيسي، أفاد المعرض بأنه **لا** خطوات قادمة، بدلًا من إعادة سرد المسار الرئيسي كله.

أنواع الخطوات (`StepType`)

القيمة	الاسم	المعنى
0	`Entry`	بداية سير العمل، وفيها تستقرّ المدخلات المُنشأة حديثًا. مالكها دائمًا منشئ المدخل (انظر محرّك التحديد).
1	`Exit`	الخطوة الختامية التي تعلن اكتمال العملية. يتّسم تحديد المالك في خطوة النهاية بالتساهل (لا يُطلَق `OriginalOwnerNotFoundException`).
2	`Intermediate`	خطوة عادية بين البداية والنهاية، كمرحلة مراجعة أو موافقة.
3	`AdHoc`	خطوة تُدرَج وقت التشغيل لطلب موافقة إضافية أو معلومات إضافية.
4	`SubWorkflowStep`	خطوة تفوّض التنفيذ إلى سير عمل آخر عبر `subWorkflowId` .
5	`ExternalStep`	خطوة تسلّم المدخل إلى نظام خارجي، ولا تكتمل إلا حين يعاود ذلك النظام الاتصال. انظر الخطوة الخارجية. `Since 1.8`

وفي قائمة **نوع الخطوة** في المصمّم تظهر هذه الأنواع باسم **عادي** (Intermediate) و**تدفق فرعي** و**خطوة خارجية**؛ أما البداية والنهاية فهما الخطوتان الافتراضيتان، ولا تُدرَج الخطوات المخصّصة إلا وقت التشغيل.

`Since 1.8` – وجود المدخل في خطوة النهاية دليلٌ كافٍ بذاته على أن سير العمل **مكتمل**. ويتغيّر أيضًا علَم منفصل هو `lifeCycleStatus` ليشير إلى الاكتمال، غير أن هذا التحديث يجري بصورة غير متزامنة، ولا تنتظره نقاط نهاية الحالة.

يجب أن يبدأ سير العمل بخطوة بداية وينتهي بخطوة نهاية. ولأن التنقّل النسبي (الأولى، والسابقة، والحالية، والتالية، والأخيرة) يُحسب من موضع الخطوة في التسلسل، فإن ترتيب الخطوات مهمّ.

التدفق الفرعي `Since 1.4`

تفوّض **خطوة التدفق الفرعي** المدخلَ إلى سير عمل منفصل متداخل، ثم تستأنف سير العمل الأصل حين ينتهي سير العمل المتداخل. وبذلك تتيح إعادة استخدام عملية مشتركة (مثل «اعتماد الميزانية» القياسي) ضمن العديد من عمليات سير العمل الأكبر، دون إعادة بنائها في كل مرة. وتشير `SubWorkflowStep` إلى سير عمل آخر في الكيان **نفسه** عبر `subWorkflowId` : فيُشغَّل سير العمل المُشار إليه بكامله، وحين يبلغ `Exit` يستأنف المحرّك سير العمل الأصل عند `mergeStep` .

قواعد تصميم خطوة التدفق الفرعي

لأن خطوة التدفق الفرعي تسلّم المدخل إلى سير عمل آخر بدلًا من أن تنجز العمل بنفسها، يفرض المصمّم عليها قواعد خاصة:

- **لا إجراءات خاصة بها** – إذ يحدث الانتقال تلقائيًا حين يكتمل سير العمل المتداخل، ويُرفَض أي إجراء يُضاف إلى خطوة التدفق الفرعي.
- يجب أن تشير إلى **تدفق فرعي** موجود ومعرَّف على الكيان نفسه، وأن تكون الخطوة الأولى في التدفق الفرعي خطوة بداية.
- يجب أن تحدّد **خطوة دمج**: وهي الخطوة في سير العمل الأصل التي يعود إليها المدخل حين يكتمل التدفق الفرعي. ويمكن أن تكون خطوة الدمج خطوة محدّدة أو نسبية (الأولى، والسابقة، والحالية، والتالية، والأخيرة).

كيف يعبر المدخل حدود التدفق الفرعي

- **الدخول – يُعلَّق سير العمل الأصل.** حين ينتقل المدخل إلى خطوة تدفق فرعي، يُعلَّق سير العمل الأصل ويبدأ المدخل سير العمل المتداخل من خطوة البداية فيه. وتبقى خطوة سير العمل الفرعي في الأصل معلّقة حتى ينتهي سير العمل المتداخل.

- **الاكتمال – يُستأنف سير العمل الأصل عند خطوة الدمج.** حين يبلغ سير العمل المتداخل خطوة النهاية فيه، يعود المدخل إلى سير العمل الأصل عند خطوة الدمج المُعدّة، وتُوسَم خطوة التدفق الفرعي بأنها نُفِّذت، ويُحسب مالك خطوة الدمج في تلك اللحظة. ثم يُغلَق سير العمل المتداخل ويُزال.
- **التسلسل والتصاعد.** إذا كانت خطوة الدمج *نفسها* خطوة تدفق فرعي، بدأ سير العمل الفرعي التالي تلقائيًا؛ وإذا أوصل اكتمالُ تدفق فرعي المدخلَ إلى خطوة النهاية في سير عمل أصلٍ ينتمي بدوره إلى سير عمل أعلى، تصاعد الاكتمال إلى الأعلى حتى يبلغ سير عمل غير متداخل.
- **التفاعلات عند الحدود** `Since 1.8` **.** الخطوة التي يغادرها المدخل حين يبدأ سير العمل الفرعي، وخطوة التدفق الفرعي نفسها حين ينفَّذ إجراء من داخلها يعيد المدخل إلى الأصل (مثل «عودة» أو «رفض»)، تبلّغ كلتاهما عن تقدّم الخروج الخاص بها بالترتيب المعتاد، فيُطلَق أي تفاعل مُعدّ على الإجراء أو الخطوة المُغادَرة عند أيٍّ من الحدّين وفق الإعداد.

صمّم SLA خطوة التدفق الفرعي مع مراعاة المدة الكاملة لسير العمل المتداخل؛ فالعدّ التنازلي لسير العمل الأصل في تلك الخطوة يستمرّ طوال بقاء المدخل داخل سير العمل الفرعي، ولا يُستوفى إلا حين يُستأنف الأصل.

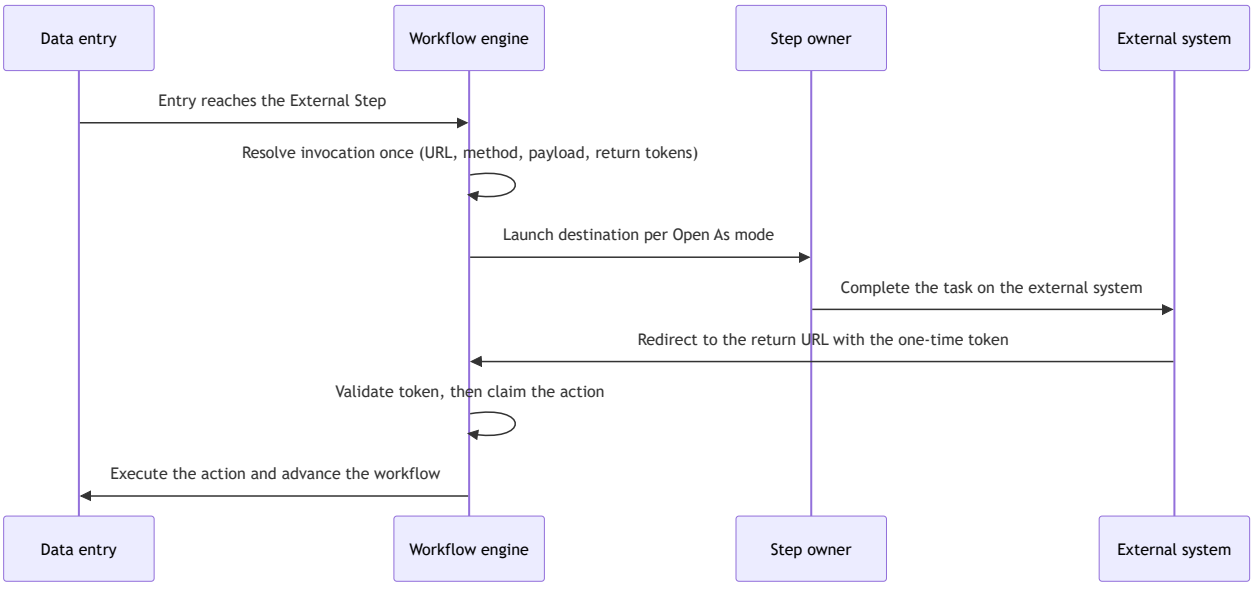
وراثة المالك عند خطوة بداية التدفق الفرعي

عادةً ما يملك منشئُ المدخل خطوةَ البداية. أما داخل التدفق الفرعي، فترث خطوة البداية **المالكَ المحسوب مسبقًا لخطوة التدفق الفرعي في الأصل**، لا المنشئ. وإذا كانت خطوة التدفق الفرعي في الأصل مُعدّة للموافقة المتوازية أو للاختيار اليدوي («أخذ الملكية»)، تُترك خطوة بداية التدفق الفرعي **دون إسناد**، كي يتولّى ملكيتها مستخدم مؤهّل.

`Since 1.8` – تسري هذه الوراثة أيضًا عند **انتقال لاحق يعود** إلى خطوة البداية في سير العمل الفرعي نفسه (كإجراء «عودة» يُتّخذ من داخله)، لا عند بدء التدفق الفرعي فحسب. ولا يعود إسناد الملكية إلى منشئ المدخل إلا حين يتعذّر حلّ الارتباط بالأصل، أو حين لا يكون المالك الموروث صالحًا للاستخدام (كعودة آلية أطلقها النظام دون وجود مالك مُعدّ للوراثة).

الخطوة الخارجية `Since 1.8`

تسلّم **الخطوة الخارجية** (`StepType.ExternalStep` ، القيمة `5`) المدخلَ إلى نظام خارجي – خدمة توقيع إلكتروني، أو بوابة دفع، أو بوابة خارجية – ولا تُستأنف إلا حين يعاود ذلك النظام الاتصال. ولا يمكن إتمام أيٍّ من إجراءاتها بالنقر على زر في البوابة؛ فاستدعاء العودة (أو تفاعل أتمتة) هو السبيل الوحيد لتقدّم الخطوة.



الإعداد (`ExternalStepConfig`)

يُضبَط السلوك الخارجي للخطوة في `WorkflowStep.externalStepConfig`، وهو مطلوب كلما كان `type = ExternalStep` .

الحقل	النوع	مطلوب	ملاحظات
`url`	string	نعم	الوجهة. قد يحتوي على عناصر نائبة `{{token}}` تُحَلّ حين يبلغ المدخل الخطوة. تسميته في المصمّم **الرابط**.
`method`	`ExternalStepHttpMethodEnum`	نعم	`0 = GET` ، `1 = POST` . تسميته **الطريقة**.
`openAs`	`ExternalStepOpenAsEnum`	نعم	طريقة عرض الوجهة – انظر أوضاع "فتح كـ". تسميته **فتح كـ**.
`payloadEnabled`	boolean	لا	الافتراضي `false` . يفعّل **حمولة مخصصة**؛ ولا يسري إلا مع `POST` .
`payload`	string	لا	الحمولة المخصّصة: نص JSON يحوي عناصر نائبة `{{token}}` ، ويُحلَّل إلى كائن مرة واحدة عند حلّ الاستدعاء.
`tokenBindings`	`Record<string, TokenSource>`	لا	`TokenSource` واحد (`type` + `source`) لكل عنصر نائب `{{token}}` مستخدَم في `url` / `payload` . الافتراضي `{}` .

ترسل خطوة `POST` التي لا تحمل حمولة مخصّصة متنًا افتراضيًا بالصيغة `{entry: <entry details>, actions: [{id, name, returnUrl}, …]}` – أي ببنية المدخل نفسها التي تعيدها نقطة نهاية قراءة تفاصيل المدخل، وليس بصيغة المستند الخام المخزَّن.

أوضاع "فتح كـ"

يحدّد `openAs` كيفية انتقال المالك إلى الوجهة. وله خمسة أوضاع:

القيمة	التسمية	ما يراه المالك
0	كامل الشاشة	ينتقل المتصفّح مباشرةً إلى الوجهة الخارجية.
1	صفحة داخلية	تُحمَّل الصفحة الخارجية داخل المدخل نفسه، في إطار قابل لتغيير الحجم.
2	شريط جانبي	تُفتَح الصفحة الخارجية في لوحة جانبية منزلقة.
3	مربع حوار	تُفتَح الصفحة الخارجية في مربع حوار مشروط (modal).
4	نافذة منبثقة	تُفتَح الصفحة الخارجية في نافذة متصفّح جديدة أو تبويب جديد.

تدعم بوابة الويب الأوضاع الخمسة كلها. أما **تطبيق الجوّال فلا يدعم إلا كامل الشاشة والصفحة الداخلية**، ويستخدم كامل الشاشة بدلًا من الشريط الجانبي ومربع الحوار والنافذة المنبثقة؛ لذا اضبط الخطوة التي يجب أن تظهر بالطريقة نفسها في كلا العميلين على أحد هذين الوضعين المدعومين.

رابط العودة واستبدال القيم

يُحَلّ `url` (ومعه `payload` في حالة `POST`) **مرة واحدة بالضبط**، حين يبلغ المدخل الخطوة. ويشمل الاستبدال نوعين من العناصر النائبة:

- **العناصر النائبة لرابط العودة.** يُضمِّن `{{@returnUrl}}` رابط العودة للإجراء الوحيد في الخطوة، ويُضمِّن `{{@returnUrl:actionId}}` رابط العودة لإجراء بعينه. ولا يُستبدَل `{{@returnUrl}}` المجرّد تلقائيًا إلا حين يكون للخطوة إجراء واحد بالضبط؛ أما في خطوة بلا إجراءات أو بعدّة إجراءات فيُترك دون حلّ بدلًا من تخمين الإجراء المقصود. ويعرض المصمّم التلميح *«استخدم `{{@returnUrl}}` في الرابط أو الحمولة أعلاه لتضمين رابط العودة (أو*

`{{@returnUrl:actionId}}` *إذا كان لهذه الخطوة أكثر من إجراء واحد).»* ويظهر **معرف الإجراء** لكل إجراء للقراءة فقط في نموذج الإجراء، مع زر لنسخه إلى الحافظة ولصقه في العنصر النائب.

- **ربط الرموز.** يُحَلّ كل عنصر نائب `{{token}}` آخر في الرابط أو الحمولة باستخدام `tokenBindings` استنادًا إلى المدخل. ويُستبعَد العنصران النائبان المحجوزان لرابط العودة عمدًا من قائمة الرموز القابلة للربط، كي لا يُربَطا خطأً كرمزين عاديين.

تُرمَّز القيم المستبدَلة داخل **الرابط** ترميزَ URL قبل إدراجها، إلا إذا كان موضع العنصر النائب *هو* الرابط كاملًا بعد إزالة المسافات من طرفيه (أي إنك جعلت رابط العودة هو الوجهة كلها وليس قيمةً داخل رابط أكبر). وبذلك، إذا كانت القيمة المستبدَلة رابطًا، فإنها لا تُفسد سلسلة الاستعلام في الرابط الخارجي. أما الاستبدال في الحمولة فيبقى دون ترميز، لأن الحمولة نص خام أو JSON، وليست رابطًا.

عقد العودة

تُعالَج العودة لكل إجراء في كل خطوة خارجية عبر نقطة نهاية واحدة هي `POST /external-step/return`، ومتنها `{ token }`. وهذه النقطة **غير مصادَق عليها عمدًا**: فالمستدعي نظام خارجي يعيد توجيه متصفّح المستخدم، وليس جلسة بوابة مسجّلة الدخول، والرمز المعتم ذو الاستخدام الواحد هو نفسه بيانات الاعتماد. وتعيد عمليات التكامل توجيه المستخدم إلى مسار الواجهة الأمامية `{APP_BASE_URL}/external-step-return?token=…`، الذي يستدعي نقطة النهاية نيابةً عنها.

يُصدَر رمز واحد **لكل إجراء**، مرة واحدة، حين يُحَلّ الاستدعاء. وهو سلسلة معتمة غير موقَّعة: بادئة بترميز base64url تعرّف المدخل وسير العمل والنسخة والخطوة والإجراء، يليها سرّ عشوائي. ويُقارَن السرّ في زمن ثابت بالقيمة المحفوظة على الخطوة الحالية للمدخل، فلا يمكن تخمين الرمز ولا إعادة استخدامه من إجراء آخر.

مدة صلاحية رمز العودة `Since 1.8` – تنتهي صلاحية رمز العودة بعد مدة ثابتة من إطلاق الخطوة. والمدة قابلة للضبط لكل بيئة، وقيمتها الافتراضية **24 ساعة**. وإذا كان الطابع الزمني للإطلاق مفقودًا أو تعذّرت قراءته، **تُرفض العودة احتياطًا** بدلًا من قبولها بصلاحية غير محدودة.

وتنتهي معالجة العودة بإحدى أربع نتائج: النجاح وإدراج الإجراء في الطابور؛ أو تبيّن أنها عولجت سابقًا (تجاوز المدخل الخطوة، أو استأثرت بالإجراء عودةٌ شبه متزامنة – وهذه عملية متساوية الأثر لا تغيّر شيئًا ولا تُعدّ خطأً)؛ أو عدم صلاحيتها (رمز مشوَّه أو منتهي الصلاحية أو غير مطابق، أو لم يعد المدخل أو الخطوة مؤهّلًا)؛ أو الاستئثار بالإجراء مع إخفاق إرسال إجراء سير العمل.

القواعد التي يفرضها المصمّم

يُتحقَّق من استيفاء خطوة من نوع `ExternalStep` للقواعد التالية في المصمّم وعند الحفظ معًا:

القاعدة	يُرفض بـ
`url` (غير فارغ) و `method` و `openAs` كلها مطلوبة.	`ExternalStepConfigRequiredException`
يجب أن يُحَلّ مالك الخطوة إلى مستخدم واحد محدّد بالضبط – نوع المالك **مستخدم**، والمصدر **ثابت**، مع الإشارة إلى الكيان `USER`. ويُرفض عند الحفظ المالك من نوع مجموعة أو المالك الديناميكي أو المالك المحدّد بتعبير.	`ExternalStepOwnerMustBeUserException`
لا تُقبل الحمولة المخصّصة (`payloadEnabled = true`) إلا حين يكون `method = POST`.	`ExternalStepCustomPayloadRequiresPostException`

ويطبّق المصمّم هذه القواعد نفسها على جانب العميل: فعند تحويل خطوة إلى خطوة خارجية، يضبط `method` على `GET` إن لم يكن مضبوطًا، ويشترط الرابط والطريقة و«فتح ك»، ويمسح الحمولة المخصّصة حين لا تكون الطريقة `POST`، ويقصر منتقي المالك على مستخدم ثابت واحد.

ولأن مرجع هذا المستخدم الثابت هو مرجع مالك **مطلوب**، فإنه يُحمى أيضًا عند انتقال سير العمل بين البيئات: فإذا كان استيراد الحزمة سيؤدي إلى مسح مرجع مطلوب لمالك خطوة ثابت، أوقفه فحصٌ مسبق بدلًا من أن يُخفق في منتصف الاستيراد – وهي القاعدة الثابتة نفسها التي تمنع حفظ الكيان إذا كان المالك الأساسي الثابت `null`.

قواعد الإبطال والحماية

- **الإتمام اليدوي ممنوع.** يُرفض تنفيذ إجراء خطوة خارجية من البوابة ما لم يأتِ الأمر من استدعاء العودة أو من تفاعل أتمتة. ولا تعرض بوابة المستخدم النهائي أزرار إجراءات الخطوة أصلًا.
- **المدخلات المؤرشفة وغير الصالحة والمتقادمة لا تُتمّ الخطوة.** يشترط مسار العودة أن يكون المدخل نشطًا وغير مُبطَل، ثم يعيد التحقق من هذا الشرط ضمن عملية الكتابة التي تستأثر بالإجراء، فيسدّ الفجوة التي قد يُؤرشَف فيها المدخل أو يُبطَل بين التحقّقين.
- **يُعاد التحقق من نوع الخطوة الحالي عند العودة.** حتى مع رمز صالح لم يُستأثر به بعد، يقرأ مسار العودة تعريف سير العمل *الحالي* للمدخل ويشترط أن تظل الخطوة خطوة خارجية. فإذا غُيّر نوع الخطوة (إلى Intermediate مثلًا) والرمز ما زال قائمًا، لم يعد ذلك الرمز قادرًا على تنفيذ أي شيء.
- **يُمسح الاستدعاء المتقادم عند كل انتقال.** يُحذف الاستدعاء المحلول كلما غادر المدخل خطوة، وليس فقط حين يغادرها عبر إجراءاتها؛ وبذلك تحصل الخطوة الخارجية التي يصل إليها لاحقًا بقفزة بين عمليات سير العمل أو بنقل إداري على رابط وحمولة ومجموعة رموز عودة تُحَلّ جميعها من جديد في كل مرة.

الانتقالات والإجراءات (`WorkflowAction`)

الإجراءات هي الأزرار التي يراها مالك الخطوة؛ ويوجّه كل إجراء المدخل إلى خطوة هدف. وتحتاج كل خطوة – عدا خطوة النهاية وخطوات التدفق الفرعي – إلى إجراء واحد على الأقل، ولا يجوز أن يكون للخطوة أكثر من إجراء أساسي (رئيسي) واحد.

الحقل	النوع	ملاحظات
`id`	UUID	معرّف الإجراء.
`name`	`LocalizedValue`	تسمية الزر.
`description`	nullable localized	اختياري.
`isMainAction`	boolean	يميّز الإجراء الأساسي (إجراء المسار المثالي). يستخدمه تصويت القرار الجماعي (انظر `mainActionExecutionCondition`).
`dialogOptions`	`WorkflowActionDialogOptions`	سلوك مربع حوار التعليق / التأكيد.
`targetStep`	`ActionTargetStep`	الخطوة التي ينتقل إليها المدخل حين يُنفَّذ هذا الإجراء.
`message`	`WorkflowActionMessage`	رسالة اختيارية تظهر بعد التنفيذ (`enabled` + قيمة `value` متعدّدة اللغات).
`visible`	`FieldHiddenOption`	قاعدة ظهور اختيارية – لا يظهر الإجراء إلا إذا تحقق شرطه.
`conditionalTargetSteps`	`ConditionalTargetStep[]`	بدائل لـ `targetStep` تحدّدها التعبيرات. اختياري.
`stepUp`	`StepUpRequirement`	اختياري – يُلزم المنفِّذ بتأكيد هويته من جديد قبل تشغيل هذا الإجراء، لمدخل واحد أو – `Since 1.10` – لمجموعة المدخلات المحدّدة الخاضعة للتحقق بأكملها. انظر التحقق الإضافي قبل تنفيذ الإجراء. `Since 1.10`

الخطوة الهدف (`ActionTargetStep`)

الحقل	النوع	ملاحظات
`isRelative`	`TargetStepTypeEnum` (number)	`0 = Relative` ، `1 = Absolute` ، `2 = Dynamic` .
`targetStep`	`RelativeStepEnum` | UUID	في Absolute: معرّف UUID للخطوة. وفي Relative: رمز نسبي.

يحدّد الهدف **الثابت (المطلق)** خطوة بعينها بمعرّفها (مثل «موافقة المدير المباشر»). أما الهدف **النسبي** فيشير إلى خطوة بموضعها دون أن تكون معروفة مسبقًا. والرموز النسبية (`RelativeStepEnum`) سلاسل حارسة تُحَلّ وقت التشغيل:

الرمز	القيمة
الخطوة التالية	`___NEXT_STEP___`
الخطوة السابقة	`___PREVIOUS_STEP___`
الخطوة الأولى	`___FIRST_STEP___`
الخطوة الأخيرة	`___LAST_STEP___`
الخطوة الحالية	`___CURRENT_STEP___`

الإجراء الأساسي والعودة والرفض المدمجة

ينشئ مربع الحوار **إنشاء خطوة جديدة** الإجراءات الأولية للخطوة:

- **الإجراء الأساسي** (مطلوب) – إجراء باسم *الإجراء الأساسي* يحمل `isMainAction = true` ، وهدفه الخطوة التي تختارها (خطوة نسبية أو خطوة موجودة).
- **رجوع** (خانة اختيار) – إجراء باسم *عودة* هدفه **الخطوة الأولى** النسبية (`___FIRST_STEP___`)، أي العودة إلى خطوة البداية.
- **رفض** (خانة اختيار) – إجراء باسم *رفض* هدفه **الخطوة الأخيرة** النسبية (`___LAST_STEP___`)، أي خطوة النهاية، فينتهي سير العمل.

وتصبح الإجراءات الثلاثة بعد إنشائها إجراءات `WorkflowAction` عادية؛ فيمكنك إعادة تسميتها أو تغيير أهدافها كأي إجراء آخر. ويظهر إجراءا العودة والرفض كذلك ضمن خيارات `failureActions` في تصويت القرار الجماعي.

الخطوات الهدف الشرطية (`ConditionalTargetStep`) `Since 1.4`

إلى جانب الهدف الثابت أو النسبي الواحد، يمكن للإجراء أن يوجّه المدخل إلى **خطوات تالية مختلفة بحسب شرط** يُقيَّم وقت التشغيل. وعند وجود هذه الأهداف، تُقيَّم **بحسب** `order` ؛ فأول شرط يُقيَّم إلى `true` يحدّد الخطوة الهدف، وتكون له الأولوية على `targetStep` الثابت للإجراء. وإذا لم يتحقق أي شرط، استخدم الإجراء `targetStep` الافتراضي.

الحقل	النوع	ملاحظات
`order`	number	ترتيب التقييم (تصاعدي).
`condition`	string	تعبير يجب أن يُقيَّم إلى قيمة منطقية.
`targetStep`	`ActionTargetStep`	الهدف المستخدَم حين يتحقق الشرط.

خيارات مربع الحوار (`WorkflowActionDialogOptions`)

الحقل	النوع	الافتراضي	ملاحظات
`commentValidationMode`	`WorkflowActionCommentValidationModeEnum`	`Optional (0)`	`0 = Optional` ، `1 = Required` ، `2 = Disabled` (تسميته في المصمّم **مخفي**).

الحقل	النوع	الافتراضي	ملاحظات
`showConfirmationDialog`	boolean	`false`	طلب التأكيد قبل التنفيذ.
`title`	nullable localized	–	عنوان مربع الحوار.
`message`	nullable localized	–	نص مربع الحوار.

التحقق الإضافي قبل تنفيذ الإجراء `Since 1.10`

يمكن إعداد الإجراء بحيث يُلزم من ينفّذه بأن **يؤكّد هويته من جديد** قبيل تشغيله، بمعزل عن الجلسة التي سجّل دخوله بها أصلًا. ويُضبَط هذا الإعداد على **الإجراء** (`stepUp`)، وليس على الخطوة أو سير العمل؛ لذا يمكن أن يختلف إجراءان على الخطوة نفسها: قيّد *اعتماد الصرف*، واترك *طلب معلومات إضافية* بلا قيد.

وللإعداد جزآن: **تفعيله**، ثم – اختياريًا – **تحديد تعبير** يقرّر اشتراط التحقق لكل مدخل على حدة.

الحقل	النوع	الافتراضي	ملاحظات
`enabled`	boolean	`false`	يفعّل الاشتراط لهذا الإجراء.
`expression`	string | null	`null`	شرط اختياري يُقيَّم استنادًا إلى المدخل الذي يُنفَّذ عليه الإجراء.

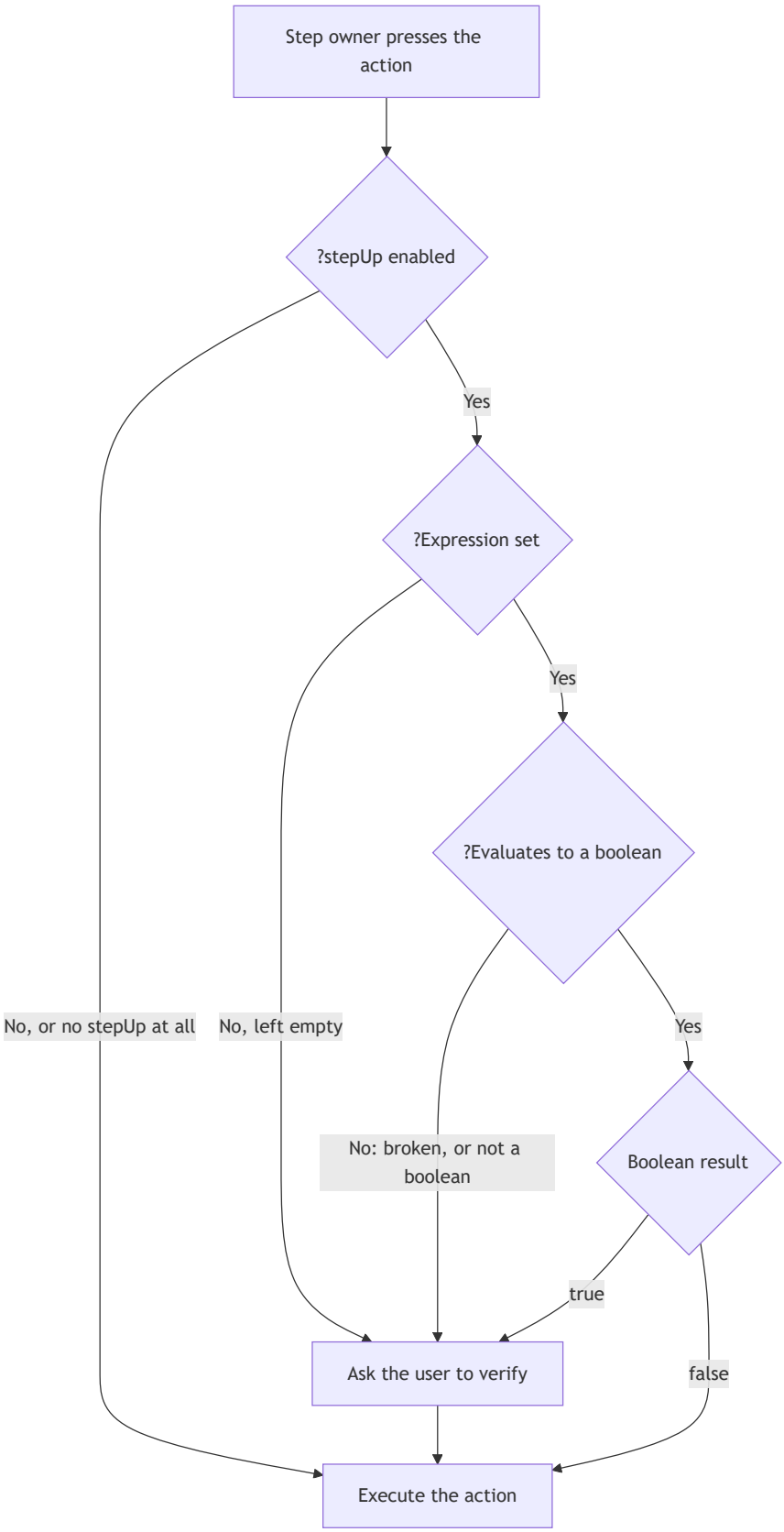
ويكون السلوك في الحالات الثلاث كما يلي:

طريقة الإعداد	متى يُطلب التحقق
معطَّل – `enabled` قيمته `false` ، أو لا يحمل الإجراء `stepUp` أصلًا	**أبدًا.**
مفعَّل والتعبير متروك فارغًا	**دائمًا.**
مفعَّل مع تعبير	**حين يكون ذلك التعبير صحيحًا** للمدخل الذي يُنفَّذ عليه الإجراء.

ويظل سلوك الإجراء الذي لا يحمل `stepUp` مطابقًا لسلوكه قبل إتاحة هذه الميزة، دون الحاجة إلى أي ترحيل أو تعطيل.

⚠ التعبير المعطوب يوجب التحقق ولا يتخطّاه

التعبير الذي يتعذّر تقييمه، أو الذي يُرجع قيمة غير منطقية، يجعل التحقق **مطلوبًا**. وهذا السلوك معاكس عمدًا لسلوك تعبير الظهور `visible` للإجراء: فقاعدة الظهور المعطوبة تُخفق *منفتحةً* إلى «مخفي»، أما قاعدة التحقق الإضافي المعطوبة فتُخفق *منغلقةً* إلى «تحقّق أولًا». لذا اكتب التعبير بحيث يُرجع قيمة منطقية دائمًا؛ فالتعبير المشوَّه لا يُتجاهَل، بل يجعل تنفيذ الإجراء مشروطًا بالتحقق في كل مدخل.



وسائل التحقق المتاحة. يعيد التحقق الإضافي استخدام وسائل التحقق المتوفّرة للحساب أصلًا، ولا يضيف آلية جديدة. والوسائل المقبولة هي:

- رمز من **تطبيق المصادقة**، أو
- **رمز لمرة واحدة يُرسَل برسالة نصية (SMS)**، أو

- **رمز لمرة واحدة يُرسَل بالبريد الإلكتروني**، أو
- **كلمة مرور الحساب**، ولكن فقط حين لا تكون أيٌّ من الوسائل السابقة مُعدّة للحساب – فهي وسيلة أخيرة تلجأ إليها المنصة، وليست خيارًا يختاره المستخدم.

ولا يصلح مفتاح المرور (passkey) لهذا التحقق، حتى لمن اعتاد تسجيل الدخول به؛ إذ يتطلّب الفحص رمزًا يُكتب يدويًا، ولذلك يستخدم الحساب الذي يفضّل مفتاح المرور أول وسيلة رموز متاحة له. فاذكر ذلك صراحةً في أي إرشادات تكتبها لمستخدميك: لن يظهر طلب استخدام مفتاح المرور الذي يتوقعونه.

ويُقيَّد نطاق الإثبات عمدًا. ينتج عن التحقق الناجح إثبات **لاستخدام واحد**، **تنتهي صلاحيته بعد دقيقتين**، و**مقيَّد بذلك المستخدم وذلك المدخل وذلك الإجراء** – وتُقارَن العناصر الثلاثة كلها عند استهلاكه، ويصبح غير قابل للاستخدام بعد ذلك. ولا يُعدّ الإثبات جلسةً؛ فلا يسهّل تنفيذ الإجراء التالي، ولا يكشف الحقول الحسّاسة المقنَّعة. (فكشف القيمة الحسّاسة فحص منفصل له بيانات اعتماده الخاصة؛ واجتياز أحد الفحصين لا يغني عن الآخر أبدًا، وإن طلب كلاهما النوع نفسه من الرموز.) ويُرسَل الإثبات في الحقل `stepUpProof` ضمن استدعاء تنفيذ الإجراء – انظر سطح API للمطوّرين.

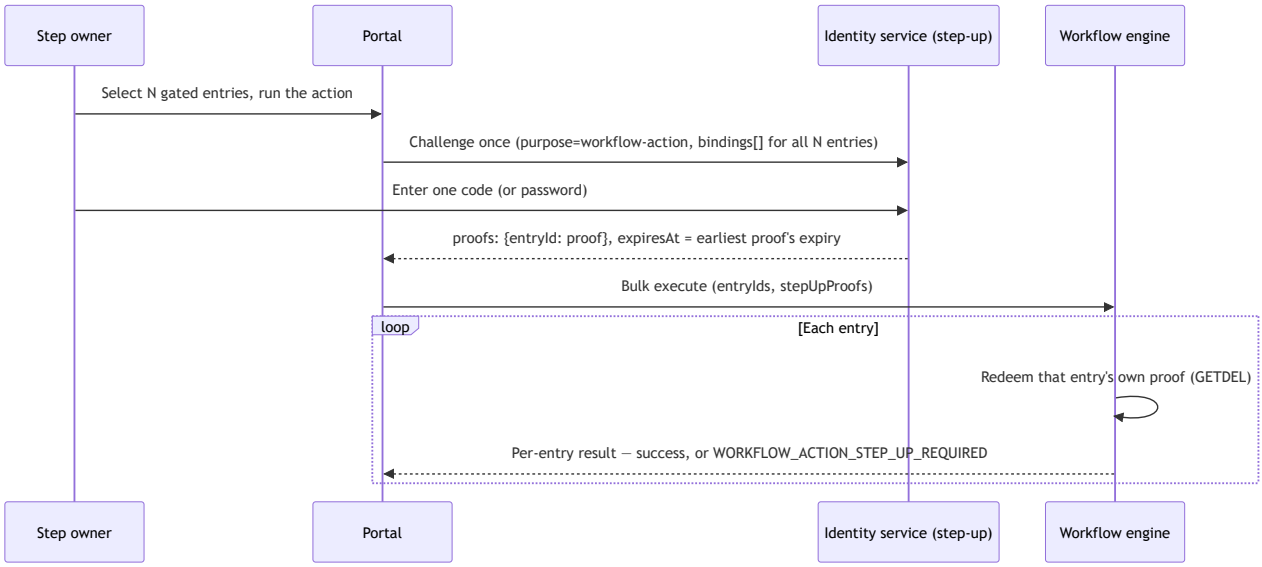
التنفيذ الجماعي `Since 1.10` **– تحقّق واحد يغطّي مجموعة المدخلات المحدّدة الخاضعة للتحقق بأكملها.** يُدعَم تنفيذ إجراء خاضع للتحقق على دفعة دعمًا كاملًا: يتحقق المنفِّذ **مرة واحدة** للمجموعة المحدّدة كلها، ويُنفَّذ كل مدخل في الدفعة باستخدام **خريطة من الإثباتات أحادية الاستخدام، إثبات لكل مدخل خاضع للتحقق** – دون استخدام إثبات واحد مشترك أو تنفيذ أيٍّ من هذه المدخلات بلا إثبات.

وأُضيف إلى متن طلب نقطة النهاية الجماعية (`POST /v1/entities/{entity}/{workflowId}/{actionId}/bulk` ، `BulkExecuteWorkflowActionRequestDto`) حقل اختياري:

```
class BulkExecuteWorkflowActionRequestDto {
  entryIds: string[];
  notes: string;
  // entryId -> single-use step-up proof, one per entry, all minted by one
  // verification. Absent when the action needs no step-up.
  stepUpProofs?: Record<string, string>;
}
```

يقرأ الخادم `stepUpProofs?.[entryId]` لكل مدخل في الدفعة، ويمرّره إثباتًا خاصًا بذلك المدخل إلى مسار التنفيذ الفردي نفسه الذي يستخدمه استدعاء المدخل الواحد؛ وبذلك يعادل التنفيذ الجماعي، من منظور المحرّك، تنفيذ الإجراء فرديًا N مرة مستقلة، يحمل كلٌّ منها إثباته الخاص (أو لا يحمل إثباتًا). **ويُرفض المدخل الذي لا مفتاح له في الخريطة تمامًا كما يُرفض استدعاء مدخل واحد بلا إثبات** (`WORKFLOW_ACTION_STEP_UP_REQUIRED`)، بمعزل عن بقية مدخلات الدفعة؛ فالخريطة الناقصة لا يمكن أن توسّع الوصول أبدًا، وأقصى ما تسبّبه هو إخفاق تنفيذ بعض المدخلات.

إصدار إثباتات الدفعة. يرسل العميل طلب التحقق مرة واحدة مع `purpose: 'workflow-action'` ومصفوفة `bindings[]` بدلًا من `binding` واحد – عنصر `{entityId, entryId, actionId}` لكل مدخل محدّد، وتشترك جميع العناصر في قيمتي `entityId` / `actionId` نفسيهما (إجراء واحد ومدخلات كثيرة، دون السماح بدفعة مختلطة)، بحدّ أقصى **100** عنصر، ودون تكرار أي `entryId` . ويُصدر تحقق ناجح واحد **إثباتًا لكل عنصر ربط**، ويعيد `{proofs: {entryId: proof}, expiresAt}` ؛ وقيمة `expiresAt` هي وقت انتهاء صلاحية **أول** إثبات صادر وليس آخرها، لأن الدفعة لا تبقى صالحة للاستخدام إلا ما دام أقدم إثبات فيها صالحًا. أما آلية الإصدار والاستهلاك الكاملة للأغراض الثلاثة – التي يشترك فيها غرض التحقق الإضافي `Decrypt` المضاف في الدورة نفسها – فموثّقة في موضع واحد هو الحقول الحسّاسة: التحقق الإضافي لـ WorkflowActiong Decryptg Unmask [بناء التطبيقات › حماية حقل]، تجنّبًا لاختلاف التوثيق بين الصفحتين.



ويظهر هذا السلوك للمستخدمين على النحو التالي، لتراعيه في التصميم:

- عند تحديد **عدة مدخلات** وتشغيل إجراء خاضع للتحقق، يُفتَح مربع تحقق **واحد** للمجموعة المحدّدة كلها، بدلًا من طلب تحقق لكل صف.
- عند نجاح التحقق، يُصدَر **لكل** صف محدّد إثباته الخاص، وتُنفَّذ الدفعة باستخدام الخريطة التي تربط كل مدخل بإثباته. راجع تنفيذ إجراءات سير العمل: التحقق من هويتك قبل تنفيذ الإجراء [استخدام أسس › تنفيذ إجراءات خطوات سير العمل] للاطّلاع على مسار المستخدم النهائي.
- إذا انتهت صلاحية إثبات مدخل أو أصبح غير مطابق بين لحظة إصداره ولحظة تنفيذ الإجراء على ذلك المدخل (كأن تتغيّر خطوته في الأثناء)، تُعاد نتيجة **إخفاق خاص بذلك المدخل** في خريطة نتائج الدفعة، دون أن يُفشل ذلك بقية الدفعة.
- عند تحديد **مدخل واحد بالضبط**، يظل السلوك كما كان: يظهر مربع التحقق الواحد نفسه الذي يظهر عند التنفيذ من صفحة المدخل.
- مَن ينفّذ الإجراء عبر صفحة تمنح **التنفيذ بصفة مدير**، أو من منطقة الإدارة حيث لا توجد صفحة مستخدم نهائي في السياق، **لا يُطلب منه التحقق أبدًا** – سواء في التنفيذ الجماعي أو في المدخل الواحد. ولا يخضع التنفيذ الآلي (تفاعل، أو استدعاء عودة من خطوة خارجية) لهذا القيد أيضًا.

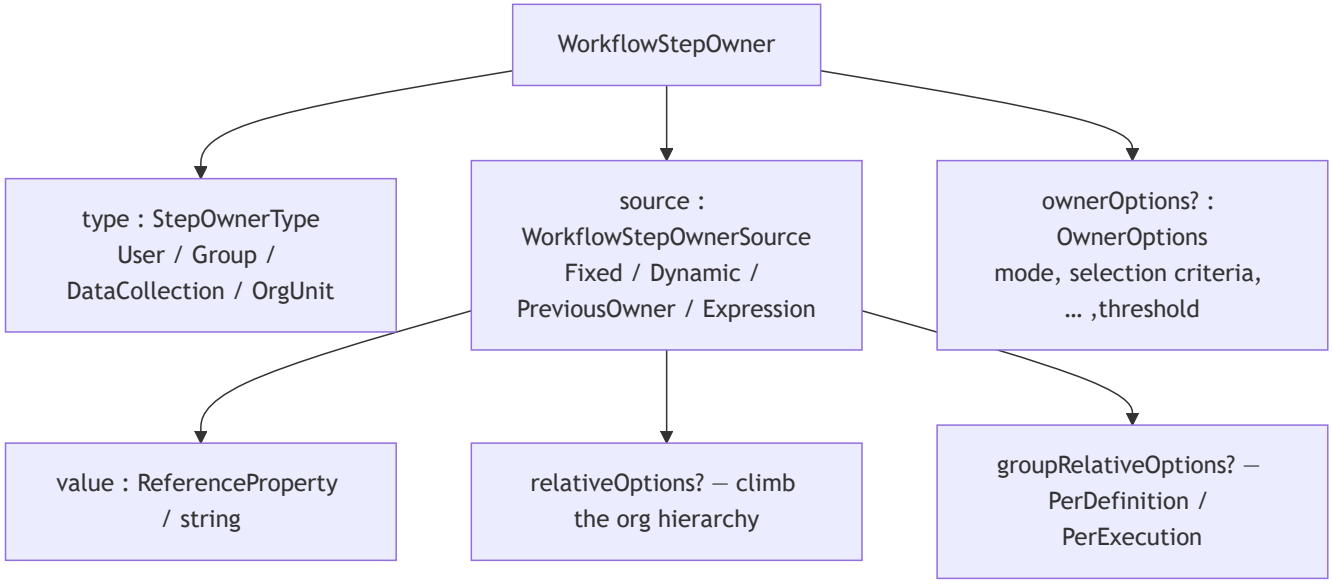
إذا كان عمل الخطوة يُنجَز فعلًا على دفعات، فأزل القيد عن ذلك الإجراء وضعه على إجراء آخر؛ إذ لا يمكن أن يكون الإجراء الخاضع للتحقق والإجراء المعتمَد على دفعات إجراءً واحدًا.

يُعرَف الرفض برمز الخطأ – فاعتمد في عقد التكامل على الرمز وليس على حالة HTTP. التنفيذ الذي يبلغ الخادم دون إثبات صالح يُرفض برمز الخطأ `WORKFLOW_ACTION_STEP_UP_REQUIRED`، وتتضمّن تفاصيله `actionId`. أما حالة HTTP المصاحبة لهذا الرفض فهي **موروثة من صنف خطأ عام، وليست مختارة لهذه الحالة**: فهي ليست `401` ولا `403` كما قد يتوقع المُكامِل، ولا تضمن المنصة ثباتها، ولذلك سيختلّ عمل العميل الذي يبني قراره عليها. ابنِ قرارك على رمز الخطأ، كما تفعل البوابة نفسها. وينطبق هذا على الدفعة أيضًا: فرفض أي مدخل داخل نتيجة الدفعة يحمل رمز الخطأ نفسه، دون حالة HTTP خاصة به.

يُسجَّل التحقق الناجح أو المخفق بمعزل عن سجل تدقيق إجراء سير العمل نفسه – انظر المراقبة: السجلات والتدقيق [الإدارة] للاطّلاع على سجل التدقيق الذي يراه المسؤول.

إسناد المالك

يتولّى مالك الخطوة متابعة المهام المرتبطة بها وإنجازها: اتخاذ القرارات، وتنفيذ الإجراءات، وتمرير المدخل إلى المراحل التالية. ويُمثَّل المالك بـ `WorkflowStepOwner`، الذي يتألّف من ثلاثة أجزاء: **النوع** (فئة الجهة التي تملك الخطوة)، و**المصدر** (كيفية العثور على المرجع)، و**خيارات المالك** (السلوك بعد التحديد).



نوع المالك (`StepOwnerType`)

القيمة	الاسم	سلوك التحديد
0	`User`	يشير المرجع إلى مستخدم ويُستخدَم مباشرةً. تسميته في المصمّم **مستخدم**.
1	`Group`	تُستخرج منه قائمة أعضاء المجموعة، ويُختار العضو الأول ما لم تنطبق معايير التوازي أو الاختيار اليدوي أو المدير. تسميته في المصمّم **مجموعة**.
2	`DataCollection`	يصبح أول عنصر في مجموعة البيانات المُشار إليها هو المالك. تسميته في المصمّم **قائمة**.
3	`OrgUnit`	مرجع إلى وحدة تنظيمية، وإلى مديرها اختياريًا. لا يظهر في قائمة **مالك الخطوة** في المصمّم (التي تسرد مستخدم ومجموعة وقائمة)؛ ويمكن الوصول إليه عبر API.

مصدر المالك (`WorkflowStepOwnerSource`)

`source.type` من النوع `StepOwnerSourceType`، ويحدّد *كيفية* العثور على مرجع المالك:

القيمة	الاسم	ما يحمله `value`	الاستراتيجية التي تحلّه
0	`Fixed`	`ReferenceProperty` (أو مصفوفة) يشير إلى مستخدم أو مجموعة تُختار وقت التصميم	`FixedOwnerStrategy`
1	`Dynamic`	الاسم الآلي لحقل أو مسار في المدخل (قد يتبع علاقة)	`DynamicOwnerStrategy`
2	`PreviousOwner`	**معرّف الخطوة** التي يُؤخذ منها المالك	`PreviousOwnerStrategy`
3	`Expression`	تعبير يُرجع مرجعًا (أو مصفوفة)	`ExpressionOwnerStrategy`

تسمياتها في المصمّم: **ثابت**، و**حقل**، و**مالك خطوة سابقة**، و**تعبير برمجي**. ويعرض المصمّم الأربعة كلها لمالك من نوع **مستخدم** أو **مجموعة**، وكلَّ المصادر عدا **حقل** لمالك من نوع **قائمة**. ويستخدم إجراء التفاعل **تغيير مالك الخطوة الحالية** النموذج نفسه دون **مالك خطوة سابقة**.

`source.value` معرَّف بالنوع `ReferenceProperty | string`، ومعناه بحسب تعليق النموذج: مرجع (Fixed)، أو الاسم الآلي لحقل (Dynamic)، أو معرّف خطوة يُستخدم نقطةً مرجعية نسبية (المالك السابق).

> **تنبيه بشأن واجهة وقت التصميم.** في نموذج مالك الخطوة، لا يدعم منتقي كيان المصدر (`entityIdForSourceType`) إلا مصدرَي **User** و**Group**؛ أما المصدران **DataCollection** و**OrgUnit** فدعمهما في واجهة وقت التصميم أضيق نطاقًا.

المالك النسبي (`relativeOptions`)

عند تفعيله، يحلّ مدير المالك المحدَّد محلّه، صعودًا في التسلسل الإداري، ولا ينطبق ذلك إلا حين يكون المالك من نوع **مستخدم**.

الحقل	النوع	ملاحظات
`enabled`	boolean	يفعّل التحديد النسبي.
`relationType`	`StepRelativeRelationTypeEnum`	لا يوجد حاليًا إلا `0 = Manager`.
`relationGrade`	number	عدد المستويات التي يُصعَد إليها، بدءًا من 1.

ينفّذ `OwnerTypeProcessor` ذلك باستدعاء `userService.getUserDirectManager(userId, relationGrade)`؛ فإن لم يُعثر على مدير، يُمسح المالك وتنتقل السلسلة إلى المرشّح أو البديل التالي. فإذا كان «محمد» مثلًا هو المالك الأساسي والعلاقة «مدير» بالدرجة 1، أصبحت «سلمى» (مديرة محمد المباشرة) مالكة الخطوة.

الوضع النسبي للمجموعة (`groupRelativeOptions`)

لا يتضمّن إلا `mode` (`GroupRelativeOptionsModeEnum`): `0 = PerDefinition`، `1 = PerExecution`. ويستخدمه `PreviousOwnerStrategy` للمالكين من نوع DataCollectiong Group:

- **PerDefinition (حسب التعريف)** – يأخذ تعريف المالك كما هو مُعدّ في الخطوة النسبية (ويعمل حتى لو لم تُنفَّذ تلك الخطوة بعد).
- **PerExecution (حسب التنفيذ)** – يأخذ **المنفّذين الفعليين** لتنفيذ الخطوة النسبية أو تنفيذاتها؛ فتصبح قائمة المنفّذين الكاملة هي `originalOwner`. وإذا كانت الخطوة النسبية هي الخطوة الحالية، يُضاف المستخدم الحالي إليها. وإذا كان المالك المحدَّد هو مستخدم النظام، يُمسح المالك. ويسري «حسب التنفيذ» حين يتدخّل مدير، أو تتغيّر عضوية المجموعة، أو تُؤخذ الملكية، أو ينتقل الإسناد الأساسي إلى بديل.

خيارات المالك (`OwnerOptions`)

الحقل	النوع	ملاحظات
`allowChangeOwner`	boolean	يسمح بإعادة إسناد المالك وقت التشغيل.
`mode`	`StepOwnerMode` | null	`0 = SingleUser`، `1 = AllUsers`.
`returnTo`	`TakeOwnershipReturnToModeEnum` | null	لخطوات أخذ الملكية: `0 = Group`، `1 = LastOwner`.
`accessPermission`	`StepOwnerMode` | null	يحدّد مَن يحقّ له الوصول إلى الخطوة. القيمة الافتراضية `SingleUser` (`0`).
`selectionCriteria`	`StepOwnerSelectionCriteria` | null	`0 = First`، `1 = Manual`، `2 = Manager`.
`mainActionExecutionConditionEnum`	`MainActionExecutionConditionEnum`	قاعدة القرار الجماعي: `0 = AllUsers`، `1 = SomeUsers`.
`failureActions`	`StepOwnerFailureActions[]`	الإجراء أو الإجراءات التي تُتّخذ حين يُخفق تصويت SomeUsers في بلوغ العتبة.

الحقل	النوع	ملاحظات
`threshold`	`StepOwnerThreshold` | null	عتبة التصويت (انظر أدناه).
`isParallel`	boolean	الإسناد إلى جميع الأعضاء في آن واحد (خطوة متوازية).

تسميات هذه الخيارات في المصمّم: `mode` هو **اختر نمط المستخدم** (**مستخدم واحد**، **جميع المستخدمين**)؛ و`isParallel` هو **طريقة الإسناد المتعدد** (**متتالي**، **متوازي**)؛ و`selectionCriteria` هو **اختر نمط الانتقاء** (**المستخدم الأول**، **أخذ الملكية**، **مدير وحدة تنظيمية**)؛ و`accessPermission` هو **أذونات الوصول**؛ و`returnTo` هو **إرجاع إلى** (**المجموعة**، **المالك الأخير**)؛ و`mainActionExecutionCondition` هو **شرط تنفيذ الإجراء الرئيسي** (**اختياره من بعض المستخدمين**، **اختياره من جميع المستخدمين**)؛ و`failureActions` هو **الإجراء عند الفشل** (ومن خياراته **الإجراء الثانوي الأكثر اختياراً**)؛ و`threshold` هو **عتبة المتابعة**؛ و`allowChangeOwner` هو **السماح بتغيير المستخدم**؛ و`relativeOptions` هو مفتاح **نسبي** مع **العلاقة** و**درجة العلاقة**.

معايير الاختيار تحكم السلوك بعد التحديد:

- `First (0)` – يُستخدَم أول عضو مؤهّل.
- `Manual (1)` – يُمسح المالك المحدَّد تلقائيًا، ويتسلّم مستخدم مؤهّل الخطوة بالنقر على **أخذ الملكية** في واجهة تفاصيل المدخل.
- `Manager (2)` – عندما يكون المالك من نوع Group، يصبح **مدير الوحدة التنظيمية** للمجموعة هو المالك بدلًا من العضو الأول.

أذونات الوصول تحدّد مَن يرى الخطوة: مع `SingleUser` لا يراها إلا المستخدم المالك، ومع `AllUsers` يراها جميع أعضاء المجموعة. والخطوة التي فيها `selectionCriteria = Manual` و`returnTo = LastOwner` تستعيد آخر مالك لها عند العودة إليها؛ ومع `returnTo = Group` تعود إلى المجموعة.

`Since 1.8` – تخضع قدرة المالك الحالي على إعادة إسناد الخطوة (`allowChangeOwner`، وتسميتها في المصمّم **السماح بتغيير المستخدم**) دائمًا لتصميم سير العمل **الحالي** متى تضمّن هذا التصميم إعدادًا للمالك؛ فإذا عطّلتها، سرى ذلك أيضًا على المدخلات الموجودة على الخطوة بالفعل. ولا يُرجَع إلى لقطة `ownerDefinition` المثبّتة على المدخل إلا حين لا يتضمّن تصميم الخطوة الحالي أي إعداد للمالك.

خطوات المجموعة بلا مالك `Since 1.8`

الخطوة التي يكون مالكها من نوع **Group** أو **DataCollection** مع `mode = SingleUser` و`selectionCriteria = Manual` مصمَّمة لتبقى **بلا مالك**: إذ يمسح المحرّك المالك المحدَّد تلقائيًا ويحتفظ بالمجموعة المؤهّلة في `originalOwner` الخاص بالخطوة، فيستطيع أي عضو أن يتولّى ملكيتها لاحقًا عبر **أخذ الملكية** (انظر تنفيذ إجراءات سير العمل [استخدام أسس › تنفيذ إجراءات خطوات سير العمل]).

ويمكن أيضًا **حفظ هذه الحالة على خطوة كان لها مالك**: فأمر تغيير المالك الداخلي يقبل مالكًا `null`، والمعالِج الذي يطبّقه يتخطّى البحث عن المالك والتحقق منه، ويمسح `owner` الخاص بالخطوة الحالية بدلًا من رفض الطلب.

ويبقى تعريف مالك الخطوة متزامنًا مع المالك. إذ تُكتب لقطة `ownerDefinition` على الخطوة إلى جانب المالك الجديد، لأن فحص أهلية أخذ الملكية **يعطي الأولوية** لهذه اللقطة عند قراءة `mode` و`selectionCriteria`، ولا يرجع إلى التصميم الثابت لسير العمل إلا حين لا تتضمّن الخطوة لقطةً أصلًا.

من أين يأتي المالك `null`. نقطة نهاية تغيير المالك وقت التشغيل (`PATCH /v2/…/change-owner`، أدناه) لا ترسله أبدًا: فحذف `owner` من متن الطلب يجعل الاستدعاء في وضع أخذ الملكية، الذي ينتهي دائمًا إلى إسناد الملكية إلى المستخدم

المستدعي. أما المستدعي الذي تحقّقنا منه فهو إجراء التفاعلات والأتمتة **تغيير مالك الخطوة الحالية** – إذ يشغّل مسار تحديد المالك نفسه الموصوف أدناه، فإذا انتهى إلى `originalOwner` جماعي بلا مالك فردي (حالة Manual أو التوازي) ترك الخطوة بلا مالك بدلًا من تخطّي المدخل.

ويتيح سجل إعادة إسناد المالك المحفوظ مع حدث تغيير مالك الخطوة أن تكون قيمة كلٍّ من المالك القديم والجديد `null`.

تصويت القرار الجماعي والعتبات

حين تكون الخطوة مملوكة لـ**جميع مستخدمي** مجموعة، يتطلّب الانتقال إلى الخطوة التالية أن ينفّذ الإجراءَ الأساسي عددٌ كافٍ منهم، وفق `mainActionExecutionCondition`:

- `AllUsers` – إذا اتخذ أي مستخدم إجراءً غير أساسي، كانت النتيجة إجراء إخفاق.
- `SomeUsers` – يُقارَن عدد مرات تنفيذ الإجراء الأساسي بـ `threshold`؛ فإن كان دون العتبة، طُبّقت `failureActions`. وما إن يُبلَغ حدّ العتبة حتى يتوقف إرسال الإجراء إلى بقية الأعضاء.

يمكن أن تكون `failureActions` الإجراءَ الثانوي الأكثر اختيارًا، أو الإجراء الأساسي، أو عودةً، أو رفضًا؛ وإذا أخفق الشرط ولم يوجد إجراء إخفاق صالح، نُفّذ أول إجراء ثانوي اختاره أول مستخدم.

العتبة (`StepOwnerThreshold`)

الحقل	النوع	ملاحظات
`type`	`ThresholdValueTypeEnum`	`0 = Percentage` (القيمة 1-100)، `1 = Number` (عدد المستخدمين). لا يعرض **نوع عتبة المتابعة** في المصمّم حاليًا إلا **قيمة ثابتة**؛ ويمكن ضبط عتبة النسبة المئوية عبر API.
`value`	number	نسبة مئوية أو عدد مستخدمين بحسب `type`.

شرط الانتقال	المستخدمون الذين ينفّذون الإجراء الأساسي	الإجراء المنفَّذ
موافقة جميع المستخدمين	بعض المستخدمين فقط	إجراء الإخفاق
موافقة جميع المستخدمين	جميع المستخدمين	الإجراء الأساسي
موافقة بعض المستخدمين	العتبة أو أكثر	الإجراء الأساسي
موافقة بعض المستخدمين	أقل من العتبة	إجراء الإخفاق

المالكون البدلاء (الاحتياطيون)

`fallbackOwner` قائمة مرتّبة من تعريفات المالك الاحتياطية، تُجرَّب واحدًا تلو الآخر حين يتعذّر إسناد المالك الأساسي – كأن يكون المرجع غير مؤهّل، أو حساب المستخدم معطَّلًا، أو يتعذّر العثور على مدير نسبي. ويُعَدّ كل بديل بالبنية نفسها التي يُعَدّ بها المالك الأساسي، ويمكن تكرار ذلك حتى يُحَلّ إلى مستخدم ثابت. وتظهر البدائل في المصمّم على هيئة أقسام باسم **إسناد بديل** 1، و2، …؛ ويُضاف أحدها تلقائيًا، ويصبح مطلوبًا، كلما كان المالك الأساسي غير مستخدم ثابت بلا خيار **نسبي** (أي مالكًا من حقل، أو من خطوة سابقة، أو بتعبير، أو مجموعة، أو قائمة، أو مستخدمًا ثابتًا يُصعَد منه إلى مديره). وتنتهي السلسلة عند أول بديل يكون **مستخدمًا** ثابتًا؛ وإذا انتهت الخطوة دون مالك صالح، أخفق الانتقال.

تحديد مالك الخطوة (السلوك الملحوظ)

المحرّك الذي يحوّل هذا الإعداد إلى مالك موثّق بالكامل في المحرّك: سير العمل وتحديد المالك (سلسلة المعالِجات، واستراتيجيات المصدر، والمعالجة اللاحقة، ومدقّق الأهلية). أما السلوك الذي يستطيع المُكامِل ملاحظته فهو:

- **ترتيب الأسبقية.** تتقدّم إعادةُ الإسناد **الدائمة** (`permanentChange`) في تغيير المالك) على كل شيء؛ تليها خطوةُ أخذ الملكية التي فيها `returnTo = LastOwner` فتستعيد آخر مالك لها؛ ثم المالكون المتبقّون في خطوة جميع المستخدمين؛ أما خطوة **البداية** فيملكها دائمًا منشئ المدخل؛ وفيما عدا ذلك يُجرَّب `owner` الخاص بالخطوة، ثم كل `fallbackOwner` بالترتيب، ويُعتمد أول ما يُنتج مالكًا صالحًا.
- **المدير النسبي.** مع `relativeOptions` على مالك من نوع User، يحلّ محلّ المستخدم المحدَّد مديرُه الواقع على بُعد `relationGrade` من المستويات صعودًا؛ فإن لم يوجد مدير، أُسقط المرشّح وجُرّب البديل التالي. ولا يُطبَّق ذلك على خطوة النهاية.
- **مصدر التعبير** يُقيَّم ضمن نطاق يضمّ `entry` و`dataEntity` و`currentUser` و`currentStepId` و`history` و`workflow`؛ ولا يُحتفظ إلا بالمراجع الصالحة.
- **مالك خطوة سابقة** على خطوة أخذ ملكية يرث المستخدم الذي تولّى ملكيتها فعلًا (`Since 1.8`)، لا المجموعة التي عُرضت عليها.
- **الأهلية.** يُتخطّى المرشّح (ويُجرّب البديل التالي) إذا كان المستخدم أو المجموعة أو الوحدة التنظيمية المُشار إليها في حالة أرشفة، أو كان المستخدم معطَّلًا، أو كان المرجع سجلًا متقادمًا. ويُتحقَّق من أهلية المالك من نوع **قائمة** بالرجوع إلى القائمة نفسها. والمالك من نوع **مجموعة** أو **قائمة** الذي يتوسّع إلى **صفر أعضاء مؤهّلين** ينتقل إلى المالكين البدلاء – إلا مع اختيار **مدير وحدة تنظيمية**، الذي يأخذ المدير لا الأعضاء.
- **انعدام المالك.** ما لم تكن الخطوة خطوة النهاية، فإن الخطوة التي لا يُحدَّد لها مالك تُفشل الانتقال (`OriginalOwnerNotFoundException`). أما خطوات النهاية فيجوز أن تبقى بلا مالك.

المالكون المؤهّلون لخطوة بلا مالك `Since 1.6`

يستمدّ منتقي تغيير المالك بياناته من `eligible-owners` (انظر سطح API للمطوّرين)، الذي يحصر المرشّحين المتاحين للمستدعي غير المسؤول في `originalOwner` الخاص بالخطوة. أما الخطوة المملوكة لمجموعة ثابتة **لا أعضاء فيها** – حين يُمسح `originalOwner` إلى `null` (خطوة مجموعة بلا مالك)، أو يكون مرجع المجموعة فارغًا – فتُعامَل معاملة المالك من مستخدم واحد: فحين يكون `originalOwner` معدومًا، أو لا يوجد أعضاء في المجموعة أو مجموعة البيانات، يسرد المنتقي كل مستخدم **نشط** – وهي بالضبط مجموعة المستخدمين التي تقبلها عملية تغيير المالك. أما المجموعة التي لها أعضاء كلهم غير مؤهّلين فتظل تُنتج قائمة فارغة. وقد أُدرج هذا السلوك في خط 1.6 أيضًا ضمن تحديث.

`Since 1.10` – يُردّ على أربعة أخطاء في تصميم مالك الخطوة (`FallbackOwnerLastItemOwnerType`، `PreviousOwnerStepIsExitStep`، `PreviousStepNotFound`، `RelativeOptionsOwnerType`) بالحالة **400**، بوصفها أخطاء تحقّق تحدّد المشكلة.

ملكية المجموعة على خطوة مملوكة لمجموعة

ينطبق على خطوط 1.4 و1.6 و1.8 على حدّ سواء – إصلاح لصحة السلوك، لا ميزة مرتبطة بإصدار.

وُصف `originalOwner` أعلاه بأنه مجموعة المالكين المؤهّلين كاملة، وهو في خطوة من نوع Group أو DataCollection عادةً مرجعُ المجموعة نفسها. لكن المحرّك، في كل خطوة مجموعة غير متوازية وغير يدوية، **يوسّع المجموعة إلى أعضائها** ويُبقي أحدهم مالكًا فعليًا – فحين تُقرأ خطوة مكتملة لاحقًا، قد يحمل `originalOwner` ذلك العضو وحده دون أي مرجع للمجموعة.

لذلك تستعيد المنصة المجموعة من المالك **المُعدّ** للخطوة: فحين لا يعود `originalOwner` يشير إلى مجموعة أو مجموعة مستخدمين أو دور أو وحدة تنظيمية أو مساحة أو مجموعة بيانات، ترجع عملية التحديد إلى `source.value` في تعريف المالك – أي المجموعة التي أُعدّت بها الخطوة.

ولا يعمل هذا الرجوع إلا مع مصدر مالك من نوع **`Fixed`**؛ أما المصدر `Dynamic` أو `Expression` فيخزّن في `source.value` مسار حقل أو نص تعبير، لا مرجع مجموعة؛ لذا لا تُستعاد أي مجموعة، ولا تُجرى أي محاولة للرجوع معهما.

وبهذا يظل أعضاء المجموعة الذين لم يتخذوا إجراءً ظاهرين في لوحة سير العمل لخطوة مكتملة مملوكة لمجموعة.

مراقبة SLA وزمن الخطوة (`WorkflowStepTimeMonitoring`)

يمكن ضبط **اتفاقية مستوى خدمة (SLA)** لكل خطوة، تحدّد مهلة زمنية تُعَدّ الخطوة بعدها متأخرة، مما يتيح مراقبة المدة التي تستغرقها الخطوات والتعامل مع التأخير. وتُضبط إعدادات توقيت كل خطوة في `WorkflowStep.timeMonitoring` .

الحقل	النوع	ملاحظات
`enabled`	boolean	يفعّل مراقبة الخطوة.
`workingTimeReference`	nullable `ReferenceProperty`	مرجع إلى تعريف ساعات العمل / التقويم المستخدَم في حساب الوقت المنقضي.
`monitoringUnit`	`TimeMonitoringUnitEnum` | null	`0 = Minutes` ، `1 = Hours` ، `2 = Seconds` .
`value`	number | string | null	زمن التنفيذ: عدد أو تعبير أو حقل مصدر بحسب `mode` .
`fallbackValue`	number	يُستخدَم حين يتعذّر تحديد `value` .
`mode`	null | `TokenSourceType`	طريقة الحصول على `value` : `0 = Fixed` ، `1 = Field` ، `2 = Expression` ، `3 = Pointer` . الافتراضي `Fixed` .

يُحسب تاريخ الاستحقاق عبر **خدمة ساعة** خارجية تراعي ساعات العمل والعطلات المُعدّة، فيستثني العدّ التنازلي الوقت خارج الدوام (وتُعرَّف ساعات العمل والعطلات نفسها بوصفها بيانات في المنصة). وعلى مستوى المنصة، يحفظ `workflow-service` سجلّ SLA لكل خطوة مراقَبة (`WorkflowSlaConfig`)، وينسخ فيه `timeMonitoring` (`timeToExecute`) إلى جانب `module` و `entity` و `workflowId` و `stepId` . ولا توجد نقطة نهاية عامة لـ SLA؛ إذ تظهر تواريخ الاستحقاق التي جرى تجاوزها في السجل التاريخي للمُدخل على هيئة عناصر من النوع `SlaOverdue` – انظر مرجع واجهة برمجة السجل التاريخي والشريط الزمني.

`Since 1.8` – يقتصر إغلاق سجلّ SLA للخطوة عند الخروج منها على **المدخل الذي خرج تحديدًا**، ولا يُطبَّق على الخطوة وحدها.

`Since 1.10` – ينطبق التغييران التاليان على طريقة حفظ سجلات SLA (ولا يغيّر أيٌّ منهما معنى *Pending* و*Achieved* و*Overdue* للخطوة التي صمّمتها):

- **الـ SLA المُحقَّق يبقى محقَّقًا.** ينقل فحصُ التأخر السجلاتِ واحدًا تلو الآخر، ولا ينقل السجل إلا ما دام معلّقًا، ولا يسجّل مخالفة إلا إذا وقع ذلك الانتقال فعلًا؛ فالسجل الذي صار *Achieved* أثناء الفحص لا يُعاد تسجيله بحالة *Overdue* أبدًا، ولا تُسجَّل مخالفة زائفة.
- **المدخل الذي لا خطوة نشطة له يُتخطّى** في كلٍّ من الفحص الدوري للتأخر والمعالِج الذي يبدأ ساعة الخطوة، دون أن تتأثر المدخلات الأخرى.

الخطوات المخصّصة

تتيح الخطوات المخصّصة (Ad hoc) للمستخدم وقت التشغيل إدراج خطوة موافقة أو معلومات إضافية بعد خطوة ضُبط فيها `allowAddAdHoc = true` . وتكون الخطوة المُدرَجة خطوة فرعية تابعة للخطوة الحالية؛ يتوقف سير العمل عندها حتى يتصرّف مالكها، ثم يستأنف الخطوة الأصلية تلقائيًا. ويُختار مالكها باستخدام خيارات اختيار المالك نفسها الموصوفة أعلاه.

`CreateAdhocWorkflowStepDto` (منسوخ دون اختلاف في `data-service` و `workflow-service`، وتُستخدَم النسخة في الأخير متنًا للطلب العام إلى `POST …/adhoc-step` أدناه):

الحقل	النوع	ملاحظات
`adhocStepType`	`AdhocStepTypeEnum`	`0 = AdditionalApproval`، `1 = AdditionalInformation`.
`owner`	`WorkflowStepOwner`	تعريف مطلوب لمالك الخطوة المُدرَجة.
`fallbackOwner`	`WorkflowStepOwner[]`	بدائل اختيارية.
`notes`	string	اختياري.
`description`	string	ملاحظة نصية حرّة اختيارية توضّح سبب إضافة الخطوة. `Since 1.8`

`description` `Since 1.8`. تُثبَّت القيمة النصية الحرّة الوحيدة على الخطوة المُنشأة، وتُنسخ في كلٍّ من `en` و`ar` ضمن `NullableLocalizedValue` الخاص بها (إذ لا يجمع مربع حوار الإنشاء إلا قيمة واحدة، ولا تُختار القيمة بحسب لغة المشاهد)، وتبقى `undefined` إن لم تُقدَّم قيمة. ويعرضها التلميح الخاص بكل خطوة. راجع إضافة خطوة مخصّصة [استخدام أسس › تنفيذ إجراءات خطوات سير العمل] للاطلاع على ما يراه المستخدم النهائي.

متطلبات حقول سير العمل

تضبط هذه الإعدادات سلوك حقول الكيان في كل خطوة، حيث تختار لكل حقل خيار عرض واحدًا:

1. **مخفي** – لا يظهر الحقل في هذه الخطوة إطلاقًا.
2. **مقفل** – يظهر الحقل لكن لا يمكن تعديله.
3. **قابل للتغيير** – يمكن تعديل الحقل خلال هذه الخطوة.
4. **مطلوب** – يجب ملء الحقل قبل أن يتقدّم سير العمل.

ففي سير عمل لتقرير مثلًا، قد يكون حقل «تاريخ الإنشاء» مخفيًا عند إدخال البيانات ومقفلًا عند المراجعة، بينما يكون حقل «تقديم التقرير» مطلوبًا في خطوة التأكيد النهائي.

يُخزَّن هذا الربط على **الحقل** لا على الخطوة؛ إذ يحمل كل حقل جدولًا اختياريًا لكل سير عمل ولكل خطوة (`RequirementSettingEnum`: `Hidden`، `Locked`، `Editable`، `Required` – وتسمية الأخير في المصمّم **مطلوب**). وإذا لم يحمل الحقل قيدًا لخطوة ما، عُدّ قابلًا للتغيير فيها. أما حقل **الكتلة** فيمكن أن يكون إعداده الخاص بكل خطوة `PerBlock` بدلًا من ذلك، فيُحيل إلى متطلب **الكيان الداخلي** للكتلة على مستوى الحقل للخطوة نفسها (ويعود إلى قابل للتغيير إذا لم يكن للكيان الداخلي أي ربط بسير العمل النشط أصلًا).

أي خطوة تنطبق متطلباتها على المشاهد

تُحدَّد المتطلبات **لكل مشاهد**، لا لكل مدخل فحسب، لأن شخصين يطّلعان على المدخل نفسه قد يرتبط كلٌّ منهما بخطوة مختلفة منه: مالك يتصرّف في الخطوة الحالية، وشخص يقرأ المدخل لاحقًا. وتُحدَّد الخطوة المستخدَمة بهذا الترتيب:

1. مصدر المالك السابق لخطوة **مخصّصة**، إذا كانت الخطوة التي يملكها المشاهد حاليًا هي تلك الخطوة المخصّصة نفسها.
2. الخطوة التي **يملكها حاليًا** المشاهد (أو مجموعته أو قائمته).
3. آخر خطوة **غير مخصّصة** نفّذها المشاهد (أو استطاع الوصول إليها عبر مجموعته أو قائمته).
4. `Since 1.8` – إذا تعذّر كل ما سبق، تُستخدَم الخطوة **الحالية الفعلية** للمدخل، أيًّا كان مالكها.

والبند 4 هو الذي يطبّق المتطلبات – بما فيها الإخفاء – على مشاهد لم يملك أي خطوة في سير العمل ولم ينفّذها قط، كطالبٍ يقرأ مدخله بعد انتقاله إلى **تدفق فرعي** اقتصر دوره على إطلاقه.

تبقى الحقول المخفية مخفية بعد العودة أو الرفض `Since 1.8`

لا يحمل نوعان من الخطوات متطلبات حقول إطلاقًا؛ إذ يستبعد منتقي المتطلبات خطوة **النهاية** وخطوات **التدفق الفرعي**، لأن أيًّا منهما لا يمثّل موضعًا مناسبًا يضبط فيه المسؤول الإخفاء أو الإلزام أو القفل. غير أن **الرفض** و**العودة** التي تُنهي تدفقًا فرعيًا قد يوقفان المدخل على إحدى هاتين الخطوتين تحديدًا: فالرفض يرسل المدخل إلى خطوة النهاية الرئيسية في سير العمل، والعودة التي تُغلق تدفقًا فرعيًا تتركه على خطوة النهاية في ذلك التدفق الفرعي. ولأن خطوة كهذه، حين تُحَلّ مباشرةً، لا تحمل أي ربط للمتطلبات، تطبّق المنصة آلية احتياطية **للإخفاء وحده**: فحين لا يكون لخطوة المدخل الفعلية ربط للمتطلبات، يبقى الحقل مخفيًا إن كان مخفيًا في **أي** خطوة مرّ بها المدخل فعلًا في سير العمل ذاك – بما في ذلك تدفق فرعي أُغلق منذ ذلك الحين. وتقتصر هذه الآلية عمدًا على استعادة `Hidden`: فلا يُطبَّق بهذه الطريقة على المدخل المكتمل أو المرفوض أي إعداد `Locked` / `Mandatory` قديم.

القراءات بشكل القائمة تبلّغ عن الحقول المخفية لكل صف عبر `hiddenFields` `Since 1.8`

تعالج الآليات السابقة الإخفاء بتعديل **تعريف الكيان** الذي تُسلسَل الاستجابة وفقه، وهذا يصلح لقراءة مدخل واحد، حيث يصف تعريف كيان واحد صفًا واحدًا بالضبط. لكنه لا يستطيع التعبير عن حالة **خاصة بكل صف**: فالقراءة بشكل القائمة (Table وBadge وMiniCard وKanban وTree) تخدم صفوفًا كثيرة باستخدام **تعريف كيان واحد مشترك**، وقد تختلف الحقول المخفية بين صفّين يقفان على خطوتين مختلفتين من سير العمل، ويكون هذا الاختلاف صحيحًا.

لذلك يبلّغ `DataEntryService` عن هذه الحالة الخاصة بكل صف على نحو منفصل، في مصفوفة جديدة خاصة بالاستجابة `hiddenFields?: string[]` يحملها كلا نموذجي قراءة القائمة (`DataEntry`، `DataEntryV2Body`)؛ وتضم قيم `fullNormalizedName` للحقول التي يجب ألا تُعرض على القارئ الحالي **في هذا الصف**. وتشمل هذه الآلية جميع مسارات القراءة بشكل القائمة (`findWithResolves`، `findKanbanWithResolves`، `findTree`)، وتُتخطّى في معارض `viewAsAdmin` / `editAsAdmin` ولمستدعي النظام، وتعتمد الإخفاء عند الإخفاق: فإن تعذّر تحديد الحقول المخفية لصف ما، أُبلغ بأن جميع حقول ذلك الصف مخفية، بدلًا من الإبلاغ عن عدم وجود حقول مخفية.

حماية التزامن في عمليات الكتابة على سير العمل `Since 1.10`

تُشترط في كل عملية كتابة على سير العمل **مطابقة حالة سير العمل التي جرى التحقق منها**، سواء أكانت تنفيذ إجراء (بما في ذلك الفروع التي تبدأ تدفقًا فرعيًا أو تُكمله)، أم إدراج خطوة مخصّصة، أم تغيير مالك الخطوة، أم تغيير الخطوة. لذا، إذا تسابق استدعاءان على مدخل واحد، سُجّل انتقال واحد ونُشرت `WorkflowActionExecuted` / `WorkflowStepExited` / `WorkflowStepReached` مرة واحدة، ورُفض الاستدعاء الخاسر بدلًا من السماح له بالكتابة فوق نتيجة الآخر بصمت. وتشمل الحالة المحمية الخطوة الحالية، ونسخة سير العمل، والمالك، وعدد المنفّذين، وطول السجل التاريخي للخطوات، لكل نسخة سير عمل على المدخل. وهذا ما يراه المُكامِل:

- **يُجاب عن الاستدعاء الخاسر في السباق بـ `409 UPDATE_CONFLICT`**، ولا يُسجَّل له شيء ولا يُنشر. وتُمرَّر الحالة 409 إلى المستدعي كما هي (ويُجاب بـ 502 عن الإخفاق الوارد من الخدمة الأصلية دون حالة محدّدة)، فيستطيع العميل تمييز حالة «سبقك أحدهم إلى نقله» من الخطأ الحقيقي: أعد تحميل المدخل وقرّر من جديد. ويعرض عميل الويب مربع حوار تعارض التحديث.
- **الكتابة جزئية.** لا تُضبط أو تُزال إلا المفاتيح التي غيّرها الانتقال، فلا يضيع تعديل متزامن على حقل خارج سير العمل.

- **تقوم الحماية على حالة سير العمل، لا على `updateIdentifier`.** فعمليات الكتابة التي لا علاقة لها بسير العمل تعيد ختم `updateIdentifier` بصورة مشروعة (كعودة الخطوة الخارجية ولقطات التفاعلات)، فلا يكفي وحده لكشف سباق على سير العمل؛ وتتحقّق عملية تغيير المالك، إضافةً إلى ذلك، من مطابقة قيمة `updateIdentifier` التي حمّلتها.
- **يبقى إثبات التحقق الإضافي صالحًا بعد خسارة السباق.** يُستهلَك الإثبات قبل حساب الانتقال؛ فإذا خسرت الكتابة المحمية، أُعيد للمدة المتبقّية من صلاحيته كي يعيد المستدعي المحاولة. أما الإثبات غير المطابق فيُبطَل كالمعتاد.
- **ترتيب الأحداث.** ينشر فرع تغيير الخطوة الذي يستهدف تدفقًا فرعيًا أحداث خطواته بعد الكتابة المحمية.
- **الأرشفة والاستعادة** تتّبعان النمط نفسه بالاستناد إلى حالة دورة حياة المدخل: فلا تُنفَّذ كتابة الأرشفة إلا ما دام المدخل نشطًا، ولا تُنفَّذ كتابة الاستعادة إلا ما دام مؤرشفًا. فإذا تسابقت عمليتا أرشفة، نُشر `DataEntryArchived` مرة واحدة. ويُجاب عن العملية الخاسرة – مثل أي أرشفة لمدخل مؤرشف أصلًا – بـ **422** `ENTRY_ARCHIVED`؛ ويُجاب عن الاستعادة الخاسرة بـ `ENTRY_NOT_ARCHIVED`، ويُبلَّغ عن الكيان المؤرشف بالحالة 422 كذلك.
- **لا تستطيع لقطة المدخل القديمة التراجع عن نقل خطوة** `Since 1.4`. فالحفظ الذي لا يُسمح له بتجاوز بيانات سير العمل الوصفية يُبقي `_metaCurrentSteps` و`_metaWorkflowHistory` و`_metaAdhocSteps` و`_metaOrder` و`lifeCycleStatus` كما خُزّنت، ويبدأ نقل الخطوة (كتفاعل **نقل سير العمل إلى الخطوة**) من حالة سير العمل المخزّنة للمدخل، لا من لقطة الحدث المُطلِق. ونتيجةً لذلك، يرفض هذا الحفظ أي مفتاح قيمة يحتوي `.` أو يبدأ بـ `$` (`EXTRA_ENTRY_VALUE`). أما **الخطوة المخصّصة** فما زالت تُحسب من لقطتها؛ فتُخفق بتعارض بدلًا من إعادة الحالة إلى الوراء.
- **يُجاب عن حالتي رفض بـ 422** `Since 1.8`. تنفيذ إجراء مخفي على المدخل (`WORKFLOW_ACTION_EXECUTION_NOT_ALLOWED`) وأخذ ملكية خطوة لها مالك أصلًا (`CURRENT_STEP_ALREADY_OWNED`) حالتا رفض يُجاب عنهما بالحالة **422** مع رمزَي الخطأ هذين. اعتمد في المطابقة على `errorCode` لا على الحالة.

مثال عملي في صورة تعريف: اعتماد مصروف من خطوتين

فيما يلي العملية نفسها التي يشرحها دليل البناء في المصمّم، بصياغة تستند إلى مصطلحات النموذج: سير عمل مبسّط لاعتماد المصروفات على كيان `Expense` (حقوله: المبلغ، والوصف، ومرجع `submitter`، ومرجع القسم).

1. **البداية (Entry) – "مُقدَّم".** يشير مصدر المالك **Dynamic** إلى الحقل `submitter` في المدخل، فتُسنَد ملكية كل مدخل ديناميكيًا إلى مَن قدّمه. وتتضمّن هذه الخطوة إجراءً أساسيًا واحدًا **تقديم** ← **الخطوة التالية** النسبية.
2. **وسيطة (Intermediate) – "مراجعة المدير".** المالك هو **مدير** مقدّم الطلب: مصدر ضُبطت فيه `relativeOptions.enabled = true` و`relationType = Manager` و`relationGrade = 1`، فيصعد المحرّك مستوى واحدًا في الهيكل التنظيمي. اضبط `commentValidationMode = Required` للإجراء كي يُلزَم المراجع بكتابة ملاحظة، وأضف إلى الخطوة **SLA** مدته **يوما عمل** (`timeMonitoring`، يُحسب بالاستناد إلى مرجع لساعات العمل فتُستثنى عطلات نهاية الأسبوع والعطلات الرسمية). الإجراءات: **اعتماد** ← التالية؛ **رفض** ← إجراء عودة (الخطوة السابقة النسبية، رجوعًا إلى «مُقدَّم»).
3. **النهاية (Exit) – "معتمَد".** خطوة ختامية؛ لا تحتاج إلى مالك.
4. **التصعيد عند المخالفة.** اربط تفاعل سير عمل بالحدث **SlaTimeExceeded** لخطوة «مراجعة المدير» ليصعّد الأمر إلى **مدير المدير** – بإشعار مثلًا، مع إعادة إسناد ملكية الخطوة اختياريًا (مدير نسبي بالدرجة 2).
5. **صيغة متوازية N من M (اختيارية).** لاشتراط عدة مراجعين بدلًا من مراجع واحد، اجعل «مراجعة المدير» خطوة من نوع **Group** فيها `mode = AllUsers` و`isParallel = true` و`mainActionExecutionCondition = SomeUsers` و`threshold` (مثلًا 2 من 3). وأضف عنصرًا إلى `failureActions` (الإجراء الثانوي الأكثر اختيارًا) لحالة عدم بلوغ العتبة.

قواعد وقت التصميم التي تُفرض عند الحفظ

عند حفظ سير العمل، تُفحص هذه القواعد:

- يجب أن يحتوي سير العمل على **خطوتين على الأقل**، **أولاهما من نوع Entry** و**آخرتهما من نوع Exit**.
- **تحتاج كل خطوة إلى إجراء واحد على الأقل**، باستثناء خطوة النهاية وخطوات سير العمل الفرعي.
- **لا يجوز أن يكون للخطوة أكثر من إجراء أساسي (رئيسي) واحد.**
- **يجب أن تشير خطوة التدفق الفرعي إلى تدفق فرعي موجود وأن تحدّد خطوة دمج، وألّا تحمل أي إجراءات خاصة بها.**
- **يجب أن تكون أسماء الخطوات وأسماء عمليات سير العمل فريدة** ضمن نطاقها.
- في تفاعلات أتمتة سير العمل: يجب أتمتة إجراء واحد على الأقل من إجراءات خطوة، ولا يمكن أتمتة الخطوة الأخيرة (Exit).

سطح API للمطوّرين

تُحرَّر تعريفات سير العمل وقت التصميم ضمن **كيان البيانات** (المصفوفة `workflows`)؛ فلا توجد نقطة نهاية مستقلة لـ«إنشاء سير عمل». وتوفّر نقاط النهاية أدناه واجهات الاستعلام ووقت التشغيل، أما دلالات التنفيذ وقت التشغيل (قواعد التنقّل، ومن يحقّ له التصرّف) فتغطّيها صفحة تنفيذ إجراءات سير العمل [استخدام أسس › تنفيذ إجراءات خطوات سير العمل].

`WorkflowController` — `workflow-service`

انتقالات وقت التشغيل الموجّهة إلى المستخدم (في سياق المالك):

الطريقة والمسار	الغرض
`PATCH /v1/entities/{entity}/entries/{entry}/{workflowId}/execute-action`	ينفّذ إجراء سير عمل (انتقالًا) على مدخل. ويتضمّن الطلب الحقل `stepUpProof` حين يشترط الإجراء التحقق الإضافي. `Since 1.10`
`PATCH /v1/entities/{entity}/entries/{entry}/workflow/{workflowId}/change-step`	ينقل المدخل إلى خطوة أخرى (`ChangeWorkFlowStepDto`).
`PATCH /v2/entities/{entity}/entries/{entry}/{workflowId}/change-owner`	يعيد إسناد ملكية الخطوة (`ChangeStepOwnerDto`).
`POST /v1/entities/{entity}/entries/{entry}/{workflowId}/adhoc-step`	يُدرج خطوة مخصّصة (`CreateAdhocWorkflowStepDto`).
`GET /v1|v2/entities/{entity}/entries/{entryId}/main-flow-status`	يعيد تقدّم المدخل على المسار الرئيسي.
`GET /v1/entities/{entity}/workflow/{scope}`	يعيد سير العمل (التعريف / المعاينة) لنطاق معيّن.

الطريقة والمسار	الغرض
`POST /v1/entities/{entity}/{workflowId}/{actionId}/bulk`	ينفّذ إجراءً واحدًا على كل مدخل في `entryIds` (`BulkExecuteWorkflowActionRequestDto` : `{entryIds, notes, stepUpProofs?}` – و `stepUpProofs` خريطة `{entryId: proof}` ، تتضمّن إثباتًا أحادي الاستخدام لكل مدخل خاضع للتحقق، وتصدر الإثباتات كلها عن تحقق واحد، `Since 1.10` ، انظر التحقق الإضافي قبل تنفيذ الإجراء)؛ ويجب أن تكون المدخلات كلها حاليًا على الخطوة نفسها، وإلا أُطلق `WorkflowStepsNotSameException` . يعيد خريطة نتائج لكل مدخل.

تغيير الخطوة (`ChangeWorkFlowStepDto`) – `stepId` (UUID أو سلسلة فارغة) إضافةً إلى `changeStepMode` (`ChangeStepModeEnum`): `0 = Normal` ، `1 = ReturnToStep` ، `2 = MoveToStep` ، `3 = ResetWorkflow` ، `4 = EndWorkflow` ، `5 = AdHoc` ؛ و `notes` اختياري.

تغيير المالك (`ChangeStepOwnerDto`) – `owner` (`UserReferenceProperty`)، و `note` (الافتراضي `''`)، و `permanentChange` (boolean). ويقرأ `PermanentChangedOwnerHandler` لاحقًا إعداد `permanentChange` ليُبقي إعادة الإسناد ثابتة كلما عاد المدخل إلى الخطوة.

التنفيذ الجماعي (`BulkExecuteWorkflowActionRequestDto`) – إجراء واحد يُطبَّق على كل مدخل في `entryIds` . يُرسَل كل مدخل إلى ناقل الخدمة، ويُنتظر الردّ عليه بصورة مستقلة، فلا تعطّل نتيجةُ مدخل نتيجةَ غيره؛ وتكون الاستجابة خريطة نتائج لكل مدخل. `Since 1.8` – يُحسم ردّ كل مدخل على نحو مستقل عند الإرسال، فلا يُبلَّغ عن إخفاق مدخل واحد أو انتهاء مهلته إلا بوصفه نتيجةً لذلك المدخل وحده، ولا يؤثر ذلك أبدًا في طلبات سير العمل الأخرى الجارية. راجع التعامل مع عدة مدخلات دفعة واحدة (الإجراء المجمّع) [استخدام أسس › تنفيذ إجراءات خطوات سير العمل] لشرح الاستخدام.

`Since 1.10` – المسار الجماعي مُدرَج في قائمة السماح ضمن **خطة التنفيذ** لكل معرض بيانات، مما يتيح تنفيذ إجراء خطوة على مدخلات محدّدة في قائمة عبر أي معرض. وبصرف النظر عن ذلك، يُرفض هنا دائمًا الإجراء الخاضع للتحقق الإضافي ما لم يحمل إثباتًا صالحًا؛ انظر التحقق الإضافي قبل تنفيذ الإجراء.

مسارات أُضيفت إلى خطة التنفيذ في كل معرض `Since 1.8`

يحمل كل معرض بيانات **خطة تنفيذ**، وهي قائمة السماح بالمسارات التي يجوز لطلب يمرّ عبر ذلك المعرض استدعاؤها. وتوجد هذه المسارات في كل خطة (وقد أُضيفت إلى الخطط القائمة، أما المعرض الذي أخفقت إعادة حساب خطته فيكتسب المسار عند حفظه التالي):

- **بيانات حقول "سؤال المستخدم"** `Since 1.8` (أُتيحت في خط 1.8 بوصفها تحديثًا). يُخاطَب نموذج «سؤال المستخدم» الخاص بزر نظام مرتبط بإجراء مخصّص بوصفه الكيان `TRASK_<entityId>_<triggerId>` ، وتحمّل حقوله المرجعية خياراتها من نقاط نهاية بيانات الحقول (`POST /v2/entities/TRASK_…/fields/{fieldId}/data` و `…/data/count` ، و `GET /v1/entities/TRASK_…/fields/{fieldId}/data`). ويُسمح بنقاط نهاية بيانات الحقول هذه وحدها لكل معرّف `TRASK_` في كل معرض، كي تُحمَّل منتقيات المراجع في النافذة المنبثقة؛ ويظل المعالِج يرفض المشغّل المفقود أو غير الصالح أو المعطَّل.
- **حالة قوالب التصدير** `Since 1.10` . يعيد `POST /v1/entities/{entity}/templates/status/entries` ، لكل مدخل مطلوب، حالةَ كل قالب تصدير مشروط من نوع المدخل الواحد، كي تحدّد أزرار التصدير القوالب التي يعطّلها شرط المدخل أو يخفيها. ولا يُضاف إلا إلى المعارض التي تعرض قالب تصدير. وعند استدعائه عبر معرض، لا يُبلَّغ إلا عن قوالب ذلك المعرض، ولا تُقيَّم الشروط إلا حين يستطيع المعرض قراءة جميع المدخلات المطلوبة؛ وإلا أُبلغ عن كل قالب بأن شرطه غير متحقق، فلا تكشف الاستجابة شيئًا عن صفوف خارج المعرض.

ملاحظات الإصدارات

القدرة	التوفّر
خطوات التدفق الفرعي (`subWorkflowId` ، `mergeStep`)	Since 1.4
الخطوات الهدف الشرطية (`conditionalTargetSteps`)	Since 1.4
الخطوات الخارجية (`StepType.ExternalStep` ، `ExternalStepConfig` ، استدعاء العودة)	Since 1.8
خطوات المجموعة بلا مالك (أمر تغيير مالك يقبل القيمة المعدومة، وإبقاء `ownerDefinition` متزامنًا)	Since 1.8
استعادة ملكية المجموعة على خطوة مملوكة لمجموعة (الرجوع إلى مصدر مالك `Fixed`)	إصلاح لصحة السلوك – السلوك الحالي في 1.4 و1.6 و1.8
رجوع مالك الخطوة إلى `fallbackOwner` حين لا يكون للمجموعة أعضاء مؤهّلون	Since 1.8
الحفاظ على المالك الموروث عند العودة إلى خطوة البداية في تدفق فرعي	Since 1.8
إذن إعادة الإسناد (`allowChangeOwner`) يعكس تصميم الخطوة الحالي لا لقطة مجمّدة	Since 1.8
إغلاق SLA مقصور على المدخل الخارج، لا على الخطوة وحدها	Since 1.8
إرسال الإجراء الجماعي: لم يعد إخفاق مدخل واحد يُسقط العامل	Since 1.8
أحداث الخروج من الخطوات وتنفيذها عند حدود التدفق الفرعي تُنشر بموثوقية وبالترتيب	Since 1.8
المدخل الواقف على خطوة النهاية يُبلَّغ عنه بأنه مكتمل	Since 1.8
متطلبات الحقول: الرجوع أخيرًا إلى الخطوة الحالية للمشاهدين الذين لا سجل خطوات لهم	Since 1.8
استمرار تطبيق متطلبات الإخفاء عند الوصول إلى خطوة النهاية أو خطوات التدفق الفرعي (العودة/الرفض)	Since 1.8
القراءات بشكل القائمة (`findWithResolves` / `findKanbanWithResolves` / `findTree`) تبلّغ عن `hiddenFields` لكل صف	Since 1.8
إضافة حقل `description` نصي حرّ اختياري إلى الخطوة المخصّصة	Since 1.8
التحقق الإضافي على إجراء خطوة (`stepUp` ، `stepUpProof`)، لمدخل واحد	Since 1.10
توسيع التحقق الإضافي ليشمل تنفيذ الإجراء الجماعي (خريطة `stepUpProofs` ، تحقق واحد لكل تحديد)	Since 1.10
إدراج مسار تنفيذ الإجراء الجماعي في قائمة السماح ضمن خطط تنفيذ معارض البيانات	Since 1.10
لم يعد SLA الذي يتحقق خلال نافذة الفحص يُسجَّل مخالفةً	Since 1.10
لم يعد غياب الخطوة النشطة يُجهض دورة فحص SLA أو يمنع ساعة SLA للخطوة	Since 1.10
السماح بمسارات بيانات حقول "سؤال المستخدم" (`TRASK_*`) في خطة تنفيذ كل معرض	Since 1.8
منتقي تغيير المالك يسرد كل مستخدم نشط لخطوة بلا مالك	Since 1.6
عمليات الكتابة على سير العمل مشروطة بحالته المتحقَّق منها؛ ويُجاب عن الاستدعاء الخاسر في السباق بـ 409 `UPDATE_CONFLICT`	Since 1.10
مسار حالة قوالب التصدير في خطط تنفيذ المعارض	Since 1.10
يُجاب عن أخطاء تصميم مالك الخطوة بـ 400؛ وعن أرشفة مدخل مؤرشف بـ 422 `ENTRY_ARCHIVED`	Since 1.10
يُجاب عن تنفيذ إجراء مخفي وأخذ ملكية خطوة مملوكة بـ 422 (`WORKFLOW_ACTION_EXECUTION_NOT_ALLOWED` ، `CURRENT_STEP_ALREADY_OWNED`)	Since 1.8
حفظ المدخل يُبقي حالة سير العمل المخزّنة؛ ونقل الخطوة يبدأ من الحالة المخزّنة، فلا تستطيع لقطة قديمة التراجع عنه	Since 1.4
المالكون البدلاء المرتّبون، والخطوات المتوازية، والجهة التي تعود إليها خطوة أخذ الملكية، ومراقبة زمن كل خطوة، وتصويت القرار الجماعي، والأوضاع النسبية للمجموعة، والخطوات المخصّصة	Long-standing
محرّك تحديد مالك الخطوة (Chain + Strategy، ومصدر Expression)	Long-standing (سطح الإعداد)؛ وقد أُعيد تنظيم المحرّك داخليًا في إصدار لاحق

مرجع واجهة معارض البيانات البرمجية

تستهدف هذه الصفحة **مطوّري التكامل والمطوّرين** الذين يقرؤون معارض البيانات أو يكتبونها عبر الواجهة البرمجية، أو يستدعون مسارات لوحة المهام أو المرشحات السريعة أو التصدير، أو يحتاجون إلى معرفة القواعد الدقيقة التي يطبّقها الخادم عند حفظ المعرض. كما توثّق سجلّ التغييرات في مصمّم المعرض، وتتّبع أقسامها ترتيب أقسام المصمّم.

تصمّم معرضًا من الواجهة؟

ينبغي لبُناة التطبيقات أن يبدؤوا من معارض البيانات والواجهات [بناء التطبيقات › معارض البيانات وأوجهها]، فهي تشرح الميزات نفسها بالتسميات التي يعرضها مصمّم المعرض.

المعارض وGrafana

يمتدّ أثر المعرض إلى ما وراء البوّابة: فبدءًا من `Since 1.6`، أصبح أيضًا الوحدة الإلزامية التي تُنفَّذ عليها لوحة **Grafana**، لأن بوابة الـ API تجيز خطة تنفيذ المعرض. ولإظهار البيانات على لوحة معلومات Grafana، لا بدّ من وجود معرض لها أولًا – انظر لوحات معلومات Grafana.

المصطلحات

بعض المصطلحات المستخدمة في هذه الصفحة:

المصطلح	المعنى
معرض البيانات	معرض مقيّد بكيان بيانات واحد، يعرض مُدخلاته.
الواجهة / أسلوب العرض	تمثيل بصري واحد لقائمة المُدخلات، يُحدَّد بحسب نوعه (جدول، بادج، …).
الواجهة الافتراضية	الواجهة الوحيدة التي تظهر أوّلًا عند فتح المعرض.
ربط الحقول	تحديد حقول الكيان التي تُغذّي خانات العنوان أو العنوان الفرعي أو الصورة أو التاريخ أو التصنيف في الواجهة.
معلومات العرض	إعداد خاص بكل حقل يتيح ربطه داخل الواجهة.

معارض الوكيل

معارض الوكيل · Since 1.10

يمكن أن يكون المعرض أيضًا **معرض وكيل** (Proxy view) داخليًا (`kind: Proxy`) – يُنشَأ تلقائيًا لضبط ما يُسمَح لكيان وكيل افتراضي [بناء التطبيقات › الكيانات الوهمية] بقراءته وكتابته، بدلًا من أن يكون معرضًا تصمّمه عبر الخطوات المعتادة أدناه. ويقتصر محرّره عمدًا على خيارات أقل من محرّر المعرض العادي (الهوية، والمصدر، والحقول، ومفاتيح تبديل عمليات الإنشاء/التحرير/الأرشفة فقط – بلا حالة نشر أو جمهور أو أذونات خاصة به)، ولا يمكن تغيير نوع المعرض إطلاقًا بعد ضبطه. وتصف بقية هذه الصفحة نوع المعرض المعتاد الذي يصمّمه المستخدم؛ انظر الكيانات الافتراضية [بناء التطبيقات › الكيانات الوهمية] للاطّلاع على نموذج حوكمة معرض الوكيل نفسه، بما في ذلك كيفية انتقال أثر تضييق قائمة حقول معرض الوكيل إلى مُنتقي حقول الكيان نفسه.

تسميات المصمّم والأسماء المخزَّنة

تسمية المصمّم	الاسم المخزَّن / المعنى في هذه الصفحة
وجه البطاقات	واجهة البادج
عرض العداد (تحت **خيارات أخرى**)	واجهة البطاقة الصغيرة
خانات الواجهة **العنوان الاساسي**، و**العنوان الفرعي**، و**صورة/ملف**، و**ارتباط رئيسي**، و**الفئة الفرعية**، و**الرقم الاساسي**، و**البيانات الوصفية**	العنوان الأساسي، والعنوان الفرعي، وصورة / ملف، والارتباط الرئيسي، والتصنيف الفرعي، والرقم الرئيسي، و«عرض البيانات الوصفية»
أعمدة لوحة المهام: **جميع الأعمدة** / **الأعمدة المستخدمة** / **أو اختر أعمدة محددة**	`columnsMode`: `all` / `used` / `specific`
السماح للمستخدم بتغيير الإعداد (لوحة المهام)	`allowUsersToChangeColumns`
حقل الارتباط، **تتبع سلسلة الآباء**	حقل المرجع الذاتي للشجرة، وتتبّع الأصول
فتح المدخل: **الإجراء الافتراضي للنظام** / **عرض تفاصيل المدخل** / **تعديل المدخل**	سلوك النقر على صف الجدول
الإستعلامات السريعة › **مكان الظهور داخل قائمة الحالة** / **زر مستقل**؛ وللمجموعات **قائمة منسدلة مستقلة**؛ **افتراضي**	`displayMode`، `isAppliedByDefault`، `order`
المحصلات: **التجميع** › **إضافة تجميع**، و**طي المجموعات افتراضيًا**، و**تغيير التجميع**	المحصّلات لكل معرض؛ ومستويات التجميع، وطيّ المجموعات عند البدء، والسماح للمستخدمين النهائيين بالتجميع
السماح بالتصدير	`viewConfiguration.allowExport`
ترتيب التكرارات، **إضافة خيارات التصنيف**، **تعديل خيارات التصنيف**، **ربط إعدادات بطاقتي الإنشاء والتعديل**	`allowRepeatableReorder`، `allowAddCategoryOptions`، `allowEditCategoryOptions`، `linkCategoryOptionSettings`
طريقة عرض الأزرار، **إظهار إجراءات سير العمل فقط**	`actionButtonsConfig.layout`، `actionButtonsConfig.applyWorkflowActionsOnly`
الشاشة الافتراضية للمعرض	ترتيب الشاشات
إعدادات الويب (تبويب)	النصوص البرمجية لكل معرض / لكل كيان

لا يُعرض خيار **إظهار إجراءات سير العمل فقط** إلا مع التخطيطين الأفقي والعمودي. وهو يعرض إجراءات سير العمل الخاصة بالخطوة الحالية في صفّ (أو الإجراء الأساسي إذا لم تكن للخطوة إجراءات)، وينقل كل إجراء آخر إلى قائمة **المزيد** للإجراءات الإضافية (`libs/feature/entry-content/.../entry-header.component.ts`)، ولا يُخفي تلك الإجراءات. وكانت نسخة سابقة من صفحة بُناة التطبيقات تذكر أنه «يقصر أزرار إجراءات المعرض على إجراءات سير العمل، مخفيًا سواها».

الواجهات في لمحة

يمكن أن تكون للكيان الواحد معارض عديدة، ويمكن تفعيل أكثر من واجهة للمعرض الواحد. يلخّص الجدول الآتي كل واجهة ومواضع توفّرها:

الواجهة	الأنسب لـ	الويب	الجوال
الجدول	البيانات الكثيفة وأعمال الإدارة؛ إعادة ترتيب الأعمدة، والتعديل المباشر، والإضافة السريعة، والتحديد، والإجراءات الجماعية	نعم	نعم
لوحة المهام	العمل القائم على الحالة؛ بطاقات مجمّعة في أعمدة بحسب حقل تصنيف، مع السحب بين الأعمدة	نعم	نعم
التقويم	البيانات المرتبطة بالتواريخ؛ عرض شهري أو أسبوعي أو يومي أو قائمة	نعم	نعم
الشريط الزمني	المنشورات والخلاصات؛ تظهر المُدخلات المثبّتة أوّلًا، ثم تُرتَّب بحسب التاريخ	نعم	نعم
الشكل الشجري	البيانات الهرمية والهيكل التنظيمي؛ توسيع العُقد عند الطلب	نعم	نعم
البادج	شرائح أو شارات مدمجة	نعم	نعم
البطاقة الصغيرة	قائمة بطاقات مدمجة مع العدادات	نعم	لا

ملاحظة: يؤدي اختيار واجهة **البطاقة الصغيرة** في تطبيق الجوال إلى ظهور منطقة فارغة – استخدم واجهة أخرى على الجوال. ويتوفّر ترتيب المُدخلات وإعادة ترتيبها في الجدول والبادج ولوحة المهام؛ وعلى الجوال، يمكنك سحب بطاقة لوحة المهام إلى عمود آخر (ويُحفَظ ذلك على الخادم)، غير أن إعادة ترتيب أعمدة لوحة المهام بالكامل غير مدعومة بعد.

تصميم كل واجهة

تقرأ كل واجهة **ربط حقول** يحدّد لها حقول الكيان التي تُغذّي خانات العنوان أو العنوان الفرعي أو الصورة أو التاريخ أو التصنيف. ولكي يكون الحقل قابلًا للربط، يجب أوّلًا أن يكون متاحًا في **معلومات العرض** الخاصة بالمعرض. ولا يجري التحقّق من جهة الخادم إلّا عند تفعيل شاشة قائمة المُدخلات، ولواجهات الجدول ولوحة المهام والتقويم والشريط الزمني والشكل الشجري فقط – أمّا **البادج** و**البطاقة الصغيرة** فلا تخضع واجهتاهما للتحقّق من جهة الخادم.

الجدول

يعرض المُدخلات في صفوف وأعمدة. لتصميمه:

1. فعّل شاشة قائمة المُدخلات.
2. فعّل واجهة **الجدول**.
3. اضبط:
   - **أعمدة الجدول** (مطلوب) – الحقول التي تظهر كأعمدة؛ يلزم اختيار حقل واحد على الأقل. ويجب تفعيل خيار **العرض في الجدول** على الكيان لكل حقل يُستخدم كعمود، وإلّا فلا يمكن استخدامه كعمود.
   - **السماح بتغيير الحقول** (اختياري) – يتيح للمستخدم النهائي إظهار الأعمدة المرئية أو إخفاءها.
   - **إعادة ترتيب الأعمدة** (اختياري، مُعطّل افتراضيًا)، وحدًّا لعدد القيم التي تعرضها الخلية متعدّدة القيم.

يتحكّم **سلوك النقر على الصفّ** على مستوى القائمة في ما يحدث عند نقر المستخدم على صفّ – الوضع الافتراضي للنظام، أو فتح تفاصيل المُدخل، أو فتح المُدخل للتحرير. **Since 1.6**

البادج

يعرض المُدخلات على هيئة شرائح أو شارات مدمجة، ويناسب القوائم المختصرة. اربط:

- **العنوان الأساسي** (مطلوب)
- **العنوان الفرعي** (اختياري)
- **صورة/ملف** (اختياري)

ويُظهر مبدّل **عرض البيانات الوصفية** (مفعّل افتراضيًا) بياناتٍ وصفية مثل تاريخ الإنشاء أو التحديث. وفي تطبيق الجوال، توفّر واجهة البادج تخطيطات فرعية قابلة للاختيار (بادج أو معرض صور). ولا يخضع البادج للتحقّق من جهة الخادم.

التقويم

يعرض المُدخلات المرتبطة بالتواريخ على تقويم شهري أو أسبوعي أو يومي أو قائمة. اربط:

- **العنوان الأساسي** (مطلوب)
- **العنوان الفرعي** (اختياري)
- **صورة/ملف** (اختياري)

- **حقول التاريخ** (مطلوب) – التواريخ المعروضة على التقويم؛ ويجب أن يشير كل حقل تاريخ إلى حقل في الكيان من نوع `Date` أو `DateTime`، وإلّا رُفضت واجهة التقويم.

لوحة المهام

تعرض المُدخلات على هيئة بطاقات مجمّعة في أعمدة بحسب حقل تصنيف. اربط:

- **العنوان الأساسي** (مطلوب)
- **التصنيف** (مطلوب) – الحقل الذي تشكّل قيمه أعمدة اللوحة: **حقل تصنيف** أحادي القيمة، أو `Since 1.10` **حقل مرجع** أحادي القيمة (انظر حقل الأعمدة وأنماط الأعمدة).
- **العنوان الفرعي**، و**صورة/ملف**، و**الارتباط الرئيسي**، و**التصنيف الفرعي**، و**التاريخ**، و**الرقم الرئيسي** (اختيارية).
- **أعمدة لوحة المهام** – القيم التي تشكّل الأعمدة. كانت قبل 1.10 قائمة مطلوبة تُختار عناصرها يدويًا (وتُخفى المُدخلات غير المصنّفة عند توفير قائمة)؛ وصارت `Since 1.10` **نمطًا** – *كل الأعمدة* أو *الأعمدة المستخدمة* أو اختيارًا محدّدًا – كما يلي.

ويُظهر مبدّل **عرض البيانات الوصفية** (مفعّل افتراضيًا) البيانات الوصفية للمُدخل. ويتطلّب التحقّق وجود ربط للحقول وحقل تصنيف.

ويمكن أن يكون حقل التصنيف **داخل كتلة** – بما في ذلك الكتلة القابلة للتكرار – وتظلّ اللوحة قادرة على تحديده وبناء أعمدتها وتوزيع البطاقات عليها. وكان الفحص يقتصر سابقًا على حقول المستوى الأعلى، فكانت اللوحة المربوطة بحقل تصنيف متداخل تظهر بلا أعمدة. `Since 1.8`

يُعدّ سحب بطاقة إلى **عمود مختلف** تغييرًا للتصنيف لا إعادة ترتيب – وهو يعمل الآن بصرف النظر عمّا إذا كان المعرض **قابلًا لإعادة الترتيب**. وكان هذا الانتقال بين الأعمدة يُرفَض سابقًا كليًّا على أي لوحة يكون فيها خيار إعادة الترتيب مُعطَّلًا، مع أنّ هذا الخيار مخصّص فقط لضبط إعادة الترتيب **داخل** عمود واحد. وما زال هذا الخيار يتحكّم في إعادة الترتيب داخل العمود كما كان؛ أمّا نقل بطاقة بين الأعمدة فلم يعد يخضع له. `Since 1.8`

حقل الأعمدة وأنماط الأعمدة `Since 1.10`

أيّ الحقول يمكنها تقسيم اللوحة. يجوز أن يكون حقل الأعمدة:

- **حقل تصنيف** – لكن لا يجوز أن يكون *متعدّد القيم* مطلقًا: فالبطاقة لا تنتمي إلى عمودين، ونقلها يستبدل المصفوفة كلها. ويُرفض الحفظ بالخطأ `MULTI_VALUE_CATEGORY_FIELD_NOT_ALLOWED` (`details.usage: 'kanban'`)؛ وينطبق الأمر نفسه على خانة **التصنيف الفرعي**.
- **حقل مرجع** – إلى مُدخلات كيانات أخرى، أو إلى **قائمة سجلات**، أو إلى **قائمة قيم**. ويجب أن يكون المرجع أحادي القيمة، وغير قابل للتكرار، وألّا يكون داخل كتلة قابلة للتكرار، وألّا يشير إلى كيان المعرض نفسه؛ وإلّا فشل الحفظ بالخطأ `KANBAN_REFERENCE_FIELD_NOT_ALLOWED` (`details.reason` : `multiValue` أو `repeatable` أو `repeatableBlock` أو `selfReference`). ويُطبَّق هذا القيد في الاتجاه المعاكس أيضًا: يُرفض تحويل حقل مرجع تستخدمه لوحة مهام إلى حقل متعدّد القيم أو قابل للتكرار بالخطأ `KANBAN_REFERENCE_FIELD_IN_USE`، ويسرد `details.usages[]` المعارض التي يجب تعديلها أولًا.

أنماط الأعمدة. تعرض أداة التصميم تحت **أعمدة لوحة المهام**:

النمط (`columnsMode`)	الأعمدة المعروضة	متاح لـ
جميع الأعمدة (`all`)	كل خيار حالي – أو، لحقل المرجع، كل مُدخل يجوز أن يشير إليه – **بما في ذلك القيم المضافة لاحقًا**.	حقول التصنيف وحقول المرجع.
الأعمدة المستخدمة (`used`)	القيم التي تحملها بطاقة واحدة على الأقل فقط.	حقول المرجع فقط (حفظه على حقل تصنيف ← `KANBAN_COLUMNS_MODE_NOT_ALLOWED`). وهو النمط **الوحيد** حين يكون مصدر مُدخلات الحقل استعلامًا ذا معاملات، لأن مجموعة المُدخلات المرشّحة تتوقّف حينئذٍ على قيمة لا تعرفها اللوحة.
أعمدة محددة (`specific`)	القيم التي تحدّدها (مخزّنة في `categories`).	حقول التصنيف وحقول المرجع.

وتحفظ أداة التصميم دائمًا النمط الذي جرى تحديده. ولا يحتوي المعرض المحفوظ قبل 1.10 على `columnsMode`؛ فيعمل وفق نمط **محدّد** إن كانت له قائمة أعمدة، وإلّا وفق نمط **مستخدم** لحقل المرجع ونمط **الكل** لحقل التصنيف – فتظهر اللوحات القائمة كما كانت تمامًا.

وتظهر البطاقات التي لا قيمة لها في عمود **غير محدّد** (عنوانه على اللوحة *التصنيف غير محدد*). ولا تحمّل اللوحة أكثر من **20 عمودًا** في المرّة الواحدة؛ وعند تجاوز هذا العدد، يرى المستخدم النهائي إشعار «بعض الأعمدة مخفية» يقترح واجهة أخرى. وتحتفظ قيمة المرجع التي حُذف مُدخلها (أو عنصر القائمة) بعمودها موسومًا بـ *سجل محذوف*، لتبقى بطاقاتها متاحة.

ويتيح خيار **السماح للمستخدم بتغيير الإعداد** (`allowUsersToChangeColumns`) للمستخدمين النهائيين اختيار أعمدتهم من *إعدادات العرض ← الأعمدة* في المعرض، مع عدد البطاقات لكل عمود وإمكانية العودة إلى إعداد المعرض. ويسري الاختيار على الجلسة فقط، ويُهمَل عند تغيير المعرض أو حقل أعمدته أو النمط المعروض.

واجهة البرمجة. يقبل `GET /v1/entities/{entity}/entries/kanban/{fieldId}` حقل تصنيف أو حقل مرجع للوحة المهام. ولحقل المرجع، يجب أن يحدّد المُستدعي الأعمدة المطلوب تحميلها في `columns` (20 على الأكثر، بلا تكرار؛ غيابها ← `KANBAN_COLUMNS_REQUIRED`، وعدم صلاحيتها ← `KANBAN_COLUMNS_INVALID`)؛ وما زال `categoryId` يحمّل عمودًا واحدًا، و`CATEGORY_NOT_SET` هو معرّف العمود غير المحدّد. ويُعيد المسار الجديد `GET /v1/entities/{entity}/entries/kanban/{fieldId}/columns` قيم الأعمدة مع أعدادها (`values[]`) بالحقول `id` و`label` و`archived` و`deleted` و`count`؛ و`notSetCount`؛ و`parameterized` لحقل المرجع). والمساران، ونظيراهما `/draft`، مدرجان في قائمة السماح لخطة تنفيذ معرض لوحة المهام، ويقبل كلاهما `quickFilterParameters` (انظر الاستعلامات [بناء التطبيقات ‹ الاستعلامات والمرشّحات]).

فحوص نقل كل بطاقة `Since 1.10`

يستدعي إفلات بطاقة على عمود آخر `PUT /v1/entities/{entity}/entries/{id}/position`. وفي لوحة مقسّمة بحسب **حقل مرجع**، يفحص الخادم الهدف قبل الكتابة:

1. يجب أن يكون الهدف عنصرًا من قائمة الحقل، أو مُدخلًا من أحد كيانات مصدر الحقل – وإلّا `KANBAN_COLUMN_NOT_IN_SOURCE`.
2. يجب ألّا يكون المُدخل الهدف مؤرشفًا – وإلّا `KANBAN_COLUMN_ARCHIVED`.
3. إن كان المرجع **متتاليًا** (مُرشَّحًا بحقل آخر من المُدخل نفسه) أو كان مصدره **استعلام ذو معاملات**، فيجب أن يكون الهدف ممّا يعرضه منتقي المرجع الخاص بالبطاقة نفسها – وإلّا `KANBAN_COLUMN_NOT_ALLOWED_FOR_ENTRY`. ولا يُفحص مطلقًا الإفلات داخل عمود البطاقة نفسه (إعادة ترتيب) ولا الإفلات على **غير محدّد**.

ثم تكتب عملية النقل قيمة العمود (`{id}` لقائمة القيم، و`{dataEntity, dataEntry}` للمُدخل، و`null` لغير المحدّد) **دون إعادة تشغيل تحقّق الحقل نفسه**، ويحميها `updateIdentifier` الخاص بالمُدخل، فيفشل النقل المتزامن مع تعديل آخر

بتعارض التحديث المعتاد بدل الكتابة فوق ذلك التعديل. وترتدّ البطاقة المرفوض نقلها إلى مكانها مع رسالة. ويحلّ هذا محلّ قاعدة مؤقتة في إصدارات 1.10 السابقة للإطلاق كانت ترفض *كل* نقل على لوحة ذات مرجع متتالٍ أو ذي معاملات. وما زالت إعادة الترتيب داخل العمود تتطلّب خيار **قابل لإعادة الترتيب**.

الشريط الزمني

يعرض المُدخلات على هيئة خلاصة زمنية، بحيث تظهر المثبّتة أوّلًا، ثم تُرتَّب بحسب التاريخ. **شرط مسبق:** يجب أوّلًا تفعيل المساهمة في الشريط الزمني على الكيان (انظر إعدادات الشريط الزمني للكيان)؛ وإلّا رُفضت واجهة الشريط الزمني. الخيارات:

- **السماح بنشر/حجب المنشورات** – يتيح لأي شخص لديه وصول إلى المعرض نشر مُدخل أو حجبه.
- **السماح بالتثبيت** – إبقاء منشور مهم مثبّتًا في الأعلى.

والشريط الزمني هو الواجهة الوحيدة التي يقتصر التحقّق منها على فحص الكيان دون إعداد الواجهة.

الشكل الشجري

ينظّم المُدخلات في تسلسل هرمي يشبه الشجرة. **شرط مسبق:** أضِف **حقل ارتباط يشير إلى الكيان نفسه** وفعّل خيار منع الحلقات (منع العلاقة الدائرية) الخاص به. اربط:

- **حقل الارتباط** (مطلوب) – العلاقة ذاتية الارتباط؛ ويجب أن يكون منع العلاقة الدائرية مفعّلًا فيها.
- **العنوان الأساسي** (مطلوب) – ويجب أن يكون من نوع بيانات صالح للعنوان.
- **العنوان الفرعي**، و**صورة/ملف** (أيقونة أو صورة)، و**الارتباط الرئيسي** (اختيارية).

ويخضع الشكل الشجري لأشدّ فحوص التحقّق: يجب أن يكون مصدر بيانات المعرض **كل المدخلات** أو **استعلامًا**، ولا يمكن الجمع بين واجهة الشجرة و**الترشيح المسبق**. ويتتبّع خيار **تتبّع الأصول** أسلاف العُقدة ويعرضها. **Since 1.2**

وعندما يضيّق المستخدم النهائي نطاق الشجرة **بالحالة أو الصلاحية وحدها** – المؤرشفة فقط، أو البالية فقط، وما إلى ذلك، دون أي عبارة بحث أو شروط ترشيح أو مرشّح سريع – تتوسّع الشجرة الآن لتجلب المُدخلات المطابِقة الأعمق في التسلسل الهرمي وتعرضها. وكان اختيار الحالة وحده لا يُحتسَب سابقًا مرشّحًا نشطًا، فلا تُعاد سوى المُدخلات الجذرية وتعرض الواجهة رسالة «لا نتائج مطابقة» رغم وجود مطابقات في مستويات أعمق. `Since 1.8`

إصلاح في البوّابة: الشريط الجانبي للعلاقة الواردة يُحلّ الآن بشكل صحيح `Since 1.8` **.** يفتح النقر على علاقة واردة لبطاقة في واجهة الشكل الشجري شريطًا جانبيًا يعرض المُدخلات المرتبطة عبر معرض العرض الخاص بتلك العلاقة. وتُنفَّذ طلبات الشريط الجانبي باستخدام كائن وصول المعرض **الرئيسي**، لكن الشريط كان يطلب سابقًا معرض العرض الخاص بالعلاقة عبر المسار غير المقيَّد لجلب المعرض بمعرّفه، وهو مسار لا تُدرجه خطة تنفيذ المعرض الرئيسي في قائمة السماح مطلقًا، فكانت بوابة الـ API ترفضه بالخطأ `INVALID_ACCESS` وتعرض البوابة صفحة «لا إذن وصول». أمّا الآن فيجري تحديد المعرض عبر المسار المقيَّد بالعلاقة، باستخدام كيان المعرض الرئيسي مفتاحًا له. كما جرى إصلاح عرض الشريط الجانبي نفسه: إذ كان يستخدم سابقًا قائمة مخصّصة من نحو 780 سطرًا لا تُعيد عرض محتواها على نحو موثوق فور وصول بياناتها؛ وهو يستخدم الآن مكوّن قائمة العلاقات نفسه الذي تستخدمه تبويبة العلاقات في شاشة تفاصيل المُدخل (أعلاه)، بحيث يتطابق الاثنان شكلًا وسلوكًا.

البطاقة الصغيرة

تعرض قائمة بطاقات مدمجة تتمحور حول العدادات. اربط:

- **العنوان الأساسي** (مطلوب)

- **العنوان الفرعي** (اختياري)
- **صورة/ملف** (أيقونة) (اختياري)
- **التاريخ** (اختياري)

وواجهة البطاقة الصغيرة **غير متوفّرة على تطبيق الجوال** ولا تخضع للتحقّق من جهة الخادم. ولإضافة العدادات، انظر إعداد عدّادات الكيان.

> لملء الخانات المربوطة في أي واجهة، يجب أوّلًا تحديد الحقول في معلومات العرض الخاصة بالمعرض.

تصميم شاشة قائمة المدخلات

تضمّ شاشة قائمة المُدخلات الواجهات، إضافةً إلى عناصر التحكّم التي يستخدمها المستخدمون النهائيون للعثور على المُدخلات. فعّل **معرض قائمة المدخلات** لتشغيلها؛ وتعتمد عليها الخيارات الآتية.

الترشيح والبحث

- **السماح بالبحث** (مفعّل افتراضيًا) و**السماح بالمرشحات** – تتطلّب المرشحات تفعيل البحث أوّلًا.
- **مرشحات أوصاف المدخل** – الترشيح حسب: الإنشاء بواسطة، وتاريخ الإنشاء، والتحديث بواسطة، وتاريخ التحديث، ومعرّف المُدخل.
- **مرشحات تفاصيل أوصاف المدخل** – الترشيح حسب المجموعة الأوسع من حقول تفاصيل «الإنشاء بواسطة».
- **مرشحات الخطوة الحالية** و**مرشحات الخطوة السابقة** – حقول خطوات سير العمل، مثل الخطوة الحالية أو السابقة، ومالكي الخطوات، وتاريخ الوصول، وفترة التنفيذ.
- **حفظ الاستعلامات** – يتيح للمستخدمين النهائيين حفظ مرشحاتهم (يتطلّب تفعيل المرشحات).
- **إنشاء القوائم (التجميعات)** – يتيح للمستخدمين النهائيين إنشاء قوائم مُدخلات خاصة بهم وفق أذوناتهم.

إذا كنت تُعدّ معايير المعرض أو المرشحات السريعة عبر الواجهة البرمجية، فانتبه إلى أن صيغة مسارات حقول المعايير تختلف باختلاف السطح – إذ تتوقّع معايير المعرض مسارات حقول مسبوقة بـ `value.`، ويُعيد المسار الخاطئ قائمة فارغة بصمت بدلًا من خطأ. انظر صيغ المعايير ومسارات الحقول للاطّلاع على المرجع الخاص بكل سطح.

الفرز والمرشحات السريعة وأشكال التصنيف

- **الفرز الافتراضي** – اختر الحقول واتجاه الفرز المُطبَّق عند الفتح؛ ويكون الفرز تصاعديًا حسب حقل الترتيب هو الإعداد الافتراضي الأوّلي.
- **المرشحات السريعة** – أظهِر الاستعلامات السريعة للكيان بوصفها مرشحات تُطبَّق بنقرة واحدة، واختر موضع كلٍّ منها، وطبّق بعضها افتراضيًا إن شئت. وقد أصبح هذا `Since 1.10` إعدادًا أشمل، موضّحًا في المرشحات السريعة ومجموعاتها وترتيبها.
- **أشكال تصنيف المدخل** – اختر شكلًا يُطبَّق عند الفتح، وأتِح للمستخدمين النهائيين تبديل الأشكال اختياريًا، واختر الأشكال المتاحة. عندئذٍ يرى المستخدمون النهائيون قائمة **الأشكال** في المعرض.

موقع خيار الفرز `Since 1.10`. في تبويب إعدادات المعرض، أصبح الخيار الذي يتيح للمستخدمين النهائيين تغيير الفرز يقع مباشرةً تحت **الفرز الافتراضي**، ليظهر خيارا الفرز معًا؛ وتليهما خيارات المحصّلات والتجميع. وأصبح ذلك الخيار والمفتاحان الخاصان بالتجميع مربّعات اختيار بدلًا من مفاتيح تبديل. ولم يتغيّر شيء في الفرز نفسه.

المرشحات السريعة ومجموعاتها وترتيبها `Since 1.10`

يعرض المعرض مرشحًا سريعًا **إذا وفقط إذا كان مسرودًا** في إعدادات المرشحات السريعة للمعرض (`viewConfiguration.quickFilters[]`). ولكل استعلام سريع مُدرَج، تضبط ما يأتي:

- **مكان الظهور** (`displayMode`) – **داخل قائمة الحالة** (الافتراضي: يظهر داخل قائمة الحالة في المعرض) أو **زر مستقل** (مفتاح خاص به يُوضَع قبل الحالة مباشرةً؛ وتنتقل المفاتيح التي لا يتّسع لها الصف إلى صفّ أدناه).
- **افتراضي** (`isAppliedByDefault`) – يُطبَّق مسبقًا عند فتح المعرض. ويُسمح بتعيين عدّة استعلامات افتراضية. ولا يمكن أبدًا أن يكون الاستعلام السريع الذي يتضمّن شروط **معامل** افتراضيًا (`QUICK_FILTER_PARAMETERIZED_DEFAULT`): فهو يطلب قيمه من المستخدم كلّما شُغّل. ولا يُفحص إلّا التغيير – استعلام أُضيف حديثًا أو إعداد افتراضي فُعّل حديثًا – فلا يُرفض تعديل غير ذي صلة على معرض أقدم.
- **الترتيب** (`order`) – اسحب الصفوف؛ ويظهر أوّلًا ما كانت قيمة ترتيبه أصغر. ويحتفظ المعرض المخزَّن بلا `order` بترتيب مصفوفته.

والعنوان الذي يراه المستخدمون النهائيون هو **اسم المرشح السريع** للاستعلام (مترجَم)، ويُضبط على الاستعلام نفسه؛ وإن تُرك فارغًا، نسخ نموذج الإدارة اسم الاستعلام إليه، ثم يستقلّ كلٌّ منهما عن الآخر.

المجموعات. يمكن تصنيف الاستعلامات السريعة في **مجموعات مرشحات سريعة**، تُدار في تبويب *الاستعلامات* للكيان (التبويب الفرعي *مجموعات المرشحات السريعة* – انظر الاستعلامات [بناء التطبيقات › الاستعلامات والمرشّحات]). وينتمي الاستعلام إلى مجموعة واحدة على الأكثر. فعّل مجموعة في معرض عبر `viewConfiguration.quickFilterGroups[]` ، بإعدادات **افتراضي** و**الترتيب** و**مكان الظهور** نفسها (وتظهر المجموعة المستقلّة في صورة **قائمة منسدلة مستقلة**). وتطبّق المجموعة الافتراضية جميع الاستعلامات المنتمية إليها مسبقًا. وفي قائمة الحالة، تظهر المجموعة بعنوان ذي ثلاث حالات فوق الاستعلامات المنتمية إليها؛ ولا يعني تطبيق مجموعة سوى تطبيق تلك الاستعلامات، فلا يتغيّر طلب المُدخلات.

وعند الاستدعاء من خلال المعرض، يُعيد `GET /v1/entities/{entity}/queries` المرشحات السريعة للمعرض مع الاستعلامات المنتمية إلى مجموعاته المفعّلة، فلا يحتاج المستخدمون النهائيون إلى مسار إضافي. ويُتخطّى المرشح السريع الذي حُذف استعلامه بدلًا من أن يتسبّب في فشل القائمة. ويلتزم تطبيق الجوال بالأسماء والمواضع والترتيب والإعدادات الافتراضية نفسها.

المحصّلات والتجميع على بطاقة قائمة المدخلات `Since 1.10`

يُضبط هنا خياران آخران من خيارات قائمة المُدخلات، غير أنّهما **مُنمذَجان على الكيان**، لذا لا تتناول هذه الصفحة سوى جانب المعرض منهما. أمّا النموذج نفسه – الدوال الخمس، وأنواع الحقول التي تقبلها، والحقول التي يمكن التجميع حسبها وسبب تعذّر ذلك على بقيّتها، وكيفية ترتيب المجموعات، وسلوك الحقول المحمية – فانظر تجميعات الكيان وتجميع المدخلات [بناء التطبيقات › المجاميع والتجميع].

- **المحصّلات** – قائمة مربّعات اختيار للمحصّلات المعرَّفة على الكيان؛ حدّد ما ينبغي لهذا المعرض حسابه وإظهاره. **والمعرض الذي لا يؤشّر على شيء لا يُظهر شيئًا.** فهو لا يرث مجموعة محصّلات الكيان، ولا تُضاف إليه محصّلة لاحقًا لمجرّد أن أحدهم أضافها إلى الكيان. ويُحدَّد هذا التقاطع على الخادم، فهو قيد لا مجرّد ترشيح للعرض – ويترتّب عليه أن تبقى المحصّلة الجديدة غير مرئية حتى تفعّلها في كل معرض ينبغي أن يتضمّنها.
- **التجميع** – قائمة مرتّبة بمستويات التجميع (أوّلها المجموعة الخارجية)، إضافةً إلى **البدء مطويًّا** و**السماح للمستخدمين النهائيين بالتجميع**. وكلاهما معطّل افتراضيًا. ويجب أن يكون مستوى التجميع من حقول المعرض المُدرَجة؛ فالتجميع حسب حقل يخفيه المعرض مرفوض.

ويُخفي تعطيل بطاقة قائمة المُدخلات الإعدادين دون مسحهما – فيعودان كما هما عند تفعيلها من جديد.

وتصل إجماليات المعرض من **استدعاء ملخّص مُقيَّد بنطاق المعرض**، يُنفَّذ منفصلًا عن طلب المُدخلات، وتُحسب على أساس كل ما يطابق نطاق ذلك المعرض، بما فيه مرشحاته وبحثه ومرشحاته السريعة وترشيحه المسبق. واستدعاء المسار المُقيَّد بالمعرض بدلًا من المسار على مستوى الكيان هو تحديدًا ما يقصر النتيجة على المحصّلات التي فعّلها هذا المعرض.

ولا تُعرَض المحصّلة المفعَّلة إلا إذا كان حقلها من أعمدة المعرض المعروضة. إذا فعّلت *مجموع المبلغ* في معرض لا تتضمّن قائمة أعمدته المبلغ، يُستبعَد الرقم من الإجمالي الكلي ومن كل مجموع فرعي بدلًا من إدراجه في عمود لا صلة له به. أمّا ما إذا كان *يجوز* للمعرض حساب محصّلة، فيبقى قرارًا على الخادم؛ وتحدّد هذه القاعدة موضع عرض الرقم فحسب. فإن غاب إجمالي تتوقّعه، فابدأ بمراجعة أعمدة المعرض.

الترشيح المسبق

اضبط المعرض على **الترشيح المسبق** بحيث لا تُحمَّل أيّ مُدخلات حتى يملأ المستخدم نموذج ترشيح. اضبط ما يصل إلى أربعة حقول ترشيح، واختياريًا **املأ تلقائيًا** تلك القيم في المُدخلات الجديدة. ويضيف الترشيح المسبق خطوة تحقّق بشري (Captcha) للزوار المجهولين، وهو **متنافٍ مع واجهة الشجرة ومع لوحات المعلومات**.

إصلاحان في اللوحة `Since 1.8` . كانت العودة إلى معرض ذي ترشيح مسبق سبق للمستخدم أن أرسل بحثه فيه – كالعودة من صفحة تفاصيل مُدخل مثلًا – تُعيد عرض نموذج الترشيح كاملًا فوق النتائج التي جُلبت بالفعل، بدلًا من طيّه في الشريط المدمج «نتائج البحث عن… / غيّر بحثك»؛ وكان العيب نفسه يؤثّر في معاينة أداة بناء المعرض. كما كانت قائمة **نوع الاستعلام** («المطابقة») المنسدلة في أداة البناء تعرض مفاتيح الترجمة الخام (`OPERATORS.And` ، `DATA_VIEW.CUSTOM_QUERY`) بدلًا من نصوصها المترجمة. ولم يغيّر أيّ من الإصلاحين طريقة عمل الترشيح المسبق نفسها – فكلاهما إصلاح في العرض فقط.

المدخلات البالية والمؤرشفة

- **إظهار المدخلات البالية** – إظهار المُدخلات التي تحتاج إلى تحديث.
- **إظهار المدخلات المؤرشفة** – إظهار المُدخلات التي جرت أرشفتها.

تخطيط أزرار الإجراءات

اختر كيفية عرض أزرار الإجراءات داخل المعرض – **افتراضي** أو **أفقي** أو **عمودي** – مع تمييز الزر الأساسي في كل تخطيط. كما يمكنك تفعيل الأزرار المنشورة المراد إظهارها وسحبها لترتيبها؛ ويتيح لك الزرّ المرتبط بمعرض ذي علاقة اختيار معرض الكيان المرتبط. ويقصر خيار **الاقتصار على إجراءات سير العمل** (apply-workflow-actions-only) أزرارَ إجراءات المعرض على إجراءات سير العمل فقط، ويُخفي سواها. **Since 1.8**

تسوية أزرار النظام الضمنية عند الحفظ `Since 1.8`

هناك خمسة معرّفات لأزرار النظام تُعدّ **ضمنية**: `VIEW` و `EDIT` و `ARCHIVE` و `CREATE` و `DUPLICATE` . وهي ليست أزرارًا مخزَّنة، بل تُشتقّ من إعداد المعرض نفسه (ما إذا كانت شاشة قائمة المُدخلات مفعّلة، وعمليات الإنشاء والتحرير والأرشفة التي يسمح بها إعداد التحرير، وما إذا كان التكرار مسموحًا).

وعندما تحفظ معرضًا لا تزال حمولته تتضمّن زرًّا ضمنيًا لم يعد الإعداد الحالي يسمح به – والحالة الشائعة هي بقاء زرّ `EDIT` أو `DUPLICATE` بعد تعطيل العملية المقابلة – **ينجح الحفظ ويُقتطَع الزرّ الضمني غير المسموح به من الحمولة** قبل تخزين المعرض. وكان الحفظ يُرفَض سابقًا بخطأ «زرّ النظام غير موجود»، لأن فحص وجود الزرّ كان يبحث عن المعرّف الضمني في مستودع الأزرار المخزَّنة، حيث لا وجود له أصلًا.

ولا يخضع للاقتطاع سوى تلك المعرّفات الخمسة. أمّا **أزرار النظام المخصّصة وأزرار القوالب فلا تُقتطَع أبدًا**: إذ يستمرّ التحقّق من مرجعها في الأزرار المخزَّنة، وتظلّ الإشارة إلى زرّ غير موجود خطأً.

تصدير المدخلات

يمكن للمستخدمين النهائيين تصدير مُدخلات المعرض بثلاث طرق:

1. **التصدير إلى XLSX** – تصدير مُدخلات المعرض الحالي إلى ملف Excel بصيغة `.xlsx`؛ وتتبع الأعمدة المُصدَّرة حقول المعرض المُعدَّة (`GET /v1/entities/{entity}/views/{id}/entries/export/xlsx`، على الويب والجوال).
2. **التصدير عبر قالب** – تنسيق المُدخلات المحدَّدة ضمن **قالب** طباعة أو تصدير (مُنشأ بصيغة **DOCX** أو **HTML** أو **XLSX**)، وتحويل النتيجة إلى **PDF** حيثما أُعِدّ ذلك.
3. **التصدير السريع** `Since 1.10` – زرّ **تصدير المدخلات** في المعرض، ولا يُعرض إلّا عند تفعيل **السماح بالتصدير**؛ ويُنتج **XLSX أو CSV**. انظر التصدير السريع من المعرض.

لا توجد صيغة CSV إلّا في التصدير السريع (ونظيره في الإدارة، *تصدير المدخلات* في صفحة الكيان). أمّا تصدير XLSX، وتصدير القوالب (بصيغة قالبها، مع تحويل اختياري إلى PDF)، والتقارير فتبقى **XLSX / PDF فقط**. وتُنفَّذ مهام القوالب الكبيرة في الخلفية ليتمكّن المستخدم من متابعة عمله؛ أمّا المهام الأصغر فتُنفَّذ في المقدّمة ويبدأ تنزيلها مباشرةً. وتُراعى أذونات الحقول الحسّاسة في المُخرَج. ولا يطبّق التصدير عبر قالب المرشحات السريعة التي تتضمّن قيم معاملات؛ ويُبلَّغ المستخدم بذلك عند بدء التصدير.

التصدير السريع من المعرض `Since 1.10`

يضيف خيار **السماح بالتصدير** (`viewConfiguration.allowExport`، معطّل افتراضيًا) زرّ النظام القابل للترتيب **تصدير المدخلات** (`EXPORT`) إلى قائمة المُدخلات. ولا يسري هذا الخيار إلّا ما دامت شاشة قائمة المُدخلات مفعّلة: فحفظ معرض قائمته معطّلة يُعيد قيمة الخيار إلى `false`، ويُقتطَع الزرّ متى عُطّل التصدير – وهي التسوية نفسها المطبّقة على أزرار النظام الضمنية. وعند تفعيله، تُدرج خطة تنفيذ المعرض مسار التصدير في قائمة السماح.

يختار المستخدمون النهائيون **Excel (XLSX)** أو **CSV** في نافذة صغيرة، ويحتوي الملف على **كل مُدخل يطابق القائمة كما يرونها** – البحث والمرشحات والمرشحات السريعة (بقيم معاملاتها) ومرشحات الوسوم وقيم الترشيح المسبق وترتيب التجميع – بأعمدة الجدول الحالية وترتيبها (أو بحقول المعرض بترتيبها حين لا تكون للمعرض واجهة جدول). والمسار هو `GET /v3/entities/{entity}/views/{id}/entries/export` مع `format=xlsx|csv`؛ ويعيد استخدام نطاق القائمة نفسه، فتنطبق عليه فحوص الترشيح المسبق واستعلام المعرض وحماية الحقول الحسّاسة. ويحتاج المُستدعي إلى إذن الوصول العادي إلى مُدخلات المعرض؛ وعند تعطيل القائمة أو **السماح بالتصدير**، يستجيب المسار بـ `EXPORT_NOT_ALLOWED`.

المدخلات المطابقة	ما يحدث
حتى `MAX_ENTRIES_RESULT_LIMIT` (افتراضيًا **10,000**)	يُعاد الملف مباشرةً.
أكثر، لمستخدم **مسجّل الدخول**	يُوضَع التصدير في الطابور؛ ويتلقّى المستخدم إشعار *الملف جاهز*. ولا يتجاوز عدد صفوف الملف المُنشأ في الخلفية `MAX_BACKGROUND_EXPORT_ROWS` (افتراضيًا **100,000**) صف، ويُسمّى `<n>_first-…` عند اقتطاعه.
أكثر، لزائر **مجهول**	يُرفض (`EXPORT_TOO_LARGE_FOR_ANONYMOUS`) ما لم يُختر في النافذة **تصدير أول 10,000** (`truncate=true`).

ويُرمَّز ملف CSV بترميز UTF-8 مع علامة ترتيب البايت، وتُفصَل قيمه بفواصل، وتنتهي أسطره بـ CRLF؛ وتُسبق الخلية النصية التي تبدأ بـ `=` أو `+` أو `-` أو `@` أو مسافة جدولة أو رجوع عربة بـ `'` ما لم تكن نصًا رقميًا صرفًا (حماية من حقن الصيغ). ويتحوّل حقل الكتلة إلى أعمدة `Block › Field` في CSV وترويسة مدموجة من صفّين في XLSX. ويحتوي الملف على المُدخلات فقط – بلا ترويسات مجموعات ولا إجماليات.

ويُتاح للمسؤولين التصدير نفسه في صفحة الكيان (**تصدير المدخلات**، `GET /v3/entities/{entity}/entries/export`): للمستخدمين المسجّلين الذين يحقّ لهم سرد مُدخلات ذلك الكيان، مع جميع حقول الكيان بترتيب تعريفها افتراضيًا، ويُسمّى

الملف باسم الكيان. و**لا يملك تطبيق الجوال تصديرًا سريعًا**؛ ويحتفظ بتصدير XLSX وتصدير القوالب.

تصميم شاشة تفاصيل المدخل

تعرض شاشة تفاصيل المُدخل مُدخلًا واحدًا. صمّمها بتفعيل **عرض تفاصيل المدخل** واختيار **الحقول** المراد عرضها (يجب ضبطها في معلومات العرض أوّلًا). ويجب تفعيل شاشة قائمة المُدخلات قبل شاشة تفاصيل المُدخل – إذ يصل المستخدمون النهائيون إلى تفاصيل مُدخل باختياره أوّلًا من القائمة. وتشمل اللوحات الإضافية ما يأتي:

- **المناقشة** – أتِح للمستخدمين النهائيين إضافة التعليقات والتفاعل معها. يمكنك السماح بـ**الإشارة إلى الأشخاص** (وتقييد نطاق الإشارة إلى جميع المستخدمين، أو المستخدمين ذوي إذن المعرض، أو نطاق مخصّص ثابت أو متغيّر). وعند تفعيل **المرفقات**، تختار أنواع الملفات المسموح بها **لذلك المعرض** وتضبط الحجم الأقصى للمرفق (نحو **1 ميغابايت** افتراضيًا) – فالأنواع المسموح بها تُضبط لكل معرض على حدة، ولا تُحدَّد بقائمة ثابتة.
- **السجل التاريخي للمدخل** – إظهار التعديلات، ومن أجراها، والتفاعلات، و**تفاصيل التفاعلات** (التفاعلات، ومَن تفاعل، ومتى)، مع تفعيل كلٍّ منها أو تعطيله على حدة.
- **مسار سير العمل** – إظهار مخطّط سير عمل المُدخل، مع التحكّم اختياريًا في كيفية عرض أسماء مالكي الخطوات (كل الخطوات مقابل الخطوات التي يملكها المستخدم فقط).
- **رسائل البريد الإلكتروني** – إظهار رسائل تفاعلات الواجهة وتفاعلات الخلفية المرتبطة بالمُدخل، مع تحديد نطاقها بالكل أو المُرسَلة من المستخدم أو المُستقبَلة من المستخدم.
- **العلاقات** – إظهار المُدخلات المرتبطة على هيئة **تبويبة** أو **معرض فرعي** مضمَّن، حسب أسلوب العرض المختار. وعند اختيار المعرض الفرعي، يظهر الخيار الفرعي **وضع العرض** (**العرض ككتلة**).
- **التنقّل بين المدخلات** – أتِح للمستخدم النهائي التنقّل بين المُدخلات دون العودة إلى القائمة.

عرض سير العمل الافتراضي (المسار/السجل التاريخي) `Since 1.8`

عندما تتيح لوحة سير العمل للمستخدم النهائي الوصول إلى عرض التنفيذ (السجل التاريخي) إضافةً إلى المسار الرئيسي، يمكنك أيضًا اختيار العرض الذي **يفتح عليه** التبويب افتراضيًا: **المسار** أو **السجل التاريخي**. وكان التبويب سابقًا يفتح دائمًا على المسار دون وسيلة لضبط ذلك.

ولا يُطبَّق هذا الإعداد الافتراضي إلّا بعد تفعيل الوصول إلى السجل التاريخي نفسه في اللوحة – فعند تعطيله، يظلّ السجل التاريخي غير متاح ويفتح التبويب على المسار مهما كانت القيمة الافتراضية، لذا لا يؤثّر تفعيل هذا الإعداد في معرض لا يُظهِر السجل التاريخي أصلًا. ويلتزم كلٌّ من الويب والجوال بالإعداد الافتراضي نفسه الذي يضبطه المسؤول. انظر تنفيذ إجراءات سير العمل [استخدام أسس › تنفيذ إجراءات خطوات سير العمل] لمعرفة كيفية قراءة المستخدمين النهائيين للمسار والسجل التاريخي والتبديل بينهما أثناء التشغيل – ولا يُكرَّر ذلك هنا.

تصميم شاشات الإنشاء والتحرير والأرشفة

يتحكّم إعداد التحرير في كيفية إضافة المستخدمين النهائيين للمُدخلات وتحريرها وأرشفتها. فعّل العملية المعنيّة، واضبط خيارات كلٍّ منها:

- **إنشاء المدخلات** – اختر الحقول التي يجوز للمستخدم النهائي تعبئتها (يلزم حقل واحد على الأقل)، ثم عيّن **الإجراء بعد الحفظ**: عرض رسالة، أو العودة إلى قائمة المُدخلات، أو تفريغ الحقول لإضافة مُدخل آخر، أو الانتقال إلى تفاصيل المُدخل.
- **تحرير المدخلات** – يتيح اختيار الحقول نفسه وخيارات ما بعد الحفظ؛ ويُظهر التحرير أيضًا قسم **الأذونات الإدارية** للكيانات ذات سير العمل، إذ يتيح للمستخدمين المخوّلين التحرير، أو تنفيذ إجراء سير عمل بالنيابة عن مالك خطوة، أو

إنهاء سير العمل أو نقله أو إرجاعه أو إعادة تعيينه، أو تغيير مالك خطوة مؤقتًا أو دائمًا.

- **أرشفة المدخلات** – عيّن الإجراء بعد الأرشفة.

وتشمل خيارات التحرير الأخرى التعديل المباشر، والإنشاء السريع، والحفظ مسوّدةً، والتكرار. لاحظ أن حقول أوصاف المُدخل لا يمكن أن تظهر في قائمة حقول العملية، وأن إعداد التحرير غير مدعوم لكيانات الوكيل الافتراضي (virtual-proxy).

منح بعض أعمدة كتلة قابلة للتكرار دون غيرها. قد تقتصر الحقول التي تُدرجها العملية على بعض الحقول الفرعية لكتلة قابلة للتكرار. وتُحفظ صفوف الكتلة في مصفوفة واحدة تحلّ محلّ المصفوفة المخزَّنة، ثم يستعيد الخادم، صفًّا صفًّا، القيم المخزَّنة للأعمدة التي لا تمنحها العملية، وبذلك لا تُمحى عند الحفظ `Since 1.4` . وابتداءً من الإصدار 1.10.1، لا تجري مطابقة الصف إلا بمعرّفه `__id` ، ويُعدّ الصف الذي لا معرّف له صفًّا جديدًا لا تُستعاد له أيّ قيم `Since 1.10` . ومنح أي حقل فرعي من الكتلة يتيح للمستخدم أيضًا إضافة الصفوف وإزالتها. ويوثّق الكتلة: حفظ الصفوف عبر معرض [بناء التطبيقات › حقل الكتلة] عقد العميل.

ترتيب التكرارات `Since 1.8`

ترتيب التكرارات (`allowRepeatableReorder` ، مُعطّل افتراضيًا) هو **خيار إعداد معرض لكل عملية على حدة**: فهو موجود في كل عنصر من قائمة العمليات المدعومة ضمن إعداد التحرير، إلى جانب اختيار حقول تلك العملية وإجرائها بعد الحفظ. وهو **ليس** خيارًا على حقل الكيان ولا يُخزَّن في تعريف الحقل، ولذلك يمكن أن يكون الحقل نفسه قابلًا لإعادة الترتيب في شاشة الإنشاء وثابتًا في شاشة التحرير للمعرض نفسه – أو قابلًا لإعادة الترتيب في معرض دون آخر.

ولا يصبح الخيار قابلًا للاختيار إلّا عندما تضمّ قائمة حقول العملية هدفًا **قابلًا للتكرار** واحدًا على الأقل – حقلًا قابلًا للتكرار أو كتلة قابلة للتكرار. وتجري المطابقة على اسم الحقل الكامل، فلا يُخلَط أبدًا بين حقلين متداخلين يشتركان في جزء من الاسم فقط (`blockA.title` و `blockB.title`). وإذا أزلت لاحقًا آخر حقل مؤهَّل من العملية، عاد الخيار تلقائيًا إلى حالة الإيقاف.

وعند تفعيله، يظهر للمستخدم النهائي في نموذج الإنشاء أو التحرير مقبض سحب بجانب كل تكرار من قيم الحقل القابل للتكرار، وبجانب كل تكرار من الكتلة القابلة للتكرار. ويُعيد سحب المقبض ترتيب التكرارات في مواضعها ويضع علامة التعديل على النموذج، فيُحفَظ الترتيب الجديد مع المُدخل. ولا يظهر المقبض إلّا عند وجود **أكثر من تكرار واحد** – إذ لا يوجد ما يُعاد ترتيبه عند وجود تكرار واحد أو عدم وجود تكرارات، ويكون السحب معطّلًا.

> **يحترم العميلان كلاهما هذا الإعداد** `Since 1.8` ، إذ يقرأ تطبيق الجوال علامة `allowRepeatableReorder` نفسها الخاصة بكل عملية من إعداد المعرض، ويعرض مقبض السحب الخاص به على كل تكرار بعد الأول، فيعمل المعرض المُعدّ لإعادة الترتيب بالطريقة نفسها في بوابة الويب وعلى الهاتف. (وكانت نسخة سابقة من هذه الصفحة تصف ترتيب التكرارات بأنه خاص بالويب؛ ولم يعد ذلك صحيحًا.)

خيارات التصنيف من بطاقة المُدخل `Since 1.10`

يمكنك أن تتيح للمستخدمين النهائيين **إضافة** خيارات حقل تصنيف أو **تعديلها** أثناء إنشاء مُدخل أو تحريره، دون أن يتولّى مسؤول الكيان تعديلها. ويجب أن تتوافق ثلاثة إعدادات، ويفرض الخادم أوّل إعدادين منها:

الإعداد	الموضع	المعنى
السماح للمستخدمين بإدارة الخيارات (`allowUsersToManageOptions`)	إعدادات حقل التصنيف على الكيان	يقبل الحقل أصلًا خيارات يديرها المستخدمون.
إضافة خيارات التصنيف / تعديل خيارات التصنيف (`allowAddCategoryOptions` ، `allowEditCategoryOptions` ، معطّلان افتراضيًا)	كل عملية إنشاء أو تحرير في المعرض	يحدّد ما إذا كان يجوز إضافة الخيارات أو تعديلها من تلك البطاقة. ويجب أيضًا أن يكون الحقل قابلًا للتحرير في تلك العملية.
ربط إعدادات بطاقتي الإنشاء والتحرير (`linkCategoryOptionSettings` ، مفعّل افتراضيًا)	إعداد التحرير في المعرض	تشترك بطاقتا الإنشاء والتحرير في قيمتَي المفتاحين. وتفرض أداة التصميم وحدها هذا الربط.

وتُرسَل الخيارات الجديدة أو المعدّلة ضمن عملية كتابة المُدخل نفسها (`categoryOptionChanges` في جسم الإنشاء أو التحديث، ويُستخدم الاسم الكامل للحقل مفتاحًا لها)، فتُحفظ مع المُدخل. وتُحدَّد العناوين لكل لغة، واللون الافتراضي هو `#81C784` . ويُرفض الحفظ بأحد الرموز الآتية: `CATEGORY_OPTION_CHANGE_NOT_ALLOWED` (غير مسموح هنا)، أو `INVALID_CATEGORY_OPTION_CHANGE` (صيغة خاطئة، أو حقل مجهول، أو عنوان فارغ)، أو `CATEGORY_OPTION_CONFLICT` (غيّر شخص آخر ذلك الخيار أو حذفه)، أو `CATEGORY_OPTIONS_CHANGED` (تغيّرت قائمة خيارات الحقل منذ تحميل البطاقة) – وعندئذ يعيد المستخدم التحميل ويحاول مجدّدًا.

حقول التصنيف متعدّدة القيم. يمكن لحقل التصنيف الآن أن يحمل عدّة قيم. ولا يصلح مثل هذا الحقل حقلًا للأعمدة أو تصنيفًا فرعيًا للوحة المهام، ولا حقلًا للفرز الافتراضي، ولا أساسًا لتجميع إحصائية في لوحة المعلومات، ولا تصنيفًا للشريط الزمني، ولا تصنيفًا رئيسيًا أو فرعيًا للكيان؛ انظر حقول التصنيف متعدّدة القيم [بناء التطبيقات › المجاميع والتجميع] للاطلاع على القاعدة كاملة.

الطلب من المستخدم

فعّل **الطلب من المستخدم** لاختيار مجموعة من الحقول يطلب المعرض من المستخدم النهائي تعبئتها عند تنفيذ حدث في الواجهة (مثل زرّ إجراء). وأثناء التشغيل، تُعرَض تلك الحقول في مربّع حوار ليتمكّن الحدث من جمع مُدخلات إضافية قبل تنفيذه.

شاشة لوحة المعلومات

فعّل **عرض لوحة المعلومات** ليتمكّن المستخدمون النهائيون من رؤية خلاصات الكيان وإحصائياته، ويمكنك جعل لوحة المعلومات شاشة الوصول الافتراضية للمعرض. ولا يمكن الجمع بين لوحة المعلومات و**الترشيح المسبق**. ويتطلّب **النقر للتصفّح** على لوحة المعلومات تفعيل شاشة قائمة المُدخلات، ويتطلّب تجاوز المصدر لاستعلام إحصائي أن يكون مصدر المعرض استعلامًا.

خلاصات نظرة الكيان العامة تحترم «التوافق مع مصدر المعرض» `Since 1.8` . عند تفعيل خيار **عرض الخلاصات المتوافقة مع مصدر الواجهة فقط**، كانت لوحة نظرة الكيان العامة تسرد مع ذلك جميع خلاصات الكيان، ثم يفشل تشغيل كل خلاصة غير متوافقة برسالة «Invalid summary source» . أما الآن فتُطبَّق عند جلب القائمة (`GET` `/v1/entities/{entity}/statistic-queries` في طلب يمرّ عبر معرض) الفحوص نفسها التي كانت تُطبَّق أصلًا عند تشغيل الخلاصة – أن تكون الخلاصة على كيان المعرض، وأن تكون لوحة معلومات المعرض مفعّلة، وأن يتطابق مصدرها حين يقيّده المعرض – وتُقسَّم القائمة إلى صفحات **بعد** ذلك الترشيح، فلا يحتسب `skip` و `limit` إلا الخلاصات التي يستطيع المعرض عرضها. أما الطلبات التي لا تمرّ عبر معرض، والطلبات التي تمرّ عبر صفحة، فلم تتغيّر.

تثبيت المعرض في الشريط الجانبي

ثبّت المعرض المُصمَّم في الشريط الجانبي عبر **التثبيت في الشريط الجانبي** ليصل إليه المستخدمون النهائيون مباشرة. ويتحكّم **سلوك عرض الشريط الجانبي** في حالة المعرض المثبّت عند فتحه – **مغلق** أو **مفتوح** أو **الإبقاء** (الاحتفاظ بالحالة الأخيرة).

استخدام الواجهات والتبديل بينها

أثناء التشغيل، يُفتح المعرض على **واجهته الافتراضية** – الواجهة الوحيدة المحدَّدة بوصفها افتراضية عند تصميم المعرض. وينتقل المستخدم النهائي إلى أي واجهة أخرى مفعّلة عبر مبدّل العرض. ويحدّد **ترتيب الشاشات** الخاص بالمعرض الشاشةَ

التي يُوجَّه إليها المستخدم (قائمة المُدخلات، أو الإنشاء، أو تفاصيل مُدخل واحد، أو لوحة المعلومات، أو التفاصيل مع التنقّل).
Since 1.4

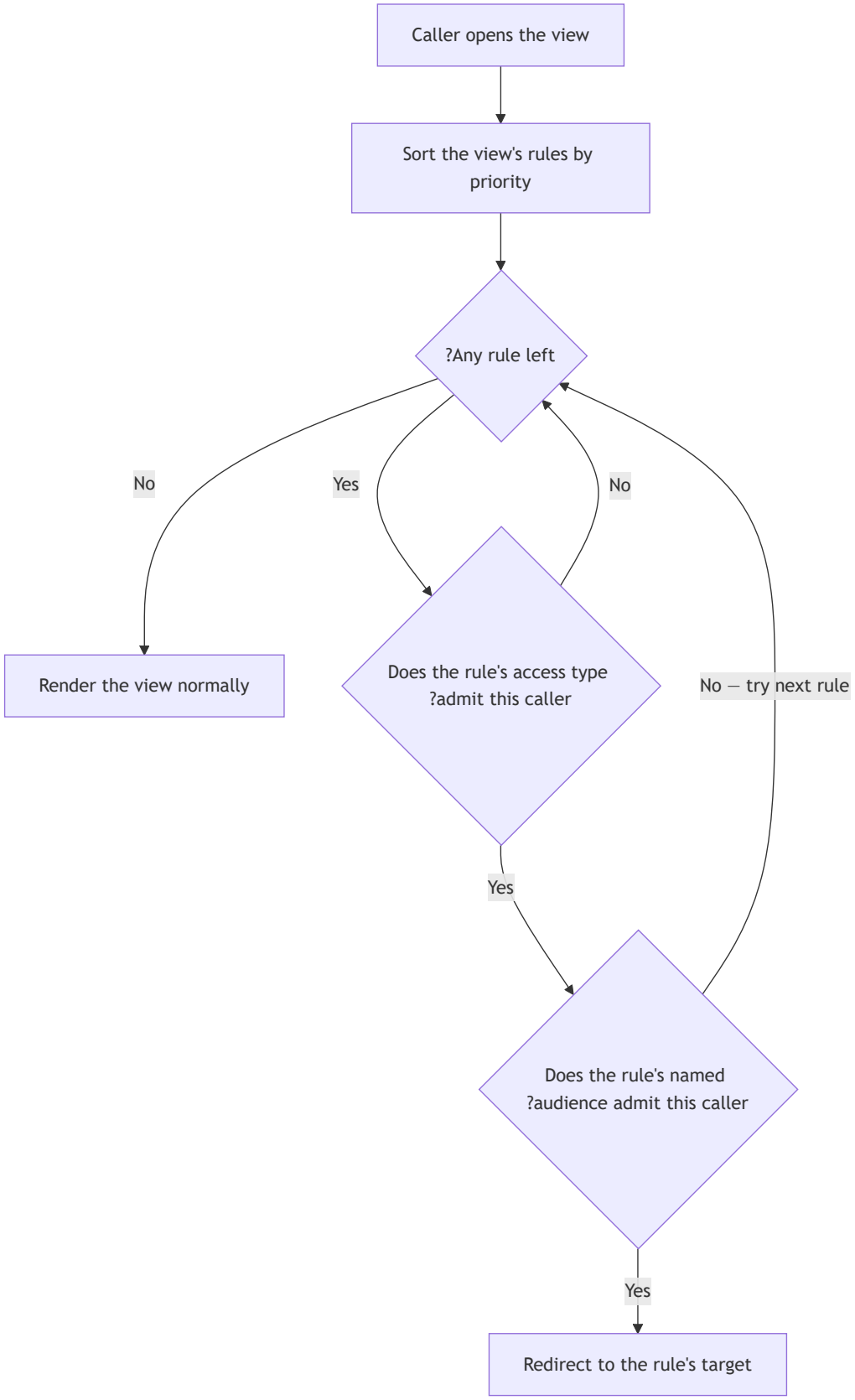
ويتوفّر ترتيب المُدخلات وإعادة ترتيبها في واجهات الجدول والبادج ولوحة المهام. وعلى الجوال، يُحفَظ على الخادم نقل بطاقة لوحة المهام بالسحب إلى عمود آخر، وإن كانت إعادة ترتيب أعمدة لوحة المهام بالكامل غير مدعومة بعد.

قواعد إعادة التوجيه

يمكن أن يتضمّن المعرض **قواعد إعادة توجيه** تُرسِل المستخدم النهائي مباشرةً إلى وجهة مختلفة – عنوان URL ثابت، أو معرض بيانات آخر، أو صفحة، أو تقرير – قبل عرض المعرض نفسه. وتُقيَّم القواعد بترتيب **الأولوية** (الأدنى أولًا)؛ وتُطبَّق أول قاعدة تطابق المستدعي، ويتوقّف التقييم عندها. أما المستدعي الذي لا تطابقه أي قاعدة فيرى المعرض بصورة معتادة.

وتجمع كل قاعدة بين شرطين يجب أن يتحقّقا **كلاهما** لكي تطابق المستدعي:

- **نوع الوصول** – `Anonymous` أو `Authenticated` أو `Everyone` أو `Restricted`. وهذا هو معيار الترشيح العام: فقاعدة `Anonymous` لا تشمل إلا زائرًا لم يسجّل الدخول، وقاعدة `Authenticated` لا تشمل إلا مستخدمًا مسجّل الدخول، وقاعدة `Everyone` تقبل الجميع دائمًا، أما `Restricted` فتقبل الجميع في مرحلة الترشيح الأولى هذه، إذ يقتصر الغرض منها على تضييق نطاقها في مرحلة الترشيح الثانية.
- **الجمهور المسمّى** – مستخدمون أو مجموعات أو كلاهما، مُدرَجون في القاعدة بأسمائهم. فالقاعدة التي لا تسمّي أحدًا تطابق كل من يقبله نوع الوصول فيها أصلًا، **باستثناء** قاعدة `Restricted`: فعدم تسمية أحد فيها يعني أنها لا تطابق أحدًا إطلاقًا، لأن مطابقة قاعدة `Restricted` فارغة للجميع كانت ستُبطل الغرض من اختيار `Restricted` أساسًا. أما القاعدة التي تسمّي مستخدمين أو مجموعات فلا تطابق إلا مستدعيًا مدرجًا في تلك القائمة، وذلك في جميع أنواع الوصول، لا في `Restricted` وحده.



`Since 1.10`، أصبح الشرطان يُفحَصان لكل قاعدة، في معارض البيانات والصفحات على السواء. وقبل هذا الإصلاح، كانت قاعدة `Restricted` وحدها تتحقّق من المستخدمين أو المجموعات المسمّاة فيها، فكانت قاعدة `Authenticated` التي تسمّي مجموعة أيضًا تطابق **كل** مستدعٍ مسجّل الدخول بصرف النظر عن عضويته، وتبقى المجموعة المُسمّاة إعدادًا بلا أثر، دون أي تنبيه. وأُصلِح معه عطل مرتبط: كانت الصفحة التي تتضمّن قواعد إعادة توجيه تُعرَض **فارغة تمامًا** للزوار المجهولين، لأن أداة

التقييم كانت تفشل عند غياب قائمة مجموعات الزائر المجهول بدل معاملتها بوصفها قائمة فارغة. وطُبِّق كلا الإصلاحين بأثر رجعي أيضًا على خطّي الإصدار الأخيرين، بوصفهما إصلاحًا على مستوى التصحيح، لا قدرة جديدة في هذين الخطّين.

وتعمل قواعد إعادة التوجيه بالآلية نفسها في معرض البيانات وفي الصفحة [بناء التطبيقات › الصفحات والقطع] – انظر تلك الصفحة لمعرفة كيف يتّسق إعداد الصفحة الافتراضية مع قواعد إعادة التوجيه الخاصة بها.

معاينة المعرض أثناء التصميم

أثناء بناء المعرض، استخدم **المعاينة** في أداة البناء للاطلاع على مظهره باستخدام بيانات تجريبية (وهمية) محدَّدة مسبقًا – دون الحاجة إلى مُدخلات حقيقية. ويتيح لك تبويب **معاينة الجوال** التحقّق من مظهر الأعمدة أو البطاقات المختارة وتخطيط أزرار الإجراءات على الهاتف قبل النشر. والمعاينة وضعٌ مقيَّد بنطاق الطلب (تُفعّله ترويسة معاينة)، وليست خاصيّة مخزّنة للمعرض، والتخفيف الذي تطبّقه على التحقّق من الأذونات **ضيّق**: فهو يسمح بمرور مورد غير موجود فعليًا كي تُعرَض بيانات العيّنة، ولا يتجاوز الأذونات بصورة شاملة. انظر وضع المعاينة [بناء التطبيقات] للاطلاع على عقد الطلب، وبيانات العيّنة المُولَّدة، ومعاينة الجوال الحيّة. **Since 1.8**

تنسيق المعرض ومساعدته ونصوصه البرمجية

إلى جانب خيارات العرض السابقة، يمكن أن يتضمّن المعرض تعديلات تنسيق خاصة به، ومحتوى مساعدة مضمَّنًا، ومنطقًا يعمل داخل المتصفّح – وتُؤلَّف جميعها في أداة بناء المعرض دون أي نشر.

التنسيق المخصّص (CSS) وأنماط العناصر

اربط قواعد تنسيق CSS بالمعرض، أو بعناصر مفردة داخله، لضبط مظهره (`appCssStyler` ؛ وتُقدَّم قواعد المعرض عبر `get-view-config-css-rules`). ويختلف هذا عن **أشكال تصنيف المدخل** السابقة – إذ تُعيد تلك تلوين المُدخلات بحسب حقل تصنيف، بينما يتيح لك التنسيق المخصّص تنسيق أي أجزاء من المعرض. ولأن هذه القواعد واسعة النطاق، اختبر أي تنسيق مخصّص في المظهرين الفاتح والداكن وفي معاينة الجوال قبل النشر.

إصلاح: عادت أصناف CSS المحفوظة تظهر في إعدادات التنسيق المخصّص `Since 1.8` . كانت الأصناف المحفوظة للمعرض وحقوله ولوحات معلوماته وأزراره وخطوات سير عمله تُخزَّن وتُطبَّق، غير أن لوحة الإعدادات كانت تعرضها فارغة بعد إعادة التحميل. أما الآن فتعرض اللوحة كل صنف محفوظ، ويبقى الصنف الذي تمسحه محذوفًا. ويصل هذا الإصلاح في خط 1.10 مع الإصدار 1.10.2.

وثمة إصلاح ذو صلة (`Since 1.8`) صحّح كيفية عمل التنسيق الذي يجريه **نصّ برمجي** يعتمد على DOM عندما يكون المعرض نفسه مضمَّنًا أكثر من مرة في صفحة – انظر النصوص البرمجية لكل معرض ولكل كيان أدناه.

المساعدة السياقية

ألّف **محتوى مساعدة لكل معرض ولكل حقل** بصيغة markdown؛ ويعرضه أسس في شريط مساعدة جانبي بجوار المعرض (ميزة `help-and-documentation`). استخدمه لشرح الغرض من المعرض أو لتوضيح حقول مفردة مباشرةً، دون إحالة المستخدمين النهائيين إلى مستند منفصل.

ويمكن أيضًا تأليف ملاحظة مساعدة الحقل **لكل شاشة على حدة** – إذ تسترجع كل شاشة من شاشات قائمة المُدخلات وتفاصيل المُدخل والإنشاء والتحرير و«الطلب من المستخدم» مساعدتها الخاصة بصورة مستقلة، لأن الحقل قد يظهر في بعض الشاشات دون غيرها. وتتكوّن لوحة **المساعدة السياقية** في أداة البناء من أقسام قابلة للطي: قسم للمعرض بأكمله ثم

قسم لكل شاشة، وتسرد اللوحة كل حقل يمكن أن تسترجعه أيٌّ من الشاشات الخمس – بحيث تبقى مساعدة الحقل الذي يظهر (مثلًا) في عملية التحرير، لكنه غير موجود في إسقاط المعرض الأساسي، متاحة للتأليف ومرتبطة بالشاشة الصحيحة.

إصلاح: ملاحظة مساعدة حقل فرعي داخل كتلة لم تعد تُعطّل المعرض `Since 1.8` . كانت إضافة مساعدة سياقية إلى حقل متداخل داخل كتلة تؤدّي إلى فشل جميع شاشات المعرض وتعذّر فتحها. وقد أُصلِح ذلك بربط ملاحظة مساعدة الحقل الفرعي بالحقل الذي ينتمي إليه من المستوى الأعلى (بحيث تُعرَض الملاحظة متداخلة تحت ذلك الحقل، بدل الاستعلام عن قيمة لم تكن تتضمّن الحقل الفرعي مفتاحًا لها أصلًا)، وببناء خيارات المساعدة لكل شاشة من قائمة حقول تلك الشاشة نفسها بدلًا من إسقاط حقول المعرض الأساسي – وبذلك لم تعد مساعدة حقل لا يوجد إلا في شاشة التحرير، مثلًا، تُعطّل شاشات لا يوجد فيها ذلك الحقل.

النصوص البرمجية لكل معرض ولكل كيان

للمنطق الذي يعمل بالكامل في متصفّح المستخدم النهائي – دون أي وظيفة تتطلّب نشرًا – يمكنك ربط رماز JavaScript يضبطه المستخدم بمعرض أو بكيانه. وأثناء دورة حياة عرض المُدخل أو تحريره، تشغّل الواجهةُ الأماميةُ نصّ **واجهة المعرض** (view-face) أوّلًا، ثم نصّ **واجهة الكيان** (entity-face) عبر `scriptAttributes[face]` ، من خلال `EntityScriptExecutorService.run` . وهذه إحدى طرائق المنصّة لتشغيل منطقك الخاص دون نشر، إلى جانب وظائف المستخدم والتفاعلات (reactions).

إصلاح: تأثيرات النصّ البرمجي على DOM تستهدف الآن نسخة القطعة الصحيحة `Since 1.8` . كثيرًا ما ينسّق نصّ برمجي عنصرًا أو يبدّل حالته عبر معرّفه (`document.querySelector('#some-id')`). وعندما يكون المعرض نفسه مضمَّنًا أكثر من مرة في صفحة بوصفه قطعة معرض بيانات، تتكرّر في كل نسخة معرّفات العناصر نفسها التي ضبطها المدير، فكان البحث غير المقيَّد يطابق دائمًا **النسخة الأولى** في الصفحة؛ ولذلك كان التنسيق الذي يُحدثه النصّ البرمجي قد يُطبَّق على القطعة الخطأ، أو لا يظهر أصلًا على القطعة المعروضة فعليًا. أما الآن فعمليات البحث التي يجريها النصّ البرمجي للقطعة (`document.querySelector` أو `querySelectorAll` أو `getElementById`) لا تبحث إلا ضمن الشجرة الفرعية الخاصة بتلك القطعة؛ ولا تتأثر النصوص البرمجية التي تعمل خارج نسخ القطع.

حفظ المعرض: التعارضات وحالات الرفض `Since 1.10`

- **لا يستقرّ الحفظ إلا على النسخة التي قرأتها.** يُحفظ المعرض بوصفه مستندًا كاملًا تحميه قيمة `updateIdentifier` ؛ وإذا كانت القيمة متقادمة، يُرفض الحفظ بالرمز `409 UPDATE_CONFLICT` . وقبل 1.10، كانت المقارنة بين القيمتين تقتصر على قراءة مسبقة، مما كان يتيح لإحدى عمليتَي حفظ اجتازتا تلك القراءة أن تكتب فوق نتيجة الأخرى. أما الآن، فالكتابة نفسها مشروطة – لا تُستبدل بيانات المعرض إلا ما دامت قيمة `updateIdentifier` المخزّنة تطابق القيمة التي قرأتها – فتفشل العملية الثانية من عمليتَي حفظ متزامنتين بالرمز 409، ويلزمها إعادة القراءة. وتُحفظ الصفحات بالطريقة نفسها.
- **تعديلات المدخلات تحتاج إلى عملية مسموحة.** يُرفض حفظ معرض يفعّل تعديلات المُدخلات دون أي عملية مدعومة بالرمز `400` (`DATA_VIEW_INVALID_ALLOWED_OPERATIONS` ، «Allowed Operation must be provided»). وكان يظهر سابقًا بالرمز `500` .
- **المعرض الذي لا يستطيع أي جمهور فتحه 403 لا 500.** المعرض أو الصفحة أو التقرير الذي لا يفعّل جمهور الزوار المجهولين **ولا** جمهور المستخدمين المصادَق عليهم بيانات صحيحة – إذ يُخزَّن معرض الوكيل بهذه الصورة عمدًا – غير أن الاستعلام من المنصّة عمّا إذا كان بإمكان المستدعي فتحه (`GET /v2/access-object/{id}`) كان يفشل بالرمز `500` . أما الآن، فتستجيب المنصّة بالرمز `403 ACCESS_OBJECT_NO_AUDIENCE` ، وتمرّر البوابة وخدمات التدقيق والتعليقات والشريط الزمني وسير العمل الرمز 403 دون تغيير.

- **حفظ المعرض بلا جسم 400** `Since dev` . أصبح `PATCH /v1/entities/{entity}/views/{id}` يحدّد جسم الطلب بوصفه مطلوبًا، فيُرفض الطلب الخالي من الجسم بالرمز `400 MISSING_REQUIRED_PARAMETER` قبل أن يصل إلى المتحكّم؛ وكان يفشل سابقًا بالرمز `500` . ويظلّ الجسم الجزئي مقبولًا كما كان.
- **تُتتبَّع الملفات مع المعرض.** عند حفظ معرض أو صفحة، تُسجَّل الملفات التي يشير إليها قبل الكتابة، وتُحرَّر الملفات التي لم يعد يستخدمها بعدها؛ وعند حذفه، تُحرَّر جميعها.

المعارض المثبَّتة (من حزمة) `Since 1.10`

يكون المعرض الذي أنشأه **تثبيت** حزمة مقفلًا ومرتبطًا بتلك الحزمة: فتُرفض الكتابة بالرمز `403` `PACKAGE_RESOURCE_READ_ONLY` («belongs to installed package …. fork it to edit»). ويظلّ بإمكان العميل تعديل بعض خصائصه، و**يحتفظ بها تحديث الحزمة**:

ما هو	القاعدة
منشور / مخفي (`isPublished`)	يُغيَّر من قائمة معارض الكيان؛ ولا يُتحقَّق منه أبدًا بمقارنته بقيمة `updateIdentifier` .
الاسم والوصف والأيقونة	قابلة للتحرير في محرّر المعرض (باستثناء اسم المعرض الضمني). ويعرض المحرّر قيمة الحزمة بجانب قيمتك، وتُسجَّل قيمة الحزمة نفسها في `packageLabels` أول مرة تختلف فيها قيمتك عنها؛ وتؤدي إعادة كتابة قيمة الحزمة إلى إزالة ذلك السجل. ويتطلّب تغيير التسمية قيمة `updateIdentifier` الحالية (وإلا فالرمز `409`). وتمسح القيمة `description: null` الوصف دائمًا؛ أما `icon: null` فلا تمسح الأيقونة إلا مع `?clearIcon=true` .
ترتيب العرض والجمهور وحقوق الوصول	يُحتفظ بها عند التحديث (تُطبَّق قيم العميل مجددًا فوق قيم المعرض الوارد).

وعند الحفظ عبر `PATCH /v1/entities/{entity}/views/{id}` في معرض مقفل، لا تُكتب **إلا** تلك المفاتيح – بكتابة جزئية لا تمسّ شيئًا آخر في المعرض المخزَّن – ويُتجاهل ما سواها في الجسم؛ ويُرفض الجسم الذي لا يغيّر أيًّا منها بالرمز 403 الأصلي. وتكون بقية عناصر المحرّر غير مفعّلة (انظر شاشات التحزيم)؛ وتبقى تجاوزات العلاقات والنصوص البرمجية قابلة للاستخدام. ولتغيير أي شيء آخر، اشتقّ التثبيت.

مراجع التخطيط على مسار المستخدم النهائي `Since 1.10`

في المعرض المبني على قالب العرض **Layout builder**، يحلّ `GET /v2/views/end-user/{id}` مراجع مُدخلات SHELL التي يحدّدها القالب (`layout_list` و `layout_view` و `layout_edit`) ضمن الخاصية `layouts` – `{listEntry, viewEntry, editEntry}`، وتكون قيمة كلٍّ منها `{id, name, definition}` لمُدخل SHELL، أو `null` إذا لم يُربط أي مُدخل أو كان المُدخل مفقودًا أو غير نشط أو غير صالح. ولا يخزّن القالب إلا المراجع، لذا يسري تعديل التخطيط على كل معرض مرتبط به دون إعادة حفظه، ويتلقّى الزوار المجهولون التعريف أيضًا. ولا يتضمّن مسار v1 الخاصية `layouts` . **النصف الخلفي موجود في 1.10، لكن لا عميل مشحون يستخدمه بعد:** فقالب Layout builder ومنشئ الهيكل اللذان يقرآن `layouts` هما `Since dev` .

ملاحظات الإصدار

المكوّنات التأسيسية – الواجهات السبع، والترشيح المسبق، ولوحات المعلومات، وشاشات المُدخل والتحرير – **Long-standing** (موجودة منذ ما قبل 1.2) ولا تحمل شارة. أمّا الإضافات المرتبطة بالتصميم منذ ذلك الحين، فهي الآتية:

الميزة	Since
تتبّع الأصول في الشكل الشجري	1.2
ترتيب الشاشات (إعادة التوجيه بين الشاشات)	1.4

الميزة	Since
سلوك النقر على صفّ الجدول	1.6
تخطيط أزرار الإجراءات	1.8
وضع المعاينة	1.8
ترتيب التكرارات (خيار لكل عملية)	1.8
اقتطاع أزرار النظام الضمنية عند الحفظ بدلًا من رفضه	1.8
حقل تصنيف لوحة المهام داخل كتلة	1.8
السماح بتغيير عمود لوحة المهام بصرف النظر عن خيار إعادة الترتيب	1.8
توسّع الشكل الشجري عند الترشيح بالحالة وحدها	1.8
عرض سير العمل الافتراضي (المسار أو السجل التاريخي)	1.8
حلّ الشريط الجانبي للعلاقة الواردة في الشكل الشجري بصورة صحيحة	1.8
إصلاحات استعادة لوحة الترشيح المسبق وتسميات نوع الاستعلام	1.8
إصلاح تعطّل المساعدة السياقية عند حقل فرعي داخل كتلة	1.8
إصلاح نطاق DOM للنصّ البرمجي عند تضمين معرض أكثر من مرة في صفحة	1.8
تفعيل المحصّلات لكل معرض على بطاقة قائمة المُدخلات	1.10
ضبط تجميع المُدخلات على بطاقة قائمة المُدخلات (المستويات، والبدء مطويًّا، والسماح للمستخدمين النهائيين بالتجميع)	1.10
إعادة وضع خيار الفرز مباشرةً تحت الفرز الافتراضي في إعدادات المعرض	1.10
نوع معرض الوكيل (`kind: Proxy`) الذي يضبط أذونات الكتابة لكيان وكيل افتراضي	1.10
قواعد إعادة التوجيه: مطابقة الجمهور بشرطين، وإصلاح عطل الزائر المجهول	1.10
لوحة المهام: حقل أعمدة مرجعي، وأنماط الأعمدة (الكل أو المستخدمة أو المحدّدة)، واختيار الأعمدة لكل مستخدم، وفحوص نقل كل بطاقة	1.10
المرشحات السريعة: الاسم، والموضع (قائمة الحالة أو مستقلّ)، والترتيب الصريح، والمجموعات، وتعدّد الخيارات الافتراضية	1.10
التصدير السريع من المعرض (**السماح بالتصدير**، XLSX أو CSV) و«تصدير المُدخلات» في الإدارة	1.10
إضافة خيارات التصنيف وتعديلها من بطاقة المُدخل؛ وحقول التصنيف متعدّدة القيم	1.10
الحفظ المشروط: لا يُكتب المعرض أو الصفحة إلا ما دامت قيمة `updateIdentifier` مطابقة	1.10
المعرض المثبَّت: إعادة التسمية وتغيير الوصف ومسح الأيقونة، مع الاحتفاظ بها عبر تحديثات الحزمة	1.10
المعرض الذي يفعّل تعديلات المُدخلات بلا عملية مسموحة يستجيب بالرمز 400؛ والمعرض الذي لا يستطيع أي جمهور فتحه يستجيب بالرمز 403	1.10
ترشيح خلاصات النظرة العامة للكيان حسب مصدر المعرض عند ضبط «التوافق مع مصدر المعرض»	1.8
الحفظ عبر معرض يُبقي أعمدة الكتلة القابلة للتكرار التي لا يمنحها المعرض	1.4
عودة ظهور أصناف CSS المحفوظة في إعدادات التنسيق المخصّص للمعرض	1.8
اقتران صفوف الكتلة القابلة للتكرار بـ `__id` عند الحفظ عبر معرض (ابتداءً من الإصدار 1.10.1)	1.10
طلب `PATCH` للمعرض بلا جسم يستجيب بالرمز 400 بدلًا من 500	dev
حلّ مراجع التخطيط على `GET /v2/views/end-user/{id}` (الخلفية فقط؛ والعميل الذي يستخدمها `Since dev`)	1.10

واجهة الأذونات البرمجية

تُوجَّه هذه الصفحة إلى فرق التكامل والمهندسين الذين يديرون الوصول برمجيًا، أو يحتاجون إلى فهم دقيق للعمليات التي تنفّذها شاشات الأذونات في تطبيق الإدارة. أما المسؤول، فيجد شرحًا مبسّطًا للنموذج نفسه مع الشاشات في الأذونات والوصول [الإدارة]. في حين يرد توثيق محرّك التقييم نفسه (المُخوِّل، والمصوّتون، والدمج، والخبء، وفحص OpenFGA الظلّي) لمهندسي المنصة في المحرك: الأذونات والمصوّتون، وترد قائمة نقاط النهاية المولَّدة لكل عملية في data-service – Access.

جميع المسارات أدناه نسبية إلى واجهة البيانات (`/api`)، وتَرِد كل استجابة ضمن الغلاف القياسي `{success, result}` الموصوف في استدعاء واجهة HTTP.

سجلّ المنحة

يُمثَّل كل إذن بسجلّ `AccessRight` واحد (في المجموعة `access_right`)، بحيث يُخصَّص سجلّ لكل منحة. ويحتوي الحقل `value` من النوع `AccessRightValue` على ما يلي:

الحقل	المعنى
`entitled_entity`	الطرف الممنوح له الوصول: مرجع `{dataEntity, dataEntry}` يكون فيه `dataEntity` إحدى القيم `USER` أو `GROUP` أو `ROLE` ، و `dataEntry` هو المعرّف.
`entitlement`	المورد الذي تنطبق عليه المنحة: مرجع `{dataEntity, dataEntry}` يكون فيه `dataEntity` نوع الاستحقاق (وحدة، كيان، معرض، …) و `dataEntry` معرّفه. **فارغ أو غائب في المنح على مستوى النظام.**
`level`	مستوى الوصول (انظر أدناه).
`permission_type`	اختياري. يميّز منحة `NORMAL` من منحة القراءة `SENSITIVE_FIELD` ، ويُعامَل غيابه على أنه `NORMAL` .
`scope`	مسار معياري اختياري (مثل `module/entity/field`) تستخدمه منح الحقول الحسّاسة.
`expiry`	نافذة صلاحية اختيارية تُخزَّن بالصيغة `[startIso, endIso]` ؛ وتعني القيمة `null` أن المنحة لا تنتهي أبدًا.

فمثلًا، يُمثَّل الأمر «امنح الدور R مستوى التعديل على الوحدة M» بسجلّ واحد فيه `entitled_entity = {ROLE, R}` و `entitlement = {DataModule, M}` و `level = EditAccess` .

النماذج المرتبطة:

النموذج	دوره
`AccessRight`	`access_right` : سجلّ الإذن المخزَّن.
`AccessRightValue`	`entitled_entity` + `entitlement` + `level` (+ `permission_type` و `scope` و `expiry`).
`EntitledEntityPermission`	مرجع الطرف: `dataEntity` ∋ `{USER, GROUP, ROLE}` .
`EntitlementReferenceProperty`	مرجع المورد: `dataEntity` ∋ `EntitlementEnum` .
`UserDto` / `GroupDto`	مجموعة أو دور أو مستخدم مع قائمة أذوناته من `AccessRightDto` .
`RoleDataEntry` / `RoleDataEntryValue`	دور مسمّى: الاسم، والوصف، والارتباط بالوحدات التنظيمية، والنوع، ونوع الإسناد (انظر نموذج بيانات الأدوار).
`GrantAccessDto`	جسم طلب الكتابة: مستوى `level` واحد.

لمن تُمنح المنحة (الكيانات المُستحِقة)

تستهدف المنح أحد ثلاثة أنواع من الكيانات المُستحِقة (`EntitledEntityEnum`):

القيمة	المعنى
`USER`	مستخدم فرد.
`GROUP`	مجموعة. وتحمل المجموعات قيمة من `GroupTypeEnum`: `UserGroup` (0)، و`Role` (1)، و`OrgUnit` (2)، و`Virtual` (3)، و`QueryGroup` (4).
`ROLE`	دور.

الأدوار والوحدات التنظيمية مجموعاتٌ تميّزها قيمة `GroupTypeEnum`، ولذلك يمكن لسجلّ منحة واحد أن يستهدف مجموعة مستخدمين عادية، أو دورًا، أو مجموعة وهمية، أو مجموعة استعلام، أو وحدة تنظيمية. ولكل دور مجموعة دور وهمية تُنشأ تلقائيًا وتضم جميع المستخدمين الذين يحملون الدور. وفي واجهات سرد الأذونات يكون نوع الكيان المُستحِق في المسار `user` أو `group` (دون تمييز بين الأحرف الكبيرة والصغيرة، إذ يُحوَّل إلى أحرف كبيرة قبل الاستخدام).

على ماذا تنطبق المنحة (نطاقات الاستحقاق)

النطاق (`entitlementType`)	قيمة `dataEntity` المخزَّنة	ما يتحكم فيه
`System`	`System`	القاطن كله، وتكون `entitlementId` فارغة.
`Module`	`DataModule`	وحدة واحدة، وتمتد منحة الوحدة أيضًا إلى الكيانات التي تحتويها.
`DataEntity`	`DataEntity`	كيان بيانات واحد.
`DataView`	`DataView`	معرض بيانات واحد.
`GenericView`	`GenericView`	صفحة.
`Report`	`Report`	تقرير.

تُظهر منحة الوحدة الكيانات التي تحتويها. أما منحة الكيان الواحد فتُظهر وحدته الأم بوصفها حاوية للتنقّل فقط، و**لا** تكشف الكيانات الشقيقة. و`Since 1.8` تُعاد إلى العملاء معرّفات الوحدات الأم التي يمكن الوصول إليها بهذه الطريقة في الحقل `entityParentModuleIds` ضمن استجابة المستخدم الحالي (انظر استجابة المستخدم الحالي).

مستويات الوصول

`AccessRightLevelEnum` سُلّم **رقمي مرتّب**، وتعتمد عدة فحوص على المقارنة العددية بين مستوياته.

المستوى	القيمة	المعنى
`None`	`-1`	علامة «الإزالة». إرسالها **يحذف** المنحة المطابقة (انظر سلوك upsert)، ولا تخزّنها عملية upsert أبدًا.
`Restricted`	`-0.5`	«المستوى الأساسي للانضمام»، وهو خاص بالنظام وحده. لا يمنح شيئًا بذاته، لكنه يتيح سريان منح المستخدم على الموارد، ويتيح الوصول للقراءة فقط إلى البنية التحتية الأساسية (المستخدمين والمجموعات والوحدات التنظيمية) لمن يحمل منحة واحدة على الأقل على وحدة أو كيان. `Since 1.8`.
`ViewAccess`	`0`	قراءة تعريف الكائن.
`DeveloperAccess`	`0.5`	وصول يتيح البناء باستخدام أدوات الاستوديو (مصمّم الكيانات، والمعارض، والسمات، والخطوط، واللغات). ويقع *بجانب* مستوى التعديل: فمستوى التعديل يستوفي كل فحص يستوفيه مستوى المطوّر، لكن مستوى المطوّر لا يمنح الوصول إلى المُدخلات. وهو مستوى قديم يسبق 1.2.
`EditAccess`	`1`	القراءة والكتابة، بما في ذلك بيانات المُدخلات.
`AdminAccess`	`2`	تحكّم كامل، بما في ذلك إدارة الأذونات. لا يُتاح إلا على **النظام**، و`Since 1.6` لا تتيح منحة المسؤول على وحدة أو كيان أي وصول.

المستويات المتاحة لكل هدف (`GRANTABLE_LEVELS_BY_ENTITLEMENT`): النظام: عرض، ومطوّر، وتعديل، ومسؤول (إضافةً إلى Restricted)؛ والوحدة والكيان: عرض، ومطوّر، وتعديل؛ والمعرض والصفحة والتقرير: عرض.

بيانات المُدخلات تتطلب مستوى التعديل. يتطلب سرد مُدخلات كيان مستوى التعديل على الكيان أو على وحدته، أو مستوى التعديل أو المسؤول على النظام؛ ولا يكفي مستوى العرض. و`Since 1.8` تتطلب قراءة مُدخل واحد بمعرّفه (من الشريط

الجانبي للمعاينة السريعة لعلاقة مثلًا) الوصول نفسه بمستوى التعديل، وإلا رُفض الطلب بالرمز `ENTRY_READ_FORBIDDEN`. أما القراءات التي تمرّ عبر معرض أو صفحة أو خدمة فتخضع لقواعد وصول ذلك الكائن نفسه، وتظل الكيانات من نوع Service معفاة.

يُرفض وصول المستخدم الذي لا يملك منحة على النظام. إذا كانت حصيلة منح النظام المدمجة فارغة (لا منحة مباشرة على النظام ولا منحة عبر مجموعة)، يرفض المُخوِّل الأعلى كل طلب لا يُنفَّذ ضمن سياق معرض. ولا تمنح خدمة المستخدمين أي إذن افتراضيًا عند إنشاء المستخدم، لذلك يجب أن تمنح عملية الانضمام كل مستخدم مستوى وصول على النظام، `Restricted` على الأقل، مباشرةً أو عبر مجموعة.

المنح والتغيير والإبطال

تُنشأ المنح وتُغيَّر عبر عملية **upsert** لا يتغيّر أثرها بتكرار تنفيذها (idempotent)، ولها صيغتان: إحداهما تستخدم نوع الاستحقاق ومعرّفه في المسار، والأخرى تستخدم معرّف كائن نظام:

```
PUT /v1/permissions/{entitlementType}/{entitlementId}/{entitledEntityType}/{entitledEntityId}
PUT /v1/permissions/{systemObjectId}/{entitledEntityType}/{entitledEntityId}
```

جسم الطلب (`GrantAccessDto`):

```
{ "level": 1 }
```

`level` مطلوب، ويجب أن يكون إحدى قيم `AccessRightLevelEnum` الرقمية. وتتمثل معاملات المسار فيما يلي:

- `entitlementType`: `System` | `Module` | `DataEntity` | `DataView` | `GenericView` | `Report`.
- `entitlementId`: معرّف الكائن الهدف. ويكون فارغًا مع `System` (يحوّل الموجّه القيمة الغائبة إلى السلسلة `"null"`، فتُقبل وتُعامَل على أنها System).
- `entitledEntityType`: `user` أو `group` أو `role` (يُحوَّل إلى أحرف كبيرة ويُتحقق من مطابقته لقيم `EntitledEntityEnum`).
- `entitledEntityId`: معرّف المستخدم أو المجموعة أو الدور.

تبحث صيغة كائن النظام عن الكائن في المجموعة `SystemObject`، وتحوّل نوعه إلى استحقاق (`DataView → DataView`، و `Page → GenericView`، و `Report → Report`)، ثم تنفّذ عملية upsert نفسها. وإذا لم يكن الكائن موجودًا أو لم يكن نوعه أحد هذه الأنواع الثلاثة، يفشل الاستدعاء بالرمز `ENTITY_NOT_FOUND` / `INVALID_ENTITILED_ENTITY`. وتتوفر صيغة كائن النظام، وقراءات DTO لأذونات كل مستخدم وكل مجموعة، كلها `Since 1.6`.

سلوك upsert

ينفّذ `AccessRightsService.upsertPermission` الخطوات التالية بالترتيب:

1. `Restricted` **خارج** `System` **يعني إبطال المنحة** `Since 1.8`. لا يُخزَّن `Restricted` أبدًا على منحة وحدة أو كيان أو معرض؛ بل يُحوَّل طلب هذا المستوى على تلك الموارد إلى `None` ويُوجَّه إلى مسار الإبطال (الخطوة 4)، إذ إن إبطال منحة على مورد هو ما يُعيد الطرف إلى مستواه الأساسي `Restricted` على النظام. (قبل 1.8 كان هذا الطلب يفشل بالاستثناء `RestrictedLevelSystemOnlyException`، ولم يكن خفض منحة وحدة أو كيان ذات مستوى أعلى ممكنًا إلا بحذفها.)
2. **يجب أن يكون الكيان المُستحِق نشطًا**: يحمّل المتحكّم المستخدم أو المجموعة، ويرفض الطلب بالاستثناء `EntitledEntityNotActiveException` إن لم تكن قيمة حالته `state` هي `Active`.

3. **البحث عن منحة قائمة**: يبحث عن منحة مطابقة غير مؤرشفة (`Archived`) للزوج (الكيان المُستحِق، الاستحقاق)؛ فتُطابَق منح النظام بغياب الاستحقاق أو بكون قيمته null، بينما تُطابَق منح الموارد بمرجع الاستحقاق. ويجري تخطّي السجلات المؤرشفة لمنع إعادة استخدام منحة غير نشطة دون علم. و `Since 1.8` تستبعد مطابقة منح النظام أيضًا سجلات `permission_type = SENSITIVE_FIELD` : فكلا النوعين يحمل استحقاقًا فارغًا، وقبل الإصلاح كان منح مستوى على النظام لمستخدم لا يحمل إلا منحة حقل حسّاس يؤدي إلى مطابقة تلك المنحة، فيُبلغ الاستدعاء عن نجاحه دون إنشاء إذن على النظام، وتُفسَد منحة الحقل الحسّاس بقيمة `level` لا تخصّها.
4. `level === None` ⇒ **حذف**: تُزال المنحة المطابقة. وينطبق ذلك على جميع النطاقات، بما فيها System. وإن لم توجد منحة مطابقة، فلا يكون لـ `None` أي أثر.
5. **حارس التخفيض**: إن وُجدت منحة مطابقة، وكان المستوى الجديد *أدنى* من الحالي، واختلف نوع الاستحقاق عن نوعه في المنحة المخزَّنة، ولم يكن النطاق System، يُطلق الاستدعاء الاستثناء `UserGroupAlreadyHasHigherPermissionLevelException` .
6. **وإلا فالتحديث أو الإنشاء**: يُحدَّث مستوى المنحة القائمة (مع تسجيل بيانات التحديث في `updateInfo`)، وإلا أُنشئ `AccessRight` جديد بالحالة `Active` في المخزن العام.
7. **الحدث**: يُنشر `AccessRightChangedEvent(entitlementType, entitlementId)` لتحديث الخبء والمكوّنات المعتمدة عليه. وطريقة إبطال الخبء مشروحة في المحرك: الأذونات والمصوّتون – إبطال الأجيال (epoch).

لا توجد نقطة نهاية منفصلة لحذف المنحة: أرسل `level: -1` في طلب `PUT` نفسه. أما الاستبدال الجماعي فيُعالَج داخليًا عبر `storePermissionsInAccessRights(..., isUpdate = true)` ، الذي يقارن المجموعة المرسلة بالمنح الحالية ويضيف المنح أو يزيلها بحسب الفرق.

من يحق له الإسناد

تختلف ضوابط الوصول إلى صيغتي upsert، وتتطلب كلتاهما كذلك رخصةً تشمل الإدارة.

الصيغة التي تستخدم نوع الاستحقاق ومعرّفه في المسار يحرسها `voteByEntitlementFactory(0, 1)` ، الذي يشترط في طلب `PUT` أن يكون المستدعي قادرًا على **تعديل الاستحقاق الذي يمنح الوصول إليه**:

- منح `System` ← يجب أن يكون المستدعي مطوّر نظام أو أعلى (`isSystemDeveloper`).
- منح `Module` ← يجب أن يكون المستدعي قادرًا على تعديل تلك الوحدة (`canEditModule`).
- منح `DataEntity` ← يجب أن يكون المستدعي قادرًا على تعديل ذلك الكيان (`canEditEntity`).

صيغة كائن النظام لا تتضمن فحص استحقاق خاصًا بها، ولذلك فهي متاحة **لمسؤولي النظام وحدهم**: إذ لا تقبل المنصة من غير مسؤول النظام طلبًا إلى مسار كتابة لا يحدّد الأدوار المسموح لها. (وكان على هذه الصيغة مصوّت استحقاق لا يتوصّل إلى أي قرار، فأُزيل في 1.8.)

لا تتجاوز المنحة مستوى المستدعي `Since 1.8`

ما لم يكن المستدعي **مسؤول** نظام، ترفض الصيغة التي تستخدم نوع الاستحقاق ومعرّفه في المسار:

- أي مستوى مطلوب **يتجاوز مستوى المستدعي الفعّال** على ذلك الهدف،
- و**أي تغيير لمنحة تتجاوز** مستوى المستدعي أصلًا، أو إزالة لها.

ويأتي الرفض بالرمز `403 PERMISSION_GRANT_EXCEEDS_CALLER_ACCESS` ("You cannot grant, change or revoke access above your own level")، وتتضمن تفاصيله `details` القيم `entitlementType` و `entitlementId` و `callerLevel` و `requestedLevel` و `currentLevel` . ويُحتسب المستوى الفعّال للمستدعي من منحه على النظام، ومنحه المباشرة ومنح

مجموعاته على الهدف، وما يرثه من وحدته أو كيانه؛ وعند العمل بالتفويض تُحتسب أذونات التفويض. وتُعفى الاستدعاءات الداخلية للمنصة (انضمام المستخدمين، وإجراءات التفاعلات). وقبل 1.8 كان فحص التعديل هو الفحص الوحيد، فكان بوسع مطوّر النظام أن يجعل نفسه مسؤول النظام. وترد تفاصيل المحرك في تقييد منحة الإذن بمستوى المستدعي نفسه.

التفاعلات التي تمنح الأذونات أو تزيلها `Since 1.10`

يمكن أن تتضمن التفاعلات الإجراءين *منح إذن* و*إزالة إذن*. وعند **حفظ** تفاعل كهذا، تتحقق المنصة من أن الشخص الذي يحفظه يملك أصلًا الإذن الذي سيمنحه الإجراء: فإجراء *منح إذن* يتطلب على الأقل المستوى الممنوح على ذلك المورد، وإجراء *إزالة إذن* يتطلب مستوى **المسؤول** عليه. ولا تُقبل لكل نوع مورد إلا المستويات التي يعرضها محرّر الأذونات (للنظام: عرض، ومطوّر، وتعديل، ومسؤول؛ وللوحدة والكيان: عرض، ومطوّر، وتعديل؛ وللمعارض والصفحات والتقارير: عرض). وإن فشل الفحص رُفض الحفظ بالرمز `PERMISSION_ACTION_EXCEEDS_SAVER_ACCESS`، كما يُرفض الحفظ إن تعذّرت قراءة أذونات المؤلّف. ويُعفى مستخدم النظام من هذا الفحص. وعند تشغيل التفاعل، لا يخفّض *منح إذن* منحةً مباشرة قائمة أبدًا (ويُبلَّغ عن تخطّي الطرف بالوصف «أعلى أصلًا»)، أما *إزالة إذن* فتخفّض المستوى بطبيعتها. وترد تفاصيل المحرك في قراءات الوصول الداخلية وسقف التفاعلات.

التحقق من المعاملات

قبل تنفيذ أي عملية منح أو سرد، يفحص `validatePermissionsParams` مسار الطلب:

- يجب أن يكون `entitlementType` قيمة صالحة من `EntitlementEnum`، وإلا أُطلق الاستثناء `PermissionScopeNotSupportedException`.
- يجب أن يكون `entitledEntityType` قيمة صالحة من `EntitledEntityEnum` (`ROLE` / `GROUP` / `USER`)، وإلا أُطلق الاستثناء `PermissionScopeNotSupportedException`.
- يجب تقديم `entitlementId` مع أي استحقاق غير `System`، وإلا أُطلق الاستثناء `EntitlementTypeRequiredException`.

قراءة الأذونات وتدقيقها

تقبل نقاط نهاية السرد معاملات الاستعلام القياسية `filter` / `where` في LoopBack، ولكلٍّ منها صيغة `/count`.

نقطة النهاية	الحارس	ما تُعيده
`GET /v1/permissions/{entitlementType}/{entitledEntityType}`	`voteByEntitlementFactory`	المستخدمون أو المجموعات وأذوناتهم على نظام أو وحدة أو كيان أو معرض محدّد.
`GET /v1/permissions/{entitlementType}/{entitledEntityType}/count`	`voteByEntitlementFactory`	عدد ما سبق.
`GET /v1/permissions/list/{entitledEntityType}`	مصوّت الوحدة أو النظام	جميع المستخدمين أو المجموعات في النظام، مع إمكانية حصرهم في وحدة عبر `moduleId`.
`GET /v1/permissions/list/{entitledEntityType}/count`	مصوّت الوحدة أو النظام	عدد ما سبق.
`GET /v1/permissions/current_user`	يُتخطّى التخويل	الأذونات الفعّالة للمستدعي نفسه.
`GET /v1/permissions/user/{userId}`	`systemLevelVoterFactory`	أذونات مستخدم على الموارد (`UserPermissionsDto[]`).
`GET /v1/permissions/user/{userId}/admin`	`systemLevelVoterFactory`	أذونات مستخدم بمستوى الإدارة.

نقطة النهاية	الحارس	ما تُعيده
`GET /v1/permissions/group/{groupId}`	`systemLevelVoterFactory`	أذونات مجموعة أو دور أو وحدة تنظيمية.
`GET /v1/permissions/group/{groupId}/admin`	`systemLevelVoterFactory`	أذونات مجموعة بمستوى الإدارة.
`GET /v1/permissions/anonymous`	`systemLevelVoterFactory`	الأذونات المطبّقة على الجمهور المجهول.
`GET /v1/permissions/system-object/{systemObjectId}`	`systemLevelVoterFactory`	كل مستخدم أو مجموعة له وصول إلى كائن معيّن (`ObjectPermissionsDto[]`). `Since 1.6` .

يشترط الحارس `systemLevelVoterFactory` مستوى عارض نظام للقراءة ومستوى مطوّر نظام للكتابة. و `Since 1.8` يحمل هذا الاسم؛ إذ كان اسمه سابقًا `systemAdminVoterFactory` ، وقد اقتصر التغيير على الاسم، لأن الفحص لم يشترط مستوى المسؤول قط. ولكلٍّ من نقاط النهاية `user` و `group` و `admin` صيغة `/count` .

أعلام الاستعلام في نقاط نهاية السرد:

- `include_virtual` : تُستبعد الكيانات الوهمية (التي يولّدها النظام) افتراضيًا (`virtual: {neq: true}`)؛ مرّر `true` لتضمينها.
- `only_none_permission` : يقصر النتيجة على الكيانات المُستحِقة التي **لا** تملك إذنًا (أو تملك قيمة `None` مخزَّنة)، وهو ما يستخدمه منتقي **إضافة**. `Since 1.8` .
- `entitlementId` : يحصر النتيجة في كائن محدّد عند السرد بحسب نوع الاستحقاق.
- `moduleId` : يحصر نتائج نقاط نهاية `list` في وحدة واحدة.

عرض تدقيق المستخدم

تُعيد `GET /v1/permissions/user/{userId}` مُدخلات لا تصف الكائن والمستوى فحسب، بل توضّح أيضًا **كيف** اكتُسب الوصول:

الحقل	المعنى
`object` (`ResourceObjectDto`)	الهدف: المعرّف، والاسم، و `type` ، و `dataModule` / `dataEntity` الخاصان به، و `abilities` المحسوبة، والأعلام `isAuthenticated` / `isAnonymous` / `isRestricted` .
`assignmentType` (`UserPermissionAssignmentTypeEnum`)	`Direct` (0)، و `group` (1)، و `authenticated` (2)، و `delegation` (3)، و `anonymous` (4).
`acquiredThroughGroupType` (`GroupTypeEnum`)	نوع المجموعة إن اكتُسب الوصول عبرها: UserGroup أو Role أو OrgUnit أو Virtual أو QueryGroup.
`acquiredThrough`	المجموعة أو الدور الذي اكتُسب الوصول من خلاله بعد الحسم.
`adminLevel`	مستوى الإدارة الفعّال، حيثما ينطبق.
`delegationId`	يُضبط حين يُكتسب الوصول عبر تفويض (`getAllDelegateePermissions` يحسم ما يملكه المفوَّض إليه نيابةً عن المفوِّض).

يبيّن `abilities` (`ResourceObjectAbilitiesDto`) القيم المنطقية التي يستخدمها العميل لإظهار الإجراءات أو إخفائها: `create` و `edit` و `archive` و `entriesList` و `entryDetails` ، وللكائنات القابلة للمشاركة `audience` و `observer` . وتظهر هذه في شاشات الإدارة ضمن الأعمدة **نوع الإذن** (بقيمه **إذن مُعطاة مباشرة**، و**إذن موروثة من مجموعة**، و**إذن موروثة لجميع المستخدمين المصادق عليهم**، و**إذن موروثة من التفويض**، و**إذن موروثة للمستخدم المجهول**)، و**مكتسبة عن طريق**، و**الإجراءات المتاحة**.

أما `ObjectPermissionsDto` (نقطة النهاية `system-object`) فيقدّم المنظور المعاكس: إذ يسرد لكائن واحد كل كيان مُستحِق مع نوعه `type` (`EntitledEntityTypeEnum` GROUP/USER)، و`assignmentType`، و`accessType` (`ObjectPermissionAccessTypeEnum`: `Default` (0)، و`Audience` (1)، و`Observer` (2)).

استجابة المستخدم الحالي

تبني `GET /v1/permissions/current_user` قائمة أذونات المستدعي نفسه. فإن كان محرّر نظام أو مسؤول نظام، أعادت منح النظام وحدها (إذ لا حاجة إلى تعداد جميع الموارد). وإلا أعادت منح النظام والوحدات، ومنح الكيانات **غير** المشمولة أصلًا بمنحة وحدة، وجميع منح المعارض والصفحات والتقارير. كما تُعيد `entityParentModuleIds` (`Since 1.8`) ليتمكن العميل من فتح وحدة لا يمكن الوصول إليها إلا عبر منحة كيان، وتكون هذه القائمة فارغة لعارضي النظام ومحرّريه ومسؤوليه.

المشاركة وظهور الكائنات

بمعزل عن منح المستخدمين والمجموعات، يحمل **الكائن القابل للمشاركة** (معرض البيانات، والصفحة، والتقرير) إعدادًا يحدّد أين يظهر ومن يستطيع فتحه بعد نشره. وتطبّق المنصة الشرطين عند كل قراءة: فلا يرى المستخدم إلا الكائنات الظاهرة له **والتي** يحق له فتحها. وتتوفر التقارير على الويب وحده، أما على الجوال فيؤدي رابط التقرير إلى عنصر نائب يفيد بأن الرابط غير مدعوم.

مستويات المشاركة

يحمل كل كائن **مستوى مشاركة** واحدًا بين المساحات (`ApplicationSharingLevelEnum`)، يُختار عند إنشاء وحدة أو تطبيق:

المستوى	المعنى
`PRIVATE`	ظاهر داخل مساحة الكائن وحدها.
`SHARED`	ظاهر داخل مجموعة المساحات المرتبطة بالكائن.
`PUBLIC`	ظاهر في جميع المساحات (ويُستخدم عادةً للعناصر المشتركة على مستوى النظام، كالصفحات الافتراضية للمساحة).

عند القراءة، يظهر السجلّ إن كان مستواه `PUBLIC`، أو كان `SHARED` / `PRIVATE` وكانت المساحة الحالية ضمن مساحات الكائن. وتُعيد `GET /v2/access-object/{id}` كائن `AccessObjectDto` يتضمن `authenticationMode` بعد حسم قيمته، و`domainId`، و`spacesId`، و`isPublicSpace` (تكون قيمته صحيحة حين تكون `sharing.level` هي `PUBLIC`)، و`executionPlan`، و`published`، والكائن الأصلي. وترد تفاصيل الحسم في المحرك: الأذونات والمصوّتون – حسم مستوى المشاركة.

الظهور داخل المساحة

تدعم بعض الكائنات، كالمنشورات وعناصر الشريط الزمني، أيضًا **مستوى ظهور** داخل المساحة:

مستوى الظهور	المعنى
خاص	لا يراه إلا مالكه والمستخدمون المُسنَد إليهم صراحةً.
المنظمة	يراه جميع مستخدمي المنظمة داخل المساحة.
عام	يظهر أيضًا للزوّار غير المصادَق عليهم (وفق ما يسمح به الوصول المجهول).

خيارات الجمهور

تضبط لوحة المشاركة جمهور الكائن القابل للمشاركة:

- `anonymous` (`AnonymousOption`): يحدّد الحقل `enabled` ما إذا كان الزوّار غير المصادَق عليهم يستطيعون فتح الكائن.

- `authenticated` (`AuthenticatedOption`):
  - `enabled` : يستطيع المستخدمون المسجّلون فتحه؛
  - `isRestricted` : حين تكون قيمته صحيحة، لا يفتحه إلا المستخدمون والمجموعات المُدرَجون في `entitledEntries` (قائمة سماح خاصة بالكائن)؛
  - `entitledEntries` : قائمة السماح، ولا تُطلب إلا حين تكون قيمة `isRestricted` صحيحة.

يمثّل نموذج `Permission` حاوية لـ `users[]` و `groups[]` ، إضافةً إلى `accessType` القديم (`AccessTypeEnum` : `ReadOnly` / `Modify` ، وقد أُهمل لصالح `AccessRightLevelEnum`). ويستند إليه `UpdateResourcePermissionsDto` (الأذونات وخيارا الوصول المجهول والوصول المصادَق عليه) لمشاركة المعارض والصفحات والتقارير، بينما يحمل `UpdateServicePermissionsDto` الحقلين `audience` و `observers` للخدمة (`PATCH` `/v1/entities/{id}/meta/permissions`، لكيانات النوع `Request` وحدها). وتؤدي إضافة مستخدم أو مجموعة إلى كائن من شاشة **أذونات وصول المجموعة** أو **أذونات وصول المستخدم** إلى كتابة منحة عرض على الكائن وتحويله إلى الوضع المقيّد؛ أما في الخدمة فتُضيف الطرف إلى قائمة السماح في جمهور الخدمة أو إلى متابعيها بدلًا من ذلك.

قواعد التحقق من الجمهور

يفرض `PermissionsService.validatePermissions` القواعد الثابتة التالية عند حفظ الجمهور:

القاعدة	الاستثناء
`anonymous.enabled` **و** `authenticated.enabled` **و** `authenticated.isRestricted` كلها صحيحة	`AnonymousAndRestrictedEnabledException` : لا يمكن أن يكون الكائن مفتوحًا للزوّار المجهولين ومقيّدًا للمستخدمين المسجّلين في آن واحد.
لا الوصول المجهول مفعّل ولا الوصول المصادَق عليه	`AnonymousOrRestrictedNotEanabledException` : يجب أن يتمكن طرف ما من الوصول إلى الكائن.
`entitledEntries` مقدَّمة لكن قيمة `authenticated.enabled` خاطئة	`AuthenticatedMustEnabledExcecption` .
`entitledEntries` مقدَّمة لكن قيمة `isRestricted` خاطئة	`RestrictedMustEnabledExcecption` .
قيمة `isRestricted` صحيحة لكن `entitledEntries` فارغة	`RestrictedPermissionEntitledEntriesEmptyExcecption` .

دوال مساعدة ذات صلة: `hasEmptyPermissions` (تُعدّ `Permission[]` فارغة إن غابت، أو لم تحتوِ على مُدخلات، أو خلا كل مُدخل فيها من المستخدمين **و**المجموعات)؛ و `anonymusResetEntitledEntries` (يمسح تفعيل الوصول المجهول أي قائمة سماح للمصادَق عليهم)؛ و `getDuplicatedEntitledEntries` (يكشف المراجع المكرّرة)؛ و `checkUserAndGroupExistence` / `findEntitledEntry` (يجب أن يكون كل مستخدم أو مجموعة مُشار إليه موجودًا وبالحالة `Active` ، وإلا أُطلق الاستثناء `NotExistEntitledException`).

الوصول الفعّال إلى الكائن

تُعيد `GET /v1/access-object/can-access/{id}` (يُتخطّى فيها التخويل، فتعمل مع المستدعين المجهولين) القيمة `{result, view}` ، وتُجري فحص الوصول وقت التشغيل لفتح معرض أو صفحة أو تقرير:

1. إن كان الكائن **غير منشور** ← يُرفض الوصول.
2. إن كانت قيمة `anonymous.enabled` صحيحة وكان المستدعي مجهولًا ← يُسمح بالوصول.
3. إن كانت قيمة `authenticated.enabled` صحيحة:

- غير مقيّد والمستدعي مصادَق عليه ← يُسمح بالوصول؛
- مقيّد ← لا يُسمح بالوصول إلا إذا كان المستدعي يحمل منحة مطابقة على الكائن.

4. وفيما عدا ذلك ← يُرفض الوصول.

الوصول المجهول وروابط المشاركة

ثمة آليتان لإتاحة البيانات لمن لم يسجّل الدخول:

1. **الأذونات المجهولة**: امنح الجمهور *المجهول* المستوى `ViewAccess` على كيان أو معرض (وتسردها `GET /v1/permissions/anonymous`).
2. **روابط المشاركة**: يؤدي رابط المشاركة إلى معرض لكائن محدّد، ويمكن أن يكون **مجهولًا** (بلا تسجيل دخول) أو **مصادَقًا عليه** (يتطلب تسجيل الدخول).

طلبات الخدمات

تحدّد الخدمة نفسها من يرى طلباتها، لا المنح:

من	القاعدة (`RequestService`)
من يستطيع التقديم (يرى الخدمة في الدليل)	`canCreateRequest` : الخدمة مفعّلة ولها جمهور؛ والخدمات المخصّصة للمجهولين فقط ترفض المستخدمين المسجّلين؛ والجمهور المصادَق عليه المقيّد يشترط وجود المستخدم أو إحدى مجموعاته في `entitledEntries` .
مقدّم الطلب	`canAccessMyRequest` : منشئ الطلب (والمنشئ المجهول ما دامت خطوات الطلب كلها بلا مالك).
مالكو الخطوات	`canAccessInbox` : مالك الخطوة الحالية، وعضو المجموعة أو القائمة المالكة ما دامت الخطوة غير مطالَب بها أو مضبوطة لجميع المستخدمين، وكل من نفّذ خطوة سابقة أو كان مالكًا لها؛ ولا يشمل ذلك المنشئ أبدًا.
المتابعون	`canAccessObservers` : المستخدم، أو إحدى مجموعاته، ضمن `meta.observers` للخدمة. ويُنسخ المتابعون إلى صف كل طلب، ويؤدي تغييرهم إلى تحديث الطلبات القائمة.

وعند العمل بالتفويض، يُستعاض عن كل فحص بنطاق التفويض الخاص بإجراء الخدمة المقابل (`submit` و `request` و `inbox` و `observe`). وخارج المعرض، لا تُقرأ مُدخلات كيان الطلبات إلا بوصول من مستوى التعديل (انظر مستويات الوصول).

منح الحقول الحسّاسة (نموذج البيانات)

يمكن تصنيف المنحة على أنها منحة قراءة **لحقل حسّاس** عبر `permission_type = SENSITIVE_FIELD` و `scope` يحدّد مسار الحقل (`module/entity/field` ، أو نطاقًا بالرمز البديل `module/entity/*` / `module/*/*`). ويميّز `PermissionTypeEnum` هذه السجلات من منح `NORMAL` ، وتُعامَل السجلات القديمة التي لا تحمل `permission_type` على أنها `NORMAL` . وتُدار هذه المنح لكل نطاق عبر `syncSensitiveFieldAccess` و `getSensitiveFieldGrants` و `removeSensitiveFieldAccess(scopePrefix)` ، التي توفّق سجلات `SENSITIVE_FIELD` وتحذف المتقادم منها. و`Since 1.10` تُخوَّل الحقول **المشفّرة** بمنحة `SENSITIVE_FIELD` نفسها على النطاق نفسه؛ فلا يوجد نوع منحة ثانٍ.

أما الحسم وبوّابة الكشف التي تشترط تحققًا إضافيًا، فموثّقان في المحرك: الأذونات والمصوّتون – حسم منح الحقول الحسّاسة، وإجراء المسؤول في الأذونات الحساسة [الإدارة].

انتهاء صلاحية المنحة والأرشفة التلقائية `Since 1.8`

يخزّن `AccessRightValue.expiry` نافذة صلاحية `[startIso, endIso]` ، وهي تنطبق حاليًا على منح الحقول الحسّاسة.

- لا تنتهي الصلاحية أبدًا إذا كانت القيمة `null` أو كانت النافذة بلا نهاية؛ ويترك العنصر ذو القيمة `null` أو الغائب الحدّ المقابل مفتوحًا.

- لا تُحتسب عند حسم الوصول الفعّال إلا المنح الصالحة حاليًا (`buildExpiryWindowExpr` ، و `buildActiveExpiryWindowWhere`). أما القيمة المفردة القديمة (scalar) فتُعامَل على أساس الرفض.
- يؤرشف فحص تلقائي كل منحة انقضى تاريخ انتهائها. وحين تُؤرشف منحة، تلقائيًا أو على يد مسؤول، يُسحب الوصول فورًا؛ إذ يعيد الكائن المتأثر حساب ما إذا كان مقيّدًا، وتتوقف المنحة المؤرشفة عن الظهور ضمن المنح النشطة للطرف.

نموذج بيانات الأدوار

يُمثَّل الدور الذي تُنشئه بمُدخل دور (`RoleDataEntry` / `RoleDataEntryValue`) ومجموعة من النوع `Role` (`GroupTypeEnum.Role`) في آن واحد، ويمكن أن يكون كيانًا مُستحِقًا بذاته (`EntitledEntityEnum.Role = 'ROLE'`). ويرث المستخدم المُسنَد إليه الدور منح ذلك الدور، وحين يُبلغ النظام عن المجموعة التي اكتُسب الإذن عبرها، تُوسَم مجموعة الدور بالقيمة `Role` في `acquiredThroughGroupType` .

الحقل	النوع	القيم
`name`	نص مترجَم	اسم العرض.
`description`	نص مترجَم	اختياري.
`linked_with_org_unit`	تصنيف (`RoleLinkWIthOrgUnitType`s)	`NOT_LINKED_WITH_ORG_UNITS` ، و `LINKED_WITH_ONE_ORG_UNIT` ، و `LINKED_WITH_MANY_ORG_UNITS` .
`type`	تصنيف (`RoleTypes`)	`JOB_POSITION` أو `MERE_ROLE` . ولا يمكن أن يكون الدور منصبًا وظيفيًا إن لم يكن مرتبطًا بأي وحدة تنظيمية (تحقق على مستوى الحقل).
`assignment_type`	تصنيف (`RoleAssignmentTypes`)	`UNIQUE` (حامل واحد) أو `MULTIPLE` (عدة حاملين).

يُربط المستخدمون بالأدوار عبر مُدخلات **الإسناد** (الدور، ومستوى الإسناد رئيسي/إضافي، ونوع الإسناد أصلي/تفويض، والمستخدم، والوحدة التنظيمية، وما إذا كان المستخدم مديرًا). ويُتحقق من كل إسناد قبل حفظه (قواعد المستوى، والتفرّد، والدور، والوحدة التنظيمية، وحساب المنظمة)، ويؤدي تغيير الإسنادات إلى إعادة حساب مدير الوحدة التنظيمية عند الحاجة.

أما مستويات الوصول على النظام (التعداد `Roles` : `SystemAdmin` و `SystemEditor` و `SystemDeveloper` و `SystemViewer` و `Restricted`) فهي **ليست** سجلات، بل تُحسب مع كل طلب من مستوى المستخدم على نطاق النظام. وترد الدوال الشرطية التي تحدّدها، وترتيب تطبيق المُخوِّل الأعلى لها، في المحرك: الأذونات والمصوّتون – تدفّق القرار.

أذونات المدير (القيود الحالية)

تدعم شاشة **أذونات المدير** منح مستخدمين بعينهم إذن منح الأذونات لمستخدمين ومجموعات آخرين ضمن حدود محدّدة. ولها حاليًا السلوكان التاليان:

- لا تسرد الشاشة إلا الأطراف التي لها سجلّ إذن على ذلك النطاق؛ أما المستخدم أو المجموعة اللذان لا يملكان أي سجلّ إذن عليه، فيمكن الوصول إليهما عبر منتقي **إضافة** (`only_none_permission`).

وتجري معالجة كليهما ضمن إعادة تصميم هذا النموذج.

قفل الكائنات

لمنع مستخدمَين اثنين من الكتابة فوق تعديلات بعضهما، تدعم المنصة **أقفال الكائنات** التعاونية (التحكّم التشاؤمي في التزامن)؛ إذ يمكن قفل المُدخل الجاري تحريره، فيرى المستخدمون الآخرون حالة القفل، ويُحتفظ **بسجلّ** للأقفال. وتنسّق

الأقفال تجربة التحرير دون فرض قفل صارم على قاعدة البيانات، لذلك يظل العميل الذي يتجاهل بروتوكول القفل قادرًا على الكتابة.

نقطة النهاية	العملية
`POST /v1/locks`	الحصول على قفل كائن.
`GET /v1/locks`	جلب القفل الحالي (أو الأقفال).
`PUT /v1/locks`	تمديد قفل قائم (تجديده).
`GET /v1/locks/{objectId}/history`	سجلّ الأقفال لكائن.

تتحكم في إمكانية الحصول على القفل الدالة `getLockedStatusByUser` (ما إذا كان الكائن مقفولًا، ومن قفله) وفحص `can-lock`. وفي تطبيق الإدارة، يُدار القفل عبر الحالة `locks` في العميل (`LockEntity` و `UnlockObject` و `ExtendLockObject` وإجراء للتجاهل)، ويعرض تبويب **سجلّ القفل** في الكيان بيانات نقطة نهاية السجلّ. أما كيانات التجميع الوهمية، فتطبّق حالة القفل الخاصة بها عبر `AggregatorDataEntryService.applyVirtualAggregatorLockState` . ويرد المرجع المولَّد في Locks – data-service.

موضوعات المحرك ذات الصلة

تُوثَّق هذه الموضوعات لمهندسي المنصة في المحرك: الأذونات والمصوّتون:

- كيف تُدمج المنح الآتية من المستخدمين والمجموعات والأدوار والوحدات التنظيمية ومجموعات الاستعلام في منحة واحدة لكل مورد: دمج المنح دون إخفاء أي منحة صالحة.
- خبء الأذونات عبر الخدمات وعدّادات الأجيال التي تُبطله: الخبء وإبطال الأجيال (epoch).
- تفويض «السماح للكل» وصلاحيات الإدارة: التفويض: تفويض «السماح للكل» لا يحمل صلاحيات الإدارة.
- فحص OpenFGA الظلّي ومفاتيحه وضماناته: التخويل عبر OpenFGA في وضع الظل.
- إصلاحات البيانات الصادرة مع 1.10: إصلاحات البيانات.

مرجع نقاط النهاية

مولَّد من لقطات OpenAPI المودَعة – صفحة لكل متحكِّم، مجموعةً بحسب الخدمة. أعد التوليد بالأمر `npm run api-reference`؛ ولا تحرّر صفحة مولَّدة يدويًّا أبدًا.

مولَّد (من لقطات OpenAPI)

الخدمة	المتحكِّم	عملية
`comments-service`	Comment	7
`comments-service`	Reaction	4
`data-service`	Access	18
`data-service`	BackupAdmin	10
`data-service`	CounterWidget	4
`data-service`	Counters	1
`data-service`	Dashboard	5
`data-service`	DataCollections	21
`data-service`	DataEntities	20
`data-service`	DataEntityAggregate	6
`data-service`	DataEntityBlocks	6
`data-service`	DataEntityCategories	7
`data-service`	DataEntityFields	7
`data-service`	DataEntries	53
`data-service`	DataEntryTemplateExport	2
`data-service`	DataModule	7
`data-service`	DataQuery	10
`data-service`	DataRelation	20
`data-service`	DataViews	38
`data-service`	Delegations	2
`data-service`	EntityEntriesExport	1
`data-service`	Expression	2
`data-service`	GenericView	10
`data-service`	Groups	1
`data-service`	Icons	1
`data-service`	ImportExport	4
`data-service`	Job	1
`data-service`	Locks	5
`data-service`	OrgConfig	1
`data-service`	QuickFilterGroup	6

الخدمة	المتحكّم	عملية
`data-service`	ReportEntry	12
`data-service`	Request	1
`data-service`	SensitiveFieldGrants	8
`data-service`	SideBar	1
`data-service`	Spaces	10
`data-service`	StatisticQuery	11
`data-service`	Stores	4
`data-service`	SystemButton	10
`data-service`	SystemLink	2
`data-service`	SystemObject	1
`data-service`	SystemSettings	1
`data-service`	TagGroup	6
`data-service`	Tags	2
`data-service`	Templates	1
`data-service`	UiSettings	13
`data-service`	User	11
`data-service`	UserPreferences	1
`data-service`	Version	1
`data-service`	ViewApiDocs	1
`data-service`	ViewEntriesExport	1
`data-service`	Workflow	1
`functions-service`	Functions	2
`notification-api`	Notification	3
`notification-api`	NotificationTemplate	1
`notification-api`	PushNotificationsTokens	4
`reaction-service`	Announcement	7
`reaction-service`	DataReaction	12
`reaction-service`	ReactionLog	2
`reaction-service`	SystemReaction	5
`reaction-service`	WhatsAppTemplates	1
`timeline-service`	Timeline	7
`workflow-service`	Workflow	10

مصان يدويًّا (بلا مواصفة آلية)

- `users-service`
- `storage-service`
- `scheduler-service`

هذه الخدمات تطبيقات Express عادية ولا تُصدر وثيقة OpenAPI. يوثَّق سطحها يدويًّا في الأدلة السردية – الاستيثاق في اختيار بيانات الاعتماد والملفات في الملفات.

مستخرَجة من الخلفية

- `osos-shortener-fn` → `shortener-service`
- `osos-altcha-fn` → `altcha-service`

هذه لم تعد خدمات خلفية (Backend) بعد الآن (`Since 1.8`). أصبحت كل واحدة منها دالة Knative مشتركة لكل عنقود، تُستهلَك كصورة منشورة، ولها مصدرها الخاص خارج هذا المستودع أحادي الجذر – لم يعد هناك كود خلفية يُوصَف أو يُصان يدويًّا. المسار العام الذي تخدمه كل واحدة يبقى ثابتًا عند البوابة (مثل `/api/human/challenge`)؛ ما تغيّر هو التنفيذ الأساسي فقط.

Comment — comments-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `comments-service.json` · 7 · comments 0.0.1 عملية

`GET /v1/{entity}/{relatedToId}`

`getCommentsRelatedTo`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`relatedToId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataEntry model instance

`POST /v1/{entity}/{relatedToId}`

`createComment`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`relatedToId`	path	نعم	`string`

جسم الطلب: `NewCommentDto`

الاستجابات

الحالة	الوصف
`200`	Comment DataEntryValue model instance

`PATCH /v1/{entity}/{relatedToId}/{commentId}`

`updateComment`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`relatedToId`	path	نعم	`string`
`commentId`	path	نعم	`string`

جسم الطلب: `UpdateCommentDto`

الاستجابات

الحالة	الوصف
`200`	Comment DataEntryValue model instance

`GET /v1/{entity}/{relatedToId}/attachments`

`getCommentsAttachmentsRelatedTo`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`relatedToId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataEntry model instance

`GET /v1/{entity}/{relatedToId}/count`

`getCommentsRelatedToCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`relatedToId`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Comment Data Entry model count

`GET /v1/comments/{entityId}/{entryId}/mentions-list`

`getMentionsList`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entryId`	path	نعم	`string`
`entityId`	path	نعم	`string`
`keyword`	query	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Comment DataEntry model instance

`GET /v1/comments/audience-list`

`getAudienceList`

المعاملات

الاسم	الموضع	مطلوب	النوع
`keyword`	query	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Comment DataEntry model instance

Reaction — comments-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `comments-service.json` · 4 · 0.0.1 comments عملية

`DELETE /v1/reaction/{entity}/{relatedToId}`

`deleteReaction`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`relatedToId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Delete user reaction on specific entry

`GET /v1/reaction/{entity}/{relatedToId}`

`getReactions`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`relatedToId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	ReactionDataEntryValueDto model instance

`PUT /v1/reaction/{entity}/{relatedToId}`

`createReaction`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`relatedToId`	path	نعم	`string`

جسم الطلب: `CreateReactionDto`

الاستجابات

الحالة	الوصف
`200`	Reaction added

`GET /v1/reaction/{entity}/{relatedToId}/count`

`getReactionsCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`relatedToId`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	CountSchema model instance

Access — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 18 · data-service 1.11.0 عملية

`GET /v1/access-object/{id}`

`findAccessObjectById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of AccessController.findAccessObjectById

`GET /v1/access-object/can-access/{id}`

`canAccessObject`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of AccessController.canAccessObject

`GET /v1/permissions/{entitlementType}/{entitledEntityType}`

List users/groups with system, module, entity and view permissions — `getPermissions`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entitlementType`	path	نعم	`string`
`entitledEntityType`	path	نعم	`string`
`entitlementId`	query	لا	`string`

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`include_virtual`	query	لا	`boolean`
`only_none_permission`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of UserDto model instances

`GET /v1/permissions/{entitlementType}/{entitledEntityType}/count`

Get permission count – `getPermissionsCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entitlementType`	path	نعم	`string`
`entitledEntityType`	path	نعم	`string`
`entitlementId`	query	لا	`string`
`where`	query	لا	–
`include_virtual`	query	لا	`boolean`
`only_none_permission`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataEntry model count.

`PUT /v1/permissions/{entitlementType}/{entitlementId}/{entitledEntityType}/{entitledEntityId}`

Update/revoke a access permissions that belongs to the user/group – `updateAccessPermission`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entitlementType`	path	نعم	`string`
`entitlementId`	path	نعم	`string`
`entitledEntityType`	path	نعم	`string`
`entitledEntityId`	path	نعم	`string`

جسم الطلب: `UpsertPermission`

الاستجابات

الحالة	الوصف
`200`	DataEntry PUT success

`PUT /v1/permissions/{systemObjectId}/{entitledEntityType}/{entitledEntityId}`

Update/revoke a access permissions that belongs to the user/group — `updateAccessPermissionBySystemObjectId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`systemObjectId`	path	نعم	`string`
`entitledEntityType`	path	نعم	`string`
`entitledEntityId`	path	نعم	`string`

جسم الطلب: `UpsertPermission`

الاستجابات

الحالة	الوصف
`200`	DataEntry PUT success

`GET /v1/permissions/anonymous`

Get all permissions for the user — `getAnonymousUserPermissionsDto`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of UserPermissionsDto model instances

`GET /v1/permissions/current_user`

Get all permissions for the current user — `getUserPermissions`

الاستجابات

الحالة	الوصف
`200`	Array of UserDto model instances

`GET /v1/permissions/group/{groupId}`

Get all permissions for the user – `getPermissionsDtoByGroupId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`groupId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of UserPermissionsDto model instances

`GET /v1/permissions/group/{groupId}/admin`

Get all admin permissions for the user – `getAdminPermissionsDtoByGroupId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`groupId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of UserPermissionsDto model instances

`GET /v1/permissions/list/{entitledEntityType}`

List all users/groups in the system with permissions – `getSystemPermissions`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entitledEntityType`	path	نعم	`string`
`moduleId`	query	لا	`string`
`filter`	query	لا	–
`include_virtual`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of UserDto model instances

`GET /v1/permissions/list/{entitledEntityType}/count`

Get permissions count — `systemCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entitledEntityType`	path	نعم	`string`
`moduleId`	query	لا	`string`
`where`	query	لا	–
`include_virtual`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataEntry model count.

`GET /v1/permissions/system-object/{systemObjectId}`

Get all permissions for the system object — `getObjectPermissionsDtoBySystemObjectId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`systemObjectId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of ObjectPermissionsDto model instances

`GET /v1/permissions/user/{userId}`

Get all permissions for the user — `getPermissionsDtoByUserId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`userId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of UserPermissionsDto model instances

`GET /v1/permissions/user/{userId}/admin`

Get all admin permissions for the user – `getAdminPermissionsDtoByUserId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`userId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of UserPermissionsDto model instances

`GET /v1/permissions/user/{userId}/admin/count`

Get count of admin permissions for the user – `getAdminPermissionsDtoCountByUserId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`userId`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	.DataEntry model count

`GET /v1/permissions/user/{userId}/count`

Get count of permissions for the user – `getPermissionsDtoCountByUserId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`userId`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	.DataEntry model count

`GET /v2/access-object/{id}`

`findAccessObjectByIdV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of AccessController.findAccessObjectByIdV2

BackupAdmin — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 10 · data-service 1.11.0 عملية

Since 1.10 GET /v1/admin/backups

`list`

الاستجابات

الحالة	الوصف
`200`	The tenant's backups and today's quota use

Since 1.10 POST /v1/admin/backups

`start`

الاستجابات

الحالة	الوصف
`202`	The backup, running

Since 1.10 GET /v1/admin/backups/{backupId}

`read`

المعاملات

الاسم	الموضع	مطلوب	النوع
`backupId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	One backup

Since 1.10 GET /v1/admin/exports

`listExports`

الاستجابات

الحالة	الوصف
`200`	The tenant's exports

`Since 1.10` `POST /v1/admin/exports`

`startExport`

الاستجابات

الحالة	الوصف
`202`	The export, pending

`Since 1.10` `GET /v1/admin/exports/{exportId}`

`readExport`

المعاملات

الاسم	الموضع	مطلوب	النوع
`exportId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	One export

`Since 1.10` `POST /v1/admin/exports/{exportId}/link`

`linkExport`

المعاملات

الاسم	الموضع	مطلوب	النوع
`exportId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	A single-use download link

`Since 1.10` `POST /v1/admin/restores`

`startRestore`

جسم الطلب: `object`

الاستجابات

الحالة	الوصف
`202`	The restore, running

`Since 1.10` `GET /v1/admin/restores/{restoreId}`

`readRestore`

المعاملات

الاسم	الموضع	مطلوب	النوع
`restoreId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	The restore's status

`Since 1.10` `POST /v1/admin/restores/preview`

`previewRestore`

جسم الطلب: `object`

الاستجابات

الحالة	الوصف
`200`	What the restore would do, and its confirmation token

CounterWidget — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 4 · data-service 1.11.0 عملية

`GET /v1/counter-widgets`

Get counter widgets DTO — `getCounterWidgets`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of CounterWidgetDataEntryDto model instances

`GET /v1/counter-widgets/{id}/count`

`getCounterWidgetsCountEndUser`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	CounterWidgetDataEntry model count.

`GET /v1/counter-widgets/{id}/entries`

`getCounterWidgetsEntriesEndUser`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of CounterWidgetDataEntryEndUserDto model instances

`GET /v1/counter-widgets/end-user`

`getCounterWidgetsEndUser`

الاستجابات

الحالة	الوصف
`200`	Array of CounterWidgetDataEntry model instances

Counters — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v1/auto-increment-counters`

List all auto increment counters — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataCollections model instances

Dashboard — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 5 · data-service 1.11.0 عملية

`GET /v1/dashboards`

Get all dashboards — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of Dashboard model instances

`GET /v1/dashboards/{id}`

Get dashboard by id — `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Dashboard model instance

`PUT /v1/dashboards/{id}`

Update dashboard by id — `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `DashboardPartialExcluding_creationInfo-updateInfo-stateHistory-id_`

الاستجابات

الحالة	الوصف
`200`	Dashboard PUT success

DataCollections — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 21 · data-service 1.11.0 عملية

`GET /v1/collections`

Get all collections — `findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataCollections model instances

`GET /v1/collections/{id}`

Get all collections — `findByIdWithoutEntity`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataCollections model instances

`GET /v1/collections/count`

Get collections count — `countAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataCollections model count.

`GET /v1/collections/my`

List all user collections in a specific view or in all views – `findMyCollections`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataCollections model instances

`GET /v1/collections/shared`

List all user shared collections in a specific view or in all views – `findSharedCollections`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataCollections model instances

`GET /v1/entities/{entity}/collections`

Get all entity collections – `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataCollections model instances

`POST /v1/entities/{entity}/collections`

Create new data collection — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewDataCollections`

الاستجابات

الحالة	الوصف
`200`	DataCollections model instance

`DELETE /v1/entities/{entity}/collections/{id}`

Delete specific entity collection — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataCollections DELETE success

`GET /v1/entities/{entity}/collections/{id}`

Get specific entity collection — `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataCollectionsPermissionsDto model instance

`PATCH /v1/entities/{entity}/collections/{id}`

Update specific entity collection — `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `DataCollectionsPartialExcluding_id-updateInfo-creationInfo-isImplicit-relatedTo-module_`

الاستجابات

الحالة	الوصف
`200`	DataCollections PATCH success

`PUT /v1/entities/{entity}/collections/{id}`

Update specific entity collection — `replaceById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `ReplaceDataCollections`

الاستجابات

الحالة	الوصف
`200`	DataCollections PUT success

`POST /v1/entities/{entity}/collections/{id}/duplicate`

Duplicate data collection — `duplicate`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `DuplicatedDataCollections`

الاستجابات

الحالة	الوصف
`200`	DataCollections model instance

`GET /v1/entities/{entity}/collections/{id}/entries`

Get specific entity collection — `findEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filter`	query	لا	—
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	Array of DataEntry model instances

`PATCH /v1/entities/{entity}/collections/{id}/entries`

`updateCollectionItems`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `object` array of

الاستجابات

الحالة	الوصف
`200`	Update collection items

`GET /v1/entities/{entity}/collections/{id}/entries/count`

Get specific entity collection count — `findEntriesCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	DataEntry model count.

`PATCH /v1/entities/{entity}/collections/{id}/share`

Share collection – `share`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `DataCollectionsShareModel`

الاستجابات

الحالة	الوصف
`200`	DataCollections PATCH success

`GET /v1/entities/{entity}/collections/count`

Count specific entity collections – `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataCollections model count.

`POST /v2/entities/{entity}/collections`

Create new data collection – `createV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewDataCollectionsV2`

الاستجابات

الحالة	الوصف
`200`	DataCollections model instance

`GET /v2/entities/{entity}/collections/{id}/entries`

Get specific entity collection with applied tags – `findEntriesV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filter`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of
`appliedTagsFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	DataEntriesResponseV3 model instance

`PATCH /v2/entities/{entity}/collections/{id}/entries`

`updateCollectionItemsV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `PatchDataCollectionItemsDto`

الاستجابات

الحالة	الوصف
`200`	Set or unset collection items with select all ability

`GET /v2/entities/{entity}/collections/{id}/entries/count`

Get specific entity collection count — `findEntriesCountV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`where`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	DataEntry model count.

DataEntities — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 20 · data-service 1.11.0 عملية

`GET /v1/entities`

`find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Get a list of data entities

`POST /v1/entities`

Create a new data entity. `isImplicit` and `implicitOptions` are ignored for user callers: the entity is – `create` stored with `isImplicit: false`.

جسم الطلب: `NewDataEntity`

الاستجابات

الحالة	الوصف
`200`	DataEntity model instance

`GET /v1/entities/{entity}/breaking-changes/{version}`

Get breaking changes for an entity – `getEntityVersionChanges`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`version`	path	نعم	`number`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntityVersionChanges model instances

`DELETE /v1/entities/{id}`

Delete a data entity — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntity DELETE success

`GET /v1/entities/{id}`

Get a single data entity — `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–
`includeMainTitleFields`	query	لا	`boolean`
`includeMetaDataFields`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataEntityDto model instance

`PATCH /v1/entities/{id}`

Update data entity by id. `isImplicit` and `implicitOptions` are ignored for user callers: the stored — `updateById` values are kept.

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `UpdateDataEntity`

الاستجابات

الحالة	الوصف
`200`	DataEntity model instance

`PATCH /v1/entities/{id}/breaking-changes`

Preview breaking changes for a data entity update – `previewBreakingChanges`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `UpdateDataEntity`

الاستجابات

الحالة	الوصف
`200`	BreakingChangesResultDto model instance

`Since 1.10` `GET /v1/entities/{id}/index-paths`

List the paths a custom index on the entity may use – `getIndexPaths`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Pickable index paths of the entity

`Since 1.10` `POST /v1/entities/{id}/indexes/{indexId}/retry`

Re-run the index sync for a custom index of an entity – `retryIndex`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`indexId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Index sync requested; the body is the standard envelope {success: true, result: null}

`PATCH /v1/entities/{id}/meta/permissions`

Update service permissions for a data entity – `updateServicePermissionsById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `UpdateServicePermissionsDto`

الاستجابات

الحالة	الوصف
`200`	DataEntity model instance

`PATCH /v1/entities/{id}/module`

Move a data entity to a different module – `moveToModule`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`force`	query	لا	`boolean`

جسم الطلب: `object`

الاستجابات

الحالة	الوصف
`200`	DataEntity model instance

`Since 1.10` `DELETE /v1/entities/{id}/permanent`

Permanently delete a data entity, its entries and everything it owns – `deleteEntityPermanently`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`204`	The entity and everything it owned were deleted

`GET /v1/entities/all`

List all entities in the system — `findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntity model instances

`GET /v1/entities/all/count`

Count all entities in the system — `countAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	.DataEntity model count

`GET /v1/entities/count`

Get count of data entities in the system — `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	.DataEntity model count

`Since 1.10` `POST /v1/entities/move`

Move several data entities to one module, one at a time; each entity is reported as moved or failed — `moveEntitiesToModule`

جسم الطلب: `MoveEntitiesRequest`

الاستجابات

الحالة	الوصف
`200`	MoveEntitiesResult model instance

`POST /v1/entities/virtual`

Create a new virtual data entity — `createVirtual`

جسم الطلب: `NewVirtualDataEntity`

الاستجابات

الحالة	الوصف
`200`	DataEntity model instance

`PUT /v1/entities/virtual/{id}`

Update data entity by id — `updateVirtualEntityById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `UpdateVirtualDataEntity`

الاستجابات

الحالة	الوصف
`200`	DataEntity model instance

`GET /v2/entities`

`findV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`offset`	query	لا	`number`

الاستجابات

الحالة	الوصف
`200`	Get a bounded page of data entities

`GET /v2/entities/{entity}/breaking-changes`

Get breaking changes for multiple entity versions (v2) – `getEntityVersionChangesV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`versions`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	Array of DataEntityVersionChanges model instances

DataEntityAggregate — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 6 · data-service 1.11.0 عملية

Since 1.10 `GET /v1/entities/{entity}/aggregates`

List aggregates that belong to a specific entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataEntityAggregate model instances

Since 1.10 `POST /v1/entities/{entity}/aggregates`

Create a new aggregate for a specific entity — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewDataEntityAggregate`

الاستجابات

الحالة	الوصف
`200`	DataEntityAggregate model instance

Since 1.10 `DELETE /v1/entities/{entity}/aggregates/{id}`

Delete an aggregate by id — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntityAggregate DELETE success

`Since 1.10` `PATCH /v1/entities/{entity}/aggregates/{id}`

Update an aggregate by id — `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `DataEntityAggregatePartialExcluding_id_`

الاستجابات

الحالة	الوصف
`200`	DataEntityAggregate PATCH success

`Since 1.10` `GET /v1/entity-aggregates`

List all entity aggregates in the system — `findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataEntityAggregate model instances

`Since 1.10` `GET /v1/entity-aggregates/{id}`

Get an entity aggregate by id, without an entity — `findByIdWithoutEntity`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntityAggregate model instance

DataEntityBlocks — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 6 · data-service 1.11.0 عملية

`GET /v1/entities/{entity}/blocks`

Get all blocks for specific entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of InternalEntity model instances

`POST /v1/entities/{entity}/blocks`

Create block for specific entity — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewInternalEntity`

الاستجابات

الحالة	الوصف
`200`	InternalEntity model instance

`DELETE /v1/entities/{entity}/blocks/{id}`

Delete block for specific entity — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	InternalEntity DELETE success

`GET /v1/entities/{entity}/blocks/{id}`

Get block for specific entity – `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	InternalEntity model instance

`PATCH /v1/entities/{entity}/blocks/{id}`

Update specific block in specific entity – `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `InternalEntityPartialExcluding_id_`

الاستجابات

الحالة	الوصف
`200`	InternalEntity PATCH success

`GET /v1/entities/{entity}/blocks/count`

Count blocks for specific entity – `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	InternalEntity model count.

DataEntityCategories — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 7 · data-service 1.11.0 عملية

`GET /v1/entities/{entity}/categories`

List internal collections that belongs to specific entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	—

الاستجابات

الحالة	الوصف
`200`	Array of InternalCollection model instances

`POST /v1/entities/{entity}/categories`

Create internal collection inside specific entity — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewInternalCollection`

الاستجابات

الحالة	الوصف
`200`	InternalCollection model instance

`DELETE /v1/entities/{entity}/categories/{id}`

Delete specific internal collection that belongs to specific entity — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	InternalCollection DELETE success

`GET /v1/entities/{entity}/categories/{id}`

Get specific internal collection that belongs to specific entity – `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	InternalCollection model instance

`PATCH /v1/entities/{entity}/categories/{id}`

Update specific internal collection that belongs to specific entity – `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `InternalCollectionPartialExcluding_id_`

الاستجابات

الحالة	الوصف
`200`	InternalCollection PATCH success

`PUT /v1/entities/{entity}/categories/{id}`

Update specific internal collection that belongs to specific entity – `replaceById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `InternalCollectionExcluding_id_`

الاستجابات

الحالة	الوصف
`200`	InternalCollection PUT success

`GET /v1/entities/{entity}/categories/count`

Count internal collections that belongs to specific entity – `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	.InternalCollection model count

DataEntityFields — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 7 · data-service 1.11.0 عملية

`GET /v1/entities/{entity}/fields`

Get all fields of specific entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	—

الاستجابات

الحالة	الوصف
`200`	Array of DataEntityField model instances

`POST /v1/entities/{entity}/fields`

Add field to specific entity — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewDataViewFields`

الاستجابات

الحالة	الوصف
`200`	DataEntityField model instance

`DELETE /v1/entities/{entity}/fields/{fieldId}`

Delete single field of specific entity — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`fieldId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntityField DELETE success

`GET /v1/entities/{entity}/fields/{fieldId}`

Get single field of specific entity – `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`fieldId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntityField model instance

`PATCH /v1/entities/{entity}/fields/{fieldId}`

Update single field of specific entity – `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`fieldId`	path	نعم	`string`
`updateIdentifier`	query	لا	`string`

جسم الطلب: `DataEntityFieldPartialExcluding_id_`

الاستجابات

الحالة	الوصف
`200`	DataEntityField PATCH success

`GET /v1/entities/{entity}/fields/count`

Get total fields of specific entity – `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntityField model count.

`GET /v1/entities/fields/system`

Get all system fields — `findSystemFields`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntityField model instances

DataEntries — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 53 · data-service 1.11.0 عملية

`GET /v1/currencies`

List all currencies — `findCurrencies`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of CurrencyDto model instances

`DELETE /v1/entities/{entity}/archive/entries`

Delete a all data entries that belongs to an entity — `archiveEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: array of `object`

الاستجابات

الحالة	الوصف
`204`	Data entries status updated to archive success

`PATCH /v1/entities/{entity}/bulk-update`

Bulk update data entries — `bulkUpdateEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `BulkUpdateEntries`

الاستجابات

الحالة	الوصف
`200`	BulkUpdateResponseDto model instance

`PATCH /v1/entities/{entity}/bulk-upgrade`

Bulk upgrade data entries — `bulkUpgradeEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `BulkUpgradeEntries`

الاستجابات

الحالة	الوصف
`200`	BulkUpgradeResponseDto model instance

`GET /v1/entities/{entity}/entries`

Get a list of data entries that belong to an entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	–
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	Array of DataEntry model instances

`POST /v1/entities/{entity}/entries`

Create a new data entry — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewDataEntry`

الاستجابات

الحالة	الوصف
`200`	DataEntry model instance

`DELETE /v1/entities/{entity}/entries/{id}`

Delete a single data entry that belongs to an entity – `archiveEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntry DELETE success

`GET /v1/entities/{entity}/entries/{id}`

Get a single data entry that belongs to an entity – `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`excludeResolved`	query	لا	`boolean`
`fields`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntry model instance

`PATCH /v1/entities/{entity}/entries/{id}`

Update a single data entry that belongs to an entity – `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `UpdateDataEntryDto`

الاستجابات

الحالة	الوصف
`200`	DataEntry PATCH success

`GET /v1/entities/{entity}/entries/{id}/editable/{fieldId}`

Check if field is editable or not – `isFieldEditable`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`fieldId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	IsEditableFieldDto model instance

`GET /v1/entities/{entity}/entries/{id}/emails`

`getEmails`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataEntryEmailDto model instances

`GET /v1/entities/{entity}/entries/{id}/history`

return entry history – `findEntryHistoryById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntryHistory model instances

`Since 1.10` `GET /v1/entities/{entity}/entries/{id}/owner-change-notes`

return owner-change notes for an entry — `findOwnerChangeNotesById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of OwnerChangeNote model instances

`PUT /v1/entities/{entity}/entries/{id}/position`

update data entry _metaOrder — `updateEntryPosition`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `UpdateEntryPositionDto`

الاستجابات

الحالة	الوصف
`200`	DataEntry model instance

`PATCH /v1/entities/{entity}/entries/{id}/restore`

Restore an archived data entry — `restoreEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntry PATCH success

`PATCH /v1/entities/{entity}/entries/{id}/upgrade`

Upgrad., e data entry and reset the workflow that belongs to the entity – `upgradeEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `DataEntryPartialExcluding_creationInfo-updateInfo-stateHistory-id-isInvalid-invalidReason-searchIndex_`

الاستجابات

الحالة	الوصف
`200`	DataEntry PATCH success

`GET /v1/entities/{entity}/entries/count`

Counts the number of data entries that belong to an entity – `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`where`	query	لا	–
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	.DataEntry model count

`GET /v1/entities/{entity}/entries/draft`

Get a list of draft entries that belongs to the current user and entity – `findDraftEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntriesResponseV3 model instance

`POST /v1/entities/{entity}/entries/draft`

Create or update a draft entry – `upsertDraftEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `CreateDataEntryDto`

الاستجابات

الحالة	الوصف
`200`	DataEntry model instance

`DELETE /v1/entities/{entity}/entries/draft/{id}`

Delete a single draft entry that belongs to the current user and entity – `deleteDraftEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntry DELETE success

`GET /v1/entities/{entity}/entries/draft/{id}`

Get a single draft entry that belongs to the current user – `findDraftEntryById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntry model instance

`GET /v1/entities/{entity}/entries/draft/count`

Counts the number of draft entries that belongs to the current user and entity – `countDraftEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntry model count.

`GET /v1/entities/{entity}/entries/kanban/{categoryFieldId}`

Get a kanban view of data entries that belong to an entity and applied query tags – `findKanban`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`categoryFieldId`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`collectionId`	query	لا	`string`
`categoryId`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of
`appliedTagsFilters`	query	لا	`string` array of
`quickFilterParameters`	query	لا	–
`columns`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	DataEntryKanbanViewDto model instance

`GET /v1/entities/{entity}/entries/kanban/{categoryFieldId}/draft`

Get a kanban view of draft data entries that belong to the user – `findKanbanDraft`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`categoryFieldId`	path	نعم	`string`
`filterCriteria`	query	لا	–
`collectionId`	query	لا	`string`
`categoryId`	query	لا	`string`
`search`	query	لا	`string`
`columns`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	DataEntryKanbanViewDto model instance

`Since 1.10` `GET /v1/entities/{entity}/entries/kanban/{fieldId}/columns`

Distinct kanban reference values across scoped entries, with card counts – `findKanbanColumns`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`fieldId`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`collectionId`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of
`quickFilterParameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	KanbanColumnValuesDto model instance

Since `GET /v1/entities/{entity}/entries/kanban/{fieldId}/columns/draft`

`1.10`

Distinct kanban reference values across the user's drafts — `findKanbanColumnsDraft`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`fieldId`	path	نعم	`string`
`filterCriteria`	query	لا	–
`collectionId`	query	لا	`string`
`search`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	KanbanColumnValuesDto model instance

Since 1.10 `GET /v1/entities/{entity}/entries/summary`

Aggregate totals, and group headers with subtotals, over the entries matching the current scope — `findSummary`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`aggregates`	query	لا	`string` array of
`groupBy`	query	لا	`string` array of
`groupSkip`	query	لا	`number`
`groupLimit`	query	لا	`number`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`collectionId`	query	لا	`string`
`appliedQuickFilters`	query	لا	`string` array of
`appliedTagsFilters`	query	لا	`string` array of
`quickFilterParameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	EntrySummaryResponse model instance

`GET /v1/entities/{entity}/entries/tree`

Get a list of data entries that childs to another entry – `findTree`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`startingItem`	query	لا	`string`
`field`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntryTreeDto model instances

`POST /v1/entities/{entity}/entries/versions/count`

`getEntriesVersionsCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `SelectAllDto`

الاستجابات

الحالة	الوصف
`200`	Get count of entries by version for selectAll criteria

`GET /v1/entities/{entity}/fields/{field}/data`

Get a list of users as reference fields for reference field – `fetchReferenceFieldData`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`field`	path	نعم	`string`
`relatedEntryId`	query	لا	`string`
`cascadedParentId`	query	لا	`string`
`uiContext`	query	لا	`number`
`includeId`	query	لا	`string` array of
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of ReferenceProperty model instances

`PATCH /v1/entities/{entity}/restore/entries`

`restoreEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `ArchiveDataEntryDtoModel`

الاستجابات

الحالة	الوصف
`200`	resulting string value

`GET /v1/entities/{entity}/sources/{referenceFieldId}/{referenceDataEntry}`

Get mapped fields related to a reference field – `getMappedFieldsData`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`referenceFieldId`	path	نعم	`string`
`referenceDataEntry`	path	نعم	`string`
`entryId`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	Array of MappingFieldSourcesDto model instances

`POST /v1/entities/{entityId}/entries/batch`

Create a main entry and related entries in a single operation – `batchCreate`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`

جسم الطلب: `BatchCreateEntriesRequestDto`

الاستجابات

الحالة	الوصف
`200`	BatchCreateEntriesResponseDto model instance

`DELETE /v2/entities/{entity}/archive/entries`

Delete a all data entries that belongs to an entity – `archiveEntriesV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `ArchiveDataEntryDtoModel`

الاستجابات

الحالة	الوصف
`204`	Data entries status updated to archive success

`PATCH /v2/entities/{entity}/bulk-update`

`bulkUpdateEntriesV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `BulkUpdateEntries`

الاستجابات

الحالة	الوصف
`200`	resulting string value

`PATCH /v2/entities/{entity}/bulk-upgrade`

Bulk upgrade data entries v2 – `bulkUpgradeEntriesV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `BulkUpgradeEntries`

الاستجابات

الحالة	الوصف
`200`	BulkUpgradeResponseDto model instance

`GET /v2/entities/{entity}/entries`

Get a list of data entries that belong to an entity – `findV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	Array of DataEntry model instances

`POST /v2/entities/{entity}/entries/{entryId}/fields/{field}/decrypt`

`Since 1.10`

`decryptField`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`
`field`	path	نعم	`string`
`rowIndex`	query	لا	`number`

جسم الطلب: `object`

الاستجابات

الحالة	الوصف
`200`	The decrypted field value

`GET /v2/entities/{entity}/entries/{id}`

Get the entity and entry information. – `findEntityAndEntryById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`includeNavigationLinks`	query	لا	`boolean`
`relationId`	query	لا	`string`
`parentId`	query	لا	`string`
`filter`	query	لا	–
`collectionId`	query	لا	`string`
`search`	query	لا	`string`
`parameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataEntryV2Body model instance

`PATCH /v2/entities/{entity}/entries/{id}`

Patch a single data entry that belongs to an entity – `updateByIdV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`updateIdentifier`	query	لا	`string`

جسم الطلب: array of `JsonPatchOperation`

الاستجابات

الحالة	الوصف
`200`	DataEntry PATCH success

`GET /v2/entities/{entity}/entries/{id}/history`

return entry history – `findEntryHistoryByIdV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`lastTimeStamp`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntryHistory model instances

`DELETE /v2/entities/{entity}/entries/{id}/permanent`

Permanently delete a single archived data entry – `permanentlyDeleteEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Data entry permanently deleted

`GET /v2/entities/{entity}/entries/{id}/references`

Get incoming references for a data entry – `getIncomingReferences`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	EntryIncomingReferencesDto model instance

`GET /v2/entities/{entity}/entries/count`

Counts the number of data entries that belong to an entity – `countV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`whereCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاسم	الموضع	مطلوب	النوع
`quickFilterParameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataEntry model count.

`GET /v2/entities/{entity}/entries/draft/{id}`

Get the entity and draft entry information. – `findEntityAndDraftEntryById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntryV2Body model instance

`GET /v2/entities/{entity}/entries/tree`

Get a list of data entries that child to another entry with applied tags filters – `findTreeV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`startingItem`	query	لا	`string`
`field`	query	لا	`string`
`appliedTagsFilters`	query	لا	`string` array of
`appliedQuickFilters`	query	لا	`string` array of
`expandAllChildren`	query	لا	`boolean`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	DataEntryTreeDtoV2 model instance

`POST /v2/entities/{entity}/fields/{field}/data`

Get a list of users as reference fields for reference field with Included fields – `fetchReferenceFieldDataV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`field`	path	نعم	`string`

جسم الطلب: `GetEntriesForFieldRequestDto`

الاستجابات

الحالة	الوصف
`200`	Array of ReferenceProperty model instances

`Since 1.10` `POST /v2/entities/{entity}/fields/{field}/data/count`

Count the options a reference field offers for the given filter – `countReferenceFieldDataV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`field`	path	نعم	`string`

جسم الطلب: `GetEntriesForFieldRequestDto`

الاستجابات

الحالة	الوصف
`200`	.DataEntry model count

`DELETE /v2/entities/{entity}/permanent/entries`

Permanently delete multiple archived data entries – `permanentlyDeleteEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `PermanentlyDeleteDataEntryDtoModel`

الاستجابات

الحالة	الوصف
`200`	Data entries permanently deleted

`PATCH /v3/entities/{entity}/bulk-upgrade`

`bulkUpgradeEntriesV3`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `BulkUpgradeEntries`

الاستجابات

الحالة	الوصف
`200`	resulting string value

`GET /v3/entities/{entity}/entries`

Get a list of data entries that belong to an entity and applied query tags – `findV3`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of
`appliedTagsFilters`	query	لا	`string` array of
`groupBy`	query	لا	`string` array of
`quickFilterParameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataEntriesResponseV3 model instance

`PATCH /v3/entities/{entity}/entries/{id}`

Patch data entry and related entries – `patchDataEntryV3`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: array of `PatchDataEntryDtoV3`

الاستجابات

الحالة	الوصف
`200`	DataEntry PATCH success

`GET /v3/entities/{entity}/entries/tree`

Get a list of data entries that child to another entry with applied tags filters — `findTreeV3`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`startingItem`	query	لا	`string` array of
`field`	query	لا	`string`
`appliedTagsFilters`	query	لا	`string` array of
`appliedQuickFilters`	query	لا	`string` array of
`expandAllChildren`	query	لا	`boolean`
`filterCriteria`	query	لا	—
`search`	query	لا	`string`
`relationId`	query	لا	`string`
`relationEntry`	query	لا	`string`
`quickFilterParameters`	query	لا	—

الاستجابات

الحالة	الوصف
`200`	DataEntryTreeDtoV2 model instance

DataEntryTemplateExport — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 2 · data-service 1.11.0 عملية

`GET /v1/entities/{entity}/entries/export/{templateId}`

`exportEntriesWithTemplateV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`templateId`	path	نعم	`string`
`mode`	query	لا	`number`
`itemsToExport`	query	لا	`string` array of
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`sourceCollection`	query	لا	`string`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	File will be processed in background mode; the returned value is the messageId of the queued export command, which correlates with the socket event that later delivers the file.

`Since 1.10` `POST /v1/entities/{entity}/templates/status/entries`

`getTemplatesStatusForEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `GetTemplatesStatusForEntriesRequestDto`

الاستجابات

الحالة	الوصف
`200`	Array of TemplateStatusForEntriesResponseDto model instances

DataModule — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 7 · data-service 1.11.0 عملية

`GET /v1/data-module`

Get all modules — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`includeCoreModule`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataModule model instances

`POST /v1/data-module`

Create new module — `create`

جسم الطلب: `NewModule`

الاستجابات

الحالة	الوصف
`200`	DataModule model instance

`DELETE /v1/data-module/{id}`

Delete module by id — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataModule DELETE success

`GET /v1/data-module/{id}`

Get module by id — `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataModule model instance

`PATCH /v1/data-module/{id}`

Update module by id — `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `DataModulePartialExcluding_id-creationInfo-updateInfo-purgeInfo-purgeStatus_`

الاستجابات

الحالة	الوصف
`200`	DataModule PATCH success

`Since 1.10` `DELETE /v1/data-module/{id}/permanent`

Permanently delete a module and everything in it, in the background — `deleteModulePermanently`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`202`	The delete was accepted; the module goes once its job finishes

`GET /v1/data-module/count`

Get total modules in the system — `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–
`includeCoreModule`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataModule model count.

DataQuery — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 10 · data-service 1.11.0 عملية

`GET /v1/entities/{entity}/queries`

List all queries for specific entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	—

الاستجابات

الحالة	الوصف
`200`	Array of DataQueryWithTagDto model instances

`POST /v1/entities/{entity}/queries`

Create new query for specific entity — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewDataQueries`

الاستجابات

الحالة	الوصف
`200`	DataQueryWithTagDto model instance

`DELETE /v1/entities/{entity}/queries/{id}`

Delete query by id — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataQuery DELETE success

`GET /v1/entities/{entity}/queries/{id}`

Get query by id — `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filter`	query	لا	—

الاستجابات

الحالة	الوصف
`200`	DataQueryWithTagDto model instance

`PATCH /v1/entities/{entity}/queries/{id}`

Update query by id — `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `DataQueryWithTagDtoPartialExcluding_id-relatedTo-creationInfo-updateInfo_`

الاستجابات

الحالة	الوصف
`200`	DataQueryWithTagDto model instance

`GET /v1/entities/{entity}/queries/count`

Count all queries for specific entity — `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataQuery model count.

`GET /v1/queries`

List all queries in the system – `findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataQueryWithTagDto model instances

`GET /v1/queries/count`

Count all queries in the system – `countAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataQuery model count.

`GET /v1/queries/my`

List all queries for current user in a view or in all views – `findMyQueries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of MyQueryDto model instances

`POST /v1/queries/parser`

`criteriaParser`

جسم الطلب: `filterExpressionCriteria`

الاستجابات

الحالة	الوصف
`200`	renderedCriteria

DataRelation — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 20 · data-service 1.11.0 عملية

`GET /v1/entities/{entity}/entries/{entryId}/relations/{relationId}`

Get all related entries for specific relation that belongs to an entity — `findRelatedEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`
`relationId`	path	نعم	`string`
`filter`	query	لا	–
`include_virtual`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntry model instances

`GET /v1/entities/{entity}/entries/{entryId}/relations/{relationId}/count`

Get count of related entries for specific relation that belongs to an entity — `findRelatedEntriesCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`
`relationId`	path	نعم	`string`
`where`	query	لا	–
`include_virtual`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataRelation model count.

GET /v1/entities/{entity}/entries/{entryId}/relations/{relationId}/kanban/{categoryFieldId}

Get all related entries for specific relation that belongs to an entity with applied tag filter as kanban view — `findRelatedEntriesKanbanView`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`
`relationId`	path	نعم	`string`
`categoryFieldId`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`categoryId`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of
`appliedTagsFilters`	query	لا	`string` array of
`columns`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	DataEntryKanbanViewDto model instance

GET /v1/entities/{entity}/entries/{entryId}/relations/{relationId}/kanban/{fieldId}/columns

Since 1.10

Distinct kanban reference values across one relation's entries — `findRelatedEntriesKanbanColumns`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`
`relationId`	path	نعم	`string`
`fieldId`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`

الاسم	الموضع	مطلوب	النوع
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	KanbanColumnValuesDto model instance

`GET /v1/entities/{entity}/entries/{entryId}/relations/{relationId}/summary`

`Since 1.10`

Aggregate totals, and group headers with subtotals, over the related entries for a – `findRelatedEntriesSummary` relation that belongs to an entity

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`
`relationId`	path	نعم	`string`
`aggregates`	query	لا	`string` array of
`groupBy`	query	لا	`string` array of
`groupSkip`	query	لا	`number`
`groupLimit`	query	لا	`number`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	EntrySummaryResponse model instance

`GET /v1/entities/{entity}/relation`

Get all relations for specific entity – `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of DataRelationDto model instances

`POST /v1/entities/{entity}/relation`

Create new relation for specific entity – `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewDataRelation`

الاستجابات

الحالة	الوصف
`200`	DataRelation model instance

`DELETE /v1/entities/{entity}/relation/{id}`

Delete specific relation that belongs to an entity – `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataRelation DELETE success

`GET /v1/entities/{entity}/relation/{id}`

Get specific relation for specific entity – `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataRelationDto model instance

`PATCH /v1/entities/{entity}/relation/{id}`

Update specific relation that belongs to an entity – `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `DataRelationPartialExcluding_id_`

الاستجابات

الحالة	الوصف
`200`	DataRelation PATCH success

`GET /v1/entities/{entity}/relation/count`

Count relations for specific entity – `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataRelation model count.

`POST /v1/entities/joiner/relations`

Get data entities relations – `findEntitiesRelations`

جسم الطلب: `findEntitiesRelations`

الاستجابات

الحالة	الوصف
`200`	Array of DataRelationDto model instances

`GET /v1/relations`

List all relations in the system – `findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataRelation model instances

`GET /v1/relations/{id}`

Get a single relation by ID – `findOneById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataRelation model instance

`GET /v1/relations/count`

Count all relations in the system – `countAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	.DataRelation model count

`GET /v2/entities/{entity}/entries/{entryId}/relations/{relationId}`

Get all related entries for specific relation that belongs to an entity – `findRelatedEntriesV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`
`relationId`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntry model instances

GET /v2/entities/{entity}/entries/{entryId}/relations/{relationId}/count

Get count of related entries for specific relation that belongs to an entity – `findRelatedEntriesCountV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`
`relationId`	path	نعم	`string`
`whereCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataRelation model count.

GET /v2/entities/{entity}/relation

Get all relations for specific entity – `findV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataRelationListDto model instance

`GET /v2/entities/{entity}/relation/count`

Count relations for specific entity – `countV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataRelationCountDto model instance

`GET /v3/entities/{entity}/entries/{entryId}/relations/{relationId}`

Get all related entries for specific relation that belongs to an entity with applied tag filter – `findRelatedEntriesV3`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`
`relationId`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of
`appliedTagsFilters`	query	لا	`string` array of
`groupBy`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	DataEntriesResponseV3 model instance

DataViews — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 38 · data-service 1.11.0 عملية

`GET /v1/entities/{entity}/default-view`

Get default view thats belongs to a specific entity — `findDefaultView`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataViews model instance

`GET /v1/entities/{entity}/entries/{entry}/view`

List entries that belongs to a specific public view — `getAViewForEndUser`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entry`	path	نعم	`string`
`notificationId`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	EntryViewModel model instance

`GET /v1/entities/{entity}/views`

List views for specific entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataViews model instances

`POST /v1/entities/{entity}/views`

Create views for specific entity – `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewDataViews`

الاستجابات

الحالة	الوصف
`200`	DataViews model instance

`DELETE /v1/entities/{entity}/views/{id}`

Delete specific view that belongs to an entity – `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataViewsPermissionsDto DELETE success

`GET /v1/entities/{entity}/views/{id}`

Get view by entity id – `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataViewsPermissionsDto model instance

`PATCH /v1/entities/{entity}/views/{id}`

Update specific view that belongs to an entity – `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`clearIcon`	query	لا	`boolean`

جسم الطلب: `DataViewsPermissionsDtoPartialExcluding_id-module-isImplicit-creationInfo-updateInfo_`

الاستجابات

الحالة	الوصف
`200`	DataViewsPermissionsDto PATCH success

`GET /v1/entities/{entity}/views/{id}/entries`

List entries that belongs to a specific view – `findEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filter`	query	لا	–
`filterFirst`	query	لا	–
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	Array of DataEntry model instances

`GET /v1/entities/{entity}/views/{id}/entries/count`

Count of entries that belongs to a specific view – `entriesCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`where`	query	لا	–
`filterFirst`	query	لا	–
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	DataEntry model count.

`GET /v1/entities/{entity}/views/{id}/entries/export/xlsx`

`exportEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`fileName`	query	لا	`string`
`filter`	query	لا	–
`filterFirst`	query	لا	–
`include_virtual`	query	لا	`boolean`
`selectedColumns`	query	لا	`string` array of
`collectionId`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	The file content

`Since 1.10` `GET /v1/entities/{entity}/views/{id}/entries/summary`

Aggregate totals, and group headers with subtotals, over the entries of a specific view – `entriesSummary`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`aggregates`	query	لا	`string` array of
`groupBy`	query	لا	`string` array of
`groupSkip`	query	لا	`number`
`groupLimit`	query	لا	`number`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`filterFirst`	query	لا	–
`include_virtual`	query	لا	`boolean`
`collectionId`	query	لا	`string`
`appliedQuickFilters`	query	لا	`string` array of
`parameters`	query	لا	–
`quickFilterParameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	EntrySummaryResponse model instance

`PATCH /v1/entities/{entity}/views/{id}/permissions`

Update specific view permissions that belongs to an entity – `updatePermissionsById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `UpdateDataViewPermissions`

الاستجابات

الحالة	الوصف
`200`	DataViewsPermissionsDto PATCH success

`GET /v1/entities/{entity}/views/{id}/relation/{relationId}`

Get view by entity id – `findRelationViewById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`relationId`	path	نعم	`string`
`relatedEntryId`	query	لا	`string`
`mainViewId`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	DataViews model instance

`GET /v1/entities/{entity}/views/count`

Count views for specific entity – `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`where`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataViews model count.

`GET /v1/views`

List all views in the system – `findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataViews model instances

`DELETE /v1/views/{id}/entries`

Archive multiple data entries that belong to a view – `archiveEntriesForView`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `ArchiveDataEntryDtoModel`

الاستجابات

الحالة	الوصف
`200`	AfterSaveEntryDto model instance

`POST /v1/views/{id}/entries`

Create a new data entry – `createEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `NewDataEntry`

الاستجابات

الحالة	الوصف
`200`	AfterSaveEntryDto model instance

`DELETE /v1/views/{id}/entries/{entryId}`

Archive a single data entry using a view – `archiveEntryForView`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`entryId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	AfterSaveEntryDto model instance

`PATCH /v1/views/{id}/entries/{entryId}`

Patch a data entry using a view – `patchEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`entryId`	path	نعم	`string`

جسم الطلب: array of `PatchDataEntryDtoV3`

الاستجابات

الحالة	الوصف
`200`	AfterSaveEntryDto model instance

`PUT /v1/views/{id}/entries/{entryId}`

Update a data entry using a view – `updateEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`entryId`	path	نعم	`string`

جسم الطلب: `UpdateDataEntry`

الاستجابات

الحالة	الوصف
`200`	AfterSaveEntryDto model instance

`GET /v1/views/{id}/entries/{entryId}/after-save/{operation}`

Get the after save dto for an entry – `getAfterSaveDto`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`entryId`	path	نعم	`string`
`operation`	path	نعم	`number`

الاستجابات

الحالة	الوصف
`200`	AfterSaveEntryDto model instance

`DELETE /v1/views/{id}/entries/{entryId}/draft`

Delete a single draft entry that belongs to the current user and entity – `deleteDraftEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`entryId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	AfterSaveEntryDto model instance

`POST /v1/views/{id}/entries/draft`

Create or update a draft entry – `upsertDraftEntry`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `NewDataEntry`

الاستجابات

الحالة	الوصف
`200`	AfterSaveEntryDto model instance

`GET /v1/views/{moduleId}`

.List views related to specific module – `findViewsInModule`

المعاملات

الاسم	الموضع	مطلوب	النوع
`moduleId`	path	نعم	`string`
`filter`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataViews model instances

`GET /v1/views/{moduleId}/count`

.List views related to specific module – `findViewsInModuleCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`moduleId`	path	نعم	`string`
`where`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataViews model count.

`GET /v1/views/admin`

List views for admin with entity and module info – `getViewsForAdmin`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataViewForAdminDto model instances

`GET /v1/views/admin/count`

Count all views in the system – `countAllForAdmin`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	DataViews model count.

`GET /v1/views/count`

Count all views in the system – `countAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	.DataViews model count

`GET /v1/views/end-user/{id}`

Find view by id without entity. This is used for end user to get the view info – `findViewForEndUser`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataViews model instance

`GET /v1/views/end-user/{id}/permissions`

.Find view permissions by id without entity – `findViewPermissionsForEndUser`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	EndUserViewsPermissionsDto model instance

`GET /v1/views/end-user/can-access/{id}`

returns whether the user can access this view or not – `canUserAccessView`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	the result will be true if the user can access the view false otherwise

`GET /v1/views/end-user/modules`

List the modules holding views the current user can open, each with those views – `findModulesForEndUser`

الاستجابات

الحالة	الوصف
`200`	LiteModuleWithViews model instances

`GET /v2/entities/{entity}/views/{id}/entries`

List entries that belongs to a specific view – `findEntriesV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`filterFirst`	query	لا	–
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	Array of DataEntry model instances

`GET /v2/entities/{entity}/views/{id}/entries/count`

Count of entries that belongs to a specific view – `entriesCountV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`whereCriteria`	query	لا	–
`search`	query	لا	`string`
`filterFirst`	query	لا	–
`include_virtual`	query	لا	`boolean`

الاسم	الموضع	مطلوب	النوع
`appliedQuickFilters`	query	لا	array of `string`
`parameters`	query	لا	–
`quickFilterParameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataEntry model count.

`GET /v2/views/{id}`

Get view by id – `findByViewId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataViewsPermissionsDto model instance

`GET /v2/views/end-user/{id}`

Find view with redirection rules without entity. This is used for end user to get the view info – `findViewForEndUserWithRedirect`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	EndUserViewDto model instance

`GET /v2/views/module/{moduleId}`

List views related to specific module. – `findViewsInModuleByModuleId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`moduleId`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–
`includeImplicit`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of DataViews model instances

`GET /v3/entities/{entity}/views/{id}/entries`

List entries that belongs to a specific view with applied tags – `findEntriesV3`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`filterFirst`	query	لا	–
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of
`appliedTagsFilters`	query	لا	`string` array of
`parameters`	query	لا	–
`groupBy`	query	لا	`string` array of
`quickFilterParameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataEntriesResponseV3 model instance

Delegations — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 2 · data-service 1.11.0 عملية

`GET /v1/delegations/{id}`

Returns a delegation by id — `getDelegation`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	MyDelegationsDto model instance

`GET /v1/delegations/my`

Returns my current active delegations — `getActiveDelegations`

الاستجابات

الحالة	الوصف
`200`	Array of MyDelegationsDto model instances

EntityEntriesExport — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`Since 1.10` `GET /v3/entities/{entity}/entries/export`

`exportEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`format`	query	لا	`string`
`fields`	query	لا	`string` array of
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of
`appliedTagsFilters`	query	لا	`string` array of
`groupBy`	query	لا	`string` array of
`quickFilterParameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	The file content

Expression — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 2 · data-service 1.11.0 عملية

`POST /v1/entities/{entityId}/{entryId}/tokens/resolve`

`resolveTokens`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`
`entryId`	path	نعم	`string`

جسم الطلب: `ResolveTokensRequest`

الاستجابات

الحالة	الوصف
`200`	Template fields with {{token}} placeholders replaced. Fields whose tokens could not be resolved are null.

`POST /v1/entities/{entityId}/expressions/resolve`

`resolveExpression`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`

جسم الطلب: `ResolveExpressionRequestDto`

الاستجابات

الحالة	الوصف
`200`	Resolved expression value

GenericView — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 10 · data-service 1.11.0 عملية

`GET /v1/generic-views`

List all generic views — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of GenericView model instances

`POST /v1/generic-views`

Create new GenericView — `create`

جسم الطلب: `NewGenericView`

الاستجابات

الحالة	الوصف
`200`	GenericView model instance

`DELETE /v1/generic-views/{id}`

Delete specific generic view by Id — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	GenericViewPermissionsDto DELETE success

`GET /v1/generic-views/{id}`

Return single generic view by Id – `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	GenericViewPermissionsDto model instance

`PATCH /v1/generic-views/{id}`

Update specific generic view by Id – `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`clearIcon`	query	لا	`boolean`

جسم الطلب: `GenericViewPermissionsDtoPartialExcluding_id-creationInfo-updateInfo_`

الاستجابات

الحالة	الوصف
`200`	GenericViewPermissionsDto PATCH success

`PATCH /v1/generic-views/{id}/permissions`

Update specific generic view permissions by Id – `updatePermissionsById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `UpdateGenericViewPermissions`

الاستجابات

الحالة	الوصف
`200`	GenericViewPermissionsDto PATCH success

`GET /v1/generic-views/count`

Count of GenericViews — `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	GenericView model count.

`GET /v1/generic-views/default`

return the default view for the user — `getUserDefaultView`

المعاملات

الاسم	الموضع	مطلوب	النوع
`relatedTo`	query	لا	–
`x-domain-id`	header	لا	`string`

الاستجابات

الحالة	الوصف
`200`	GenericViewPermissionsDto model instance

`GET /v1/generic-views/end-user/{id}`

Return single generic view by Id — `findByIdEndUser`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	GenericViewPermissionsDto model instance

`GET /v2/generic-views/end-user/{id}`

Return a dto containing either redirection rule or components based on matching rules — `findByIdEndUserV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	GenericViewEndUserDto model instance

Groups — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v1/group/{groupId}/users`

`getUsersInsideGroup`

المعاملات

الاسم	الموضع	مطلوب	النوع
`groupId`	path	نعم	`string`
`includeResolved`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of ReferencePropertyWithResolved model instances

Icons — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v1/icons`

`getIcons`

الاستجابات

الحالة	الوصف
`200`	Icon manifest keyed by icon name
`304`	Not modified — reuse cached body when If-None-Match matches ETag

ImportExport — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 4 · 1.11.0 data-service عملية

`POST /v1/export/entity`

`exportDataEntities`

جسم الطلب: `ExportingEntitiesDto`

الاستجابات

الحالة	الوصف
`200`	The file content

`POST /v1/export/module`

`exportDataModel`

جسم الطلب: `ExportingDataModulesDto`

الاستجابات

الحالة	الوصف
`200`	The file content

`POST /v1/import/{fileId}`

`validateFileToImport`

المعاملات

الاسم	الموضع	مطلوب	النوع
`fileId`	path	نعم	`string`

جسم الطلب: `ImportEntitiesDto`

الاستجابات

الحالة	الوصف
`200`	ImportResultDto model instance

`POST /v1/import/{fileId}/confirm`

`confirmFileImport`

المعاملات

الاسم	الموضع	مطلوب	النوع
`fileId`	path	نعم	`string`

جسم الطلب: `ImportEntitiesDto`

الاستجابات

الحالة	الوصف
`200`	ImportResultDto model instance

Job — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v1/job/{messageId}`

`getJobStatusByMessageId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`messageId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Get job status by message id

Locks — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 5 · data-service 1.11.0 عملية

`POST /v1/locks`

`createLockObject`

جسم الطلب: `CreateLockObjectDto`

الاستجابات

الحالة	الوصف
`200`	Create lock

`GET /v1/locks/{objectId}`

`getLocksByObjectId`

المعاملات

الاسم	الموضع	مطلوب	النوع
`objectId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Get locks by object id

`PUT /v1/locks/{objectId}`

`patchLockObject`

المعاملات

الاسم	الموضع	مطلوب	النوع
`objectId`	path	نعم	`string`

جسم الطلب: `PatchLockObjectDto`

الاستجابات

الحالة	الوصف
`200`	Patch lock object

`GET /v1/locks/{objectId}/history`

`getLocksObjectHistory`

المعاملات

الاسم	الموضع	مطلوب	النوع
`objectId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Get locks by object history

`GET /v1/locks/{objectId}/history/count`

`getLocksObjectHistoryCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`objectId`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Get locks by object history count

OrgConfig — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v1/configs`

Get org configs — `getOrgConfigs`

الاستجابات

الحالة	الوصف
`200`	Get new token

QuickFilterGroup — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 6 · data-service 1.11.0 عملية

`Since 1.10` `GET /v1/entities/{entity}/quick-filter-groups`

List the quick filter groups of an entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of QuickFilterGroupDto model instances

`Since 1.10` `POST /v1/entities/{entity}/quick-filter-groups`

Create a quick filter group on an entity — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewQuickFilterGroup`

الاستجابات

الحالة	الوصف
`200`	QuickFilterGroupDto model instance

`Since 1.10` `DELETE /v1/entities/{entity}/quick-filter-groups/{id}`

Delete a quick filter group; its queries become ungrouped — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	QuickFilterGroup DELETE success

`Since 1.10` `PATCH /v1/entities/{entity}/quick-filter-groups/{id}`

Rename a quick filter group or set its quick queries – `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `UpdateQuickFilterGroup`

الاستجابات

الحالة	الوصف
`200`	QuickFilterGroupDto model instance

`Since 1.10` `GET /v1/quick-filter-groups`

List all quick filter groups in the system – `findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of QuickFilterGroup model instances

`Since 1.10` `GET /v1/quick-filter-groups/{id}`

Get a quick filter group by id, without an entity – `findByIdWithoutEntity`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	QuickFilterGroup model instance

ReportEntry — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 12 · data-service 1.11.0 عملية

`GET /v1/reports`

Get all reports defined in the system — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of ReportEntry model instances

`POST /v1/reports`

Create a new report — `create`

جسم الطلب: `NewReportEntry`

الاستجابات

الحالة	الوصف
`200`	ReportEntryDto model instance

`DELETE /v1/reports/{id}`

Delete report with id — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	ReportEntry DELETE success

`GET /v1/reports/{id}`

Get report by id – `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	ReportEntryDto model instance

`PATCH /v1/reports/{id}`

Update report entry by id – `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `ReportEntryUpdateById`

الاستجابات

الحالة	الوصف
`200`	ReportEntry PATCH success

`GET /v1/reports/{id}/entries`

[]Execute Report and returns the result as DataEntry – `executeReport`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–
`include_virtual`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	Array of ReportEntry model instances

`GET /v1/reports/{id}/entries/count`

Get records count for the report execution — `executeReportCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`where`	query	لا	–
`include_virtual`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	.ReportEntryDto model count

`PATCH /v1/reports/{id}/permissions`

Update report entry by id — `updatePermissionsById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `UpdateReportEntryPermissions`

الاستجابات

الحالة	الوصف
`200`	ReportEntry PATCH success

`GET /v1/reports/count`

Get report entry count — `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	.ReportEntryDto model count

`GET /v2/reports/{id}/entries`

[]Execute Report and returns the result as DataEntry — `executeReportV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filterCriteria`	query	لا	–
`include_virtual`	query	لا	`boolean`
`search`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	Array of ReportEntry model instances

`GET /v2/reports/{id}/entries/count`

Get records count for the report execution — `executeReportCountV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`whereCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`

الاستجابات

الحالة	الوصف
`200`	.ReportEntryDto model count

`GET /v2/reports/{id}/export/{format}`

`exportReportV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`format`	path	نعم	`string`
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`include_virtual`	query	لا	`boolean`
`selectedColumns`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	The file content

Request — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v1/requests/{entity}/{id}/{mode}`

Get the implicit view thats belongs to the request if the user can access it – `canAccessRequest`.

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`mode`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	DataViews model instance

SensitiveFieldGrants — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 8 · data-service 1.11.0 عملية

`GET /v1/sensitive-field-grants`

List all active sensitive-field access grants — `listSensitiveFieldGrants`

الاستجابات

الحالة	الوصف
`200`	Array of SensitiveFieldGrantDto instances

`POST /v1/sensitive-field-grants`

Create sensitive-field grants for a scope — `createSensitiveFieldGrants`

جسم الطلب: `CreateSensitiveFieldGrantsDto`

الاستجابات

الحالة	الوصف
`200`	CreateSensitiveFieldGrantsResultDto instance

`DELETE /v1/sensitive-field-grants/{id}`

Delete a sensitive-field grant — `deleteSensitiveFieldGrant`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`204`	Sensitive-field grant deleted successfully

`PATCH /v1/sensitive-field-grants/{id}`

Update a sensitive-field grant — `updateSensitiveFieldGrant`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `UpdateSensitiveFieldGrantDto`

الاستجابات

الحالة	الوصف
`200`	SensitiveFieldGrantDto instance

`PATCH /v1/sensitive-field-grants/{id}/archive`

Archive a sensitive-field grant — `archiveSensitiveFieldGrant`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`204`	Sensitive-field grant archived successfully

`GET /v1/sensitive-field-grants/by-group/{groupId}`

List a group's sensitive-field grants — `listGroupSensitiveFieldGrants`

المعاملات

الاسم	الموضع	مطلوب	النوع
`groupId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of SensitiveFieldGrantDto instances

`GET /v1/sensitive-field-grants/by-user/{userId}`

List a user's effective sensitive-field grants (direct + indirect) — `listUserSensitiveFieldGrants`

المعاملات

الاسم	الموضع	مطلوب	النوع
`userId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of UserSensitiveFieldGrantDto instances

`GET /v1/sensitive-field-grants/inherited-holders`

List a sensitive field's inherited (entity/module wildcard) holders — `getInheritedSensitiveFieldHolders`

المعاملات

الاسم	الموضع	مطلوب	النوع
`moduleNn`	query	نعم	`string`
`entityNn`	query	نعم	`string`
`fieldNn`	query	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	InheritedSensitiveFieldHoldersDto instance

SideBar — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v2/side-bar`

Get all sidebar views (system and module) — `getViewsForSidebarV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`x-domain-id`	header	لا	`string`
`x-space-id`	header	لا	`string`
`include`	query	لا	`number`

الاستجابات

الحالة	الوصف
`200`	SidebarDto model instances

Spaces — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 10 · data-service 1.11.0 عملية

`GET /v1/spaces`

Get all spaces — `getAllSpaces`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filterCriteria`	query	لا	–
`search`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	Array of SpaceEntry model instances

`POST /v1/spaces`

Create a new space — `create`

جسم الطلب: `SpaceEntry`

الاستجابات

الحالة	الوصف
`200`	SpaceEntry model instance

`GET /v1/spaces/{id}`

Get single space by id — `getSpace`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	SpaceEntry model instance

`POST /v1/spaces/{id}/activate`

`activateSpace`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of SpacesController.activateSpace

`POST /v1/spaces/{id}/deactivate`

`deActivateSpace`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of SpacesController.deActivateSpace

`POST /v1/spaces/{id}/end-user-activate`

`spaceEndUserActivate`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of SpacesController.spaceEndUserActivate

`POST /v1/spaces/{id}/end-user-deactivate`

`spaceEndUserDeActivate`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of SpacesController.spaceEndUserDeActivate

`GET /v1/spaces/count`

Get spaces count — `getSpacesCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`search`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	.DataEntry model count

`GET /v1/spaces/my`

Get my spaces — `getSpacesForCurrentUser`

الاستجابات

الحالة	الوصف
`200`	Array of SpaceEntry model instances

`GET /v1/spaces/user/{id}`

Get user spaces — `getUserSpaces`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of SpaceEntry model instances

StatisticQuery — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 11 · data-service 1.11.0 عملية

`GET /v1/entities/{entity}/statistic-queries`

List statistic queries that belongs to a specific entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	—

الاستجابات

الحالة	الوصف
`200`	Array of StatisticQuery model instances

`POST /v1/entities/{entity}/statistic-queries`

Create new statistic query for specific entity — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewStatisticQuery`

الاستجابات

الحالة	الوصف
`200`	StatisticQuery model instance

`DELETE /v1/entities/{entity}/statistic-queries/{id}`

Delete statistic query by id — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	StatisticQuery DELETE success

`GET /v1/entities/{entity}/statistic-queries/{id}`

Get statistic query by id — `findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	StatisticQuery model instance

`PATCH /v1/entities/{entity}/statistic-queries/{id}`

Update statistic query by id — `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `StatisticQueryPartialExcluding_id_`

الاستجابات

الحالة	الوصف
`200`	StatisticQuery PATCH success

`GET /v1/entities/{entity}/statistic-queries/count`

Count statistic queries that belongs to a specific entity — `count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	StatisticQuery model count.

`GET /v1/statistic-queries`

List all statistic queries in the system – `findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of StatisticQuery model instances

`GET /v1/statistic-queries/{id}`

Get statistic query by id without entity – `findByIdWithoutEntity`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	StatisticQueryDTO model instance

`Since 1.10` `GET /v1/statistic-queries/by-ids`

Get several statistic queries by id without entity – `findByIdsWithoutEntity`

المعاملات

الاسم	الموضع	مطلوب	النوع
`ids`	query	لا	`string` array of

الاستجابات

الحالة	الوصف
`200`	Array of StatisticQueryDTO model instances

`GET /v1/statistic-queries/count`

Count all statistic queries in the system — `countAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	StatisticQuery model count.

`GET /v2/entities/{entity}/statistic-queries/{id}/execute`

`executeStatisticQueryV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	execute specific statistic query

Stores — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 4 · data-service 1.11.0 عملية

`GET /v1/entities/{entityId}/stores`

Get all stores for an entity including default stores — `getAllStores`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`
`filterCriteria`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of StoresResponseDto model instances

`POST /v1/entities/{entityId}/stores`

Create a new store for an entity — `createStore`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`

جسم الطلب: `CreateDataStoreRequestDto`

الاستجابات

الحالة	الوصف
`200`	StoresResponseDto model instance

`GET /v1/entities/{entityId}/stores/{storeId}`

Get a store for an entity — `getStore`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`storeId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	StoresResponseDto model instance

`PUT /v1/entities/{entityId}/stores/{storeId}`

`updateStore`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`
`storeId`	path	نعم	`string`

جسم الطلب: `UpdateDataStoreRequestDtoPartial`

الاستجابات

الحالة	الوصف
`200`	update a store for an entity

SystemButton — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 10 · data-service 1.11.0 عملية

`GET /v1/system-buttons`

`findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of SystemButtonDataEntryDto model instances

`POST /v1/system-buttons/{id}`

`executeSystemButton`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `ExecuteSystemButtonDto`

الاستجابات

الحالة	الوصف
`200`	resulting string value

`GET /v1/system-buttons/{id}/options`

`getSystemButtonActionWithOptions`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of DataReactionAction model instances

`GET /v1/system-buttons/{relatedToId}`

`find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`relatedToId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of SystemButtonDataEntryDto model instances

`GET /v1/system-buttons/{relatedToId}/active`

`findForUser`

المعاملات

الاسم	الموضع	مطلوب	النوع
`relatedToId`	path	نعم	`string`
`entryId`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	Array of SystemButtonDataEntryDto model instances

`POST /v1/system-buttons/{relatedToId}/status/entries`

`getSystemButtonsStatusForEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`relatedToId`	path	نعم	`string`

جسم الطلب: `GetSystemButtonsStatusForEntriesRequestDtoModel`

الاستجابات

الحالة	الوصف
`200`	Array of SystemButtonStatusForEntriesResponseDtoModel model instances

`GET /v1/system-buttons/by-id/{id}`

`findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	SystemButtonDataEntry model instance

`GET /v1/system-buttons/count`

`countAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	.SystemButtonDataEntryDto model count

`GET /v1/system-buttons/raw`

`findAllRaw`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of SystemButtonDataEntry model instances

`GET /v2/system-buttons/{id}/options`

`getSystemButtonActionWithOptionsV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	SystemButtonOptionsDto model instance

SystemLink — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 2 · data-service 1.11.0 عملية

`GET /v1/system-link/generate`

Generate random token — `generateToken`

الاستجابات

الحالة	الوصف
`200`	Get new token

`GET /v1/system-link/resolve/{token}`

Get anonymous view or generic view — `getAnonymousView`

المعاملات

الاسم	الموضع	مطلوب	النوع
`token`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Get new token

SystemObject — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v1/system-objects`

Array of system objects as reference with parent and resolved (like ApplicationComponentFetcher at – `find` SystemLevel)

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of ReferencePropertyWithParentAndIncludedFields model instances

SystemSettings — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`Since 1.10` `GET /v1/system-settings/public`

`getPublic`

الاستجابات

الحالة	الوصف
`200`	Public system settings

TagGroup — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 6 · data-service 1.11.0 عملية

`Since 1.10` `GET /v1/entities/{entity}/tag-groups`

List the tag groups of an entity — `find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Array of TagGroupDto model instances

`Since 1.10` `POST /v1/entities/{entity}/tag-groups`

Create a tag group on an entity — `create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewTagGroup`

الاستجابات

الحالة	الوصف
`200`	TagGroupDto model instance

`Since 1.10` `DELETE /v1/entities/{entity}/tag-groups/{id}`

Delete a tag group; its tags become ungrouped — `deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	TagGroup DELETE success

`Since 1.10` `PATCH /v1/entities/{entity}/tag-groups/{id}`

Rename a tag group or set its tags — `updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `UpdateTagGroup`

الاستجابات

الحالة	الوصف
`200`	TagGroupDto model instance

`Since 1.10` `GET /v1/tag-groups`

List all tag groups in the system — `findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of TagGroup model instances

`Since 1.10` `GET /v1/tag-groups/{id}`

Get a tag group by id, without an entity — `findByIdWithoutEntity`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	TagGroup model instance

Tags — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 2 · data-service 1.11.0 عملية

`GET /v1/tags/{entityId}/all`

Get all tags under data query — `getTags`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Get new token

`GET /v1/tags/{entityId}/groups`

Get tag groups — `getTagGroups`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Get new token

Templates — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v1/templates/{id}/used`

returns whether template is used or not — `isTemplateUsed`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	the result will be true if the user can access the view false otherwise

UiSettings — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 13 · data-service 1.11.0 عملية

`GET /v1/fonts/files/{fontId}/{format}`

`getFontFile`

المعاملات

الاسم	الموضع	مطلوب	النوع
`fontId`	path	نعم	`string`
`format`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	The file content
`204`	The font entry exists but this format was never uploaded. Empty body.

`GET /v1/formats/date`

`getAvailableDateFormats`

الاستجابات

الحالة	الوصف
`200`	Get available date formats

`GET /v1/formats/date-time`

`getAvailableDateTimeFormats`

الاستجابات

الحالة	الوصف
`200`	Get available date time formats

`GET /v1/formats/time`

`getAvailableTimeFormats`

الاستجابات

الحالة	الوصف
`200`	Get available time formats

`GET /v1/language`

return all available languages – `getAvailableLanguages`

الاستجابات

الحالة	الوصف
`200`	Array of languages model instances

`GET /v1/language/files/{langCode}/{platform}`

`getLanguageTranslationFile`

المعاملات

الاسم	الموضع	مطلوب	النوع
`langCode`	path	نعم	`string`
`platform`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	The file content
`204`	.The language exists but has no translation file uploaded for this platform. Empty body – callers should fall back to bundled assets

`GET /v1/settings/user`

`getUserUiSettings`

الاستجابات

الحالة	الوصف
`200`	return the ui settings for the current user

`GET /v1/settings/user/anonymous`

`getAnonymousUserUiSettings`

الاستجابات

الحالة	الوصف
`200`	return the ui settings for anonymous user

`GET /v1/themes`

`getAvailableThemes`

الاستجابات

الحالة	الوصف
`200`	Get available themes

`GET /v1/themes/{themeId}`

`getThemeDetails`

المعاملات

الاسم	الموضع	مطلوب	النوع
`themeId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Get theme details

`GET /v1/themes/anonymous`

`getAnonymousAvailableThemes`

الاستجابات

الحالة	الوصف
`200`	Get available themes for anonymous user

`GET /v1/themes/files/{themeId}/css`

`getThemeCssFile`

المعاملات

الاسم	الموضع	مطلوب	النوع
`themeId`	path	نعم	`string`
`if-none-match`	header	لا	`string`

الاستجابات

الحالة	الوصف
`200`	The file content
`204`	The theme exists but has no general_css configured. Empty body — never a JSON envelope, which would be invalid CSS if injected as a .stylesheet

الحالة	الوصف
`304`	The stylesheet named by If-None-Match is still current. Empty body.

`GET /v1/time-zone`

`getAvailableTimeZones`

الاستجابات

الحالة	الوصف
`200`	Get available time zones

User — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 11 · data-service 1.11.0 عملية

`POST /v1/users/{userId}/activate`

`activateUser`

المعاملات

الاسم	الموضع	مطلوب	النوع
`userId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of UserController.activateUser

`POST /v1/users/{userId}/deactivate`

`deactivateUser`

المعاملات

الاسم	الموضع	مطلوب	النوع
`userId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of UserController.deactivateUser

`POST /v1/users/{userId}/regenerate-app-token`

`regenerateAppToken`

المعاملات

الاسم	الموضع	مطلوب	النوع
`userId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of UserController.regenerateAppToken

`POST /v1/users/{userId}/reset-password`

`resetUserPasswordBySystem`

المعاملات

الاسم	الموضع	مطلوب	النوع
`userId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Return value of UserController.resetUserPasswordBySystem

`GET /v1/users/direct-manager`

Get current user direct manager — `getUserDirectManager`

الاستجابات

الحالة	الوصف
`200`	UserOrgUnitDto model instance

`GET /v1/users/my-groups`

Get user profile info — `getMyGroups`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntry model instances

`GET /v1/users/my-orgunit`

Returns the list of users inside the org unit of current person related to the signed in user and the parent org unit and all children org units — `getUserOrgUnitUsers`

الاستجابات

الحالة	الوصف
`200`	Array of DataEntry model instances

`GET /v1/users/profile`

Get user profile info – `getUserProfile`

الاستجابات

الحالة	الوصف
`200`	UserProfileDto model instance

`GET /v1/users/state`

return the current user state. If a state doesn't exist for current user, create one with default options – `getUserState`

and return the newly created state

الاستجابات

الحالة	الوصف
`200`	UserStateDataEntryValue model instance

`PUT /v1/users/state`

return the current user state. If a state doesn't exist for current user, create one with default – `updateUserState`

options and return the newly created state

جسم الطلب: `UserStateDataEntryValueExcluding_user-firstLogin_`

الاستجابات

الحالة	الوصف
`200`	UserStateDataEntryValue model instance

`PUT /v1/users/state/reset`

Reset the current user state – `resetUserState`

الاستجابات

الحالة	الوصف
`200`	UserStateDataEntryValue model instance

UserPreferences — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`PATCH /v1/user-preferences`

Update user preferences for a specific object (deep merge) — `updateUserPreferences`

جسم الطلب: `UpdateUserPreferencesDto`

الاستجابات

الحالة	الوصف
`200`	User preferences updated successfully

Version — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`GET /v1/version`

Get versions — `getVersion`

الاستجابات

الحالة	الوصف
`200`	Get versions

ViewApiDocs — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`Since 1.10` `GET /v1/views/{viewId}/openapi.json`

Returns a per-View OpenAPI document covering that View's entity schema and entry CRUD — `getViewOpenApiDocument` API, scoped via the /v1/v/{viewId}/ wrapped path prefix.

المعاملات

الاسم	الموضع	مطلوب	النوع
`viewId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	OpenAPI 3.0 document
`403`	The View's displayConfig.swaggerDocument flag is disabled, or the caller fails the end-user View access check (see assertViewAccessibleToEndUser) — system admins bypass both.
`404`	No View exists with that id, or it belongs to an archived entity.

ViewEntriesExport — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`Since 1.10` `GET /v3/entities/{entity}/views/{id}/entries/export`

`exportEntries`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`format`	query	لا	`string`
`fields`	query	لا	`string` array of
`filterCriteria`	query	لا	–
`search`	query	لا	`string`
`filterFirst`	query	لا	–
`include_virtual`	query	لا	`boolean`
`appliedQuickFilters`	query	لا	`string` array of
`appliedTagsFilters`	query	لا	`string` array of
`parameters`	query	لا	–
`groupBy`	query	لا	`string` array of
`truncate`	query	لا	`boolean`
`quickFilterParameters`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	The file content

Workflow — data-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `data-service.json` · 1 · data-service 1.11.0 عملية

`POST /external-step/return`

Callback invoked by an external system redirecting a browser back after completing an – `executeExternalStepReturn` ExternalStep. Authenticated via the opaque return token itself, not a user session.

جسم الطلب: `object`

الاستجابات

الحالة	الوصف
`200`	Result of processing the return token

Functions — functions-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `functions-service.json` · 2 · functions 0.51.5 عملية

`POST /v1/{functionId}/execute`

`executeFunction`

المعاملات

الاسم	الموضع	مطلوب	النوع
`functionId`	path	نعم	`string`

جسم الطلب: `object`

الاستجابات

الحالة	الوصف
`200`	Function executed
`400`	Bad request

`GET /v1/{scope}`

`getFunctions`

المعاملات

الاسم	الموضع	مطلوب	النوع
`scope`	path	نعم	`string`
`scopeId`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	–

Notification — notification-api

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `notification-api.json` · 3 · notification-api 0.0.1 عملية

`GET /v1/my`

`getMyNotifications`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	DataEntry model instance

`PATCH /v1/my/{id}/mark-as-read`

`markAsRead`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	NotificationDataEntryValue model instance

`GET /v1/my/count`

`getMyNotificationCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	NotificationDataEntryValue model count

NotificationTemplate — notification-api

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `notification-api.json` · 1 · notification-api 0.0.1 عملية

`POST /v1/template/resolve-tokens`

`resolveTokens`

جسم الطلب: `object`

الاستجابات

الحالة	الوصف
`200`	Token resolved

PushNotificationsTokens — notification-api

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `notification-api.json` · 4 · notification-api 0.0.1 عملية

`GET /v1/push-tokens`

`find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of PushNotificationToken model instances

`POST /v1/push-tokens`

`create`

جسم الطلب: `NewPushNotificationToken`

الاستجابات

الحالة	الوصف
`200`	PushNotificationToken model instance

`Since 1.10` `DELETE /v1/push-tokens/{token}`

`deleteOwn`

المعاملات

الاسم	الموضع	مطلوب	النوع
`token`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`204`	No Content

`Since 1.10` `POST /v1/push-tokens/invalidate`

`invalidate`

جسم الطلب: `object`

الاستجابات

الحالة	الوصف
`200`	Number of deleted PushNotificationToken rows

Announcement — reaction-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `reaction-service.json` · 7 عملية · reaction 0.51.5

`GET /v1/announcements`

`find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	List of all Announcement records

`GET /v1/announcements/{entityId}`

`findByEntity`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	List of Announcements for a specific entity

`POST /v1/announcements/{entityId}`

`create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`

جسم الطلب: `AnnouncementExcluding_id-entityId-reactions-creationInfo_`

الاستجابات

الحالة	الوصف
`200`	Create announcement for specific entity

`Since 1.10` `DELETE /v1/announcements/{entityId}/{announcementId}`

`deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`
`announcementId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`204`	Announcement DELETE success

`GET /v1/announcements/{entityId}/{announcementId}`

`findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`
`announcementId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Get a single Announcement by id (scoped to entity)

`PATCH /v1/announcements/{entityId}/{announcementId}`

`update`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`
`announcementId`	path	نعم	`string`

جسم الطلب: `AnnouncementPartialExcluding_id-entityId-reactions-creationInfo_`

الاستجابات

الحالة	الوصف
`200`	Update announcement for specific entity

`GET /v1/announcements/by-id/{announcementId}`

`findByIdOnly`

المعاملات

الاسم	الموضع	مطلوب	النوع
`announcementId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Get a single Announcement by id

DataReaction — reaction-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `reaction-service.json` · 12 عملية · reaction 0.51.5

`GET /v1/reaction/{entity}`

`find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of Reaction model instances, each with its execution and failure counts

`POST /v1/reaction/{entity}`

`create`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

جسم الطلب: `NewDataReaction`

الاستجابات

الحالة	الوصف
`200`	Create a new reaction

`DELETE /v1/reaction/{entity}/{id}`

`deleteById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`204`	Reaction DELETE success

`GET /v1/reaction/{entity}/{id}`

`findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Reaction model instance

`PATCH /v1/reaction/{entity}/{id}`

`updateById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`

جسم الطلب: `DataReactionPatch`

الاستجابات

الحالة	الوصف
`200`	Reaction PATCH success

`Since 1.10` `GET /v1/reaction/{entity}/{id}/history`

`history`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`id`	path	نعم	`string`
`outcome`	query	لا	`string`
`from`	query	لا	`string`
`to`	query	لا	`string`
`before`	query	لا	`string`
`limit`	query	لا	`number`

الاستجابات

الحالة	الوصف
`200`	A page of the reaction's recorded runs, newest first

`GET /v1/reaction/{entity}/count`

`count`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Reaction model count

`GET /v1/reactions`

`findAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of all Reaction model instances

`GET /v1/reactions/{id}`

`findOneById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Reaction model instance by ID

`GET /v1/reactions/count`

`countAll`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Count of all Reaction model instances

`Since 1.8` `GET /v1/reactions/custom-actions`

`findCustomActionCandidates`

المعاملات

الاسم	الموضع	مطلوب	النوع
`moduleId`	query	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Custom actions callable from the given module via Execute Custom Action

`Since 1.8` `GET /v1/reactions/module/{moduleId}`

`findByModule`

المعاملات

الاسم	الموضع	مطلوب	النوع
`moduleId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Array of Reaction model instances defined on entities within the module

ReactionLog — reaction-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}` ؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `reaction-service.json` · 2 · 0.51.5 reaction عملية

`Since 1.10` `GET /v1/reaction-logs`

`find`

المعاملات

الاسم	الموضع	مطلوب	النوع
`reactionId`	query	لا	`string`
`eventName`	query	لا	`string`
`actionType`	query	لا	`string`
`status`	query	لا	`string`
`from`	query	لا	`string`
`to`	query	لا	`string`
`doneBy`	query	لا	`string`
`search`	query	لا	`string`
`order`	query	لا	`string`
`skip`	query	لا	`number`
`limit`	query	لا	`number`

الاستجابات

الحالة	الوصف
`200`	A page of recorded reaction runs across the caller's space

`Since 1.10` `GET /v1/reaction-logs/{id}`

`findById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`reactionId`	query	لا	`string`
`eventName`	query	لا	`string`
`actionType`	query	لا	`string`
`status`	query	لا	`string`

الاسم	الموضع	مطلوب	النوع
`from`	query	لا	`string`
`to`	query	لا	`string`
`doneBy`	query	لا	`string`
`search`	query	لا	`string`
`order`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	One recorded reaction run, its full record and its neighbours

SystemReaction — reaction-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `reaction-service.json` · 5 · 0.51.5 reaction عملية

`GET /v1/schema/{entity}/{trigger}`

`getTriggerSchemas`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`trigger`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	–

`GET /v1/schema/{entity}/announcement/{announcementId}/source/{sourceType}`

`getAnnouncementSourceEventSchema`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`announcementId`	path	نعم	`string`
`sourceType`	path	نعم	`number`
`newTriggerId`	query	لا	`string`
`relationId`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	–

`GET /v1/schema/{entity}/event/{eventType}/{newTriggerId}`

`getEventSchema`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`eventType`	path	نعم	`string`
`newTriggerId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	–

`GET /v1/schema/{entity}/event/{sourceType}/{sourceId}/{newTriggerId}/source`

`getSourceEventSchema`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`sourceType`	path	نعم	`number`
`sourceId`	path	نعم	`string`
`newTriggerId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	–

`GET /v1/schema/announcement/{announcementId}`

`getAnnouncementSchema`

المعاملات

الاسم	الموضع	مطلوب	النوع
`announcementId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	–

WhatsAppTemplates — reaction-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

> اللقطة: `reaction-service.json` · 1 · reaction 0.51.5 عملية

`Since 1.10` `GET /v1/whatsapp/templates`

`listTemplates`

الاستجابات

الحالة	الوصف
`200`	Approved WhatsApp templates of the space

Timeline — timeline-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `timeline-service.json` · 7 · 0.0.1 timeline عملية

`GET /v1/posts`

`getPosts`

المعاملات

الاسم	الموضع	مطلوب	النوع
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Post model instance

`GET /v1/posts/{id}`

`getPostsById`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Post model instance

`PATCH /v1/posts/{id}/pin`

`pinnedPost`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `PinnedBody`

الاستجابات

الحالة	الوصف
`204`	pinned/unpinned PATCH success

`PATCH /v1/posts/{id}/publish`

`publishPost`

المعاملات

الاسم	الموضع	مطلوب	النوع
`id`	path	نعم	`string`

جسم الطلب: `PublishBody`

الاستجابات

الحالة	الوصف
`204`	publish/unpublish PATCH success

`GET /v1/posts/count`

`getPostsCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`where`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Post model count

`GET /v1/posts/entities/{entityId}`

`getPostsByEntity`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`
`filter`	query	لا	–

الاستجابات

الحالة	الوصف
`200`	Post model instance

`GET /v1/posts/entities/{entityId}/count`

`getPostsByEntityCount`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entityId`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	Post model count

Workflow — workflow-service

كل استجابة أدناه تصل داخل الغلاف الموحَّد `{success, result}`؛ وتتجاوزه التنزيلات والتدفّقات بنيويًّا. راجع استدعاء واجهة HTTP API.

اللقطة: `workflow-service.json` · 10 · workflow-service 0.0.1 عملية

`POST /v1/entities/{entity}/{workflowId}/bulk`

`bulkExecuteWorkflowAction`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`workflowId`	path	نعم	`string`
`actionId`	path	نعم	`string`

جسم الطلب: `BulkExecuteActionRequestDto`

الاستجابات

الحالة	الوصف
`200`	BulkExecuteWorkflowActionResponseDto model instance

`POST /v1/entities/{entity}/entries/{entry}/{workflowId}/adhoc-step`

`addAdHocStep`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entry`	path	نعم	`string`
`workflowId`	path	نعم	`string`

جسم الطلب: `CreateAdhocWorkflowStepDto`

الاستجابات

الحالة	الوصف
`200`	Return value of WorkflowController.addAdHocStep

`PATCH /v1/entities/{entity}/entries/{entry}/{workflowId}/execute-action`

Execute action of workflow that belongs to an entity — `executeWorkflowAction`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`workflowId`	path	نعم	`string`
`entry`	path	نعم	`string`

جسم الطلب: `ExecuteActionModel`

الاستجابات

الحالة	الوصف
`200`	workflow PATCH success

`PATCH /v1/entities/{entity}/entries/{entry}/workflow/{workflowId}/change-step`

Change current step of a data entry — `changeCurrentStep`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`workflowId`	path	نعم	`string`
`entry`	path	نعم	`string`

جسم الطلب: `ChangeWorkFlowStepDto`

الاستجابات

الحالة	الوصف
`200`	workflow PATCH success

`GET /v1/entities/{entity}/entries/{entryId}/main-flow-status`

`getMainWorkflowStatus`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`entryId`	path	نعم	`string`
`mode`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	–

`GET /v1/entities/{entity}/workflow/{scope}`

`getWorkFlowOwners`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`scope`	path	نعم	`string`

الاستجابات

الحالة	الوصف
`200`	find all workflow owners or original owners of an entity depends on the scope

`GET /v2/entities/{entity}/entries/{entry}/{workflowId}/adhoc-step-owners`

`getAdhocStepOwners`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entry`	path	نعم	`string`
`workflowId`	path	نعم	`string`
`skip`	query	لا	`number`
`limit`	query	لا	`number`
`search`	query	لا	`string`
`ownerType`	query	لا	`number`

الاستجابات

الحالة	الوصف
`200`	Get eligible owners for adhoc step owner selection

`PATCH /v2/entities/{entity}/entries/{entry}/{workflowId}/change-owner`

Change the workflow owner of a data entry – `changeCurrentStepOwnerV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`workflowId`	path	نعم	`string`
`entry`	path	نعم	`string`

جسم الطلب: `ChangeStepOwnerDto`

الاستجابات

الحالة	الوصف
`200`	workflow PATCH success

`GET /v2/entities/{entity}/entries/{entry}/{workflowId}/eligible-owners`

`getEligibleOwners`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entry`	path	نعم	`string`
`workflowId`	path	نعم	`string`
`skip`	query	لا	`number`
`limit`	query	لا	`number`
`search`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	Get eligible owners for change owner dialog

`GET /v2/entities/{entity}/entries/{entryId}/main-flow-status`

`getMainWorkflowStatusV2`

المعاملات

الاسم	الموضع	مطلوب	النوع
`entity`	path	نعم	`string`
`entryId`	path	نعم	`string`

الاسم	الموضع	مطلوب	النوع
`mode`	query	لا	`string`

الاستجابات

الحالة	الوصف
`200`	–

الطلبات والخدمات (الواجهة البرمجية والبنية الداخلية)

تتوجّه هذه الصفحة إلى فرق التكامل والمطوّرين الذين يقرؤون طلبات الخدمات أو ينفّذون إجراءات عليها من خارج البوابة، أو يحتاجون إلى فهم دقيق لأسباب تمكّن مستخدم من رؤية طلب أو تعذّر ذلك عليه. وتغطّي الصفحة نموذج البيانات، وقواعد الوصول في كل نمط، ونقطة نهاية الوصول الوحيدة، وقوائم النظام التي تستند إليها مناطق الطلبات في البوابة، وطريقة حساب **حالة الطلب** التي يراها المستخدم.

وتجد الميزات نفسها من منظور المستخدم النهائي في الخدمات وتقديم الطلبات [استخدام أسس]، أما تصميم سير عمل الخدمة فمشروح في سير العمل (التصميم) [بناء التطبيقات › سير العمل والاعتمادات].

النموذج

الخدمة كيان بيانات من النوع `Request` تكون فيه قيمة الراية `meta.activated` هي `true`، ولا تظهر في البوابة إلا الخدمات المفعّلة. ويُنشئ تقديم الخدمة **طلبًا**، أي مُدخلًا في ذلك الكيان يرتبط به سير عمل.

عند إنشاء المُدخل، تشغّل المنصة سير العمل النشط للكيان بدءًا من خطوته الأولى. ويجب أن يبدأ كل سير عمل بخطوة **البداية** (Entry)، ويكون منشئ المُدخل دائمًا مالك خطوة البداية في سير العمل الرئيسي. لذلك يبقى الطلب المحفوظ حديثًا في خطوة مقدّمه نفسه، بسجل تاريخي للخطوات فارغ، إلى أن ينفّذ مقدّم الطلب إجراء تلك الخطوة (الذي يسمّيه المصمّم، وغالبًا ما يكون *تقديم*). ويُعيد المعتمِد الذي «يُعيد» الطلب بإجراء *الخطوة السابقة* النسبي، أو بإجراء المدير *العودة إلى خطوة سابقة*، الطلبَ إلى خطوة البداية تلك، أي إلى منشئه. أما داخل التدفق الفرعي، فتستمد خطوة البداية مالكَها من خطوة التدفق الفرعي في سير العمل الأب.

ولكل طلب أيضًا **مُدخل تتبّع** خفيف في كيان النظام `REQUEST` (انظر مُدخل تتبّع الطلب)، وتقرأ قوائم الطلبات في البوابة هذه المُدخلات، لا المُدخلات الأصلية.

أنماط الوصول

توزّع البوابة الوصول إلى الطلبات على أربعة أنماط، لكلٍّ منها منطقة في البوابة، وفحص وصول خاص به، ومعارض ضمنية يولّدها النظام.

النمط (`RequestAccessMode` Enum)	منطقة البوابة (التسمية في الواجهة)	المسار	فحص الوصول	معرض التفاصيل (`ViewImplicitType`)	معارض القوائم
`New`	دليل الخدمات (Services Catalog)	`/s`	`canCreateRequest`	`MY_REQUEST_DETAILS`	المعرض `SERVICE_CATALOG` في الكيان `SERVICES`
`MyRequest`	الصادر (Outbox)	`/my`	`canAccessMyRequest`	`MY_REQUEST_DETAILS`	`ONGOING_MY_REQUESTS`، `COMPLETED_MY_REQUESTS`
`Inbox`	الوارد (Inbox)	`/inbox`	`canAccessInbox`	`INBOX`	`ONGOING_INBOX`، `COMPLETED_INBOX`
`Observer`	المتابعة (Watching)	`/rm`	`canAccessObservers`	`OBSERVER`	`ONGOING_OBSERVER`، `COMPLETED_OBSERVER`

إجراءات الخدمة الأربعة في `ServiceActionTypes` هي `submit` و`request` و`inbox` و`observe`، ويقابل كلًّا منها نمطٌ إذن تفويض بالصيغة `so:service:{serviceId}:{action}` (انظر التفويض).

*ما زالت الصفحات الأقدم ومفاتيح المسارات تسمّي الصادرَ «My Requests» (طلباتي) والمتابعةَ «Request Monitoring» (مراقبة الطلبات). أما تسميتاهما في الواجهة فهما **الصادر** (Outbox) و**المتابعة** (Watching)، وتُسمّى قائمة الخدمات **دليل الخدمات** (Services Catalog).*

نقطة نهاية الوصول

يوفّر `RequestController` مسارًا واحدًا يتطلب المصادقة:

```
GET /v1/requests/{entity}/{id}/{mode}
```

- `entity`: معرّف كيان الخدمة.
- `id`: معرّف مُدخل الطلب، أو العنصر النائب للطلب الجديد.
- `mode`: قيمة من `RequestAccessModeEnum`، وهي `New` أو `MyRequest` أو `Inbox` أو `Observer`.

يتطلب المسار استراتيجية المصادقة `Auth2`، ويحمل المعالِج `@authorize.skip()`، إذ تحسم فحوص الأنماط المبيّنة أدناه حق الوصول، لا المُخوِّل العام. ويُعيد المسار المعرض الضمني (`DataViews`) الذي يحق للمستدعي الوصول إليه في ذلك النمط، أو `null` إن لم يكن له حق الوصول:

النمط	الفحص المُنفَّذ	المعرض المُعاد
`New`	`canCreateRequest` (الجمهور)	`MY_REQUEST_DETAILS`
`MyRequest`	`canAccessMyRequest` (أنت منشئه)	`MY_REQUEST_DETAILS`
`Inbox`	`canAccessInbox` (المالك أو المجموعة أو السجل التاريخي)	`INBOX`
`Observer`	`canAccessObservers` (قوائم المتابِعين)	`OBSERVER`

يُبحث عن كل معرض ضمني باستخدام الشرط `{dataEntity, isPublished: true, isImplicit: true, implicitType}` (وقيم `ViewImplicitType` الخاصة بالطلبات هي `INBOX` و`MY_REQUEST_DETAILS` و`OBSERVER`، إضافةً إلى `NONE`). وإن لم يكن الكيان المستهدف من النوع `Request`، أطلقت نقطة النهاية الاستثناء `EntityNotOfTypeRequestException`.

تستدعي البوابة نقطة النهاية هذه (باسم `FetchServiceImplicitViewForRequest`) في موضعين:

- في صفحة الخدمة، لا تُنشئ زر **طلب جديد** (`action.NEW_REQUEST`) إلا إذا أعاد الاستدعاء بالنمط `New` معرضًا؛ فإن لم يُعِد معرضًا، لم يظهر الزر.
- في كل صفحة طلب، للتحقق من السماح بالوصول إلى المسار: فإن لم يُعَد معرض، يُحوَّل المستخدم إلى `/unauthorized`، باستثناء الطلب المفتوح من الدليل، إذ يعود المستخدم حينها إلى صفحة الخدمة.

من يصل إلى الطلب

التقديم: `canCreateRequest`

حين تبني البوابة الكائنات التي يستطيع المستخدم التنقّل إليها، تستدعي `getUserServicesCanAccess`، فتُرشّح الخدمات المفعّلة لتُبقي منها ما يسمح به `canCreateRequest`، وتطبّق القواعد التالية:

1. إن لم تكن الخدمة `activated` أو لم يكن لها `audience`، أُخفيت.

2. إن كان الوصول المجهول مفعّلًا (ولم يكن المستدعي يعمل بموجب تفويض): تُخفى الخدمة **المجهولة فقط** (المجهول مفعّل، والمصادَق عليه *غير* مفعّل) عن المستخدمين المسجّلين، وتُعرض في ما عدا ذلك.
3. يتطلب أي مسار مصادَق عليه مستخدمًا مسجّل الدخول (`sub`).
4. إن كان الجمهور المصادَق عليه **مقيّدًا**، فيجب أن يكون المستخدم مُدرَجًا في `entitledEntries`، إما مباشرةً بمعرّفه وإما عبر إحدى مجموعاته.
5. عند العمل بموجب تفويض، تُرشَّح النتيجة بعد ذلك وفق نطاق التفويض الخاص بالإجراء `submit` .

ولأن الخادم يطبّق هذه القواعد كلها، فلا تظهر في الدليل أي خدمة لا يستطيع المستخدم تقديم طلب لها، ولا يوجد مسار لـ«طلب الوصول».

الصادر: `canAccessMyRequest`

يُسمح بالوصول إذا كان المستدعي هو المنشئ (`creationInfo.doneBy` يساوي معرّفه)، أو إذا منحته قاعدة الوصول المجهول أدناه حق الوصول. وعند العمل بموجب تفويض، يحسم نطاق التفويض الخاص بالإجراء `request` القرار.

الوصول المجهول إلى الطلب (`isAnonymouslyAccessibleRequest`): لا يجوز للمستدعي غير المصادَق عليه فتح طلب إلا إذا أُنشئ الطلب دون تسجيل دخول (أي لا يحدّد `creationInfo.doneBy` فيه أي مستخدم؛ وإن غابت البيانات الوصفية رُفض الوصول)، وما دامت خطوات سير العمل الحالية **كلها** بلا مالك، أي قبل تسليم الطلب إلى موظف يتولّى ملكيته. ولا يستوفي الطلب الذي ليست له خطوات حالية هذا الشرط، كما لا يستفيد المستدعي المصادَق عليه من هذه القاعدة أبدًا.

الوارد: `canAccessInbox`

يُستبعد المنشئ دائمًا: فلا يصل مقدّم الطلب إلى طلبه عبر الوارد أبدًا، حتى حين يكون مالك خطوته الحالية (بعد إعادته إليه مثلًا)، وإنما يصل إليه عبر `MyRequest`، حيث تظهر حالة الطلب *بانتظار إجرائكم*.

أما لغيره، فيكفي تحقّق أيٍّ مما يلي:

- أن يكون المستخدم **مالك** الخطوة الحالية (`owner.dataEntry`)، أو أن يكون ضمن قائمة المالكين الأصليين للخطوة الحالية (بصيغة المصفوفة).
- أن ينتمي المستخدم إلى **مجموعة** تملك أصلًا خطوة حالية، وأن تكون تلك الخطوة غير مُسنَدة أو خطوة «لجميع المستخدمين» (`accessPermission === AllUsers`).
- أن ينتمي المستخدم إلى **تجميعة** تملك الخطوة أصلًا، بالشرط نفسه (غير مُسنَدة أو لجميع المستخدمين).
- أن يكون المستخدم قد **نفّذ** خطوة سابقة في السجل التاريخي للطلب (`executedBy`)، أو **كان مالكًا** لخطوة سابقة.
- أن يكون المستخدم ضمن **مجموعة مالكين** لخطوة لجميع المستخدمين في السجل التاريخي.

وعند العمل بموجب تفويض، يحسم نطاق التفويض الخاص بالإجراء `inbox` القرار.

المتابعة: `canAccessObservers`

يُسمح بالوصول إذا كان المستخدم مُدرَجًا في `meta.observers.users` للخدمة، أو منتميًا إلى مجموعة مُدرَجة في `meta.observers.groups` . ولا تُظهر البوابة منطقة المتابعة إلا للخدمات التي يستطيع المستخدم متابعتها (`getUserServicesCanObserve`)، ولا يعرضها الشريط الجانبي إلا إذا كان لدى المستخدم طلبات متابَعة جارية أو مكتملة. وعند العمل بموجب تفويض، يحسم نطاق التفويض الخاص بالإجراء `observe` القرار.

من يستطيع تعديل الطلب

لمسار التعديل (`/s/:serviceId/r/:requestId/edit`) آلية خاصة للتحقق من السماح بالوصول: فإن كان للطلب سير عمل، وجب أن يكون المستخدم مالك الخطوة الحالية، أو قادرًا على تولّي ملكية الخطوة، أو قادرًا على تنفيذ إجراء سير العمل في الخطوة (ويشمل ذلك أعضاء المجموعة التي تملك خطوة متوازية)، أو حاصلًا على منحة *التعديل كمسؤول* في المعرض، وإلا حُوِّل إلى صفحة الطلب للقراءة فقط. وبذلك لا يستطيع مقدّم الطلب تعديل طلبه إلا ما دام الطلب في خطوته.

نموذج جمهور الخدمة (`ServiceMeta`)

الحقل	النوع	المعنى
`audience.anonymous.enabled`	boolean	يجوز للمستخدمين المجهولين التقديم
`audience.authenticated.enabled`	boolean	يجوز للمستخدمين المسجّلين التقديم
`audience.authenticated.isRestricted`	boolean	التقييد بقائمة محدّدة
`audience.authenticated.entitledEntries`	array	المستخدمون والمجموعات المسموح لهم عند التقييد
`activated`	boolean	الخدمة مفعّلة (شرط لإدراجها)
`category`	string	تصنيف الدليل
`onBehalf`	string	مسار الحقل الذي تُتّخذ قيمته موضوعًا للطلب
`observers.groups` / `observers.users`	refs	من يجوز له متابعة الطلبات
`showServiceForEveryone`	boolean	مفتاح ظهور الخدمة في الدليل
`showServiceForAuthenticatedUsersOnly`	boolean	قصر ظهور الخدمة في الدليل على المستخدمين المسجّلين

موضوع التقديم بالنيابة. عند بناء مُدخل التتبّع، يُقرأ الطرف «المقدَّم بالنيابة عنه» من الحقل الذي يحدّده `onBehalf` في البيانات المقدَّمة (`meta.onBehalf`)، فإن كان الحقل فارغًا، استُخدم منشئ الطلب بدلًا منه؛ وهذا ما تعرضه البوابة بوصفه موضوع الطلب.

النماذج المجهولة والتحقق البشري. يمكن حماية الخدمة التي يسمح جمهورها بالوصول المجهول بفحص بشري قائم على إثبات العمل، مع ضبطه لكل معرض على حدة. وهذا الفحص معطّل افتراضيًا، ويُستخدم كل تحدٍّ فيه مرة واحدة، ولا يُطلب أبدًا من المستخدمين المسجّلين. وإن تعذّر الوصول إلى خدمة التحقق، عومل المستدعي على أنه غير متحقَّق منه ورُفض الإرسال إلى النموذج الذي يتطلب التحقق. وتجد مواصفات التخاطب اللازمة لتشغيل نموذج كهذا دون واجهة في كتابة البيانات، وطريقة تفعيله في التحقق البشري [الإدارة › التحقق البشري (الحماية من الروبوتات)].

قوائم الطلبات

تُعَدّ تبويبات الصادر والوارد والمتابعة في البوابة معارض ضمنية على الكيان `REQUEST` ، يستند كلٌّ منها إلى استعلام ضمني يُدرَج عند التثبيت. ويوضّح استعلامان منها أهم القواعد:

- **الصادر الجاري** (`ONGOING_MY_REQUESTS_QUERY`): `creationInfo.doneBy = @me` و `status.id != COMPLETED` . لذلك يبقى هنا الطلب الذي ينتظر إجراءً من منشئه، بالحالة *بانتظار إجرائكم* (أو *مسودة* إن لم يُرسَل قط).
- **الوارد الجاري** (`ONGOING_INBOX_REQUESTS_QUERY`): `creationInfo.doneBy != @me` والخطوة الحالية يملكها `@me` ، أو هي بلا مالك أو لجميع المستخدمين ومالكها الأصلي ضمن `@myGroups` أو `@myCollections` ، أو لها مصفوفة مالكين أصليين تتضمن `@me` .

تختلف **المسودات** عن الحالة *مسودة*: فالنموذج المحفوظ بزر **حفظ كمسودة** مُدخل في الحالة `Draft`، ليس له سير عمل بعد، ولا يُدرَج إلا عبر مفتاح المسودات في تبويب الجارية. ولا يظهر هذا الخيار للطلب الجديد (أو لمسودة قائمة) إلا إذا كان الإعداد `editConfiguration.allowCreateEntryAsDraft` مفعّلًا في المعرض.

الأعداد الحيّة. يشترك الدليل في قناة مقبس (`subscribeToModifiedRequestsCounts`) ويحدّث أعداد الطلبات الجارية والمكتملة كلما تغيّرت الطلبات، وتتلقى القوائم الثلاث تغييرات الخطوات والحالات عبر الاتصال اللحظي نفسه. أما حالة الخدمات في الشريط الجانبي فتبيّن أعداد الطلبات الجارية، وما إذا كانت كل منطقة تضم طلبات جارية **أو** مكتملة، وهذا ما يحدّد ظهور عناصر الصادر والوارد والمتابعة.

ترشيح القوائم عند العمل بموجب تفويض. ما دام المستخدم يعمل بموجب تفويض، يعيد معترِض HTTP كتابة الاستعلام في دليل الخدمات وفي نقاط نهاية المُدخلات والأعداد للصادر والوارد والمتابعة. انظر التفويض.

مُدخل تتبّع الطلب

يُمثَّل كل طلب جارٍ أو مكتمل بمُدخل في الكيان `REQUEST` من النوع `RequestDataEntryValue`، يبنيه `RequestService.buildRequestEntry`، ويشير هذا المُدخل إلى التقديم الأصلي ويحمل الحقول التي تعرضها قوائم البوابة:

- `requestNumber`، و`reference` (الكيان والمُدخل الأصليان)، و`serviceReference`.
- `service` (الاسم والوصف والأيقونة والتصنيف) و`observers`.
- `currentStep` (الاسم، والمالك، والمالك الأصلي، ووقت الاستلام، وتعريف المالك).
- `onBehalf`، و`creationInfo`، و`updateInfo`، و`entityVersion`.
- `lifeCycleStatus` (`LifeCycleStatusEnum`: `Active` (0، قيد التنفيذ)، أو `Completed` (1، انتهى بصورة طبيعية)، أو `Failed` (-1)، أو غير معرَّفة (ليس ضمن سير عمل).
- `isInvalid` / `invalidReason` (`DataEntryInvalidReasonEnum`: `Obsolete` = 0، `Invalid` = 1).
- البيانات الوصفية لسير العمل: `_metaCurrentSteps`، و`_metaWorkflowHistory`، و`_workFlowStepsHistory`، و`_metaAdhocSteps`، و`_metaOrder`.

إن لم يكن للطلب سير عمل نشط، أو تعذّر العثور على مُدخل خدمة يستند إليه، أعاد `buildRequestEntry` القيمة `null` ولم يُنشأ مُدخل تتبّع. ويُعاد بناء المُدخل من المُدخل بحالته المحفوظة حاليًا، لا من لقطة الحدث، فلا يمكن للتحديثات المتزامنة أن تُعيد صفًّا في القائمة إلى خطوة أقدم.

تُؤخذ الخطوة الحالية والسجل التاريخي للخطوات من نسخة سير العمل **النشطة** (`metaCurrentStep.workflowId` / `instanceId`)؛ لذلك يُظهر الطلب الذي يشغّل تدفقًا فرعيًا متداخلًا خطوةَ التدفق الفرعي ومالكها، لا خطوة الأب.

حالة الطلب المحسوبة

الحقل `status` (*حالة الطلب*) حقل تصنيف ضمني افتراضي، تُحسب قيمته لكل مستخدم يستعرض الطلب على حدة، وتُقيَّم الفروع بالترتيب التالي:

`status.id`	التسمية	الشرط
`COMPLETED`	مكتمل (Completed)	قيمة `lifeCycleStatus` هي `Completed` (1) أو `Failed` (-1)
`OBSOLETE`	بالية (Obsolete)	قيمة `invalidReason` هي `Obsolete` (0)

status.id	التسمية	الشرط
DRAFT	مسودة (Draft)	لا توجد خطوات منفَّذة بعد في السجل التاريخي لسير العمل الأول
PENDING	بانتظار إجرائكم (Pending Your Action)	قيمة lifeCycleStatus هي Active ومالك الخطوة النشطة هو المستخدم الذي يستعرض الطلب، أو الخطوة بلا مالك أو لجميع المستخدمين وهذا المستخدم ضمن مالكيها الأصليين (مباشرةً أو عبر مجموعة أو تجميعة)
ON_GOING	قيد التنفيذ (On Going)	قيمة lifeCycleStatus هي Active
NONE	لاشيء (None)	أي حالة أخرى

ويعتمد تقسيم القوائم إلى جارية ومكتملة على قيمة status.id : فالمكتملة تضم COMPLETED ، والجارية تضم كل ما عداها.

كانت الوثائق السابقة تذكر حالات الطلب: Draft Openg Completedg OnHoldg، وقيم النتيجة: Succeeded Failedg Rejectedg Timedoutg Cancelledg Obsoleteg. ولا وجود لأيٍّ من هذه القيم في الرماز الحالي، والجدول أعلاه هو المجموعة المعمول بها.

الإشعارات حين يصل الطلب إلى أحدهم

عند إنشاء كيان خدمة أو تحديثه، تضيف خدمة التفاعلات إليه تفاعلًا ضمنيًا: عند وقوع الحدث WorkflowStepReached ، يُرسَل إشعار داخل التطبيق إلى مالك الخطوة الحالية، عنوانه **إجراء مطلوب** ونصّه *لديك طلب خدمة جديد* (نوع الإشعار INCOMING_REQUEST_RECEIVED). وبذلك يعلم مقدّم الطلب أن طلبه أُعيد إلى خطوته، ويعلم المعتمِد أن طلبًا وصل إليه. أما إرسال البريد الإلكتروني أو الرسائل النصية عند وقوع الأحداث نفسها، فتتولّاه تفاعلات منفصلة يضيفها المصمّم.

التفويض

حين يعمل المستخدم بموجب **تفويض**، يحسم نطاقُ التفويض الوصولَ إلى التقديم والصادر والوارد والمتابعة بدلًا من قواعد الملكية المعتادة:

- **النطاق** إما ALLOW_ALL وإما DENY_ALL .
- **الاستثناءات** أنماط تحدّد الحالات المستثناة من النطاق؛ والاستثناءات العامة هي so:all و so:app و so:service ، والاستثناء الخاص بخدمة وإجراء بعينهما هو so:service:{serviceId}:{action} ، حيث الإجراء أحد submit أو request أو inbox أو observe .
- في نطاق ALLOW_ALL ، يؤدي الاستثناء المطابق إلى **منع** الإجراء، وفي نطاق DENY_ALL ، يؤدي إلى **السماح** به.

ويُرشّح المنطق نفسه القوائم التي تُجلب أثناء التفويض، إذ يعيد معترِض HTTP كتابة الاستعلام في دليل الخدمات وفي نقاط نهاية المُدخلات والأعداد للصادر والوارد والمتابعة: ففي نطاق ALLOW_ALL ، تُستبعد الخدمات المستثناة (nin)، وفي نطاق DENY_ALL ، لا تبقى إلا الخدمات المستثناة (inq). ويجري ذلك كله من جهة الخادم. أما نموذج التفويض الكامل ففي التفويض [الإدارة].

مسارات البوابة

المسار	الغرض
/s	دليل الخدمات
/s/:serviceId	صفحة الخدمة (التفاصيل وزر **طلب جديد**)
/s/:serviceId/r/new	نموذج الطلب الجديد

المسار	الغرض
`/s/:serviceId/r/:requestId`	عرض طلب
`/s/:serviceId/r/:requestId/edit`	تعديل طلب (يخضع للتحقق من السماح بالوصول، انظر من يستطيع تعديل الطلب)
`/my`	الصادر (الجارية / المكتملة)
`/inbox`	الوارد (الجارية / المكتملة)
`/rm`	المتابعة (الجارية / المكتملة)

تستخدم صفحات الطلبات المفتوحة من قائمة بادئةَ تلك القائمة، مثل `/my/:serviceId/r/:requestId` ، وتُفتح المسودة بمعامل الاستعلام `?d=true` .

يجلب دليل الخدمات كل المُدخلات التي يحق للمستخدم الوصول إليها في استدعاء واحد (تُضبط `limit` على أكبر عدد صحيح آمن، وتُثبَّت `activePage` على `0`)، لذلك لا تُقسَّم نتائجه إلى صفحات، ولا تُعرض فيه إلا مُدخلات الخدمات الصالحة غير البالية.

الخبء

تُخبَّأ مجموعة الخدمات المفعّلة مرة واحدة لكل قاطن تحت مفتاح الخبء `SERVICES` ، بمهلة انتهاء الخبء الافتراضية، فلا تعيد عمليات تحميل الدليل المتكررة الاستعلام عن كل خدمة. ولا تعتمد القيمة على هوية المستدعي، إذ يجري الترشيح الخاص بكل مستخدم (`canCreateRequest` ، `canAccessObservers`) على القائمة المخبوءة بعد ذلك. ويؤدي أي تغيير في كيان خدمة مفعّلة إلى حذف المفتاح. (كانت الإصدارات الأقدم تخبّئ نسخة لكل مستخدم، تحت مفتاح يتضمن معرّف المستخدم.)

سجلّ تغيّر السلوك

تجد ملخص هذه التغييرات في الإصدارات [عام]، وتبقى تفاصيلها هنا لأنها تفسّر سلوكًا قد يصادفه فريق التكامل في عمليات النشر الأقدم.

- **مُدخلات تتبّع تراعي التدفق الفرعي** `Since 1.4` . يأخذ مُدخل التتبّع الخطوة الحالية والسجل التاريخي للخطوات من نسخة سير العمل النشطة، فتُظهر الطلبات الجارية في تدفق فرعي متداخل الخطوة والمالك الصحيحين، لا خطوة الأب ومالكها.
- **تصحيحات قوائم الطلبات** `Since 1.4` . أُتيحت هذه التصحيحات أيضًا كتحديثات لسلاسل الإصدارات 1.4 و1.6 و1.8 (بدءًا من 1.4)، وهي موجودة في 1.10:
  - يطابق تبويب المكتملة في الوارد المستخدمَ مع السجل التاريخي *لكل* سير عمل في الطلب، لا سير العمل الذي انتهى فيه وحده. وكان الطلب يختفي من تبويب المكتملة لدى المعتمِد في تدفق فرعي إذا اكتمل لاحقًا في سير عمله الرئيسي. وثمة قيد معروف: لا يزال الطلب لا يظهر هناك لعضو المجموعة التي *كان* بإمكانها التصرّف في خطوة مملوكة لها، إذا لم يتصرّف ذلك العضو فيها.
  - تُصحَّح مُدخلات التتبّع التي لم تعد خطوتها الحالية أو حالة انتهائها مطابقة للطلب، ويبقى الطلب الذي أُعيد ضبطه أو أُنشئ للتو متوافقًا مع سجلّه.
  - حين يتغيّر متابِعو خدمة ما، تُحدَّث طلباتها القائمة أيضًا؛ فيرى المتابِع الجديد الطلبات السابقة في المتابعة، ويتوقف المتابِع المُزال عن رؤيتها.
  - تظهر عناصر المتابعة والصادر والوارد في الشريط الجانبي حين تضم المنطقة طلبات جارية **أو** مكتملة. وكانت المنطقة التي اكتملت كل طلباتها قد تختفي سابقًا، لأن عدّاد المكتملة فيها كان يعيد استخدام قائمة الجارية.
  - يستطيع مقدّمو الطلبات المجهولون فتح تفاصيل طلبهم بعد تقديمه.

- **أرشفة الخدمة تمتد إلى طلباتها** `Since 1.8`. تؤدي أرشفة خدمة من دليل الخدمات في الإدارة إلى أرشفة كل طلب نشط سبق تقديمه لها، فتختفي هذه الطلبات من الصادر والوارد والمتابعة دفعة واحدة، وتؤدي استعادة الخدمة إلى استعادتها. وقبل 1.8، كانت أرشفة الخدمة أو استعادتها لا تغيّر إلا إدراجها في الدليل، وتبقى الطلبات المقدَّمة ظاهرة أيًّا كانت حالة الخدمة.
- **الحقول الضمنية للطلب تُصلح نفسها** `Since 1.8`. إن فقد إعداد حقول كيان من النوع `Request` حقلَه الضمني `requestNumber` أو `status` (بسبب بيانات قديمة، أو تحديث يجريه النظام متجاوزًا التحقق من الحقول الضمنية)، أضافت المنصة الحقول المفقودة عند الحفظ التالي، سواء أكان إنشاءً أم تحديثًا. وقبل 1.8، كان الإصلاح يعمل عند الإنشاء وحده، فيبقى الكيان الذي فقد هذه الحقول معطوبًا.

أمان الحساب والتحقق الإضافي (التفاصيل الداخلية)

هذه الصفحة موجّهة إلى المُكاملين والمطوّرين الذين يحتاجون إلى فهم السلوك الدقيق الذي يحكم إعدادات أمان المستخدم وكشف القيم المحمية: أيّ طريقة يختارها الخادم، وما الذي يُفرَض وأين، وما الذي تفعله بوابة الويب وتطبيق الجوال قبل كل استدعاء وبعده.

أما منظور المستخدم النهائي فيرد في أمان الحساب وكشف القيم المحمية [استخدام أسس]. وتُوصَف آلية التحقق الإضافي نفسها (التحدي، والتحقق، وتخزين الجلسة، والأغراض الثلاثة `Unmask` و`WorkflowAction` و`Decrypt`) في الحقول الحسّاسة – جلسة الكشف بالتحقق الإضافي [بناء التطبيقات › حماية حقل] وثلاثة أغراض وآلية واحدة [بناء التطبيقات › حماية حقل]. أما إرسال رموز OTP، وحدود المعدّل، ومفاتيح المرور من جانب الخادم، فترد تفاصيلها في خادم الهوية.

أين توجد الإعدادات

- **الويب:** صفحة **تفضيلات المستخدم** على `/user/profile`، بأقسامها: الملف الشخصي، والحساب، والأمان (كلمة المرور، والمصادقة الثنائية، ومفاتيح المرور)، وتسجيلات الدخول (الجلسات، والمحاولات). أما المسار السابق `/user/security` فأصبح يعيد التوجيه إلى `/user/profile`، لتظل الإشارات المرجعية القديمة تصل إلى وجهتها.
- **الجوال: مركز المستخدم**، بتبوياته: الملف الشخصي · الأمان · تسجيلات الدخول `Since 1.10`. ولم تعد ورقة خيارات المستخدم تتضمّن عنصرًا مستقلًا للأمان؛ ويقود رابط فتح شاشة الأمان في معالج الكشف إلى تبويب الأمان.

يستدعي العميلان واجهة برمجة التطبيقات نفسها لخدمة الهوية، لذا تظهر في أحدهما أي طريقة أو مفتاح مرور يُعَدّ في الآخر. وفي الحساب الاتحادي (SSO)، يحلّ في قسم الأمان محلّ بطاقة مفاتيح المرور النصُّ «… عبر موفّر الهوية الخاص بمؤسستك».

طرق التحقق

الطريقة	كيف تظهر	هل يستطيع المستخدم إزالتها
تطبيق المصادقة (TOTP) `Since 1.8`	يسجّله المستخدم بنفسه في قسم الأمان.	نعم، وهي الطريقة الوحيدة التي يستطيع المستخدم إزالتها.
مفتاح المرور	يُسجَّل لكل جهاز في قسم الأمان.	يُلغى كل مفتاح على حدة من بطاقة مفاتيح المرور، لا عبر إجراء الإزالة العام.
الرسائل القصيرة / البريد الإلكتروني	توفّرهما المؤسسة باستخدام رقم الهاتف والبريد الإلكتروني المسجّلين في الحساب، حين يفعّلهما النطاق.	لا.
واتساب `Since 1.10`	يظهر حين تكون المؤسسة قد فعّلت واتساب **و**يكون للحساب رقم هاتف موثَّق.	لا، إذ يظهر ويختفي بحسب الإعداد ورقم الهاتف الموثَّق.
كلمة المرور	لا يختارها المستخدم أبدًا. راجع بديل كلمة المرور.	–

و**الخادم** هو الذي يحدّد الطرق المتاحة: TOTP بعد تسجيله، والرسائل القصيرة والبريد الإلكتروني متى فعّلهما النطاق، وواتساب وفق الشروط السابقة، ومفتاح المرور ما دام هناك مفتاح مسجَّل واحد على الأقل. وحين يُلغى آخر مفتاح مرور، يزيل العميل خيار مفتاح المرور فورًا، قبل أن تتحدّث حالة الخادم المحفوظة في الخبء.

وتكون إحدى الطرق هي **المفضّلة (الافتراضية)**، وتعرضها بطاقة الويب مع تاريخ «مفعّل منذ»؛ أما كل طريقة متاحة أخرى فتعرض **تعيين كافتراضي**.

التسجيل

يمسح المستخدم رمز QR (أو يُدخل السرّ) ويؤكّد التسجيل برمز من 6 أرقام. ويطلب معالج الويب كلمة مرور الحساب أولًا. وسرّ الإعداد قصير الصلاحية (يحدّد تطبيق الجوال مدته بعشر دقائق)؛ فإن انتهت صلاحيته قبل التأكيد، يبدأ العميل العملية من جديد بسرّ جديد ورمز QR جديد بدل الإبلاغ عن رمز خاطئ، إذ لم يعد هناك سرّ صالح للتحقق من الرمز بمقارنته به.

صفوف مفاتيح المرور

يعرض كل مفتاح مرور تسمية الجهاز وأيقونة نوعه (هاتف، أو حاسوب محمول، أو مفتاح أمان)، ولا تؤدّي الأيقونة سوى وظيفة **تلميح عرض**. وتُظهر الصفوف وقت إنشاء المفتاح وآخر استخدام له، بالتوقيت المحلي. ولم تعد البطاقة تفتح على الويب نافذة تفاصيل مفتاح المرور التي كانت تعرض عدد مرات تسجيل الدخول. ويعكس عدّاد الخادم عمليات تسجيل الدخول الفعلية، فلا يُبلَّغ عن مفتاح مرور متزامن عبر الأجهزة بأنه غير مستخدَم.

ترتيب واتساب `Since 1.10`

تكون الأولوية للطريقة الافتراضية التي حفظها المستخدم. فإن لم توجد، وظهرت الرسائل القصيرة وواتساب كلتاهما، ظهرت قناة الهاتف التي تفضّلها المؤسسة أولًا. وبمجرد إتاحة واتساب لمستخدم، يُحتسب طريقةً للمصادقة متعددة العوامل (MFA)، فيتطلّب تسجيل الدخول خطوة ثانية حتى إن لم يكن النطاق يتطلّبها من قبل. وتتخطّى قائمة الأمان في الجوال أي طريقة لا يتعرّف إليها التطبيق، ومنها حاليًا واتساب. ويُبلَّغ عن حالات إخفاق الإرسال بالرمز `MFA_WHATSAPP_SEND_FAILED`. وترد تفاصيل إعداد مشغّل المنصة في: رموز واتساب.

الرموز الاحتياطية `Since 1.10`

- يولّد تأكيد تسجيل TOTP **8** **رموز أحادية الاستخدام** ويُعيدها **مرة واحدة**، وبعد ذلك لا يتاح إلا ملخّص للحالة (متى وُلّدت، وكم تبقّى منها).
- **إعادة التوليد** تستبدل المجموعة بأكملها؛ فيتوقّف كل رمز غير مستخدَم من المجموعة القديمة عن العمل عند إصدار المجموعة الجديدة. وتتطلّب هذه العملية إثباتًا جديدًا (رمز TOTP صالح حاليًا أو كلمة مرور الحساب). ولا يكفي رمز الوصول وحده، فلا تستطيع جلسة مخترَقة استبدال الرموز الاحتياطية خفيةً.
- لا يمكن أن ينجح طلبا إعادة توليد متزامنان معًا: ينجح أحدهما ويُبطل الرموز التي كان الآخر سيُعيدها. ويعرض تطبيق الجوال للطلب الذي لم ينجح الرسالة «تم إعادة توليد الرموز للتو من مكان آخر. حاول مرة أخرى.»
- يمسح تعطيل TOTP الرموز الاحتياطية أيضًا.
- على الجوال، يفتح **تحميل** نافذة الحفظ في نظام التشغيل؛ ولا يترتّب على إلغائها أي إجراء.

الجلسات ومحاولات تسجيل الدخول `Since 1.10`

تسرد **الجلسات** كل جهاز أو متصفح مسجَّل الدخول، مع معلومات الجهاز ونظام التشغيل والمتصفح ووقت أول تسجيل دخول وآخر نشاط، وبيان ما إذا كانت الجلسة الحالية.

- إنهاء جلسة **أخرى** يُبطل رموزها على الخادم أولًا ثم يزيل صفّها، فتتوقّف عن العمل فورًا.
- إنهاء الجلسة **الحالية** يؤدّي إلى تسجيل خروج فعلي، لأن الرمز الذي تعمل به الصفحة لم يعد صالحًا.
- إن أخفق الإبطال في أثناء العملية، يبقى الصف، فلا توحي القائمة أبدًا بزوال جلسة لا تزال قابلة للاستخدام.

أما **المحاولات** فهي للقراءة فقط عن قصد: محاولات تسجيل الدخول خلال آخر 60 يومًا، بدءًا بالأحدث، مع الجهاز ونظام التشغيل والمتصفح والطابع الزمني والنتيجة، بما فيها المحاولات الفاشلة التي أجراها آخرون. وهي مقسّمة إلى صفحات

(«عرض المزيد») ولا تعرض موقعًا. وترد تفاصيل مخزنها في مسارات تدقيق المصادقة.

روابط إعادة تعيين كلمة المرور `Since 1.4`

طلب إعادة التعيين **يستبدل** الرابط السابق برابط جديد ومدة صلاحية جديدة (24 ساعة افتراضيًا)؛ فيتوقّف الرابط الأقدم عن العمل، ولا يمدّد تكرار الطلب صلاحية رابط قديم أبدًا. وتتوقّف أيضًا الروابط الصادرة قبل ترقية النشر إلى إصدار يتضمّن هذا السلوك (1.10، أو تحديثات 1.4.118 و1.6.67 و1.8.25). وترد تفاصيل جانب الخادم في: إعادة تعيين كلمة المرور.

ويمكن تسجيل الدخول كالمعتاد من متصفح كانت آخر جلسة فيه لحساب لم يعد موجودًا `Since 1.10` . راجع جلسة تحمل منحة حساب آخر.

كشف القيمة الحسّاسة (Unmask)

يُستخدم المعالج نفسه على الويب والجوال:

1. **المدة.** 2 أو 5 أو 10 أو 30 أو 60 دقيقة، أو 3 أو 4 أو 6 أو 8 ساعات. وتطابق القائمة قائمة السماح في خدمة الهوية تمامًا؛ وتُرفض أي قيمة أخرى بالرمز `INVALID_DURATION` (صحّحت دورة 1.8 قائمة غير متّسقة كانت بعض تطبيقات العميل تعرضها؛ ويدلّ ظهور أي خيارات غير متطابقة على انحراف).
2. **التحدي.** يطلب العميل تحديًا للمدة المختارة، و**الخادم هو الذي يختار الطريقة**: عادةً الطريقة المفضّلة، ثم ينتقل إلى الطرق المتاحة تباعًا إن كانت غير صالحة للاستخدام. وبعد ذلك يتّخذ المستخدم أحد الإجراءات التالية:
   - يُدخل رمزًا: في الرسائل القصيرة أو واتساب أو البريد الإلكتروني، يُرسَل رمز إلى وجهة مقنَّعة؛ وفي TOTP لا يُرسَل شيء. **وطول الرمز هو ما يبلّغ به الخادم** لهذا التحدي؛
   - يُدخل كلمة المرور، إن اختار الخادم بديل كلمة المرور (بلا رمز، ولا إعادة إرسال، ولا تبديل طريقة)؛
   - يبدّل الطريقة، فيُعاد بدء التحدي. **ولا يُعرض مفتاح المرور أبدًا هدفًا للتبديل في الكشف**، على أيٍّ من العميلين؛
   - يعيد إرسال رمز لطريقة تتيح الإرسال، بعد انقضاء مهلة الانتظار التي يبلّغ بها الخادم.
3. **النافذة.** تُعرض القيم غير مقنَّعة طوال المدة المختارة، حتى تنقضي المدة أو يُنهي المستخدم النافذة. ويسري الإنهاء المبكر على الجهاز فورًا، قبل أن يؤكّده الخادم.

وتفتح بوابة الويب المعالج نفسه من زر العين في الحقل المقنَّع (التلميح **الكشف**) ومن قائمة الملف الشخصي (**الكشف / إلغاء الكشف**).

تنبيه النطاق. يشمل الكشف حاليًا **كل** حقل حسّاس يحقّ للمستخدم رؤيته، لا الحقل الذي اختيرت أيقونة العين فيه وحده. وتوجد داخليًا آلية لتحديد النطاق لكل حقل، لكن التجربة المتاحة حاليًا لا تتيح تضييقه.

الفورية على الجوال `Since 1.8` . يُكشف الحقل المشمول بالمنحة في موضعه بمجرد نجاح الكشف، من دون الحاجة إلى إعادة فتح المُدخل أو القائمة. وترد تفاصيل الآلية في: مسح ختم القناع القديم عند الكشف.

بديل كلمة المرور

قد يطلب الخادم كلمة مرور الحساب بدل رمز MFA حين تتحقّق **كل** الشروط التالية: لا يملك المستخدم طريقة MFA صالحة للاستخدام (`MFA_NO_METHOD`)، ويكون النطاق قد فعّل هذا البديل (`allowPasswordStepUp` ، معطّل افتراضيًا: **Allow Password Step-Up** في مزوّدو الهوية [الإدارة ‹ طرق تسجيل الدخول والدخول الموحّد])، وتكون للحساب كلمة مرور محلية. ولا يعرض العميل هذا البديل خيارًا أبدًا، بل يعرض حقل كلمة المرور حين يختاره الخادم.

حين لا يتاح الكشف

إن أبلغ الخادم بعدم وجود طريقة صالحة للاستخدام، أو بأن الحساب اتحادي/SSO (`EXTERNAL_USER_FORBIDDEN`)، يوضّح المعالج أن الكشف غير متاح بدل عرض تحدٍّ غير صالح. ولا يعرض خيار فتح شاشة الأمان إلا في حالة «لم تُعَدّ أي طريقة بعد»؛ أما في حالة الحساب الاتحادي أو الطريقة غير المدعومة، فلا يعرضه، لأن التسجيل هناك لن يفيد.

الحظر المؤقت

يؤدّي تكرار إدخال رموز أو كلمات مرور خاطئة، أو كثرة التحديات في وقت قصير، إلى حظر الإدخال حتى وقت يبلّغ به الخادم. ويُطبَّق الحدّ **على مستوى الحساب**: إذ يخضع الحساب لمحدِّد المعدّل نفسه المستخدم في المصادقة متعددة العوامل عمومًا، فلا تمنح جلسة أخرى حصة جديدة. وإن لم يحدّد الخادم موعد رفع الحظر، يفترض العميل مهلة انتظار افتراضية قصيرة. وإن انتهت صلاحية التحدي المعلَّق أو أُبطل في الأثناء، يعيد المعالج بدء خطوة التحقق بدل أن يفشل.

دورة حياة نافذة الكشف

بصرف النظر عن العدّ التنازلي:

- **نقل تطبيق الجوال إلى الخلفية ثم استئنافه.** عند الاستئناف، تُعرض القيم مقنَّعة أولًا، ثم يتحقّق التطبيق من الخادم مما إذا كانت النافذة لا تزال مفتوحة، ولا يفترض أبدًا أنها بقيت مفتوحة.
- **تسجيل الخروج، أو تسجيل الخروج القسري، أو تبديل المساحات، أو بدء جلسة تفويض أو إنهاؤها** يُغلق النافذة على الجهاز فورًا دون انتظار الخادم، فلا يرث العمل نيابةً عن شخص آخر نافذةً مفتوحة أبدًا.
- **القيمة المقنَّعة التي تظهر بينما يظن العميل أنها مكشوفة** تُعدّ المرجع المعتمد؛ فيُغلق العميل النافذة ويعيد التحقق.

لا تُحفظ أي حالة كشف على الجهاز بين الجلسات؛ فكل تشغيل يبدأ بالقيم مقنَّعة ثم يستعلم من الخادم. وترد تفاصيل الربط في: ربط كشف القيم الحسّاسة.

فك تشفير الحقل المشفّر `Since 1.10`

الحقل إما **حسّاس** وإما **مشفّر**، ولا يجمع الصفتين أبدًا. وتُخزَّن القيمة المشفّرة بصيغة مشفّرة في حالة السكون (at rest)، لذا فإن كشفها يتّبع مسارًا منفصلًا بلا مدة ولا نافذة، و**يتطلّب تحققًا إضافيًا خاصًّا به** (أُضيفت الحقول المشفّرة إلى خطّ التطوير قبل 1.10، وكان فك تشفيرها يكتفي بالمنحة وحدها دون أي فحص؛ ولم يُطرح ذلك السلوك في أي إصدار).

- **مؤشّر الإجراء.** عند توافر منحة فك التشفير، يعرض الحقل أيقونة عين (التلميح **فك التشفير**) بجانب `****` . ومن دون المنحة، يعرض أيقونة قفل ولا يتيح أي عنصر قابل للاختيار. ولا يتيح الحقل المعروض بوصفه كلمة مرور فك التشفير أبدًا، حتى لو وُسم بأنه مشفّر، لأن قيمته المخزّنة تجزئة أحادية الاتجاه؛ وترفض نقطة نهاية فك التشفير هذه الحالة أيضًا.
- **الويب.** تفتح أيقونة العين مطالبة التحقق الإضافي (**تحقق من هويتك**) وفق قواعد اختيار الطريقة نفسها المتّبعة في معالج الكشف، على أن تقتصر المطالبة على عملية الكشف هذه وحدها. وعند النجاح، تحلّ القيمة محلّ القناع في موضعها. ومن دون الاستحقاق، لا يتّخذ العميل أي إجراء بدل إظهار خطأ، لأن أيقونة القفل أوضحت ذلك سلفًا.
- **الجوال.** يفتح النقر على القيمة المقنَّعة ورقة التحقق نفسها التي يستخدمها التطبيق لإجراء سير عمل يتطلّب تحققًا إضافيًا؛ وتسمية قارئ الشاشة هي «حقل مشفّر، القيمة مخفية. اضغط لفك التشفير». وتُبقي المطالبة الملغاة أو الفاشلة الحقل مقنَّعًا. وترد تفاصيل التنفيذ في: وحدة cubit لفك تشفير كل حقل.
- **أحادي الاستخدام.** يُستهلك الإثبات عند استرداده، ويبقى صالحًا نحو **120 ثانية** إن لم يُستخدم، ويرتبط بحقل واحد (وبصف واحد داخل كتلة قابلة للتكرار). ويتطلّب فك تشفير حقل ثانٍ، أو إعادة التحميل، مطالبة جديدة.

- **الظهور.** تبقى القيمة بعد فك تشفيرها ما دام الحقل على الشاشة. ويؤدّي الانتقال إلى موضع آخر، أو تمرير الحقل خارج نطاق العرض، أو إعادة التحميل، إلى إعادة تقنيعها. ولا تُحفظ أي حالة كشف على الجهاز أو على الخادم.
- **التدقيق.** تُسجَّل كل عملية فك تشفير ناجحة في مسار التدقيق بوصفها *Decrypted Data*؛ ويُسجَّل فحص الهوية على نحو منفصل، سواء نجح أم فشل. ولا تُسجَّل القيمة نفسها أبدًا، بل هوية من فكّ تشفيرها ووقت العملية والحقل المعني.
- **حالات الإخفاق.** يتلقّى المستدعي الذي لا يملك المنحة الرفض نفسه في كل مرة، ولا يُحظر أبدًا (إذ لا يوجد ما يُخمَّن). أما المستدعي الذي يملك المنحة ويُخفق في الفحص مرارًا، فيصل إلى الحدّ على مستوى الحساب الموصوف في الحظر المؤقت.

نقطة النهاية وربط الإثبات: فك تشفير قيمة.

مرجع نموذج الكيان وواجهته البرمجية

هذه الصفحة موجّهة إلى **مطوّري التكامل والمطوّرين** الذين يقرؤون تعريفات الكيانات أو يكتبونها عبر الواجهة البرمجية، أو يولّدون حمولات الكيانات، أو يحتاجون إلى معرفة القواعد الدقيقة التي يطبّقها الخادم عند إنشاء الكيان أو تغييره أو نقله أو حذفه. وتوثّق الصفحة النموذج المخزَّن، وأشكال القيم التي تحملها المُدخلات، ونقاط نهاية دورة الحياة والحالات التي ترفض فيها الطلبات، ونظام الإصدارات (التغييرات الكاسرة).

تصمّم كيانًا من الواجهة؟

ينبغي لبُناة التطبيقات البدء بصفحتي إنشاء الكيان وتصميمه [بناء التطبيقات › إنشاء كيان وتصميمه] وتغيير الكيان بأمان [بناء التطبيقات › غيّر كيانًا بأمان]، إذ تشرحان الميزات نفسها بالتسميات التي يعرضها المصمّم.

كيف تُعالَج عمليات الكتابة على الكيانات

تجري عمليات الكتابة على الكيانات **عبر الأوامر**. فمتحكّم HTTP لا يحفظ البيانات مباشرة، بل يتحقق من الطلب، ثم يرسل أمرًا (`CreateDataEntityCommand` ، و`UpdateDataEntityCommand` ، و`ArchiveDataEntityCommand` ، ونظائرها للكيانات الوهمية) إلى ناقل الخدمات، ويسجّل رمزًا للمستدعي. ثم يتولى **معالج** الأمر التحقق والتهيئة والحفظ ونشر الأحداث بصورة غير متزامنة، ويردّ عبر مقبس. ولذلك تُعيد نقاط النهاية `PUT` / `PATCH` / `POST` الكيانَ كما يُقرأ من المستودع فور إدراج الأمر في الطابور، أما النتيجة المعتمدة للعملية فتصل عبر رسالة المقبس الخاصة بتنفيذ الأمر.

هيكل الكيان

يُعرَّف المستند المحفوظ باستخدام النموذج `DataEntity` (`data-entity.model.ts`). وفيما يلي خصائصه الرئيسية:

الخاصية	النوع	ملاحظات
`id`	UUID	يولّده الخادم (`v4()`) عند الإنشاء؛ ولا يورّده العميل.
`specificationVersion`	number	إصدار المخطّط/المواصفة لنموذج الكيان. القيمة الافتراضية `1` .
`name`	LocalizedValue	مطلوب. اسم عرض مُوطَّن. يُتحقَّق من وجوده ومن طوله الأدنى/الأقصى، ويجب أن يكون فريدًا **داخل وحدته**.
`description`	NullableLocalizedValue	اختياري.
`normalizedName`	string	مطلوب، **فريد عالميًا** (فهرس فريد). يجب أن يستوفي النمط وقيود الطول في `NormalizedNameValidation` ؛ ولا يجوز أن يكون اسمًا مقيّدًا (انظر أدناه).
`version`	number	إصدار البيانات/التغييرات الكاسرة. القيمة الافتراضية `0` . لا يُزاد إلا عند إجراء تغييرات كاسرة تستوفي شروط زيادته (انظر الإصدارات).
`isImplicit`	boolean	مطلوب. يُميّز الكيانات التي يهيّئها النظام وتخضع لقواعد مقيّدة للتحرير/الحذف.
`implicitOptions`	`{allowEdit, isReadOnly}`	يحدّد ما إذا كان يجوز تحرير كيان ضمني.
`virtual`	DataEntityVirtualOptions	موجود للكيانات الافتراضية المدعومة بخط أنابيب؛ ويحمل `enabled` و`source` و`dependsOn` المحسوبة.
`type`	`DataEntityTypeEnum`	مطلوب. انظر أنواع الكيانات.

الخاصية	النوع	ملاحظات
`typeOptions`	TypeOptions | VirtualProxyEntityOptions | AggregatorVirtualEntityOptions	إعدادات خاصة بالنوع (مثل مصادر virtual/proxy/joiner/aggregator).
`icon`	Icon	مطلوب.
`module`	UUID	مطلوب. الوحدة المالكة.
`state`	`DataEntityStateEnum`	`Draft` / `Active` / `Archived`.
`stateHistory`	[]StateHistory	سجلّ لانتقالات الحالة لا يقبل إلا الإلحاق.
`creationInfo` / `updateInfo` / `archiveInfo`	ActionInfo	مَن نفّذ الإجراء ومتى، بما في ذلك المفوَّض له. لا يُضبط `archiveInfo` إلا عند الأرشفة.
`fields`	[]DataEntityField	حقول الكيان الخاصة به (انظر الحقول).
`internalEntities`	[]InternalEntity	مخطّطات داعمة لحقول **block** (مجموعات حقول متداخلة).
`internalCollections`	[]InternalCollection	مجموعات قوائم القيم المضمّنة.
`workflows`	[]DataEntityWorkflow	سير العمل المرفق للموافقة/الحالة.
`validation`	string | null	تعبير تحقّق على مستوى الكيان.
`tags`	[]string	وسوم حرّة الصياغة.
`meta`	ServiceMeta | object	لكيانات النوع `Request`: جمهور الخدمة/مراقبوها.
`displayOption`	`DataEntityDisplayOptionEnum`	التخطيط: `KeyValue` (افتراضي)، `TwoColumns`، `ThreeColumns`، `FourColumns`.
`formSettings`	DataEntityFromSettings	إعدادات عرض النموذج.
`entryNames`	`{singular, plural}`	تسميات مُوطَّنة للمُدخل المفرد وللمُدخلات المتعددة.
`treeStructure`	DataEntityTreeStructureConfig	إعداد للكيانات الشجرية/الهرمية ذاتية المرجعية.
`indexes`	[]DataEntityIndex	تعريفات الفهارس (تُحسب عند الإنشاء/التحديث).
`dataSources`	[]DataEntityFieldSources	تخطيطات مصادر الحقول المحسوبة.
`storeRules`	[]StoreRule	قواعد وضع البيانات؛ تستمد قيمتها الافتراضية من `StoreService.defaultStoreRule()` عند الإنشاء.
`sharing`	ApplicationSharing	موروثة من الوحدة المالكة.
`updateIdentifier`	string	رمز للتحكّم التفاؤلي في التزامن (`v4()`)، يُجدَّد عند كل عملية كتابة.
`displayTemplate`	EntityDisplayTemplate | null	قالب عرض مخصّص اختياري. **Since 1.4**
`scriptAttributes`	object	مجموعة اختيارية من سمات البرمجة النصية. **Since 1.8**
`deletedFields` / `breakingFields`	arrays	بيانات تتبّع الإصدارات. `breakingFields` خاصية مخفية (`hiddenProperties: ['breakingFields']`).

الحقول

كل مُدخل في `fields` هو `DataEntityField` (`data-entity-field.model.ts`):

الخاصية	النوع	ملاحظات
`id`	UUID	معرّف الحقل؛ يظل ثابتًا عند إعادة التسمية ويُستخدم لمقارنة التغييرات الكاسرة.
`specificationVersion`	number	القيمة الافتراضية `1`.
`name`	LocalizedValue	تسمية مُوطَّنة.

الخاصية	النوع	ملاحظات
`normalizedName`	string	مفتاح الحقل الفريد؛ يستوفي `NormalizedNameValidation` ؛ ولا يجوز أن يكون اسمًا مقيّدًا.
`dataType`	`DataTypeEnum`	مطلوب. أحد الأنواع أدناه.
`block`	FieldBlock	يربط حقول `Block` بكيان `internalEntity` يحمل الحقول الفرعية.
`options`	FieldOptions	أعلام سلوكية تشمل `repeatable` و `multiValue` و `showAsTitle` و `showAsMainPhone` و `showAsEntryPicture` و `sensitive` و `displayOnly` .
`source`	UUID | null	هدف ارتباط مفرد **مهمّل** (استخدم `sources`). لحقول الارتباط من نوع `Entity` .
`sources`	[]FieldSources	قائمة بمصادر الارتباط المتاحة.
`sourceOptions`	FieldSourceOptions	مثل `denyCircularRelation` .
`dataCollection`	UUID | null	مرجع إلى مجموعة بيانات (LOV).
`renderOptions`	FieldRenderOptions	يحمل `editComponent.component` (مثل `RelationComponent` و `CategoryComponent` و `SwitchComponent` و `PasswordInputComponent` و `DataQuerySelectorComponent`).
`validations`	FieldValidation	يشمل `required.enabled` .
`customValidations`	[]CustomValidationsRule	قواعد مخصّصة اختيارية.
`locked`	FieldLockedOption	التحكّم في القفل/التعبير.
`virtualField`	VirtualField	يُميّز حقلًا محسوبًا (`enabled`).
`isImplicit`	boolean	علامة لحقل النظام (تُفرَض القيمة `false` للحقول التي يورّدها المستخدم عند الإنشاء).
`mapping`	DataEntityFieldMapping	بيانات وصفية لتخطيط مصدر الحقل.
`sensitiveAccess`	SensitiveFieldAccess	**عابر**. للطلب فقط. منح وصول لقراءة حقل حسّاس، تُزامَن مع مخزن `access_right` ثم تُزال قبل الحفظ – ولا تُخزَّن أبدًا في مستند الكيان. **Since 1.8**

قيم `dataType` المدعومة (`DataTypeEnum`): `Text (0)` ، `Number (1)` ، `Date (2)` ، `Time (3)` ، `DateTime (4)` ، `Boolean (5)` ، `blob (6)` ، `Entity (7)` ، `Icon (8)` ، `Block (9)` ، `LocalizedJson (10)` ، `GeoJson (11)` ، `Json (12)` ، `ObjectJson (13)` ، `SVGIcon (14)` .

الكتل (Blocks) والحقول المتداخلة. لا يخزّن الحقل الذي تكون فيه `dataType = Block` حقوله الفرعية مضمّنةً فيه؛ بل يشير `block.internalEntity` إلى مُدخل في `internalEntities` الخاص بالكيان، وهو الذي يحمل `fields` الكتلة. وتُسطّح دالة النموذج المساعِدة `getAllFields()` هذه البنية: إذ تُخرج لكل كتلة الحقول الفرعية مع `fullNormalizedName` على هيئة `<block>.<field>` وعلَم `entityPath` و `isParentBlockRepeatable` . ويعتمد رماز الإصدارات والتحقّق على هذا التمثيل المسطَّح ذي الاسم الكامل في المقارنة – فتُطابَق الحقول بالزوج `(id, fullNormalizedName)` وليس بالاسم وحده.

مساعِدات العرض كـ / إسناد العنوان. يوفّر النموذج دوالّ مساعِدة لتحديد الحقول تُستخدم في مختلف أجزاء الخدمة: `getTitledField()` (`options.showAsTitle`)، و `getPhoneNumberField()` و `getEntryPictureField()` و `getRelationFields()` و `getSensitiveFields()` و `getVirtualFields()` و `getLocalizedFields()` و `getFieldByPath()` . ويخضع الحقل المعلَّم بـ `showAsTitle` («العنوان الرئيسي») لقواعد إضافية عند التحديث (انظر قواعد التحديث).

وإلى جانب العنوان، يمكن أن يحمل الحقل أيًّا من **أدوار خيارات العرض الدلالية** (`FieldViewOptionValueTypeEnum`) لتحدّد مُصيِّرات البطاقات/القوائم/لوحة المهام الحقل الذي تتعامل معه وكيفية التعامل معه: `showAsTitle` و `showAsSubTitle` و `showAsCategory` و `showAsSubCategory` و `showAsRelationField` و `showAsMainNumber` و `showAsMainDate` و

`showAsMainPhone` و `showAsEntryPicture` . اضبط هذه الخيارات عند إعداد الحقل – مثلًا علِّم حقل «الاسم الكامل» النصّي بـ `showAsTitle` وحقل الصورة بـ `showAsEntryPicture` ، فتُصيَّر أوجه البطاقات على نحو صحيح دون إعداد تخطيط إضافي.

الحقول الحساسة. يمكن تعليم أي حقل بأنه حسّاس (`options.sensitive`). تُقنَّع قيمة الحقل الحسّاس (`****`) لكل مستخدم ما لم يكن لديه منحة قراءة صريحة لذلك الحقل تحديدًا – وهذا مستقل عن أذونات المعرض/التعديل العادية. تُورَّد المنح مع كل طلب عبر الخاصية العابرة `sensitiveAccess` ، وتُزامَن مع مخزن `access_right` ، ولا تُحفظ أبدًا في مستند الكيان.

مصادر مرجعية ليست كيانات بيانات. ليست كل قيمة يمكن أن يحملها `source` / `sources` لحقل مرجعي (من نوع `Entity`) معرّفًا لكيان `DataEntity` – فكيان نظام مُدرَج مسبقًا، مثل قطعة العدّاد، يُسنِد إلى حقليه `query` / `view` سلسلتَي مصدر المنصة الحرفيتين `dataQuery` / `dataViews` بدلًا من ذلك، فتشيران إلى استعلام محفوظ أو معرض بيانات لا إلى كيان آخر. **Since 1.8**، أصبح مُنتقي مصدر المرجع في مصمّم الكيان يتعرّف على هذه المصادر الخاصة بالمنصة ويعرض تسمية بسيطة بدلًا من محاولة جلبها بوصفها `DataEntity` – وكانت عملية البحث تلك تفشل سابقًا برسالة «Entity not found» فوق حقل كان في الواقع مُهيَّأً على نحو صحيح.

البحث في لوحة تهيئة الحقل `Since 1.8`

قائمة الأقسام ومربع البحث في لوحة تهيئة الحقل (انظر إنشاء الكيان وتصميمه [بناء التطبيقات › إنشاء كيان وتصميمه]) **وسيلة تنقّل في الواجهة فقط**: فهما لا يضيفان شيئًا إلى `DataEntityField` / `FieldOptions` ، ولا علاقة لهما بفهرسة الحقل في قاعدة البيانات.

أشكال القيم المخزَّنة

يخزّن المُدخل قيمة كل حقل تحت `value.<normalizedName>` . ويعتمد الشكل المخزَّن على نوع الحقل – ويُميَّز نوعا الارتباط بمكوّن التحرير وبما إذا كان `dataCollection` مضبوطًا، إضافةً إلى `dataType` (`FieldService.isReferenceField` / `isADTField`):

نوع الحقل	يُميَّز بـ	القيمة المخزَّنة
نص	`dataType: 0`	سلسلة نصية خام – يُحوِّلها مُحوِّل النص باستخدام `value?.toString() ?? ''` .
رقم	`dataType: 1`	رقم خام (`Number(value)`).
تصنيف	`dataType: 12` + `editComponent.component: "CategoryComponent"`	`{"id": "<option id>"}` – لا يُحفظ سوى `id` ؛ وتُحقن `label` و `color` و `_resolved` (بالشكل `{label, color}`) عند القراءة من الخيار المطابق.
ارتباط كيان	`dataType: 7` + `"RelationComponent"` مع مصدر كيان مضبوط	`{"dataEntity": "<entity id>", "dataEntry": "<entry id>"}` – كلاهما مطلوب؛ وتُخزَّن **مصفوفة** منها عندما يكون الحقل `multiValue` .
ارتباط عنصر مجموعة	`dataType: 7` + `"RelationComponent"` مع `dataCollection` مضبوط	`{"id": "<collection item id>"}` – معرّف العنصر داخل المجموعة المرتبَط بها.
كتلة	`dataType: 9` + `"BlockComponent"`	كائن متداخل مفاتيحه هي `normalizedName` لحقول الكيان الداخلي (مثل `value.basic_info.username`)؛ وتُخزَّن **مصفوفة** من هذه الكائنات عندما يكون `options.repeatable` مضبوطًا. تُهيَّأ القيم الفارغة بالشكل `{}` – أو `Array(min).fill({})` للكتل المتكرّرة.

توجد خيارات حقل التصنيف القابلة للاختيار في `renderOptions.editComponent.options.options` – وهي مصفوفة من `{id, label, color?}` يكون فيها `id` **سلسلة نصية** خام: عادةً ما تكون UUID للخيارات المُنشأة في المنشئ، وسلسلة ثابتة

للحقول الأساسية المُدرَجة مسبقًا (مثل خيارَي حقل `type` في كيان المساحة `STANDALONE` / `SUB_SPACE`). ويرفض المدقّق عند الحفظ قيمة التصنيف إذا غاب عنها `id` أو لم يوجد معرّفها في تلك المصفوفة.

> **أشكال الارتباط الخاطئة والفشل الصامت.** لا يمكن استخدام شكلَي الارتباط أحدهما بدلًا من الآخر، ويعتمد نمط الفشل على اتجاه الاستبدال:
>
> - يُرفض إرسال شكل المجموعة `{"id": ...}` إلى حقل **ارتباط كيان** عند الحفظ بخطأ تحقّق `Invalid reference property`.
> - إرسال شكل الكيان `{"dataEntity": ..., "dataEntry": ...}` إلى حقل **ارتباط عنصر مجموعة** يجتاز فحص الشكل وقد يُحفظ، ثم **يُحلّ صمتًا إلى لا شيء** عند القراءة: إذ يتطلّب فرع حلّ المجموعة `value.id`، فتمرّ القيمة عبره دون أن تُحلّ وتعود `_resolved` كائنًا فارغًا – دون أي خطأ في أي مرحلة.
> - لا يُصلح الحفظُ الشكلَ أبدًا: إذ تُبقي عملية التنظيف عند الحفظ ما أرسلته من `dataEntity` و`dataEntry` و`id`.

قيود التسمية

- يجب أن يستوفي `normalizedName` (للكيان وللحقل) `NormalizedNameValidation` (نمط regex + الطول الأدنى/الأقصى) المفروض عبر مخطّط JSON على النموذج.
- يُفرَض على `normalizedName` للكيان **فهرسًا فريدًا**؛ وتُرفض القيم المكرّرة عند الإنشاء/التحديث (`EntityDuplicateException`، أو `11000` في Mongo `UniqueValueException`).
- يجب أن يكون `name` للكيان فريدًا **داخل الوحدة المالكة** (لا عالميًا) – ويُفرَض ذلك عبر `checkDuplicateNamesFromDatabase(..., [{module: {eq: module}}])`.
- لا يجوز استخدام الكلمات المحجوزة قيمةً لـ `normalizedName` للكيان أو لحقوله أو لحقول الكيان الداخلي. يرفض `checkRestrictedNormalizedName()`: `switch` و`new` واسم العدّادات و`default` و`class` و`break` و`private` و`public` و`protected` و`else` و`if` و`return` و`case` و`catch` و`throw` و`while` و`for` و`try` (`RestrictedNormalizedNameError`).

ولأن `normalizedName` هو المفتاح الآلي الثابت الذي ترتبط به الاستعلامات والتفاعلات، فإنه يظل ثابتًا عمليًا بعد الإنشاء – إذ يولّده المنشئ تلقائيًا ويتيح لك تعديله قبل الحفظ، بينما يمكنك تغيير اسم العرض المُوطَّن لاحقًا دون التأثير في المفتاح الآلي. ويُستخدم أيضًا في عمليات التحزيم (**الاستيراد/التصدير**) للحفاظ على تمثيل الكيان عبر البيئات المختلفة.

الحقول المُخطَّطة (Mapped fields) `Since 1.8`

يستمد الحقل الذي يحمل `mapping.enabled` قيمته من **حقل مصدر** في الكيان الذي تشير إليه `mapping.referenceFieldId`. وتُبقي ثلاث قواعد البيانات الوصفية للحقل المُخطَّط متزامنة مع مصدره – وتُطبَّق جميعها عبر سلسلة التخطيط الكاملة، بحيث تصل أيضًا إلى الحقل المُخطَّط بصورة متعدّية (C يعكس B الذي يعكس A بدوره):

- **وراثة الحساسية.** يرث الحقل المُخطَّط علم الحساسية من حقل مصدره – وحاملي منح الوصول إلى ذلك الحقل الحسّاس – كلما حُفظ الكيان المُخطَّط. وينتشر التغيير في الاتجاه المعاكس الآن فورًا أيضًا: فتغيير حساسية حقل المصدر ينقل التغيير إلى كل حقل مُخطَّط منه على الفور، بدلًا من انتظار المرة التالية التي يُحفَظ فيها الكيان المُخطَّط مصادفةً. انظر الحقول الحساسة [بناء التطبيقات › حماية حقل] لمعرفة كيفية إخفاء الحقل الحسّاس وكشفه عند القراءة.
- **إعادة مزامنة `unique` باتجاه واحد فقط.** عندما يفقد حقل المصدر صفة التفرّد، يُمسَح علم `unique` القديم من الحقل المُخطَّط ويُعاد بناء فهرس الكيان المالك كي يُحذَف الفهرس الفريد فعليًا. ولا يرث الحقل المُخطَّط `unique` أبدًا من مصدره، حتى لو أصبح حقل المصدر فريدًا حديثًا – فلا تُمسَح إلا قيمة `true` قديمة، ولا تُضبط هذه القيمة أبدًا.

- **مراجع «التحديد التلقائي للأول» المتسلسلة تملأ حقولها المُخطّطة الآن بشكل صحيح.** كان الحقل المرجعي المُهيَّأ لاختيار قيمته الأولى تلقائيًا، والذي يكون مصدره نفسه متسلسلًا على مرجع آخر مضبوط بالفعل على المُدخل (مثل تفاعل ينشئ مُدخلًا يربط أصل التسلسل فقط)، يُحلّ سابقًا إلى لا شيء عند حساب قيم الحقول المُخطَّطة عبره. وأصبحت تلك الحقول – بما في ذلك حقل مُخطَّط قابل للتكرار لم تتضمّن الحمولة أي قيمة له إطلاقًا – تُملأ الآن على نحو صحيح.

العرض وإعدادات النموذج

تُعرَض مُدخلات الكيان في نموذج تتحكّم `displayOption` في تخطيط أعمدته (`DataEntityDisplayOptionEnum`):

- `KeyValue` (افتراضي) – حقل واحد في كل صف.
- `TwoColumns` / `ThreeColumns` / `FourColumns` – حقلان أو ثلاثة أو أربعة حقول في كل صف، مع ملء مساحة العمود المتبقية عندما يحمل الصف حقلًا إضافيًا.

أمّا سلوكيات عرض النموذج الأكثر تفصيلًا (تجميع الحقول وترتيبها وتخطيط الأقسام) فتُضبط في `formSettings` وتُحرَّر من لوحة **إعدادات النموذج** ضمن تبويب هيكل الكيان في المنشئ.

سكربتات الكيان

يمكن أن يحمل الكيان **سكربتات كيان (entity scripts)** – خطّافات لدورة الحياة يكتبها المستخدم بلغة JavaScript، وتُرفق بالكيان في المنشئ وتُحفظ في `scriptAttributes` . وفي وقت التشغيل، يُشغّل العميل الرماز المخزَّن لكل **واجهة معرض (view-face)** ثم لكل **واجهة كيان (entity-face)** عبر `EntityScriptExecutorService.run` ، مما يتيح للمنشئ التفاعل مع أحداث دورة حياة المُدخل دون تفاعل من جانب الخادم.

حقول البيانات الوصفية للنظام (`_metaData`)

إلى جانب حقول الكيان الخاصة به `fields` ، يحمل كل مُدخل مجموعة من **حقول البيانات الوصفية للنظام** التي يختمها أسس ويتعهّدها تلقائيًا. وتُعرَض عبر كتلة `_metaData` افتراضية لإتاحة اختيارها داخل معرض (عبر `?includeMetaDataFields=true` ، انظر ملاحظات نموذج القراءة). وهذه الكتلة **مقفولة/للقراءة فقط** وافتراضية: تُبنى بواسطة `metaDataBlockField(withWorkflowMetaField)` ويدعمها كيان داخلي `_metaDataInternalEntity` (`META_DATA_BLOCK_NORMALIZED_NAME = '_metaData'`)؛ وتُسطَّح حقولها الداخلية إلى `_metaData.<id>` .

يوجد هذا السجلّ في `osos-shared/src/metaFields.ts` ، ويربط `META_DATA_PATH_MAP` كل معرّف حقل وصفي افتراضي بمساره الفعلي المخزَّن في المستند:

معرّف الحقل الوصفي	المسار المخزَّن	النوع
`createdBy`	`creationInfo.doneBy`	ref USER
`createdAt`	`creationInfo.doneAt`	datetime
`updatedBy`	`updateInfo.doneBy`	ref USER
`updatedAt`	`updateInfo.doneAt`	datetime
`entryId`	`id`	text
`entityVersion`	`entityVersion`	number
`invalidReason`	`invalidReason`	number enum (0 = OBSOLETE_ENTRY، 1 = INVALID_REQUIREMENT_ENTRY)

فالمعرض الذي يختار `_metaData.createdBy` يقرأ في الواقع `creationInfo.doneBy` – إذ يتيح السجلّ حلّ أسماء `_metaData.<id>` المسطَّحة إلى ختمَي `creationInfo` / `updateInfo` الداخليَّين الموصوفَين في جدول النموذج أعلاه.

حقول البيانات الوصفية لسير العمل. عندما يكون للكيان سير عمل، يُستدعى `metaDataBlockField` مع `withWorkflowMetaField = true`، مما يضيف مجموعة إضافية من حقول `_metaData` القابلة للاختيار تصف خطوة سير العمل الحالية للمُدخل (تُقرأ من الخطوة الحالية النشطة للمُدخل تحت `_metaCurrentSteps.*`؛ ويوجد السجلّ التاريخي تحت `_metaWorkflowHistory.*` / `_metaAdhocSteps.*`):

معرّف الحقل الوصفي	المعنى	النوع
`currentStep`	اسم الخطوة الحالية مُوطَّنًا	نص مُوطَّن
`currentStepOwner`	مالك الخطوة الحالية	ref USER
`currentStepOriginalOwner`	المالك الأصلي للخطوة الحالية	ref USER, GROUP
`currentStepReceivedDate`	وقت استلام الخطوة	datetime
`currentMainStep`	ما إذا كانت الخطوة الحالية خطوة رئيسية	boolean
`currentAdhocStep`	ما إذا كانت الخطوة الحالية خطوة مخصّصة (ad-hoc)	boolean

ويُحدِّد `CURRENT_STEP_OWNER_META_DATA_FIELDS = [ _metaData.currentStepOriginalOwner ، _metaData.currentStepOwner ]` الحقول الوصفية التي تحمل بيانات المالك وتُستخدم في فحوص ملكية الخطوة.

أنواع الكيانات (أساسية مقابل مخصّصة)

عند إنشاء كيان، تختار **نوع الكيان (Entity Type)**. وتكون الكيانات التي تعرّفها بنفسك عادةً من النوع `Custom`؛ وتضمّ المنصة أنواعًا مدمجة أخرى كثيرة يظهر لك بعضها، لكنك نادرًا ما تنشئها يدويًا. الحقل `type` من نوع `DataEntityTypeEnum`: تُصنّف قيمته الكيان وتحدّد في الوقت نفسه العمليات المسموح بها. و**نوع الكيان يُثبَّت عند الإنشاء ولا يمكن تغييره بعد ذلك** – فهو، مثل `normalizedName`، جزء من هوية الكيان الدائمة. أبرز القيم:

الاسم	القيمة	المعنى
`Custom`	0	كيان اعتيادي يعرّفه المنشئ.
`ADT`	1	كيان نوع بيانات مجرّد/بدائي.
`Service`	2	كيان خدمة.
`Request`	3	كيان نوع الطلب؛ النوع الضمني الوحيد الذي يبقى قابلًا للتحرير ويدعم تحديثات أذونات الخدمة.
`Contact`، `TimelinePost`، `CalendarEvent`، `Notification`، `Task`، `User`، `Group`، `AllRequests`، `Person`، `Role`، `Assignment`، … `Comment`، `PostCategory`، `Report`، `Dashboard`، `Configuration`، `OrganizationUnit`	4-29	أنواع كيانات المنتج المدمجة.
`Core`	30	كيان منصّة أساسي.
`System`	40	كيان نظام.
`VirtualProxy`	50	كيان وهمي يستند إلى مصدر أساسي واحد.
`JoinerProxy`	51	كيان وهمي يربط مصادر متعددة عبر العلاقات.
`VirtualAggregator`	52	كيان وهمي يجمّع مصادر متعددة (مع تخطيطات الحقول).
`Space`، `SpaceUsers`، `Delegation`، `WorkingHours`، `Interceptors`، `Templates`، `Validators`، `SystemObject`، `Audit`	60-90	أنواع كيانات البنية التحتية/النظام.

دليل الخدمات وكيانه المرتبط `Since 1.8`

يحمل المُدخل الخاص بخدمة في دليل الخدمات مؤشرًا إلى الكيان الفعلي (عادةً من نوع `Request`) الذي يستند إليه. **Since 1.8**، أصبحت أرشفة مُدخل في دليل الخدمات أو استعادته تمتدّ بالتسلسل إلى المُدخلات نفسها في الكيان المرتبط: فأرشفة مُدخل الدليل تؤرشف أيضًا المُدخلات النشطة في الكيان الذي يستند إليه، واستعادته تستعيدها – ويمتدّ ذلك بدوره إلى صفوف الطلبات المرآتية، بالطريقة نفسها التي كانت تتّبعها بالفعل أرشفة الكيان المرتبط مباشرةً أو استعادته (من صفحة إدارة كيانات البيانات في واجهة المسؤول). وقبل ذلك، كانت أرشفة خدمة من دليل الخدمات تغيّر حالة صف الدليل وحده، وتُبقي الطلبات المُقدَّمة مسبقًا لتلك الخدمة ظاهرة في طلباتي/صندوق الوارد.

تقتصر هذه الملاحظة على تسلسل الأثر في نمذجة البيانات؛ أما سير عمل إدارة دليل الخدمات نفسه فهو موثّق ضمن الخدمات وتقديم الطلبات [استخدام أسس]، والأثر على الطلبات موثّق ضمن الطلبات والخدمات (واجهة API والآليات الداخلية).

الكيانات المخصّصة

تُنشأ عبر `POST /v1/entities` باستخدام `type = Custom` (أو نوع آخر غير افتراضي وغير نظامي). وتُنشأ مُدخلاتها يدويًا، وتكون قابلة للتحرير والحذف بالكامل، وفقًا لمُصوِّتي الأذونات.

الكيانات الأساسية / الضمنية

يُهيّئ النظام الكيانات التي تكون فيها `isImplicit = true` – وهي أنواع مدمجة كالمستخدمين والمجموعات والوحدات التنظيمية والتدقيق، وغالبًا ما تكون محميّة أو للقراءة فقط جزئيًا. ويُقيّد مُسنَدان (predicates) في الخدمة قابليتها للتحرير والحذف:

```
isEntityEditDenied(e)   = e.isImplicit && !e.implicitOptions?.allowEdit && e.type !== Request
isEntityDeleteDenied(e) = e.isImplicit && e.type !== Request
```

فلا يمكن تحرير الكيان الضمني إلا إذا كان `implicitOptions.allowEdit` مضبوطًا (أو كان كيان `Request`)، ولا يمكن حذفه إلا إذا كان كيان `Request`. وتؤدي المخالفات إلى إطلاق `ImplicitEntityUpdateException` / `ImplicitEntityDeleteException`. إضافةً إلى ذلك، يحافظ المعالج عند تحديث كيان ضمني على التعريف المحفوظ، ولا يقبل من الخصائص الواردة إلا مجموعة فرعية **مُدرَجة في القائمة البيضاء** (`fields` و`internalEntities` و`displayOption` و`formSettings` و`workflows` و`indexes` و`internalCollections` و`storeRules` و`templates`) – أمّا سائر الخصائص فيحتفظ بقيمها من الكيان المخزَّن.

الوحدة الأساسية (Core module) (`CORE_MODULE_ID = 00000000-0000-0000-0000-000000000000`) محميّة هي أيضًا: فإنشاء كيان فيها أو نقل كيان إليها يُطلق `ImplicitModuleException`.

الكيانات الوهمية

الكيانات الوهمية **للقراءة فقط**: تُحسب مُدخلاتها من كيانات مصدر أخرى بدلًا من إدخالها يدويًا. ولها ثلاثة أنواع:

- **الوكيل (Proxy)** (`VirtualProxy`) – يُظهر كيان مصدر واحدًا لإتاحة الوصول إليه والتعامل معه من جزء أو وحدة أخرى، دون تكرار بياناته.
- **الموصِّل (Joiner)** (`JoinerProxy`) – يربط المصدر الأساسي بكياناته المرتبطة عبر العلاقات، ويعرض معلومات الكيان الأساسي مع الحقول المرتبطة.
- **المجمِّع (Aggregator)** (`VirtualAggregator`) – يحسب قيمًا مُجمَّعة (العدّ والمجموع والحد الأدنى/الأقصى والمتوسط) من كيان مصدر، ويجمّعها حسب حقل أو أكثر، فيكون كل مُدخل صفًّا مُلخَّصًا للقراءة فقط لا يُحرَّر يدويًا.

لا تُنشأ الكيانات الوهمية عبر `POST /v1/entities` – إذ يرفض معالجا الإنشاء والتحديث صراحةً `VirtualProxy` / `VirtualAggregator` / `JoinerProxy` بإطلاق `DataNotSupportedException`. ولها نقاط نهاية خاصة بها:

- `POST /v1/entities/virtual` – الإنشاء.
- `PUT /v1/entities/virtual/{id}` – التحديث.

يقبل `typeOptions` للكيان الوهمي شكلين، تُعالجهما `createVirtual`:

1. **مسطَّح / تفاعلي** – `{sourceEntityNormalizedName, sourceQuery}`. يُغلَّف هذا الشكل في كائن `VirtualProxyEntityOptions` واحد بمصدر `Primary` واحد.
2. **محفوظ / مُحرَّم** – `{sources: [...]}`. يُمرَّر دون تعديل (يُستخدم عندما تعيد عملية استيراد التحزيم تشغيل كيان وهمي مُصدَّر سابقًا، بما في ذلك مصادر aggregator ذات `fieldMappings`)، لأن إعادة بنائه ستزيل الحقول الخاصة بـ aggregator.

يميّز `VirtualProxyEntitySourceTypeEnum` بين المصادر `Primary (0)` و `Additional (1)`. يستبين `findById` كيانات المصدر ويُعيد كائنات نقل بيانات (DTOs) تفصيلية بحسب النوع: `DataEntityAggregatorDetailsDto` للمجمِّعات، و`DataEntityJoinerDetailsDto` لـ joiner/proxy (مع نطاقات العلاقات المستبانة). وتتضمّن استجابات aggregator أيضًا `mappedSources`، التي تحسبها `getVirtualEntityMappedSources` – مُدخل واحد لكل تخطيط كيان مصدر/معرض. **Since 1.8**

لا توجد نقطة نهاية منفصلة لحذف الكيانات الوهمية؛ لذا تُستخدم `DELETE /v1/entities/{id}` (الأرشفة).

دورة الحياة

الحالات

`DataEntityStateEnum`: `Draft (0)` و `Active (1)` و `Archived (2)`. يظلّ الكيان في حالة Draft قيد البناء؛ انشره ليصبح Active حتى يتمكّن المستخدمون النهائيون من إنشاء مُدخلات ضمنه. يُلحَق كل انتقال إلى `stateHistory` بـ `changedAt` و`newState` و`changedBy` و`delegatee` الاختياري.

الإنشاء

`POST /v1/entities` ← `CreateDataEntityHandler.handle`. ينفّذ المعالج ما يلي:

1. يرفض الطلب إذا كان المعرّف موجودًا مسبقًا (`EntityDuplicateException`) أو كان النوع افتراضيًا (`DataNotSupportedException`).
2. إذا كان `virtual.enabled` مفعَّلًا، يُعيد ضبط `workflows` إلى `[]`.
3. يختم `creationInfo` و`updateInfo`، ويُنشئ `updateIdentifier` جديدًا، ويضبط `storeRules` الافتراضية.
4. مرحلة التحقق: التحقق من الكيان، وفحص تكرار normalizedName، والتحقق من الاسم وفحص تكراره ضمن الوحدة، والتحقق من الحقول الضمنية (ما لم يكن `isSystemModification` مفعَّلًا)، وفحوص الحقول (الحقول المكرّرة، وإعدادات repeatable، ومتطلّب المصدر، وخيارات «العرض ك»، والتحقق من كل حقل)، والتحقق من مساهمة الشريط الزمني، والتحقق من سير العمل، والتحقق من الكيان الداخلي.
5. مرحلة التحضير: `defineWorkflowHappyPath` و`handleFieldsOptions` و`appendImplicitConfigs` وإنشاء تعريف الفهرس و`calculateEntityMappingFieldSources`.
6. يزامن منح الوصول للحقول الحسّاسة ويزيل `sensitiveAccess` العابر.

7. يحفظ عبر `repository.create`، وينشر `DataEntityCreatedEvent`، ويُصدر `CommandExecutedMessage` عبر المقبس.

تبدأ الكيانات الجديدة عند `version = 0` (القيمة الافتراضية للنموذج)، وتأخذ حالتها من النموذج ما لم ترد قيمة صريحة لـ `state` في الحمولة. الأذونات المورَّدة في DTO الإنشاء **لا** تُخزَّن على الكيان؛ إذ يزيلها المتحكّم ويوجّهها إلى مخزن منح الوصول (`EntitlementEnum.DataEntity`).

مثال عملي: حمولة الإنشاء

جسم `POST /v1/entities` هو مستند `DataEntity` (إضافةً إلى مصفوفة `permissions` اختيارية)، مع استبعاد حقول دورة الحياة من مخطّط الطلب. وفيما يلي زوج `fields` + `internalEntities` بالحدّ الأدنى الصالح – حقل نصي وحقل كتلة تقع حقوله الفرعية في كيان داخلي شقيق؛ أمّا الخصائص على مستوى الكيان (`name` و`normalizedName` و`type` و`icon` و`module` وغيرها) فموثَّقة في جدول الهيكل. يجب أن يتضمّن كل حقل `dataType` و`options` و`renderOptions` و**`validations`**. ويتطلّب `renderOptions` **كلًّا من** `editComponent` و`viewComponent`، وكلٌّ منهما واصف `{component, options}` تكون فيه قيمة `component` سلسلة نصية خام. ويجب أن تكون الوحدة المالكة `module` وحدة قائمة غير أساسية – احصل على قائمة الوحدات عبر `GET /v1/data-module` (عبر البوابة: `/api/v1/data-module`).

```
{
  "fields": [
    {
      "specificationVersion": 1,
      "name": {"en": "Title", "ar": "العنوان"},
      "normalizedName": "title",
      "dataType": 0,
      "options": {
        "hideForEmptyValues": false,
        "showInTableView": true,
        "showAsTitle": false,
        "showAsSubTitle": false,
        "showAsCaption": false,
        "showAsMainDate": false,
        "showAsMainNumber": false,
        "showAsEntryPicture": false
      },
      "renderOptions": {
        "editComponent": {"component": "TextInputComponent", "options": {}},
        "viewComponent": {"component": "ViewTextComponent", "options": {}}
      },
      "validations": {
        "required": {"enabled": false, "expression": null},
        "min": {"enabled": false, "value": null, "valueSourceType": 0},
        "max": {"enabled": false, "value": null, "valueSourceType": 0}
      }
    },
    {
      "specificationVersion": 1,
      "name": {"en": "Basic Info", "ar": "المعلومات الأساسية"},
      "normalizedName": "basic_info",
      "dataType": 9,
      "block": {"internalEntity": "<uuid-A>"},
      "options": {
        "hideForEmptyValues": false,
        "showInTableView": false,
        "showAsTitle": false,
        "showAsSubTitle": false,
        "showAsCaption": false,
        "showAsMainDate": false,
        "showAsMainNumber": false,
        "showAsEntryPicture": false
      },
      "renderOptions": {
        "editComponent": {"component": "BlockComponent", "options": {}},
        "viewComponent": {"component": "BlockComponent", "options": {}}
      },
      "validations": {
        "required": {"enabled": false, "expression": null},
        "min": {"enabled": false, "value": null, "valueSourceType": 0},
        "max": {"enabled": false, "value": null, "valueSourceType": 0}
      }
    }
```

```
  ],
  "internalEntities": [
    {
      "id": "<uuid-A>",
      "validation": "",
      "fields": [
        {
          "specificationVersion": 1,
          "name": {"en": "Username", "ar": "اسم المستخدم"},
          "normalizedName": "username",
          "dataType": 0,
          "options": {
            "hideForEmptyValues": false,
            "showInTableView": true,
            "showAsTitle": false,
            "showAsSubTitle": false,
            "showAsCaption": false,
            "showAsMainDate": false,
            "showAsMainNumber": false,
            "showAsEntryPicture": false
          },
          "renderOptions": {
            "editComponent": {"component": "TextInputComponent", "options": {}},
            "viewComponent": {"component": "ViewTextComponent", "options": {}}
          },
          "validations": {
            "required": {"enabled": false, "expression": null},
            "min": {"enabled": false, "value": null, "valueSourceType": 0},
            "max": {"enabled": false, "value": null, "valueSourceType": 0}
          }
        }
      ]
    }
  ]
}
```

ملاحظات على البنية:

- **`options`** – القيم المنطقية الثماني الظاهرة هي أعضاء `FieldOptions` ذات `required: true`؛ أمّا البقية فاختيارية أو لها قيم افتراضية. وتُملأ الخصائص المطلوبة *مع* قيمة افتراضية (`specificationVersion` وعدد من هذه القيم المنطقية) تلقائيًا عند إنشاء النموذج، لكن الحمولات الفعلية – بما فيها الكيانات الأساسية المزروعة – تُضمّنها صراحةً.
- **`validations`** – يتطلّب `required` (بالصيغة `{enabled, expression?}`) و `min` / `max` (بالصيغة `{enabled, value?, valueSourceType}`، والقيمة الافتراضية لـ `valueSourceType` هي `0`).
- **واصفات المكوّنات** سلاسل نصية خام. تستخدم الكيانات المزروعة مثلًا `TextInputComponent` و`LocalizedTextInputComponent` و`BooleanComponent` و`CategoryComponent` و`RelationComponent` و`BlockComponent` في جانب التحرير، و`ViewTextComponent` و`ViewLocalizedTextInputComponent` و`ViewCategoryComponent` و`ViewRelationComponent` في جانب العرض.
- **`id` الحقل** اختياري (القيمة الافتراضية من الخادم: UUID v4 جديد). ويأخذ `internalEntities[].id` أيضًا قيمة UUID افتراضية – حدّده صراحةً كي يستطيع `block.internalEntity` الإشارة إليه؛ وتستخدم الكيانات الداخلية المزروعة

`validation: ""`.

- **حقول التصنيف** تستخدم `dataType: 12` مع `"CategoryComponent"` وتضع خياراتها القابلة للاختيار في `renderOptions.editComponent.options.options`:

```
"renderOptions": {
  "editComponent": {
    "component": "CategoryComponent",
    "options": {
      "options": [
        {"id": "STANDALONE", "label": {"en": "Standalone", "ar": "مستقل"}, "color": "#FF0000"}
      ]
    }
  },
  "viewComponent": {"component": "ViewCategoryComponent", "options": {}}
}
```

يتطلّب إنشاء كيان رخصةً تشمل الإدارة إضافةً إلى صلاحية التعديل على الوحدة (غير الأساسية)، ويجري بصورة غير متزامنة؛ لذا فإن استجابة `2xx` تعني أن `CreateDataEntityCommand` وُضع في الطابور، لا أن الكيان أصبح موجودًا (انظر الملاحظة المعمارية في أعلى هذه الصفحة).

التحديث

`PATCH /v1/entities/{id}` ← `UpdateDataEntityHandler.handle`. تُفحص الشروط المسبقة في **المتحكّم** قبل وضع الأمر في الطابور:

- `isEntityEditDenied` ← `ImplicitEntityUpdateException`.
- يجب أن تطابق قيمة `updateIdentifier` لدى العميل القيمة المخزَّنة، وإلّا يُطرح `UpdateIdentifierConflictException` (التزامن التفاؤلي، انظر أدناه).
- يجب ألّا يكون الكيان مقفولًا من قِبل مستخدم آخر (`LockedObjectException`).

ثم يعيد المعالج التحقق (تكرار normalizedName، وفحص الحالة المؤرشفة، ودمج القائمة البيضاء الضمنية، وفحوص الاسم، والتحقق من الحقول/التفرّد/الحقول الضمنية، والتحقق من سير العمل)، ويعيد حساب الخيارات/الفهارس/التخطيطات، ويعيد حساب الحقول المحذوفة وإصدار الكيان، ويُجدّد `updateIdentifier`، وينفّذ **كتابة شرطية ذرّية** (انظر أدناه). وينشر `DataEntityUpdatedEvent` (مع أعلام إعادة الفهرسة والتغيير ومقارنة الفروق)، وينشر – إذا رُفع الإصدار – `DataEntityVersionChangedEvent`.

عبر إجراء **تحرير الكيان (Edit Entity)** في المنشئ، تشمل بنية الكيان القائم التي يمكنك تغييرها اسم العرض والأيقونة، والاسمين المفرد والجمع، والوصف، والحقول، والتفاعلات (reactions)، وسير العمل، والتحقق، والشريط الزمني (timeline)؛ بينما يبقى `normalizedName` و`type` ثابتَين (و`module` أيضًا في التحديث العادي).

قواعد تحديث إضافية يفرضها المعالج:

- **الكيانات المؤرشفة** لا يمكن تحديثها ما لم يُضبط `skipArchiveCheck` (`EntityArchiveException`).
- لا يُغيَّر `normalizedName` أبدًا عند التحديث؛ إذ يُعاد ضبطه على القيمة المخزَّنة.
- لا يمكن تغيير `module` في تحديث عادي؛ إذ يُعاد ضبطه على القيمة المخزَّنة ما لم يُضبط `allowModuleUpdate` (تستخدمه نقطة نهاية نقل الوحدة).

- **قيود العنوان الرئيسي** (`validateMainTitleNotNestedReference`): لا يجوز أن يكون حقل `showAsTitle` من نوع `multiValue` / `repeatable` (`MainTitleCannotBeRepeatableOrMultipleException`)، وإذا كان حقل ارتباط (`Entity`) فلا يجوز أن يُنشئ عنوان ارتباط *متداخل*؛ فلا يجوز للكيان المصدر ولا لأي كيان يشير إلى هذا الكيان أن يستخدم بدوره حقل ارتباط عنوانًا رئيسيًا له (`NestedReferenceMainTitleSourceIsReference` / `NestedReferenceMainTitleReferencedByOtherEntity`).

التزامن التفاؤلي (updateIdentifier)

تُجدّد كل عملية كتابة `updateIdentifier` بقيمة `v4()` جديدة. ويجري فحص المساواة في المتحكّم **قبل** وضع الأمر في الطابور، مما يترك نافذة بين الفحص والتنفيذ يستطيع خلالها كاتب متزامن (عميل آخر، أو أحد المشتركين الكثيرين في الأحداث الذين يعيدون الكتابة إلى الكيان) أن يغيّر الرمز. ولإغلاق نافذة التحديث المفقود، ينفّذ معالج التحديث `updateAll` **شرطي** بمطابقة المعرّف المقروء أصلًا:

```
updateAll(dataEntity, {id, updateIdentifier: oldEntity.updateIdentifier})
```

إذا لم يحدّث Mongo أي مستند، تُجهَض الكتابة ويُطرح `UpdateIdentifierConflictException` إلى المستدعي. **Since 1.8**

`Since 1.10` – صار **المدخل** يحمل رمزًا خاصًّا به أيضًا، وهو `updateIdentifier` منفصل يُجدَّد بصورة مستقلة في مستند المُدخل، وليس الرمز الموجود في تعريف الكيان الموصوف هنا، وإرساله مجددًا اختياري لكل طلب كتابة، لا إلزامي كما هو أعلاه. أمّا نقاط النهاية التي تقبله، و `409 UPDATE_CONFLICT` الناتج عنه، ففي استدعاء واجهة HTTP.

حلقة إعادة المحاولة، وإعادة الإسقاط على قراءة حديثة `Since 1.10`

يمكن للكاتب إخضاع كتابة مُدخل لإعادة المحاولة تلقائيًا عند التعارض بدلًا من أن تفشل مباشرةً بـ `409 UPDATE_CONFLICT` . فإلى جانب `updateIdentifier` الخاص بالمُدخل أعلاه، تقبل حمولة طلب الكتابة حقلين اختياريين إضافيين: `expectedUpdateIdentifier` و `changedPaths` (مسارات الحقول التي يُراد لهذه الكتابة تغييرها تحديدًا). وعند توفّر كليهما:

- عند حدوث تعارض، **يُعاد قراءة المدخل من جديد**، وتُطبَّق مسارات `changedPaths` الخاصة بالمستدعي فقط مجددًا على تلك القيمة الحديثة – ولا تُستخدم نسخة المستدعي الأصلية أبدًا، إذ يمكن أن تُسقط دون تنبيه تغيير كاتب متزامن في حقل لم يلمسه المستدعي أصلًا.
- تُعاد محاولة الكتابة مقابل الرمز المقروء حديثًا، حتى **3 محاولات إجمالاً** (محاولة أولى + محاولتا إعادة)، بفاصل **250 ملّي ثانية ثم 500 ملّي ثانية** بينها.
- تعيد كل محاولة تشغيل تسلسل القراءة ← التحقّق ← المقارنة ← الحفظ كاملًا، لا الكتابة الأخيرة فقط، بحيث يُحظَر كل ما يراقب التغييرات (الروابط، المُجمّعات، حقول الخرائط) دائمًا استنادًا إلى المُدخل الحالي، لا إلى نسخة قديمة.

لا يمكن إعادة بناء `changedPaths` بأمان من مقارنة تُجرى بعد وقوع تعارض؛ فمقارنة نسخة قديمة بقراءة حديثة تخلط نيّة المستدعي بتغيير كاتب متزامن، دون إمكان الفصل بينهما؛ لذا على المستدعي أن يُصرّح بالمسارات المقصودة مسبقًا. والحقلان اختياريان وإضافيان فحسب: فالكتابة التي تُغفلهما تتصرّف تمامًا كما كانت قبل هذا التغيير.

الكتابات المدفوعة بالتفاعلات تكتسب الحماية نفسها، خلف راية ميزة `Since 1.10`

يمكن لإجراء **تحديث مدخل** في التفاعلات والأتمتة [بناء التطبيقات] أن يُمرِّر كتاباته عبر حارس إعادة المحاولة نفسه، ولكن فقط عندما تتحقّق الشروط التالية جميعها لتلك الكتابة بعينها:

- راية البيئة `REACTION_ENTRY_OPTIMISTIC_LOCKING_ENABLED` مفعَّلة على المنشأة.
- يحمل المُدخل المستهدف بالفعل `updateIdentifier` – فالمُدخل الخالي من رمز لا يُحمى أبدًا حتى مع تفعيل الراية، تجنّبًا لإرسال كتابة غير مشروطة بينما يظنّ المشغّل أنّ الحماية نشطة.
- تغيّر مسار حقل واحد على الأقلّ فعليًا نتيجة الكتابة.

وهذا النطاق ضيّق عمدًا: فهو يحمي إجراء تحديث مُدخل في التفاعلات تحديدًا، لا كتابات التفاعلات عمومًا، ولا يغيّر كيفية حماية تعديل محرّر بشري لعمله الخاص. وعند تعطيل الراية، أو في كتابة لا تستوفي الشروط أعلاه، يكون السلوك مطابقًا تمامًا لما كان عليه قبل هذا التغيير.

النقل إلى وحدة مختلفة

`PATCH /v1/entities/{id}/module` مع `{module, updateIdentifier}` (و `?force=true` اختياري). يتطلّب تطابق `updateIdentifier` وعدم وجود قفل، ويرفض الوحدة الأساسية. يرسل `UpdateDataEntityCommand` مع `allowModuleUpdate = true` و `force` بوصفه `skipArchiveCheck`. وعندما تتغيّر الوحدة فعليًا، يمرّر المعالج معرّف الوحدة الجديد تباعًا إلى **معارض** و**مجموعات** الكيان التابعة له. **Since 1.8** (تتالي الوحدة)

تغيّر في 1.10 مَن يحقّ له استدعاء المسار: فلم يعد مُصوِّت الوحدة هو المرجع في ذلك، وصار المسار محصورًا في مسؤولي النظام (انظر أدناه).

النقل بصلاحية المسؤول، والنقل الجماعي، وما يصلحه النقل `Since 1.10`

مَن يحقّ له النقل. صار `PATCH /v1/entities/{id}/module` **حصرًا لمسؤول النظام**: الدور `SystemAdmin` (ويجب أن تشمل الرخصة الإدارة). ويُسمح باستدعاءات الخدمة الموثوقة (استيراد حزمة) لأنها لا تحمل أذونات مستخدم، ويُرفض الطلب المرتبط بمعرض مستخدم نهائي. توقّع حالات الرفض الآتية:

الحالة	الاستجابة
الوجهة هي الوحدة الأساسية	`400` (`ImplicitModuleException`)
الوجهة مجهولة أو غير مرئية	`404`
الوحدة الوجهة قيد الحذف النهائي	`409 MODULE_BEING_DELETED`
الكيان موجود أصلًا في تلك الوحدة	`400 ENTITY_ALREADY_IN_MODULE` (ويحصل المستدعي الموثوق من الخدمة على `200` بلا أثر، كي يظلّ استيراد الحزمة مستقلّ الأثر عند تكراره)

ويمكن نقل الكيانات الوهمية (الوكيل والرابط والمجمِّع) أيضًا. ولا يغيّر نقلها شيئًا سوى الوحدة؛ وكان قبل 1.10 يُعيد `500 "The data not supported"`.

النقل الجماعي. ينقل `POST /v1/entities/move` بالجسم `{entityIds[], module}` من **1 إلى 50** كيانًا مميّزًا إلى وحدة واحدة، بالتخويل نفسه. تُفحص الوجهة مرة واحدة؛ فإذا كانت الوحدة أساسية أو مجهولة أو قيد الحذف، يُرفض الطلب كله ولا يُنقل شيء. ثم تُنقل الكيانات **واحدًا واحدًا وبترتيب الطلب** (لأن تنظيف صفحات الوحدة القديمة يعيد كتابة صفحات كاملة، فالنقل المتوازي كان سيؤدي إلى فقدان عمليات الإزالة). وتكون الاستجابة دائمًا `200`:

```
{
  "module": "<target module id>",
  "moved":  [{ "id": "<entity id>", "repairIncomplete": false }],
  "failed": [{ "id": "<entity id>", "code": "ENTITY_ARCHIVE", "message": { "en": "…", "ar": "…" } }]
}
```

تسرد `moved` و `failed` النتائج بترتيب الطلب. وتكون قيمة `repairIncomplete` هي `true` فقط حين يكتمل النقل نفسه وتفشل بعده خطوة إصلاح لاحقة (ويُسجَّل الإخفاق). ويتخطّى المسار الجماعي فحص `updateIdentifier` (وتلتقط الكتابة الشرطية التي ينفّذها الكاتب أي تحرير متزامن)، ولا يفرض النقل أبدًا، فيفشل نقل الكيان المؤرشف دائمًا بـ `ENTITY_ARCHIVE`. أمّا الرموز التي لا رسالة خاصة بها، مثل `LOCKED_OBJECT` و `UPDATE_CONFLICT`، فتأخذ النص العام لـ `ENTITY_MOVE_FAILED`.

ما يعيد النقل كتابته. يتجاوز النقل تغيير حقل في الكيان؛ إذ تعيد المنصة ربط كل ما يشير إلى الوحدة القديمة وتصلحه:

ماذا	الأثر
المعارض والمجموعات	تتبع الكيان إلى الوحدة الجديدة.
نطاقات الحقول الحسّاسة	يُعاد ترميزها في موضعها للوحدة الجديدة؛ وتبقى أحرف البدل للوحدة (wildcards) دون مساس.
روابط النظام في الشريط الجانبي	يُعاد ربطها بعنصر أب جديد؛ وتُؤرشف المكرّرات.
قطع العدّادات ومنشورات الشريط الزمني	يُعاد وسمها بالوحدة الجديدة.
كائنات وصول المعارض وحقب الأذونات والملفات	تُحدَّث.
صفحات الوحدة القديمة ومبياناتها	**تفقد** ما يرتبط بالكيان المنقول (معرّفه ومعرّفات معارضه ومخططاته واستعلاماته وتقاريره): تُزال عناصر القوائم والقطع المطابقة من صفحات تلك الوحدة، وتُسحب ملخصاته من مبيان الوحدة. ولا تُمسّ صفحات النظام ولا صفحات الوحدات الأخرى ولا مبيان الكيان نفسه. ويتلقّى محرّر الصفحة المفتوح حينها تعارض تحديث.
التفاعلات	يُعاد وسم إجراءات `SendNotification` بالوحدة الجديدة.
التدقيق	يحمل `DataEntityUpdatedEvent` القيمة `EntityChanges.MODULE_CHANGED` و `previousModule` .

الأثر على التصميم: نقل كيان خارج وحدة **يزيل** قطعه وملخصاته من صفحات تلك الوحدة ومبيانها – ولا ينقلها معه. أعد بناءها في الوحدة الوجهة بعد النقل. وتنبّه واجهة الإدارة إلى ذلك في تحذيرها.

في واجهة الإدارة. يتوفّر النقل المفرد في قائمة بطاقة الكيان وفي صفحة تفاصيل الكيان (الكيانات النظامية وتبويب الكيانات في الوحدة)، مع قائمة وجهة تتيح أيضًا إضافة وحدة جديدة. ويتوفّر النقل الجماعي في قائمة ترويسة صفحة الوحدة: يحوّل القائمة إلى وضع التحديد (ويُحتفظ بالتحديد عبر الصفحات) ويعطّل **Move** عند تجاوز 50 كيانًا.

تحديث أذونات الخدمة (كيانات Request)

يحدّث `PATCH /v1/entities/{id}/meta/permissions` قيمتي `meta.audience` / `meta.observers` على كيان من نوع `Request` فقط؛ وأي نوع آخر يؤدي إلى طرح `EntityNotOfTypeRequestException` . ويخضع لفحوص التحرير الضمني والقفل.

الأرشفة (الحذف)

`DELETE /v1/entities/{id}` ← `ArchiveDataEntityCommand` ← `ArchiveDataEntityHandler` . لا ينفّذ هذا المسار حذفًا نهائيًا أبدًا؛ إذ يضبط المعالج `state = Archived` ، ويختم `archiveInfo` ، ويُضيف إلى `stateHistory` ، ويحفظ، وينشر `DataEntityArchivedEvent` (الذي يؤرشف لاحقًا مُدخلات الكيان). الحذف محظور للكيانات الضمنية غير `Request` (`isEntityDeleteDenied` ← `ImplicitEntityDeleteException`)، ويخضع لضوابط إضافية عبر `Helper.checkEntityForArchiving` . ومنذ `Since 1.10` يوجد مسار منفصل لمسؤول النظام يزيل الكيان نهائيًا – انظر الحذف النهائي.

أرشفة كيان مؤرشف أصلًا تُجيب بـ `ENTITY_ARCHIVE 422` `Since 1.10` (الرسالة *The data entity has been archived*). وكانت تظهر سابقًا بالحالة `500` وتُدرَج في تتبّع الأخطاء؛ ولم يتغيّر الرمز ولا الرسالة، بل صارت الحالة وحدها خطأ عميل. وينطبق الرفض نفسه على **تحديث** كيان مؤرشف **أو نقله**، فيُجيبان أيضًا بـ `422` ، ولم يعد النقل الجماعي للوحدة يسجّل الكيان المؤرشف بوصفه فشلًا في الخادم. ولا تخلط بينه وبين `ENTITY_ARCHIVED` (بصيغة الماضي، وهو `422` أيضًا)، الذي يرفعه مسار الحذف النهائي أدناه لكيان مؤرشف بلا علامة حذف نهائي.

الحذف النهائي (purge) `Since 1.10`

يزيل `DELETE /v1/entities/{id}/permanent` كيانًا **وكل ما يملكه** إزالةً لا رجعة فيها. وينطبق على الكيانات **المخصّصة وكيانات Request والكيانات الوهمية** (الوكيل والرابط والمجمِّع)، وهو متاح لمسؤول النظام فقط (الدور `SystemAdmin`؛ ويجب أن تشمل الرخصة الإدارة). أمّا `DELETE /v1/entities/{id}` المعتاد أعلاه فلم يتغيّر، ويقتصر على الأرشفة.

افصل الروابط أولًا. لا يسمح الحذف النهائي بترك مراجع معلّقة. فقبل أن يزيل أي شيء، يستعلم من خدمة التبعيات عمّا لا يزال يشير إلى الكيان من *خارجه*، ويرفض الحذف ما دامت تلك القائمة غير فارغة. الفحوص بالترتيب:

الفحص	الرفض
نوع غير قابل للحذف النهائي، أو ضمني، أو في الوحدة الأساسية، أو كيان مقيَّد / إعدادي / مملوك للرماز	`400 ENTITY_PURGE_NOT_ALLOWED`
مؤرشف عبر المسار المعتاد (بلا علامة حذف نهائي)	`422 ENTITY_ARCHIVED`
مثبَّت من حزمة	`403`
مقفول من قِبل مستخدم آخر	`409 LOCKED_OBJECT`
شيء خارج الكيان لا يزال يشير إليه	`409 ENTITY_HAS_DEPENDENTS` (`details.count`)
تعذّر الحصول على نتيجة فحص التبعيات	`503 DEPENDENCY_CHECK_UNAVAILABLE` (دون إعادة محاولة)

وفي واجهة الإدارة، يسرد مربع حوار الحذف تلك المراجع مجمّعة حسب الفئة، ومع كل منها رابط **Open**، ويُبقي **Delete** معطَّلًا حتى لا يبقى منها شيء؛ وحين لا تُدرَج أي مراجع، يُفعَّل بتحديد مربع الاختيار "I understand…".

ما يُزال. يُؤرشف الكيان أولًا ويُوسَم للحذف النهائي (دون حدث أرشفة، فلا تُطلَق تفاعلات أرشفته أبدًا). ثم تُزال وظائفه ومعارضه وصفحاته ومبيانه وتقاريره وأزراره وقطع عدّاداته وتعريفاته ومنحه ومجموعة مُدخلاته وكل صف مفتاحه الكيان – ولكيان Request أيضًا صف دليله ومراياه ومسوداته وكائنات النظام الخاصة بخدمته واستثناءات التفويض. وقبيل حذف مستند الكيان نفسه، يُنشر `DataEntityDeletedEvent` (لقطة مع المعرّفات المحذوفة) ويُسجَّل أثر التدقيق `DATA_ENTITY_DELETED`؛ وتزيل كل خدمة لاحقة صفوفها الخاصة.

قابل للاستئناف لا معاملاتي. كل خطوة مستقلة الأثر. فإن فشلت إحداها بقي الكيان مؤرشفًا وموسومًا، و**يُنهي استدعاء Delete مرة أخرى العمل**. وتكون استجابة النجاح `200 {success: true, result: null}` (ويحدّد مستند OpenAPI القيمة `204`).

ولإزالة وحدة كاملة بكياناتها، استخدم الحذف النهائي الخاص بالوحدة نفسها؛ انظر الوحدات.

سقف الكيانات وعلم `isImplicit` `Since 1.10`

قد يُحدَّد للقاطن حدّ أقصى لعدد الكيانات. فإذا بلغه، رُفض إنشاء كيان أو كيان وهمي بـ `403 ENTITY_CAP_EXCEEDED` مع `details: {cap, current}` (الرسالة *Creating this entity would exceed this application's limit of N entities (it has M).*).
ما يتوقّعه الباني أو المكامِل:

- **أيّ حدّ.** الأصغر من حدّين، قد يغيب أيّ منهما: `maxEntities` في الخطة `Since 1.10` (انظر حدود الخطّة)، وسقف الكيانات لكل قاطن الذي يستطيع المشغّل ضبطه (سجلّ القاطن في سجلّ القاطنين، أو إعداد في تثبيت القاطن الواحد). وفي غيابهما لا سقف إطلاقًا.
- **ما يُعدّ.** كل كيان **غير ضمني** و**ليس في الوحدة الأساسية**، أيًّا كان نوعه، عبر جميع مساحات القاطن. وتُحتسب الكيانات الوهمية، ولا تُستثنى الكيانات المؤرشفة. ولا يُفحص الإنشاء في الوحدة الأساسية أبدًا. ويجري الفحص قبل حفظ الكيان؛ لذا قد تتجاوز عمليات الإنشاء المتزامنة السقف بمقدار عددها على الأكثر.

- يحسب **استيراد الحزمة** مسبقًا الكيانات التي سينشئها، ويرفض الاستيراد كله قبل كتابة أي وحدة أو كيان – صمّم الحزم مع مراعاة حدّ القاطن المستهدف.
- **وللمدخلات حدّ أيضًا.** يرفض `maxEntries` في الخطة المُدخلات الجديدة بـ `403 ENTRY_CAP_EXCEEDED` متى كان الاستخدام سيتجاوز 110% منه، ويُرفض تثبيت الحزمة الذي سيتجاوزه قبل أن يبدأ (انظر حدّ المُدخلات [الإدارة › رخصتك وحدود خطتك] والتثبيت). ولا تُحتسب مُدخلات كيانات الوحدة الأساسية، ويُحتسب المُدخل المؤرشف بوزن ربع.

ولأن الكيانات الضمنية معفاة، فإن `isImplicit` **يُتجاهل للمستدعين من المستخدمين**: يخزّن `POST /v1/entities` القيمة `isImplicit: false` بلا `implicitOptions`، ويُبقي `PATCH /v1/entities/{id}` القيم المخزَّنة بصرف النظر عمّا يرد في الجسم. ولا يتغيّر شيء لاستيراد الحزم وأوامر الناقل وترحيلات البذر – فهي وحدها التي تستطيع إنشاء كيانات ضمنية.

الإصدارات والتغييرات الكاسرة

تحمل الكيانات قيمة `version` عددية (القيمة الافتراضية `0`). وهي **ليست** عدّادًا للمراجعات يزداد مع كل حفظ، بل لا تُرفع إلا عندما يكون التغيير كاسرًا للبيانات القائمة، وبشرط أن تكون للكيان مُدخلات أصلًا. يُنفَّذ هذا المنطق في `DataEntityVersionService`، ويتحكّم فيه `UpdateDataEntityHandler.handleEntityVersion`.

متى يُرفع الإصدار

```
if (entriesAtAnyVersion == 0 && currentVersion == 0) → no versioning work
newVersion = (entriesAtAnyVersion > 0 && breakingChanges.length) ? current+1 : current
```

ومن ثَمّ لا يرتفع الإصدار **إلا** إذا (أ) كان للكيان مُدخل واحد على الأقل – في **أي** إصدار، بما في ذلك المُدخلات التي أصبحت بالية بسبب تغيير كاسر سابق لم يُعالَج – و(ب) تضمّنت مجموعة التغييرات تغييرًا كاسرًا واحدًا على الأقل. أما التعديلات غير الكاسرة، والتعديلات على كيان ليست له مُدخلات بعد، فتُبقي الإصدار كما هو.

ما الذي يُعدّ تغييرًا كاسرًا

يقارن `calculatingBreakingChanges` بين مجموعتي الحقول المسطَّحة، القديمة والجديدة (بالمطابقة باستخدام `id` + `fullNormalizedName`)، إلى جانب تغييرات سير العمل، ويسجّل أسبابًا من `BreakingChangesReasons`:

السبب (enum)	يُطلَق عندما
`AddedRequiredField`	يكون حقل جديد كليًا مطلوبًا.
`ChangeFieldToRequired`	يصبح حقل قائم مطلوبًا.
`FieldTypeChanged`	تختلف قيمة `dataType` عن قيمتها في الحقل القديم.
`FieldEncryptionChanged`	يُفعَّل خيار **التشفير** في الحقل أو يُعطَّل `Since 1.10`. ويُضاف هذا السبب حتى لو انطبق سبب آخر على الحقل نفسه، فيُسجَّل انتقال التشفير دائمًا. انظر تبديل التشفير.
`ChangeFieldToRepeated`	يصبح الحقل `repeatable`.
`ChangeFieldToNonRepeatable`	لا يعود الحقل `repeatable`.
`ChangeFieldToMultivalue`	يصبح الحقل `multiValue`.
`ChangeFieldToSingleValue`	لا يعود الحقل `multiValue`.
`EnableDenyCircularRelationOnField`	يُفعَّل `sourceOptions.denyCircularRelation`.
`FieldRestored`	يُعاد إدراج حقل محذوف سابقًا بنوع *مختلف*.
`EntityWorkflowChanged`	تُضاف أي خطوة في سير العمل أو تُزال أو يُعاد ترتيبها أو تتغيّر، أو يُحذف سير عمل.

لا يُطلَق حاليًا السببان `AddedUniqueField` و`ChangeFieldToUnique` (إذ عُطِّلا بتعليق في المصدر). أما `ReferenceSourceChanged` فموجود في التعداد وله كاشف (`isSourceChangedOnReferenceField`)، لكنه **غير** مُدرَج في مجموعة التغييرات الكاسرة، ولذلك لا يرفع تغييرُ مصدر حقل المرجع الإصدارَ بمفرده حاليًا. و**حذف الحقل ليس سببًا كاسرًا كذلك**: فالمقارنة لا تمرّ إلا على الحقول الموجودة في التعريف الجديد، وتُسجَّل الحقول المحذوفة على حدة في `deletedFields` الخاصة بالإصدار (انظر سجلات الإصدارات). وإذا أُعيد إدراج حقل وكانت أعلامه المؤثرة في الكسر (`dataType` و`repeatable` و`required`) تطابق جميعها أعلام نسخته المحذوفة، فيُعامَل ذلك على أنه **استعادة**، و*لا* يُعدّ كاسرًا.

> قاعدة عامة: التغييرات الإضافية (حقل اختياري جديد) آمنة؛ أما تغييرات النوع، والحقول المطلوبة الجديدة، وتغييرات التكرار وتعدّد القيم، وتغييرات التشفير، وتغييرات سير العمل، فهي كاسرة وتستدعي عملية ترقية.

الأثر على المدخلات القائمة

عندما يرفع تغييرٌ كاسر إصدار كيان **منشور**، تبقى المُدخلات القائمة عند إصدارها القديم وتظهر على أنها **بالية (obsolete)** حتى تُرقَّى إلى أحدث إصدار. وإذا كان للكيان سير عمل، يتوقّف سير العمل هذا، وعلى المدير أن يقرّر كيفية التعامل مع القيم القديمة – بإبقائها أو تحديثها – ويعود سير العمل إلى خطوته الابتدائية. ولترحيل المُدخلات، استخدم **ترقية المدخلات** (أو **الترقية الجمعية**) لنقلها إلى الإصدار الجديد، و**التعديل الجمعي** لضبط قيم الحقول على نطاق واسع عبر المُدخلات.

المعاينة قبل الحفظ

يُجري `PATCH /v1/entities/{id}/breaking-changes` المقارنة نفسها مع حمولة **دون** حفظها – وهو ما يدعم إجراء **معاينة التغييرات الكاسرة** في المنشئ، ويتيح لك الاطّلاع على الأثر في المُدخلات القائمة قبل النشر. يدمج المتحكّم الحمولة الجزئية مع الكيان المخزَّن (مع استبعاد المفاتيح ذات القيمة `undefined` كي لا يُفسَّر غياب الحقول على أنه حذف)، ويثبّت id/version، ويستدعي `previewBreakingChanges`، ثم يُعيد `BreakingChangesResult`:

```
{ version, fields: DataEntityBreakingField[], hasBreakingChanges: boolean, workflowChanges?: [] }
```

سجلات الإصدارات

يُخزَّن كل إصدار في مستند `DataEntityVersion` (ضمن المجموعة `DATA_ENTITY_VERSION_NAME`) يتضمّن: `id` و`dataEntityId` و`version` و`createdAt` و`changes` (`DataEntityVersionChanges[]`) – كل عنصر منها `{changeType, field, isBreaking}`، و`diff` (ناتج `just-diff` على `fields` / `internalEntities` / `workflows`)، ومحفوظ للاحتياط. يحدّث `upsertDataEntityVersion` سجلّ الإصدار الحالي في مكانه (بحذف التغييرات الكاسرة التي لم تعد تنطبق عبر `isFieldStillBreaking`، وإضافة التغييرات الجديدة غير المكرّرة) أو ينشئ سجلًا جديدًا. تُسجَّل الحقول المحذوفة على حدة عبر `addDeletedFieldsToEntityVersion`.

قراءة تغييرات الإصدار (واجهة البرمجة)

- `GET /v1/entities/{entity}/breaking-changes/{version}` – الفرق بين الإصدار القديم المحدَّد والإصدار الحالي للكيان (`calculateDiffBetweenChanges`).
- `GET /v2/entities/{entity}/breaking-changes?versions=...` – جلب عدة إصدارات دفعةً واحدة؛ تُستبعَد من الإصدارات الأدنى التغييرات التي سبق أن أبلغ عنها إصدار أعلى (مع إزالة التكرار باستخدام `field.normalizedName`)، وتُعاد أخطاء كل إصدار ضمن الاستجابة بدلًا من إفشال الطلب كله. **Since 1.4**

مرجع واجهة البرمجة

تندرج جميع المسارات تحت `data-service` وتتطلّب استراتيجية مصادقة أسس OAuth2 إضافةً إلى مُصوِّت تخويل الكيان. وتُذكر الأدوار حيث يصرّح بها المعالج.

الطريقة والمسار	الغرض	ملاحظات
`POST /v1/entities`	إنشاء كيان (غير افتراضي)	مُصوِّت الوحدة. يرفض الوحدة الأساسية والأنواع الافتراضية. تُوجَّه الأذونات إلى منح الوصول.
`POST /v1/entities/virtual`	إنشاء كيان وهمي	مُصوِّت الوحدة. يأتي `typeOptions` مسطَّحًا أو بصيغة `sources`.
`GET /v1/entities/count`	عدّ الكيانات (بعد ترشيحها بحسب الوصول)	الأدوار: `SystemViewer`، `Restricted`.
`GET /v1/entities`	سرد الكيانات بصيغة DTOs	الأدوار: `SystemViewer`، `Restricted`. يضيف `entriesCount` و`moduleDetails` و`metaFields`.
`GET /v1/entities/all`	سرد الكيانات الخام	`?includeImplicit=` لتضمين الكيانات الضمنية.
`GET /v1/entities/all/count`	عدّ الكيانات الخام	`?includeImplicit=`.
`GET /v1/entities/{id}`	جلب كيان واحد (DTO)	الأدوار: `SystemViewer`، `Restricted` – **التعريف فقط، لا بيانات المدخلات أبدًا**. `?includeMainTitleFields=` و`?includeMetaDataFields=` (يضيف كتلة `_metaData` المقفولة). يحلّ `typeDetails` الافتراضية ويحدّد الوصول إلى الحقول الحسّاسة لكل طلب.
`PATCH /v1/entities/{id}`	تحديث الكيان	يتطلّب تطابق `updateIdentifier`، وألّا يكون الكيان مقفولًا أو خاضعًا لمنع ضمني.
`PATCH /v1/entities/{id}/breaking-changes`	معاينة التغييرات الكاسرة	دون حفظ.
`PATCH /v1/entities/{id}/module`	النقل إلى وحدة أخرى	مسؤول النظام (ويجب أن تشمل الرخصة الإدارة) `Since 1.10` (كان مُصوِّت الوحدة). `?force=`. انظر النقل بصلاحية المسؤول.
`POST /v1/entities/move`	النقل الجماعي لـ 1–50 كيانًا	`Since 1.10`. التخويل نفسه؛ نتيجة `failed` / `moved` لكل كيان.
`PATCH /v1/entities/{id}/meta/permissions`	تحديث جمهور/المتابِعين	نوع `Request` فقط.
`PUT /v1/entities/virtual/{id}`	تحديث كيان وهمي	يتطلّب تطابق `updateIdentifier`، وألّا يكون الكيان مقفولًا.
`DELETE /v1/entities/{id}`	الأرشفة (حذف ناعم)	محظور للكيانات الضمنية غير `Request`.
`DELETE /v1/entities/{id}/permanent`	الحذف النهائي (purge)	`Since 1.10`. مسؤول النظام (ويجب أن تشمل الرخصة الإدارة)؛ يُرفض ما دام أي شيء خارج الكيان يشير إليه. انظر الحذف النهائي.
`GET /v1/entities/{id}/index-paths`	الحقول التي يمكن أن يستخدمها فهرس مخصّص	`Since 1.10`. `[]` للكيانات التي لا تقبل فهارس مخصّصة. انظر فهارس الكيان.
`POST /v1/entities/{id}/indexes/{indexId}/retry`	إعادة بناء فهرس مخصّص فاشل	`Since 1.10`. صلاحية تحرير الكيان (ويجب أن تشمل الرخصة الإدارة).
`POST`/`GET /v1/entities/{entity}/tag-groups`، `PATCH`/`DELETE …/tag-groups/{id}`	إدارة مجموعات الوسوم	`Since 1.10`. انظر مجموعات الوسوم.
`GET /v3/entities/{entity}/entries/export`	التصدير السريع لمُدخلات الكيان	`Since 1.10`. `format=xlsx` أو `csv`. انظر التصدير السريع.

الطريقة والمسار	الغرض	ملاحظات
`GET /v1/entities/{entity}/breaking-changes/{version}`	فرق الإصدار من القديم إلى الحالي	
`GET /v2/entities/{entity}/breaking-changes`	فروق الإصدارات دفعةً واحدة	مصفوفة `?versions=` . **Since 1.4**

ملاحظات نموذج القراءة

- يُعيد `GET /v1/entities/{id}` عمدًا **تعريف الكيان فقط** (الحقول والأنواع والوحدة والبيانات الوصفية) – ولا يُعيد بيانات المُدخلات أبدًا – كي يتمكّن المستعرضون ومستخدمو Restricted الذين يملكون منحًا من عرض مُنتقيات حقول الارتباط لكيانات خارج نطاق منحهم. ومعرّفات الكيانات هي UUIDs مقيَّدة بالمساحة، لذا لا يتيح ذلك وسيلةً للتعداد.
- يحمل كل DTO مُعاد `sensitiveMaskedFields` ، التي تُحسَب **لكل طلب** للمستخدم الحالي (لا تُخزَّن في الخبء أبدًا، ولا تُحفظ أبدًا). **Since 1.8**
- يُلحق `?includeMetaDataFields=true` كتلة `_metaData` افتراضية مقفولة (وكيانها الداخلي الداعم) لإتاحة انتقاء حقول البيانات الوصفية للمُدخل داخل معرض. **Since 1.8**

الكيانات المشار إليها في DTO الكيان `Since 1.10`

كان العميل يستدعي `GET /v1/entities/{id}` مرة لكل كيان مشار إليه لعرض حقول المرجع. وأصبح DTO يحمل ما يحتاجه في مصفوفة `referencedEntities` **للاستجابة فقط**:

```
"referencedEntities": [
  {
    "id": "<entity id>",
    "moduleId": "<module id>",
    "isStatic": false,
    "mainTitleField": {
      "id": "…", "normalizedName": "name", "fullNormalizedName": "name",
      "dataType": "…", "renderOptions": { }
    }
  }
]
```

- **أين تُعاد:** `GET /v1/entities/{id}` و`GET /v2/entities/{entity}/entries/{id}` و`GET /v2/entities/{entity}/entries/draft/{id}` – و**لا** تعيدها مسارات قائمة الكيانات.
- **ما تغطّيه:** كل حقل مرجع في الكيان، بما في ذلك الحقول الفرعية في الكتل والحقول المخصّصة للعرض فقط والحقول الوصفية (كلٌّ من `sources[].entity` و`source` القديم المفصول بالفواصل)، مع إزالة المكرّرات وفق ترتيب أول ظهور. ويُهمَل الكيان المشار إليه إذا كان محذوفًا، لذا أبقِ آلية جلب احتياطية.
- **الأهداف الثابتة.** المراجع إلى `DataModule` و`DataEntity` و`DataView` و`DataViews` و`Report` و`GenericView` و`System` موسومة بـ `isStatic: true` وموصوفة ضمن الوحدة الأساسية بعنوان مترجَم. وأصبحت قيم المُدخلات التي تشير إلى أحدها تُعاد مع `_resolved` فعلي (العنوان ومعرّفا الوحدة والكيان) في قراءات المُدخلات التي تحلّ العلاقات وفي قراءات الشجرة. ويرى قارئ المُدخل المضيف أسماء الوحدات والكيانات والمعارض التي يشير إليها – ضع ذلك في الحسبان قبل تخزين مثل هذه المراجع في كيانات متاحة للقراءة على نطاق واسع.

لا يُخزَّن شيء، ولا يُجرى ترحيل.

الارتباط المباشر بتبويب في مصمّم الكيان `Since 1.8`

يظهر التبويب النشط في مصمّم الكيان ضمن عنوان الصفحة على هيئة معامل استعلام مسمّى `?tab=<slug>`، فأصبح للتبويب عنوان يمكن الوصول إليه، بدلًا من أن يكون مجرّد حالة عابرة في الواجهة. وعمليًا يتيح لك ذلك:

- **حفظ التبويب في المفضّلة أو مشاركته.** انسخ العنوان بينما تبويب *الاستعلامات* مفتوح، فيُفتَح تبويب *الاستعلامات* لدى كل من يتبع الرابط – وهو ما يفيد عند إحالة زميل إلى جزء بعينه من الكيان.
- **التحديث دون فقدان موضعك.** تُبقيك إعادة تحميل المصمّم على التبويب الذي كنت تعمل فيه.
- **التنقّل رجوعًا عبر التبويبات التي زرتها.** يضيف فتحُ تبويبٍ مُدخلًا إلى سجل المتصفح، فيتيح زرّا **رجوع** و**تقدّم** التنقّل عبر التبويبات التي فتحتها بدلًا من الخروج مباشرةً من الكيان.
- **الارتباط بتبويب من كتيّبات التشغيل أو أدواتك الخاصة** بإلحاق `?tab=<slug>` بعنوان مصمّم الكيان.

معرّفات التبويبات

تشكّل المعرّفات أدناه المجموعة الكاملة، وهي مرتَّبة بحسب ظهور التبويبات في المصمّم:

التبويب	معرّف `?tab=`
المُدخلات	`entries`
الاستعلامات	`queries`
المعارض	`views`
التفاعلات	`reactions`
الإعلانات	`announcements`
الأزرار	`buttons`
القوائم	`lists`
لوحة المعلومات	`dashboard`
التقارير	`reports`
العلاقات	`relations`
القوالب	`templates`
المخازن	`stores`
المعلومات	`information`
سجل القفل	`lock-history`
الوظائف	`functions`
الأذونات	`permissions`
السكربتات	`scripts`

السلوك الاحتياطي. يُفتح عنوان المصمّم الخالي من معامل `?tab=` على **المدخلات**. أما المعرّف غير الوارد في الجدول أعلاه فيُتجاهَل – ويُفتح المصمّم على آخر تبويب تذكّره لذلك الكيان، دون أي خطأ. ولاحظ أن روابط الارتباط المباشر الرقمية القديمة (`?tab=1` و`?tab=3` وغيرها) لم تعد معروفة؛ فاستبدل المعرّف المسمّى بها في أي رابط احتفظت به.

التحرير السريع والمدقّقات على مستوى الكيان `Since 1.8`

يغيّر التحرير السريع (المباشر) حقلًا واحدًا في المُدخل ويُطبَّق على هيئة **رقعة (patch)**، لا كتحديث كامل، ولذلك كان نطاق التحقق فيه أضيق من نطاق التحقق في نموذج المُدخل. فلم تكن الرقعة تُمنَع إلّا عندما يحدّد خطأ التحقق حقلًا عدّلته الرقعة

فعلًا، أو عندما يرفض التغييرَ مدقّقُ **وظيفة** مُعَدّ من نوع Standalone/Value.

وقد أدّى ذلك إلى ثغرة، إذ إن **المدقّق المخصّص على مستوى الكيان** – أي قواعد العمل والتفرّد التي يفرضها الكيان عبر حقوله – يُطلق خطأً غير مرتبط بأي حقل بعينه، وليس مدقّق وظيفة. فكانت تلك الأخطاء تُهمَل بصمت، ويُحفَظ التحرير السريع رغم عدم استيفاء القاعدة. أما مسارا الإنشاء والتحديث الكاملان فلم تكن فيهما هذه الثغرة قط.

و Since 1.8، أصبح أي خطأ يُطلقه مدقّق مخصّص على مستوى الكيان **يمنع التحرير السريع**، وتُعاد رسالة المدقّق نفسها إلى المستدعي بدلًا من إدراجها ضمن القائمة العامة لأخطاء التحقق من الحقول – فيرى المستخدم التفسير الفعلي للقاعدة بدلًا من رسالة «مُدخل غير صالح» المبهمة.

مثال. يفرض كيان الوحدة التنظيمية قاعدة «مدير واحد فقط لكل وحدة تنظيمية». وكان التحرير السريع لتغيير دور أحد الأعضاء إلى *مدير* في وحدة لها مدير بالفعل يُحفَظ دون اعتراض: فخطأ القاعدة لم يتضمّن اسم حقل، والرقعة لم تعدّل سوى حقل الدور. أما الآن فإن هذا الخطأ يمنع الحفظ ويُظهر رسالة القاعدة نفسها «لا يُسمح بتعدّد دور المدير».

فهارس الكيان (تبويب الفهارس) `Since 1.10`

يتضمّن باني الكيان (**Edit Entity**) تبويب **الفهارس (Indexes)**، بجوار سير العمل والخط الزمني وRTC، يعرّف فيه المسؤول فهارس MongoDB إضافية للمُدخلات في الكيان – عادةً لتسريع عمليات الترشيح أو الفرز التي تُجرى باستمرار على كيان كبير، أو لفرض التفرّد على مجموعة من الحقول. ويظهر التبويب للكيانات نفسها التي يظهر لها تبويبا الشريط الزمني وRTC، أي **ليس** للكيانات الأساسية أو كيانات `Request` أو الكيانات الوهمية (الوكيل والرابط والمجمِّع)، ويكون للقراءة فقط في كيان مثبَّت من حزمة.

تعريف فهرس. حدّد له اسمًا، وقائمة مرتّبة من الحقول مع اتجاه لكل منها (اسحب أو استخدم لوحة المفاتيح لإعادة الترتيب)، و**Unique** اختياريًا، ومرشِّحًا جزئيًا اختياريًا يُكتب بصيغة JSON لـ MongoDB. ويُجري مربع الحوار الفحوص نفسها التي يُجريها الخادم، فيبقى **Save** معطَّلًا ما دامت المسوّدة غير صالحة. قواعد الخادم (`422 ENTITY_INDEX_*` عند المخالفة):

القاعدة	الحدّ
الفهارس المخصّصة لكل كيان	**10** على الأكثر
الحقول لكل فهرس	من **1 إلى 8** حقول مميّزة من المسارات القابلة للاختيار (`GET /v1/entities/{id}/index-paths`): حقول `value.*` القابلة للفهرسة إضافة إلى `state` و`creationInfo.doneAt` و`creationInfo.doneBy.dataEntry` و`updateInfo.doneAt` و`updateInfo.doneBy.dataEntry`
الأسماء ومجموعات المفاتيح	فريدة، بما يشمل فهارس النظام والفهارس المعرّفة بالترحيلات أيضًا
معاملات المرشِّح الجزئي	`$eq` و`$exists: true` و`$gt` و`$gte` و`$lt` و`$lte` و`$type` فقط؛ دون معاملات مستندات مضمّنة
خيارات MongoDB المتقدمة	غير مقبولة
الفهرس الفريد	**يجب أن يحمل مرشّحًا جزئيًا** يستثني المُدخلات التي يكون الحقل فيها فارغًا (`ENTITY_INDEX_UNIQUE_REQUIRES_FILTER`)

تكتسب القاعدة الأخيرة أهميتها من أن MongoDB يفهرس القيمة المفقودة على أنها `null`، فيرفض الفهرس الفريد العادي المُدخل الثاني الذي يُترك الحقل فيه فارغًا. فمثلًا، يحتاج فهرس فريد على `value.code` إلى مرشِّح كهذا:

```
{ "value.code": { "$type": "string" } }
```

وتُستثنى الفهارس القائمة التي لا تعدّلها، حتى لو كانت تخالف القواعد الحالية.

الحفظ. تُعَدّ تعديلات الفهارس في النموذج وتُحفظ مع الكيان (`PUT` / `PATCH`). ويُبقي طلب PATCH الذي لا يتضمّن `indexes` الفهارس المخزَّنة، فلا تمحوها التكاملات الأخرى التي تحدّث الكيان. وتؤدّي إزالة حقل أيضًا إلى إزالة الفهارس المخصّصة المبنية عليه.

الحالة وإعادة المحاولة. تبني خدمة الفهرسة الفهارس بصورة غير متزامنة، خارج عملية الحفظ. ويحتفظ الخادم بـ `indexStatus[]` لكل فهرس مخصّص – `building` فور الحفظ، ثم `ready` أو `failed` مع خطأ MongoDB. ويتحدّث التبويب ما دام أيّ فهرس قيد البناء، ويعرض الخطأ على الفهرس الذي فشل بناؤه (عادةً بسبب بيانات مكرّرة ضمن فهرس فريد؛ ويُعاد خيار **Unique** الخاص بالحقل إلى وضع الإيقاف حين يفشل بناء فهرسه الفريد)، ويتيح **Retry** الذي يستدعي `POST /v1/entities/{id}/indexes/{indexId}/retry`. ويعرض الفهرس المعدَّل غير المحفوظ *Not saved yet*. وتُسمّى الفهارس المخصّصة `IXU_<entity>_<12 hex>`؛ وتُعرض فهارس النظام الخاصة بالكيان، لكنها مقفولة.

المعارض المبنيّة على هذا الكيان

لا تتناول هذه الصفحة نموذج المعرض – فمنشئ المعارض موثَّق في المعارض [بناء التطبيقات › معارض البيانات وأوجهها]. ومع ذلك، تندرج هنا أربعة تغييرات حديثة، لأنّ كلًّا منها يتعلّق بكيفية تفاعل المعرض مع حقول *الكيان* وإعداداته، ولأنّ اكتشاف كلٍّ منها سابقًا كان يتطلّب مواجهة المشكلة عمليًا.

أزرار النظام، وترتيب كل زر، وزر API

تُعرَّف أزرار إجراءات المعرض في `systemButtons`، وهي مصفوفة يحدّد كل عنصر فيها الزرّ ويحمل قيمة `order` خاصّة به. وقيمة `order` لكل زر **تحلّ محلّ** `systemButtonsOrder`، وهي الخاصّية الأقدم التي كانت مصفوفة معرّفات، وصارت الآن **مهمَلة**: فهي لا تزال تُقرأ للتوافق مع الإعدادات السابقة – فإذا ظلّ المعرض يحملها، يُعتَدّ بها ويحدّد تسلسلُها الترتيبَ النهائي – لكن ينبغي أن تحدّد الإعدادات الجديدة ترتيب كل زر. وحين لا يحمل المعرض أيّ ترتيب قديم، تتبع الأزرار تسلسلها المعياريّ وتطبّق الواجهة ترتيبها الافتراضي الخاص.

والأزرار الضمنية التي يستطيع المعرض إظهارها هي **العرض** و**التحرير** و**الأرشفة** و**الإنشاء** و**النسخ** و**API**، إضافةً إلى زرّ لكل قالب مستند على الكيان. وتتحدّد الأزرار المسموح بها وفق إعداد المعرض نفسه، وليس باختيارها من قائمة حرّة – فـ *الإنشاء* و*التحرير* و*الأرشفة* تتبع العمليات التي يسمح بها المعرض، و*العرض* يتبع إعداد المُدخلات فيه، و*النسخ* يتبع إعداد نسخ المُدخل فيه. وتُزال الأزرار التي لم يعد الإعداد يسمح بها عند حفظ المعرض، بدلًا من أن تمنع الحفظ.

`Since 1.10` – **لا يظهر زرّ API إلّا حين يكون Swagger document مُفعَّلًا على ذلك المعرض.** وهو الزرّ الذي يفتح مستند OpenAPI للمعرض الواحد الخاصّ به؛ ويؤدّي تعطيل الإعداد إلى إزالة الزرّ مجددًا مع الحفاظ على ترتيب الأزرار الأخرى.

المعرض قابل للحفظ من جديد بعد حذف أحد حقوله

كان حذف حقل من الكيان يجعل كل معرض لا يزال يشير إليه **غير قابل للحفظ** – بما في ذلك عند تغيير الأذونات وحده، وهو تغيير لا يتيح منتقي حقول لإزالة الإشارة المعلَّقة. وصار المعرض الآن يزيل تلك الإشارات المتقادمة بصمت ويُحفَظ، وفق ثلاث قواعد تمنع الإزالة من إخفاء خلل حقيقي:

- **المعارض القائمة فقط.** لا تُزال الإشارة إلّا إذا كان المعرض *المخزَّن* يتضمّنها أصلًا. أمّا مسار الحقل الذي استحدثه العميل للتوّ، فلا يزال يخضع للتحقق ويُرفض، كما أنّ إنشاء معرض لا يزيل أيّ إشارة مطلقًا.
- **ولا يُقلَّم حتى الفراغ أبدًا.** إذا كانت *كل* عناصر المجموعة متقادمة، فهذا يعني أن الميزة معطوبة، ولا يقتصر الأمر على عدم تطابق الإشارات؛ لذا يُترك التحقق ليرصد الخلل ويحدّد ما على المدير إصلاحه. والتجميع هو الاستثناء الوحيد، إذ لا يملك مفتاح تفعيل خاصًّا به – فقائمة المستويات الفارغة تمثّل بحدّ ذاتها حالة «لا تجميع» الصحيحة.

- **المجموعات وحدها.** أمّا الإعدادات التي تحدّد حقلًا *مفردًا* مطلوبًا فهي أساسية لعمل الميزة ولا تُسقَط قطّ: حقل تصنيف لوحة المهام، والعنوان الرئيسي للشجرة، وما شابههما. ولا تُزال الإشارات إلّا ممّا كان على هيئة قائمة – قائمة حقول المعرض، وحقول الترشيح أوّلًا، ومستويات التجميع، وأعمدة واجهة الجدول، وتخطيطات تواريخ واجهة التقويم.

ولا تُعَدّ حقول البيانات الوصفية (الافتراضية) متقادمة أبدًا.

أذونات الروابط المقيَّدة وترتيب التنقّل

إصلاحان لكيفية تفاعل **الرابط** – أي مُدخل التنقّل أو الوصول المقيَّد الذي يشير إلى معرض – مع حفظ ذلك المعرض:

- **لم تعد الترقية تُفرِغ جمهور الرابط المقيَّد.** كانت ترقية القاطن تمسح المستخدمين والمجموعات من كل رابط مقيَّد نشط، فتفقد الصفحات والمعارض المقيَّدة جمهورها بصمت وتصبح غير متاحة تحديدًا لمن كان ينبغي أن يروها. وصارت معالِجات الروابط الآن تقرأ الأذونات الحالية بدلًا من الاعتماد على الحدث الذي أطلق إعادة البناء، وتستعيد عملية إصلاح مشروطة قوائم الجمهور التي أفرغها السلوك القديم. أمّا الروابط المقيَّدة المؤرشفة فلم تتأثّر قطّ.
- **ويحتفظ رابط التنقّل بترتيبه عبر الحفظ.** كان حفظ المعرض المالك يعيد ضبط موضع الرابط، فيختلّ ترتيب التنقّل عند حسم الترتيب أبجديًا – وكانت الترقية، التي تُعيد حفظ كل معرض، تزيل ترتيب التنقّل كلّه دفعةً واحدة. وصار الرابط يرث ترتيب عرض المعرض نفسه بدلًا من ذلك. وهذا **يُشفى ذاتيًّا عند الحفظ التالي** لكل معرض متأثّر: فلا حاجة إلى تشغيل أيّ إجراء.

قوالب المستندات عبر معرض – قيد حاليّ

كان تصدير قالب مستند **من معرض** مبنيّ على كيان يحوي حقل كتلة يُنتج ملفًّا بـ **صفر مُدخلات**، بينما كان القالب نفسه يعمل عند تصديره من منطقة الإدارة. وقد أُصلح ذلك.

أمّا ما لم يُصلَح، وما يستحقّ التصميم حوله: فقالب المستند الذي يشير إلى حقل **لا يعرضه** المعرض الذي يُطلَق منه التصدير لا يزال يطبع ذلك الحقل **فارغًا**. فقائمة حقول المعرض تحدّد معًا العلاقات التي تُحَلّ والقيم التي تُقرأ، فتوسيع أحدهما يوسّع الآخر – وهو ما سبّب التصديرات الفارغة. وإلى أن يُفصَل الأمران، تأكّد من أنّ المعرض المستخدَم لإطلاق تصدير قالب يعرض كل حقل يطبعه القالب.

مجموعات الوسوم لكل كيان `Since 1.10`

يمكن تنظيم **وسوم** المسؤول في الكيان – وهي الاستعلامات التي ضُبط فيها `tag` ، وتُقدَّم للمستخدمين النهائيين كمرشِّحات بنقرة واحدة – في **مجموعات وسوم**. وتوجد في الواجهة ضمن تبويب فرعي **Tag groups** داخل تبويب الاستعلامات في الكيان، وحقل المجموعة في نموذج الاستعلام قائمة يمكن مسح اختيارها وتتضمّن **+ New group**.

المجموعة مستند `TagGroup` له `name` (من 3 إلى 128 حرفًا، تُزال المسافات من طرفيه، وفريد لكل كيان في كل لغة دون اعتبار حالة الأحرف)، و `normalizedName` (فريد لكل كيان، ويمثّل هويتها في الحزم)، ولون رمادي ثابت. وتُخزَّن العضوية على الوسم نفسه (`DataQuery.tag.tagGroup`)، فلا ينتمي الوسم إلا إلى مجموعة واحدة على الأكثر.

المسار	السلوك
`GET /v1/entities/{entity}/tag-groups`	المجموعات مع `queries` : معرّفات وسوم المسؤول المفعّلة للكيان في كل منها.
`POST /v1/entities/{entity}/tag-groups`	إنشاء مجموعة (صلاحية تحرير الكيان).

المسار	السلوك
`PATCH /v1/entities/{entity}/tag-groups/{id}`	`name` و`queries` فقط؛ وتضبط `queries` العضوية لتشمل الوسوم المذكورة بالضبط، فتنقلها من أي مجموعة أخرى.
`DELETE /v1/entities/{entity}/tag-groups/{id}`	يُخرج الوسوم من المجموعة ثم يحذفها. ويُعيد `200 {success: true, result: null}` . ولا تُعاد كتابة المعارض التي أشارت إلى المجموعة؛ وتُسقط الواجهة المعرّف المعلّق.

الأخطاء: `400 TAG_GROUP_NAME_DUPLICATE` (مع `details.locale`)، و`400 TAG_GROUP_INVALID_QUERY` (ليس وسم مسؤول مفعّلًا في هذا الكيان)، و`404 TAG_GROUP_NOT_FOUND` . ولاحظ تغيّر سلوك `tag.tagGroupName` عند إنشاء استعلام أو تحديثه: صار ينشئ مجموعة فقط حين يكون الاسم غير مستخدَم، ويفشل بخلاف ذلك بـ `TAG_GROUP_NAME_DUPLICATE` – ولم يعد يعيد استخدام مجموعة قائمة بالاسم نفسه بصمت. وتنتقل مجموعات الوسوم مع الكيان في الحزم، ويُستورد المعرض الذي يشير إلى وسم أو مجموعة مفقودة دون ذلك الخيار بدلًا من أن يفشل استيراده.

التصدير السريع لمدخلات كيان (للمسؤول) `Since 1.10`

تتيح صفحة الكيان الإدارية إجراء **Export Entries** (تصدير المُدخلات). وهو يوفّر التصدير السريع للمعرض دون الحاجة إلى معرض، ويعتمد على `GET /v3/entities/{entity}/entries/export` :

المعامل	المعنى
`format`	`xlsx` **أو** `csv` (دون اعتبار حالة الأحرف)؛ وأي قيمة أخرى تُرفض.
`fields`	الأعمدة. وعند غيابه: كل حقول الكيان بترتيبها المعرَّف، مع توسيع كل كتلة إلى حقولها.
`filterCriteria` و`search` و`include_virtual` و`appliedQuickFilters` و`appliedTagsFilters` و`groupBy` و`quickFilterParameters`	المرشِّحات نفسها المستخدَمة في قائمة المسؤول.

تكون الاستجابة هي الملف، أو – حين يتجاوز عدد المطابقات `MAX_ENTRIES_RESULT_LIMIT` (الافتراضي **10,000**) – `messageId` على الناقل، ويُسلَّم الملف في الخلفية. ويُسمّى الملف وورقته باسم الكيان.

الفرق عن مسار المعرض `GET /v3/entities/{entity}/views/{id}/entries/export` : تُحدَّد الصفوف وفق قائمة المُدخلات لدى المسؤول، لا وفق معرض، فلا `allowExport` ولا ترشيح أولًا ولا تحقق بشري ولا اقتطاع للمجهول. ويتطلّب الوصول إذن سرد المُدخلات في الكيان، ويُفحَص مجددًا عند تنفيذ التصدير في الخلفية؛ ويُرفض المستدعون المجهولون، لأن التصدير في الخلفية يحتاج إلى مستلم.

أين يتوفّر CSV. لا يتوفّر CSV إلا في هذين التصديرين السريعين – مسار المسؤول هذا والتصدير السريع للمعرض. أما تصدير القوالب (`/v1/entities/{entity}/entries/export/{templateId}`) فينتج ما يحدّده القالب [بناء التطبيقات › قوالب التصدير] (مستندًا أو XLSX أو PDF)، وتظلّ التقارير بصيغتَي XLSX وPDF.

المدخلات المشتقّة والحذف النهائي للمدخلات `Since 1.10`

المدخل المشتق نسخة ينتجها مُدخل مصدر في كيان آخر: مُدخل **وكيل** أو **مجمِّع** أو **رابط** في كيان وهمي [بناء التطبيقات › الكيانات الوهمية]. ويربط فهرسٌ كلّ نسخة بمصدرها. ويجب أن يزيل الحذف النهائي لمُدخل مصدر نسخَه، وإلّا بقيت يتيمة – وبمجرّد تحرير ملفات المصدر، قد يحذف كانس الملفات اليتيمة ملفاتٍ لا تزال النسخ تعرضها.

- **الحذف النهائي الجماعي يتتالى.** كان الحذف النهائي الجماعي للمُدخلات يُبقي نسخها الوكيلة والمجمِّعة والرابطة (إذ لم يكن الحذف يتتالى إلّا عند حذف المُدخل الواحد). وصار كلّ من عمليات التتالي الثلاث يعمل أيضًا عند الحذف الجماعي، لكلّ مُدخل مشمول فيه، ويؤدّي أيّ فشل إلى إعادة محاولة حدث الحذف كلّه.
- **التتالي المُعاد يُنهي كلّ نسخة.** إذا كانت نسخة قد حُذفت لكن فشل تنظيف فهرسها، فلم تعد إعادة المحاولة تتوقّف عند النسخة المفقودة؛ بل تتخطّى حذف تلك النسخة وتنظّف صفّ فهرسها مع ذلك.
- **صفّ الفهرس القديم لا يوقف التتالي.** قد يبقى صفّ فهرس بعد زوال الكيان المشتق الذي يشير إليه. وكانت عمليتا تتالي المجمِّع والوكيل تتوقّفان عند ذلك الصفّ فتتخطّيان كلّ النسخ الباقية (أو تُبقيان الصفّ)؛ أمّا الآن، كما في تتالي الرابط، فتحذفان الصفّ القديم وتسجّلان تحذيرًا وتتابعان العمل. أمّا إذا كان الكيان *المصدر* مفقودًا فعلًا، فلا يزال التتالي يُتخطّى.

حفظ مدخل في مخزن مخصّص

عند حفظ مُدخل، تسجّل المنصّة الكيانَ الذي يملك ملفاته، كي تحدّد عملية تنظيف الملفات متى لم تعد مستخدمة. وكان المُدخل المحفوظ في مخزن مخصّص [بناء التطبيقات › مشاركة مُدخلات الكيان بين المساحات]، الذي لا يحدّد **أيّ كيان إطلاقًا** (لا معرّف الكيان ولا `dataEntity` في المُدخل)، يُسجَّل تحت مالك عام لا تستطيع عملية التنظيف مطابقته بأيّ مُدخل – مما قد يؤدّي إلى حذف ملفات لا يزال المُدخل يسردها. أما الآن فهذا الحفظ **يفشل بـ** `500` (*Cannot record the files of a* `<store>` *entry: it has no entity id*) بدلًا من ذلك. أمّا عملية كتابة المُدخل العادية فتتضمّن كيانها دائمًا ولا تتأثّر؛ وتحتفظ المخازن الثابتة بنطاقاتها العامّة ونطاقات الوحدة والكيان.

ملاحظات ومحاذير للمطوّرين

- **الكتابة غير المتزامنة.** تعني استجابة `2xx` من `PUT` / `PATCH` / `POST` أن الأمر *قُبل ووُضع في الطابور*، ولا تعني أنه نجح. ويُبلَّغ عن إخفاقات التحقق داخل المعالج عبر رسالة المقبس المرتبطة بتنفيذ الأمر، لا عبر استجابة HTTP. استخدم رمز المُستدعي / ردّ المقبس لتأكيد النتيجة الفعلية.
- **هوية الحقل مهمّة.** تُبقي إعادة تسمية حقل `id` الخاص به، لذا *لا* تُعامَل على أنها حذف+إضافة في مقارنة الإصدارات. وقد تُسجَّل إعادة استخدام `normalizedName` لحقل جديد (بـ `id` جديد) بعد حذف القديم كـ `FieldRestored` .
- `source` **مهمَل** على الحقول لصالح مصفوفة `sources` ، وإن كان جزء كبير من قاعدة الرماز لا يزال يقرأ `source` لحقول الارتباط المفرد.
- تُملأ `block.internalEntity` لحقول كتل الكيان الداخلي بمعرّف الكيان الداخلي `id` أثناء التحقق، لذا لا حاجة إلى أن يضبطها العملاء بدقّة ما دام إقران الكتلة/الكيان الداخلي متّسقًا.

سجل تغيّر السلوك

تغييرات في السلوك قد يصادفها مطوّرو التكامل في بيانات أقدم أو لدى عملاء أقدم:

- **رسالة التحقق المخصصة والتحقق غير المتزامن من التفرّد (النموذج).** كانت رسالة خطأ التحقق المخصصة في الحقل قد تختفي لحظةً بعد ضبطها مباشرة، إذا كان الحقل يتضمّن أيضًا تحققًا غير متزامن من **التفرّد**: إذ كانت نتيجة التفرّد تستبدل كائن الخطأ كله، فتمحو خطأ القاعدة المخصصة مع أن الحقل ما زال غير صالح. أما الآن فتُدمج أخطاء التفرّد مع أخطاء القاعدة المخصصة.
- **صفوف باني الاستعلام المحمَّلة من استعلام محفوظ.** في باني الاستعلام المستخدَم في معايير استعلام البيانات وفي صفوف الترشيح والتعابير، كان الصف المحمَّل من استعلام محفوظ قد يبقى غير صالح إلى أن يُعدَّل يدويًا، فيمنع

استخدام Save Query / Apply على صف صالح، وكان Clear Query قد يُبقي حقلًا قديمًا محددًا في منتقي حقل الصف. وقد أُصلح الأمران.

- **مصادر المرجع في المنصة.** قبل 1.8، كان منتقي مصدر المرجع في مصمّم الكيان يحاول جلب مصادر `dataQuery` / `dataViews` على أنها كيانات، فيعرض تنبيه "Entity not found" فوق حقل مهيَّأ تهيئة صحيحة.
- **أرشفة خدمة في دليل الخدمات.** قبل 1.8، كانت أرشفة خدمة من دليل الخدمات لا تغيّر إلا صف الدليل، فتبقى الطلبات المقدَّمة سابقًا لتلك الخدمة ظاهرة في My Requests / Inbox.

مرجع واجهة برمجة الصفحات والشريط الجانبي وروابط المشاركة

هذه الصفحة موجّهة إلى **المُكامِلين والمطوّرين** الذين يقرؤون الصفحات أو يكتبونها عبر الواجهة البرمجية، أو يبنون آلية تنقّل خاصة بهم اعتمادًا على واجهة الشريط الجانبي، أو يحلّون روابط المشاركة المجهولة، أو يضيفون قطعًا إلى المنصة. وتوثّق هذه الصفحة النموذج المخزَّن، وآليات التحقّق، ونقاط النهاية، والتعدادات.

هل تبني صفحة في المصمّم؟

إن كنت من بُناة التطبيقات، فابدأ بصفحة الصفحات والقطع [بناء التطبيقات]؛ إذ تشرح مصمّم الصفحات والقطع وإعداداتها بالتسميات التي يعرضها المصمّم.

الصفحة – المعروفة داخليًا باسم **Generic View** – شاشة تتألّف من أجزاء ولا ترتبط بكيان بعينه: تضمّ قطعًا موزّعة على مجموعات ضمن **شبكة من 12 عمودًا**، وتستمدّ هذه القطع بياناتها من كيانات عديدة في آن واحد، خلافًا لمعرض البيانات الذي يعرض كيانًا واحدًا.

تغطّي الوثيقة ثلاث طبقات:

1. **المكوّنات (القطع)** – كتالوج لبنات الواجهة القابلة لإعادة الاستخدام، وكيفية ربط التعريف في الخلفية (`Component`) بمُصيِّر في الواجهة الأمامية عبر `componentName` .
2. **الصفحات (`GenericView`)** – كيفية تجميع الصفحة من المكوّنات والمجموعات، وخيارات تخطيطها وظهورها وعرضها، وأنواع الصفحات، ونقاط نهاية CRUD والتصيير للمستخدم النهائي.
3. **التنقّل وروابط النظام** – شجرة روابط الشريط الجانبي (`LinkType` = Module / View / Page / Report)، وواجهة `/v2/side-bar` ، والترشيح بحسب التفويض والوصول المقيَّد، وتدفّق رمز **system-link** المجهول.

> ملاحظة اصطلاحية: «الصفحة» في الخلفية هي الكيان `GenericView` ، وبادئة مسارها `/v1/generic-views` ، وتشير إليها طبقة الشريط الجانبي والروابط بـ `LinkType.Page` . وتستخدم هذه الوثيقة «الصفحة» و `GenericView` بالمعنى نفسه.

المكوّنات (القطع)

كتالوج المكوّنات: `Component`

`Component` (`models/component.model.ts`) هو تعريف عنصر واجهة قابل لإعادة الاستخدام – عنصر تحكّم لحقل أو قطعة صفحة. وهو مُدخل في الكتالوج له اسم وإصدار، وليس نسخة موضوعة على صفحة.

الحقل	النوع	إلزامي	المعنى
`specificationVersion`	`number` (الافتراضي `1`)	نعم	إصدار مخطط هذا التعريف (`ISpecificationVersion`).
`name`	`LocalizedValue`	نعم	اسم للعرض مقروء ومترجَم.
`type`	`number`	نعم	نوع المكوّن، وتُحدَّد قيمه في `ComponentTypeEnum` : `0` = `ViewComponent` ، `1` = `EditComponent`.

الحقل	النوع	إلزامي	المعنى
`componentName`	`string`	نعم	**معرّف ثابت** يربط هذا التعريف بمُصيِّر في الواجهة الأمامية، ويربطه أيضًا بـ**جالب** خيارات في الخلفية عند استخدامه لعنصر تحكّم في حقل.
`dataTypes`	`number[]`	نعم	أنواع بيانات الحقول التي يستطيع هذا المكوّن عرضها أو تحريرها.
`entitySource`	`string`	لا	كيان اختياري يستمدّ منه المكوّن بياناته.
`preferredPairedComponents`	`string[]`	نعم	قيم `componentName` أخرى تتلاءم مع هذا المكوّن (تلميح للواجهة).
`componentOptions`	`object[]`	نعم	مخطط تصريحي للخيارات يُعرض على باني التطبيق عند تهيئة نسخة.
`validations`	`object[]`	لا	واصفات تحقّق اختيارية.

الخاصية `idInjection` معطّلة (`settings: {idInjection: false}`)، لذا لا يحمل `Component` أي `id` مولَّد تلقائيًا؛ ويُستخدم `componentName` مفتاحًا لهويته.

`componentName` : مفتاح الربط

السلسلة `componentName` هي حلقة الوصل بين طبقات النظام كلها، وتعتمد عليها جهتان:

- **جالبو الخيارات في الخلفية.** تسجّل عناصر التحكّم في الحقول التي تحتاج إلى خيارات يزوّدها الخادم (كمنتقي المستخدمين) صنفًا فرعيًا من `Fetcher` تكون قيمة `componentName` الساكنة فيه مطابقة لقيمة المكوّن. ويفحص `FetcherBooter` (`booters/fetcher.booter.ts`) الملفات `fetchers/**/*.fetcher.js` ويربط كل صنف بالمفتاح `` `${FETCHER_BINDING_PREFIX}${cls.componentName}` ``. ويحلّ `FetcherFactoryService.getFetcher(componentName)` هذا الربط وقت الطلب (الربط اختياري – يعيد `undefined` إن لم يُسجَّل شيء). مثال: يصرّح `UserComponentFetcher` بـ `static componentName = 'UserReferenceComponent'`.

  وتنفّذ الجالبات `fetch(spaceId, data, options?)` وتتلقّى `FetcherOptions` بما يتضمّنه من خيارات وسياق (المرشِّح، و `includeIds` ، و `forceIncludeIds` ، وسياق الأب المتتالي، والبحث، والمعاملات، والمستخدم واللغة). انظر `fetchers/types.ts` .

- **سجل الواجهة الأمامية.** تحوّل `createComponentMetadata()` (`frontend/libs/shared/plugin-cdk/.../component-metadata-base.ts`) نموذج `ComponentModel` (بنية `Component`) إلى `ComponentRegisterDataModel` : فتُجري المطابقتين `componentOptions → options` و `type → componentType` ، وتطبّق القيم الافتراضية لـ `displayOptions` / `fieldQueryOptions` ، إضافةً إلى أعلام القدرات `support*` (`supportUniqueOption` ، `supportDefaultValue` ، `supportSearchable` ، `supportMultipleOptions` ، …). وبذلك يصبح مُدخل الكتالوج قطعة فعلية قابلة للتهيئة في المصمّم.

`FieldComponent` : مكوّن على حقل

`FieldComponent` (`models/field-component.model.ts`) هو الحدّ الأدنى من الربط المستخدم عند تصيير **حقل** عبر مكوّن:

```
{
  component: string;   // the componentName to render with
  options: object;     // instance configuration for that component
}
```

وهذا هو الاقتران نفسه بين `component` و`options` الذي تستخدمه قطع الصفحة (أدناه)؛ وما `FieldComponent` إلا صورته على مستوى الحقل.

القطع

القطع الأساسية المسجّلة في تطبيق الويب (`dynamic-widgets-components/metadata.ts`؛ والمجموعات في `generic-view-content/_utils/widget-group-utils.ts`) هي **فقرة** و**عمود** (مجموعتان)، و**عنوان** (مستويات العناوين 1-6)، و**نص** (متعدد الأسطر أو Markdown؛ وارتفاعه قابل للتعديل)، و**رابط** و**قائمة الروابط** (رابط خارجي، أو داخلي إلى معرض نظام أو معرض وحدة أو معرض بيانات؛ وتدعم أهداف الروابط ملء الرموز ديناميكيًا)، و**وسائط** و**قائمة الوسائط**، و**ملف** (مع صورة معاينة اختيارية) و**قائمة الملفات**، و**الخلاصة** (استعلام إحصائي)، و**معرض** (معرض بيانات؛ **عدد المدخلات** 3 / 5 / 10، والافتراضي 5، أو 3 / 5 / غير محدود حين تدعم الواجهة التقسيم إلى صفحات والبحث)، و**الشريط الزمني** (المصدر: كامل / وحدة / كيان) و**أداة التنقل** (الافتراضي 12 عمودًا، وارتفاع الصف تلقائي أو ثابت). ولكل قطعة إعدادات ظهور خاصة بها، وجميع القطع مسجّلة **داخل التطبيق** – فلا تُحمَّل قطع من أطراف ثالثة أو عن بُعد (federated)، وإضافة قطعة جديدة مهمة يتولّاها مطوّر.

وصفت نسخة سابقة من صفحة بُناة التطبيقات إعدادات القطع بتسميات أقدم (في الشريط الزمني: **Page Navigation** بخيارات Auto / Manual / Page Numbers، و**Number of Posts**، و**Show Pinned Posts**، و**Source** بخيارات Entity / Unit / General؛ وفي الخلاصة: **Entity** / **Unit** مع **View** مقفل؛ وقطعة المعرض مقفلة على عرض *Table* مع **Number of Entries**؛ و**Custom Style** و**Display** للقطعة). أما تسميات المصمّم الحالية فترد في الصفحات والقطع [بناء التطبيقات].

قطعة المعرض: تغييرات السلوك في جهة العميل

تُضمّن قطعة **معرض** معرض بيانات في صفحة، وتلتزم بواجهات ذلك المعرض وفرزه وتقسيمه إلى صفحات. وقد طرأت مؤخرًا ثلاثة تغييرات على السلوك، وجميعها في جهة العميل دون أن يتغيّر نموذج الصفحة:

- **واجهة لوحة المهام (Kanban)** `Since 1.6` – كانت قطعة المعرض التي تعرض واجهة لوحة المهام تطبّق الواجهة على العرض فقط، مع استمرارها في جلب قائمة المُدخلات المسطّحة، فتبقى اللوحة فارغة. أما الآن فتُحمِّل اللوحة نفسها، بحقل أعمدة المعرض (بما فيه، منذ 1.10، حقل أعمدة مرجعي – انظر معارض البيانات والواجهات [بناء التطبيقات › معارض البيانات وأوجهها]) و*تحميل المزيد* لكل عمود؛ ويؤدّي تبديل الواجهة في محرّر الصفحة إلى إعادة جلب الصفحة الأولى بصيغة الواجهة الجديدة. وقد سُلِّم هذا التغيير أيضًا إلى الخطّين 1.6 و1.8 بوصفه تحديثًا.
- **فرز الأعمدة** `Since 1.10` – كان فرز عمود في جدول القطعة يغيّر اتجاه سهم الترويسة دون إعادة ترتيب الصفوف. أما الآن فتحتفظ القطعة بالفرز النشط (بدءًا من الفرز الافتراضي للمعرض)، وتعود إلى الصفحة الأولى وتعيد الجلب عند كل تغيير في الفرز، وترجع إلى الفرز الافتراضي عند مسح الفرز.
- **التقسيم إلى صفحات في وجه البطاقات** `Since 1.6` – لم يعد شريط الصفحات في وجه البطاقات ملتصقًا بآخر صف من البطاقات (ووجه الجدول بلا تغيير). وقد سُلِّم هذا التغيير أيضًا إلى الخط 1.8.

حزم القطع الإضافية

تُسجَّل الحزم الاختيارية عبر ما يُصدّره (exposes) الملف `plugin-module.config.ts` في كل إضافة؛ أما القطع المخصّصة الفعلية فتوجد في التطبيق الأساسي ضمن `shared/custom-plugins/*`. وتتوقّف القطع التي تظهر لك فعلًا على الحزم المثبَّتة في بيئة النشر لديك:

- **GATE** – timeline slider، timeline hero-slider.
- **task-management** – hero-slider، quick-links، quick-services، stay-in-touch (الإعلانات)، success-stories، welcome-title، who-we-are، wiki، entity-view-summary؛ إضافةً إلى قالب مُدخل **citizen-profile**.

- **الخلاصات المركّبة** – قطعة خلاصات مجمَّعة.

ولاحظ أن **الإضافتين النموذجيتين** اللتين كانتا تُرفقان بـ SDK على سبيل المثال (`frontend/libs/plugins/gate` ، `plugins/task-mgmt`) كانتا هيكلين فارغين بلا أي سلوك مُنفَّذ، وقد أُزيل `libs/plugins/` كليًا أثناء إعادة هيكلة المكتبات؛ أما حزم القطع الفعلية المذكورة أعلاه فتوجد ضمن `frontend/apps/core/src/app/shared/custom-plugins/*` ، وهو موضعها منذ البداية.

الصفحات: نموذج `GenericView`

حقول `GenericView`

الصفحة كائن من النوع `GenericView` (`models/generic-view/generic-view.model.ts`)، ولها معرّف من النوع `UUID` . ويحمل هذا الكائن هوية الصفحة، ومحتواها (المكوّنات والمجموعات)، وسلوكها (المشاركة، والوصول المجهول، والتحديث، وإعادة التوجيه). فيما يلي حقول مختارة:

الحقل	النوع	ملاحظات
`specificationVersion`	`number` (الافتراضي `1`)	إصدار مخطط الصفحة.
`name`	`LocalizedValue`	إلزامي؛ اسم الصفحة المترجَم.
`normalizedName`	`string`	معرّف نصي آمن للاستخدام في URL، يُتحقَّق من صحته باستخدام `NormalizedNameValidation` (الحدّان الأدنى والأقصى للطول + النمط). ويُولَّد تلقائيًا من `name` عند غيابه (انظر تدفّق الإنشاء).
`description`	`NullableLocalizedValue`	اختياري.
`icon`	`Icon`	اختياري.
`anonymous`	`AnonymousViewOption | null`	إعدادات الوصول المجهول، بما فيها روابط المشاركة ورموزها. ويستند إنشاء روابط النظام إلى هذا الحقل.
`authenticated`	`AuthenticatedViewOption | null`	خيارات الوصول للمستخدمين المصادَق عليهم (مثل المعرض الافتراضي والمقيَّد).
`isPublished`	`boolean` (الافتراضي `false`)	حالة النشر.
`isPinned`	`boolean` (الافتراضي `false`)	حالة التثبيت.
`relatedTo`	`NullableReferenceProperty`	إذا ضُبط على `{dataEntity:'DataModule', dataEntry:<uuid>}` ، فإنه يقصر الصفحة على وحدة؛ أما `null` أو غياب الحقل فيعني أن الصفحة على مستوى النظام أو المساحة.
`components`	`GenericViewComponent[]` (الافتراضي `[]`)	القطع في المستوى الأعلى من الصفحة.
`groups`	`GenericViewGroup[]` (الافتراضي `[]`)	مجموعات التخطيط؛ ولكلٍّ منها `components` خاصة بها.
`sideBarConfig`	`SideBarConfig | null`	يتحكّم في فتح الشريط الجانبي وإغلاقه عند فتح الصفحة.
`displayConfig`	`GenericViewDisplayConfig | null`	أعلام إعداد إطار الصفحة (الشفافية، والصفحة الكاملة، والترويسة، والتباعد، وتخطيط الحاوية).
`displayOrder`	`number` (الافتراضي `0`)	تلميح للترتيب.
`uiAttributes`	`object | null`	بيانات وصفية حرّة للواجهة.
`allowExportPDF`	`boolean` (الافتراضي `false`)	يتيح تصدير الصفحة إلى PDF.
`domain`	`ReferenceProperty`	قصر الصفحة على نطاق، اختياريًا.

الحقل	النوع	ملاحظات
`autoRefresh`	`GenericViewAutoRefresh | null`	تحديث تلقائي `{enabled, interval}`.
`creationInfo` / `updateInfo`	`ActionInfo`	أختام التدقيق (مَن نفّذ الإجراء ومتى، والمفوَّض إليه).
`pageType`	`PageTypeEnum` (الافتراضي `Normal`)	`0 = Normal`، `1 = Embedded`.
`embeddedPageConfig`	`EmbeddedPageConfig`	`{url}` – إلزامي، ويُتحقَّق من كونه عنوان URL صالحًا عندما يكون `pageType = Embedded`.
`sharing`	`ApplicationSharing`	إعدادات المشاركة الإلزامية (Private/Public/Shared).
`redirectionRules`	`RedirectionRule[]` (الافتراضي `[]`)	عمليات إعادة توجيه شرطية تُقيَّم للصفحة – انظر قواعد إعادة التوجيه: التقييم أدناه.
`updateIdentifier`	`string`	إلزامي: يُعاد توليده (`v4()`) عند الإنشاء – رمز للتزامن التفاؤلي وتتبّع التغيير.
`help`	`PageHelp | null`	مساعدة أو توثيق اختياري داخل الصفحة. **Since 1.2.**
`executionPlan`	`ExecutionPlan | null`	خطة محسوبة للمسارات المسموح بها (يضبطها `calculateExecutionPlan`).

يهيّئ مُنشئ `GenericView` الحقلين `components = []` و`groups = []` عند غيابهما. لذلك يجب ألّا يستخدم أي ترحيل `updateById` لتعديل صفحة، إذ سيعيد بناء الكيان ويفرّغ المصفوفتين؛ بل يجب استخدام `$set` خامًا بدلًا منه.

وضع قطعة: `GenericViewComponent`

كل نسخة من قطعة على الصفحة هي كائن من النوع `GenericViewComponent` (`generic-view-component.model.ts`):

الحقل	النوع	إلزامي	ملاحظات
`componentId`	`UUID`	نعم	فريد في الصفحة – ويُفرض ذلك عند التحقق (انظر التحقق).
`component`	`string`	نعم	قيمة `componentName` المستخدمة لتصيير القطعة.
`options`	`{[key:string]: any}`	نعم	إعدادات النسخة. وقد تتضمّن `options.groupId` لوضع القطعة داخل مجموعة.
`displayOptions`	`ComponentDisplayOptions`	نعم	الحجم والعرض والمحاذاة والارتفاع (أدناه).
`description` / `title`	`NullableLocalizedValue`	لا	قيم تحلّ محلّ القيم الافتراضية على مستوى النسخة.
`icon`	`Icon`	لا	أيقونة على مستوى النسخة.
`uiAttributes`	`object | null`	لا	بيانات وصفية حرّة للواجهة.
`visibility`	`ComponentVisibility`	لا	قواعد الظهور للمستخدمين المجهولين والمصادَق عليهم (أدناه).

الشبكة والأحجام: `ComponentDisplayOptions`

الحقل	النوع	الافتراضي	ملاحظات
`size`	`ComponentSizeEnum`	إلزامي	مساحة محدّدة مسبقًا على الشبكة: `Row1Col1` (0) … `Row2Col4` (7).
`width`	`number` (1-12)	`1`	العرض من أصل 12 عمودًا.
`verticalAlginment`	`VerticalAlignmentEnum`	`0 = Stretch`	`Stretch/Start/End/Center` (ملاحظة: اسم الحقل مكتوب `verticalAlginment` في المصدر).
`showInNewRow`	`boolean`	`false`	فرض وضع القطعة في صف جديد.
`height`	`ComponentHeightConfiguration`	–	`{advanced, unit (default 'px'), value (default 100, min 0)}`.

الظهور الشرطي: `ComponentVisibility`

يقيّد `visibility` (`component-visibility.model.ts`) ظهور القطعة بحسب الجمهور:

- `custom: boolean` (الافتراضي `false`) – عندما تكون قيمته `false` ، ترث القطعة إعدادات الظهور الافتراضية للصفحة؛ وعندما تكون `true` ، تُطبَّق `settings` .
- `settings: ComponentVisibilitySettings` – يتضمّن مبدّلَي `anonymous` (`AnonymousOption`) و `authenticated` (`AuthenticatedOption`)، مما يتيح إظهار القطعة أو إخفاءها بصورة مختلفة للزوار المجهولين وللمسجّلين.

ويُتحقَّق من إعدادات ظهور المكوّن عند الإنشاء والتحديث وفق إعدادات الصفحة نفسها للمستخدمين المجهولين والمصادَق عليهم (`validateComponentVisibility`)، ويشمل ذلك المكوّنات الموضوعة داخل المجموعات. وتظهر هذه الإعدادات في المصمّم عبر المبدّل **عرض مخصص** في قسم **عام** من القطعة.

التجميع والتخطيط: `GenericViewGroup`

`GenericViewGroup` (`generic-view-group.model.ts`) حاوية للتخطيط:

- `groupId: UUID` – فريد في الصفحة.
- `options: GroupConfiguration` – `{groupId?, orientation}` ، حيث `orientation` من النوع `LayoutOrientationEnum` (`0 = Horizontal` افتراضيًا، `1 = Vertical`). (ويُعزَّف `GroupConfiguration` بالإعداد `strict:false` ، لذا قد توجد مفاتيح تخطيط إضافية.)
- `components: GenericViewComponent[]` – القطع داخل المجموعة.

وينضمّ مكوّن من المستوى الأعلى إلى مجموعة بضبط `options.groupId` على قيمة `groupId` الخاصة بالمجموعة. ويتحقّق تدفّق الإنشاء والتحديث من وجود كل مجموعة مُشار إليها.

إعدادات العرض والشريط الجانبي والتحديث على مستوى الصفحة

- **`GenericViewDisplayConfig`** (يقبل null): `transparent` ، و `fullPage` ، و `outerSpacing` ، و `showPageHeader` ، و `pageContainerLayout` الحرّ.
- **`SideBarConfig`** (يقبل null): `openMode: SideBarOpenModeEnum` – `0 = Preserve` (الافتراضي)، `1 = Hide` ، `2 = Open`، `3 = Close` . ويحدّد سلوك شريط التنقّل الجانبي عند فتح هذه الصفحة.
- **`GenericViewAutoRefresh`** (يقبل null، `strict:false`): `{enabled (default false), interval (default 0)}` .
- **`LinkConfiguration`** (`link-configuration.model.ts`) – واصف ارتباط تشعّبي قابل لإعادة الاستخدام (تستخدمه القطع والإعدادات، ويختلف عن روابط الشريط الجانبي): `type: LinkTypeEnum` (`0 = External` ، `1 = internal`)، و `url?` ، و `description` / `title` مترجَمان، و `icon` ، و `openIn: OpenLinkInEnum` (`0 = NewTab` ، `1 = SameWindow` ، `2 = NewWindow`).

أنواع الصفحات: Normal وEmbedded

تكون قيمة `pageType` (`PageTypeEnum`) إما `Normal (0)` أو `Embedded (1)` . وتعرض الصفحة **Embedded** عنوان URL خارجيًا عبر `embeddedPageConfig.url` . وعند الإنشاء والتحديث، تفرض الخدمة (`validateEmbeddedPageConfig`) ما يلي:

- وجود عنوان URL ← وإلا يُطلَق `EmbeddedPageMustHaveURLException` ؛
- صحة عنوان URL ← وإلا يُطلَق `EmbeddedPageInvalidURLException` .

تسميات المصمّم والنموذج المخزَّن

المصمّم (تبويب التفاصيل / المحتوى)	يُخزَّن في
نوع الصفحة صفحة عادية / صفحة مضمنة، و**الرابط**	`pageType` (`Normal = 0` ، `Embedded = 1`)، `embeddedPageConfig.url`
تثبيت إلى الشريط الجانبي	`isPinned`
عرض الشريط الجانبي احتفاظ / فتح / إغلاق	`sideBarConfig.openMode` (`Preserve = 0` ، `Open = 2` ، `Close = 3` ؛ ولا يوجد خيار في المصمّم لـ `Hide = 1`)
الظهور: خلفية شفافة، حشو للصفحة، حشو للمحتوى، **إظهار عنوان الصفحة**، صفحة بعرض محدد	`displayConfig.transparent` ، `fullPage` ، `outerSpacing` ، `pageContainerLayout` ، `showPageHeader`
السماح بتصدير PDF	`allowExportPDF`
التحديث التلقائي	`autoRefresh` `{enabled, interval}`
قواعد التحويل (مربع نص بصيغة JSON)	`redirectionRules`
المساعدة والتوثيق	`help`
مستوى المشاركة، المساحات	`sharing` (`ApplicationSharing`)
في القطعة: **تحرير العرض** / **المحاذاة العمودية** (**تمديد**، **الاعلى**، **الاسفل**، **وسط**) / **صف جديد** / **تحرير الارتفاع**	`displayOptions.width` ، `verticalAlginment` (`Stretch` ، `Start` ، `End` ، `Center`)، `showInNewRow` ، `height`
في القطعة: **عرض مخصص**	`visibility.settings` + `visibility.custom`
القطعتان **فقرة** / **عمود**	`GenericViewGroup`

قواعد إعادة التوجيه: التقييم

توجّه `redirectionRules` المستخدمين إلى صفحة هدف محدّدة بناءً على شروط. وتُرتَّب القواعد بحسب الأولوية، وتسري أول قاعدة مطابقة. ولا تتحقّق المطابقة إلا عندما يستوفي المستدعي **كلَّ** من متطلبات نوع الوصول للقاعدة (Anonymous/Authenticated/Everyone/Restricted) وقوائم المستخدمين والمجموعات المسمّاة فيها. و `Since 1.10` تُفرَض هذه المطابقة الثنائية بصورة صحيحة في قواعد الصفحة نفسها (إذ كان الجمهور المسمّى للقاعدة يُتجاهل بصمت خارج حالة Restricted، وكان الزائر المجهول قد يواجه عطلًا تامًا بدلًا من إعادة التوجيه أو العرض بصورة عادية). وتشترك الصفحات ومعارض البيانات في المُقيِّم نفسه تمامًا؛ انظر معارض البيانات والواجهات § قواعد إعادة التوجيه [بناء التطبيقات › معارض البيانات وأوجهها] للاطّلاع على دلالات التقييم كاملة. وتُنشأ صفحة رئيسية أو افتراضية تلقائيًا عند إنشاء وحدة أو مساحة.

واجهة الصفحات (GenericView) البرمجية

نقاط النهاية

تقع كل المسارات في `data-service` . ويُفوَّض CRUD عبر مصوِّتات `voteByRelatedToFilterFactory('relatedTo', …)` ؛ أما مسارات التصيير للمستخدم النهائي فمنفصلة.

الطريقة	المسار	الغرض
`POST`	`/v1/generic-views`	إنشاء صفحة (جسم `NewGenericView`).
`GET`	`/v1/generic-views`	سرد الصفحات.
`GET`	`/v1/generic-views/count`	عدّ الصفحات (مرشِّح `where`).
`GET`	`/v1/generic-views/default`	جلب الصفحة الافتراضية للمستخدم الحالي.
`GET`	`/v1/generic-views/{id}`	جلب صفحة بالمعرّف (عرض المصمّم).
`GET`	`/v1/generic-views/end-user/{id}`	جلب صفحة للتصيير (المستخدم النهائي).

الطريقة	المسار	الغرض
`GET`	`/v2/generic-views/end-user/{id}`	التصيير للمستخدم النهائي، الإصدار v2.
`PATCH`	`/v1/generic-views/{id}`	تحديث صفحة.
`PATCH`	`/v1/generic-views/{id}/permissions`	تحديث أذونات الصفحة فقط.
`DELETE`	`/v1/generic-views/{id}`	حذف صفحة.

تدفّق الإنشاء: التحقق والآثار الجانبية

ينفّذ `POST /v1/generic-views` (`create`) الخطوات التالية بالترتيب:

1. `validateName` على الاسم المترجَم.
2. `validateGenericView` – فلا يمكن، مثلًا، أن يكون المعرض المصادَق عليه المقيَّد هو المعرض الافتراضي أيضًا (`RestrictedViewCanNotBeDefaultException`)؛ ويجب أن يكون `relatedTo` ، إن وُجد، من النوع `ReferenceProperty` وصالحًا.
3. `validateComponentUniqueness` – يجب أن تكون قيم `componentId` فريدة في الصفحة كلها (`ComponentIdMustBeUniqueException`).
4. `validateGroupExistence` – يجب أن يطابق `options.groupId` في كل مكوّن مجموعةً موجودة (`ComponentGroupIdNotExistException`).
5. `validateGroupsUniqueness` – يجب أن تكون قيم `groupId` فريدة (`ComponentGroupIdMustBeUniqueException`).
6. `validatePermissions` و `validateComponentVisibility` (تعاوديًا داخل مكوّنات المجموعات).
7. إن كان `pageType = Embedded` ، يُنفَّذ `validateEmbeddedPageConfig` .
8. **normalizedName**: التفرّد مقيَّد بنطاق. ويكون النطاق على مستوى الوحدة حين يُضبط `relatedTo` ، وإلا فيشمل كل الصفحات على مستوى المساحة (`relatedTo` null أو غائب). ويجب أن يكون `normalizedName` المقدَّم فريدًا في نطاقه (`GenericViewNormalizedNameDuplicateException`)؛ وإلا يُولَّد تلقائيًا من `name` باستخدام `generateUniqueNormalizedName(..., 'page')` .
9. أختام التدقيق (`creationInfo` / `updateInfo`)، وقيمة جديدة لـ `updateIdentifier` (`v4()`)، و `calculateExecutionPlan` (المسارات الساكنة المسموح بها).
10. الحفظ، ثم الآثار الجانبية: إن كان `anonymous.enabled` ، تُنشأ روابط النظام عبر `SystemLinkService.upsertSystemLink` ؛ ويُفرض تفرّد المعرض الافتراضي؛ وتُخزَّن الأذونات في حقوق الوصول؛ ويُنشر `GenericViewCreatedEvent` .

تدفّق التحديث: التزامن والصفحات المثبَّتة `Since 1.10`

يقارن `PATCH /v1/generic-views/{id}` قيمة `updateIdentifier` الواردة بالقيمة المخزَّنة (`409 UPDATE_CONFLICT`)، ثم يكتب **كتابة مشروطة**: فلا تُحجز الصفحة إلا ما دام رمزها المخزَّن مطابقًا للرمز الذي قُرئ، ثم تُستبدل برمز جديد. وبذلك، إذا اجتازت عمليتا حفظ المقارنة كلتاهما، لم يعد بإمكان إحداهما الكتابة فوق الأخرى؛ إذ تتلقّى الثانية 409، وعليها إعادة القراءة. كما تسجّل عملية حفظ الصفحة الملفات التي تشير إليها الصفحة قبل الكتابة، ثم تحرّر الملفات التي أسقطتها بعدها؛ أما حذف الصفحة فيحرّرها كلها.

والصفحة **التي ثبّتتها حزمة** مقفلة (`403 PACKAGE_RESOURCE_READ_ONLY`). وعبر `PATCH` نفسه، ولجسم يغيّرها فقط، يحتفظ العميل بما يلي:

- `isPublished` – ولا يُقارن أبدًا بالرمز.
- `name` و`description` و`icon` . يسجّل التغيير الأول قيمة الحزمة الأصلية في `packageLabels` ، وتؤدي إعادة كتابة قيمة الحزمة إلى إزالة هذا السجل؛ ويُبقي تحديث الحزمة تسمية العميل. وتتطلّب كتابة التسمية قيمة `updateIdentifier` الحالية. ولا يمسح `icon: null` الأيقونة إلا مع `?clearIcon=true` ، لأن محرّر الصفحة يرسل `icon: null` لكل صفحة لا أيقونة لها؛ أما `description: null` فيمسح دائمًا.

والكتابة **جزئية**: تبقى `components` و`groups` المخزَّنة كما هي تمامًا، ويُحتفظ بتعيينات الصفحة الافتراضية القائمة بدلًا من أخذها من الحمولة، ولا تُضبط إلا المفاتيح المختارة وختم التدقيق وقيمة جديدة لـ `updateIdentifier` . وللعميل أيضًا حقّ تعديل ترتيب العرض والجمهور وصفوف حقوق الوصول (`PATCH /v1/generic-views/{id}/permissions`)، ويُبقيها تحديث الحزمة كما هي. وتسري القواعد نفسها على المعارض في معارض البيانات [بناء التطبيقات ‹ معارض البيانات وأوجهها].

الصفحات العامة في الحزم `Since 1.10`

يمكن أن تكون **الصفحة العامة** – وهي صفحة لا نطاق لها في وحدة أو كيان (`relatedTo` غائب) – جذرًا لحزمة، وحدها أو إلى جانب وحدات، وتُصدَّر بلا أبناء. ويُرفض حفظ حزمة تحدّد صفحة تنتمي إلى وحدة أو كيان بـ `422 PACKAGE_PAGE_NOT_GLOBAL` ؛ كما يُرفض نشر صفحة انتقلت منذ ذلك الحين إلى أحدهما (`PUBLISH_PAGE_NOT_GLOBAL`)، أو كانت هي نفسها مثبَّتة أو مشتقّة من حزمة أخرى (`PUBLISH_PAGE_INSTALLED` ، `PUBLISH_PAGE_FORKED`). ومع ذلك، تُثبَّت قطعة الاستعلام الإحصائي في الصفحة إذا كان مُدخل بياناتها يشير إلى خارج الحزمة؛ إذ تُترك القطعة **غير مهيّأة** – وهي الحالة نفسها التي ينتجها محرّر القطعة – بدلًا من إفشال التثبيت. وانظر شاشات التحزيم لمعرفة مواضع إضافة الصفحات إلى حزمة.

التنقّل: شجرة روابط الشريط الجانبي

الشريط الجانبي شجرة من **الروابط**. ويحمل الرابط `LinkDataEntry` قيمة `value` من النوع `LinkDataEntryValue` ، وتعيدها واجهة الشريط الجانبي في صورة عُقد `GenericLink` .

`LinkDataEntryValue` (الشكل المخزَّن)

الحقل	النوع	ملاحظات
`order`	`number`	ترتيب الفرز بين الروابط الشقيقة.
`parent`	`ReferenceProperty | null`	الرابط الأب (يبني الشجرة).
`icon`	`Icon`	
`published`	`boolean`	لا تُعاد إلا الروابط ذات `published:true` و `state:1` .
`type`	`CollectionValueModel<LinkType>`	Module / View / Page / Report / Untyped.
`ownership`	`CollectionValueModel<LinkOwnership>`	`System ('0')` أو `Manual ('1')` .
`title`	`LocalizedValue`	إلزامي.
`description`	`NullableLocalizedValue`	
`target`	`LinkTarget`	`{url, targetId, targetSectionId, targetViewAction}` .
`authentication`	`LinkAuthentication`	`{type, users[], groups[]}` – يحدّد من يستطيع رؤية الرابط.
`domain`	`ReferenceProperty[]`	المصفوفة الفارغة = كل النطاقات؛ وإلا فيجب أن تطابق نطاق الطلب.
`sharing`	`ApplicationSharing`	Private/Public/Shared مقارنةً بمساحة الطلب.

`LinkType` (`@osos/shared`): `Module = '0'` ، `View = '1'` ، `Page = '2'` ، `Report = '3'` ، `Untyped = '4'` .

`LinkAuthenticationTypeEnum` : `Anonymous = '0'` ، `Authenticated = '1'` ، `Everyone = '2'` ، `Restricted = '3'` .

`GenericLink` (الشكل المُعاد) واشتقاق URL

عقدة الاستجابة هي `GenericLink` (`generic-link.model.ts`)، وتضمّ: `id` ، و `order` ، و `title` ، و `description` ، و `icon` ، و `url` ، و `target` ، و `type` ، و `parent` ، و `children[]` ، إضافةً إلى الحقول الخاصة بالمعارض: `userQueries` و `userCollections` و `sharedCollections` . وتضبط الأصناف الفرعية `url` الخاص بالعميل انطلاقًا من `target.targetId` :

نوع الرابط	الصنف	`url` المشتق
Module	`ModuleLink`	`null` (حاوية فقط)
Page	`PageLink`	`/${targetId}`
View	`ViewLink`	`/v/${targetId}` (ويحمل أيضًا الاستعلامات المحفوظة والمجموعات)
Report	`ReportLink`	`/reports/${targetId}` (على الويب فقط)
غير ذلك	`GenericLink`	`target.url` كما هو مخزَّن

`GET /v2/side-bar`

تعيد `SideBarController.getViewsForSidebarV2` كائنًا من النوع `SidebarDtoV2` .

الترويسات والمعاملات:

- `x-domain-id` (ترويسة) – نطاق الطلب.
- `x-space-id` (ترويسة) – مساحة الطلب.
- `include` (معامل استعلام، `SidebarIncludeDataEnum`): `All = 0` (الافتراضي)، `Links = 1` ، `OnlyService = 2` . **ومحدِّد `include` متاح Since 1.4.**

حقول `SidebarDtoV2` :

الحقل	النوع	ملاحظات
`links`	`GenericLink[]`	شجرة الروابط (فارغة حين يكون `include = OnlyService`).
`serviceViewsStatus`	`SidebarServiceStatus | null`	عدّادات قسم الخدمات (null حين يكون `include = Links`).
`isRestricted`	`boolean`	هل يحمل المستخدم الحدّ الأدنى `Restricted` على مستوى النظام. **Since 1.8.**
`restrictedAccess`	`{hasModuleAccess, hasEntityAccess}`	منح على مستوى الموارد للمستخدمين المقيَّدين؛ وتكون قيمة كلٍّ منهما `true` إن وُجدت أي منحة على وحدة أو كيان ≤ `AccessRightLevelEnum.ViewAccess` . **Since 1.8.**

وتحمل نقطة النهاية `@authorize({skip: true})` ، لكنها تتطلّب المصادقة (استراتيجية `Auth2` على مستوى المتحكّم). ويُرفق ملخّص الوصول المقيَّد مع **كل** قيم `include` ليبقى شكل كائن DTO ثابتًا.

كيف تُختار الروابط وتُجمَّع

`LinksService.getLinksForSidebar(domain, spaceId, userId, delegationId?)` :

1. **ترشيح الجمهور** بحسب `authentication.type` :
   - الطلب المجهول (`userId` قيمته null): النوعان `0` (Anonymous) أو `2` (Everyone)، إضافةً إلى النوع `1` (Authenticated) فقط حين يكون `value.type.id === '0'` للرابط نفسه.
   - الطلب المصادَق عليه: النوعان `1` (Authenticated) أو `2` (Everyone)، إضافةً إلى `3` (Restricted) حين يكون المستخدم مُدرجًا في `authentication.users` أو عضوًا في إحدى `authentication.groups` .

2. **الشرط الأساسي**: `published:true` و `state:1` ؛ والنطاق مطابق (مصفوفة `domain` الفارغة = عام، وإلا `domain.dataEntry === domain`)؛ والمشاركة `Public` ، أو `Shared` / `Private` مقصورة على مساحة الطلب `spaceId` .
3. **الترتيب**: `value.order ASC` ، ثم `value.title.en ASC` .
4. **ترشيح التفويض** (حين يُضبط `delegationId`): تُسقط الروابط التي لا يملك المفوَّض إليه إذنًا على هدفها؛ إذ تُفحص أهداف `view` / `report` / `page` مقابل استحقاقات `view` / `report` / `genericView` المفوَّضة (`extractResourceInfoFromLink` + `filterLinksByDelegatedPermissions`). ويُحتفظ بالروابط التي لا تشير إلى مورد خاضع للتحكّم.
5. **إثراء المعارض**: تُرفق بروابط `View` استعلامات المستخدم المحفوظة (`userQueries`)، ومجموعاته المملوكة (`userCollections`)، والمجموعات المشارَكة (`sharedCollections`).
6. **بناء الشجرة** (`convertLinksToTree`): تُبنى علاقة التداخل بين الأب والابن عبر `value.parent` . ويُتخطّى رابط `View` الذي يكون أبوه مخفيًا. وأخيرًا، فإن روابط `Module` **التي لا أبناء لها تُسقط** (فالوحدات الفارغة لا تظهر).

روابط النظام (رموز المشاركة المجهولة)

«روابط النظام» رموز معتمة (opaque) تتيح للزوار المجهولين الوصول إلى معرض أو صفحة دون كشف معرّفها. وهذا مفهوم مختلف عن روابط الشريط الجانبي.

النموذج والتخزين

- رابط المشاركة `ViewUrl` : `{uniqueToken: string, validity?: string | null}` .
- في الصفحة أو المعرض، يحمل `anonymous` (`AnonymousViewOption`) الحقول `enabled` و `defaultView` و `urls:` `ViewUrl[]` والحقل الاختياري `humanVerification` .
- تُحفظ الرموز في صورة سجلات `SystemLinkEntry` (`value.relatedObject` ، `value.token` ، `value.validity`)، وتكتبها `SystemLinkService` .

توليد الرموز ومواءمتها

- `generateUniqToken(size)` – ينشئ رمز `nanoid(size)` ويعيد المحاولة عند حدوث تصادم (وهو نادر). `SYSTEM_TOKEN_LENGTH = 32` .
- `upsertSystemLink(relatedObject, links, user, isUpdate?)` – يُدرج الرموز الجديدة، وعند التحديث **يؤرشف** أي رمز كان نشطًا ولم يعد موجودًا في القائمة المرسلة (الحالة ← `Archived`). ويُستدعى تلقائيًا من تدفّق إنشاء الصفحة حين يكون `anonymous.enabled` .

واجهة روابط النظام

الطريقة	المسار	المصادقة	الغرض
`GET`	`/v1/system-link/generate`	مصادَق عليه	يعيد `{token}` ، وهو رمز فريد جديد من 32 محرفًا لإرفاقه بعنوان URL مجهول.
`GET`	`/v1/system-link/resolve/{token}`	`@authenticate.skip()` (عام)	يحلّ الرمز إلى الكيان الذي يشير إليه.

دلالات الحلّ (`getAnonymousView`):

1. يجب أن يساوي طول الرمز `SYSTEM_TOKEN_LENGTH` (32) ← وإلا `InvalidViewTokenExption`.
2. البحث عن `DataViews` بالرمز، وإلا عن `GenericView` بالرمز ← فإن لم يوجد أيٌّ منهما، `DataViewNotExistException`.
3. عند النجاح، يُعاد `ReferenceProperty`: `{dataEntity: 'DataViews' | 'GenericView', dataEntry: <view.id>}`. ثم يحمّل المستدعي المعرض أو الصفحة الفعلية عبر مسار التصيير المعتاد.

مرجع سريع: التعدادات

التعداد	القيم
`ComponentTypeEnum`	`ViewComponent=0`، `EditComponent=1`
`ComponentSizeEnum`	`Row1Col1=0` … `Row2Col4=7`
`VerticalAlignmentEnum`	`Stretch=0`، `Start=1`، `End=2`، `Center=3`
`LayoutOrientationEnum`	`Horizontal=0`، `Vertical=1`
`SideBarOpenModeEnum`	`Preserve=0`، `Hide=1`، `Open=2`، `Close=3`
`PageTypeEnum`	`Normal=0`، `Embedded=1`
`LinkType`	`Module='0'`، `View='1'`، `Page='2'`، `Report='3'`، `Untyped='4'`
`LinkOwnership`	`System='0'`، `Manual='1'`
`LinkAuthenticationTypeEnum`	`Anonymous='0'`، `Authenticated='1'`، `Everyone='2'`، `Restricted='3'`
`LinkTypeEnum` (LinkConfiguration)	`External=0`، `internal=1`
`OpenLinkInEnum`	`NewTab=0`، `SameWindow=1`، `NewWindow=2`
`SidebarIncludeDataEnum`	`All=0`، `Links=1`، `OnlyService=2`
`ApplicationSharingLevelEnum`	`Private='PRIVATE'`، `Public='PUBLIC'`، `Shared='SHARED'`

نموذج التفاعل ومرجع واجهته البرمجية

هذه الصفحة موجّهة إلى **مطوّري التكامل والمطوّرين** الذين يقرؤون التفاعلات أو يكتبونها عبر الواجهة البرمجية، أو يبنون أدوات للتعامل معها، أو يحتاجون إلى معرفة عقد التشغيل بدقّة: النموذج المخزَّن وتعداداته الرقمية، وكل نوع من الأحداث والإجراءات وخياراته، وكيفية تقييم الشروط، وتنفيذ عمليات التشغيل وتسلسلها وتسجيلها، ونقاط النهاية.

> **تبني تفاعلًا من الواجهة؟**
>
> ينبغي لبُناة التطبيقات الذين يضيفون التفاعلات في مصمّم الكيان أن يبدؤوا من التفاعلات والأتمتة [بناء التطبيقات]؛ فالصفحة تغطّي الميزات نفسها بالتسميات التي يعرضها المصمّم، مرتّبةً حسب الهدف.

يُسمّى التفاعل في الواجهة البرمجية وفي الرماز **trigger**، ونموذجه المخزَّن هو `DataReaction`. يرتبط التفاعل بنوع **حدث** واحد على **كيان معرَّف**، ويمكن تضييق نطاقه اختياريًا عبر **شروط** (مطابِق على مستوى الحدث، إضافةً إلى مرشّح معايير/استعلام)، ثم يشغّل **إجراءه**. وتستطيع الإجراءات إنشاء البيانات أو تحديثها أو أرشفتها، وتشغيل سير العمل، واستدعاء واجهات برمجة خارجية، وإرسال الإشعارات والبريد الإلكتروني والرسائل النصية أو رسائل WhatsApp، وتوليد الملفات، وتشغيل دوال مخصّصة، ونشر أحداث جديدة تستمع إليها تفاعلات أخرى، مما يتيح للتفاعلات أن **تتسلسل**.

البنية: خدمتان

يقع محرّك الأتمتة في خدمة مصغّرة خاصّة به، `reaction-service`. و**لا** تنفّذ خدمة `data-service` التفاعلات بنفسها؛ إذ يعمل `ReactionServiceProvider` / `ReactionServiceWrapper` فيها (`backend/services/data-service/src/services/reaction.service.ts`) بوصفه **وكيل خدمة** (service proxy) في LoopBack على مصدر البيانات `restds`، ويمرّر `getEntityReactions` / `getReaction` إلى خدمة التفاعلات المصغّرة، مع تغليف الاستدعاءات بـ `async-retry` (تُوقِف أخطاء العميل `4xx` المحاولات فورًا عبر `bail`؛ أمّا الأخطاء الأخرى فتؤدّي إلى إعادة المحاولة). ويحتوي مجلّد النماذج `data-entity-action` في `data-service` على نسخ مطابقة من كائنات نقل البيانات (DTOs) لخيارات الإجراءات، لكن النماذج المعتمَدة التي تُنفَّذ فعليًا هي الموجودة في `reaction-service/src/models`.

تُخزَّن التفاعلات في MongoDB (مجموعة `DataReaction`) وتُشغَّل بواسطة **أحداث المجال المُستهلَكة من ناقل الخدمة** (`@node-ts/bus`)، لا عبر استدعاءات REST مباشرة. ولكل قيمة من `EventTypeEnum` معالِج لحدث الناقل في `handlers/events/**` يحدّد المُدخلات المتأثّرة ويرسلها إلى `EventsHandler` المركزي.

أمّا آليات المحرّك الداخلية التي يستند إليها ذلك، كالإرسال والتوزيع المجزّأ وآليات sagas والاستعادة، فموثّقة لمهندسي المنصة في المحرّك: التفاعلات.

نموذج المشغّل (`DataReaction`)

المصدر: `models/data-reaction/data-reaction.model.ts`. الحقول الرئيسية:

الحقل	النوع	المعنى
`id`	UUID	معرّف المشغّل
`specificationVersion`	number (default 1)	إصدار مخطّط التفاعل

الحقل	النوع	المعنى
`normalizedName`	string	الاسم المعياري (النمط/الطول من `NormalizedNameValidation`)
`name`	LocalizedValue	الاسم المعروض
`description`	NullableLocalizedValue	وصف اختياري يُعرَض في تبويب التفاعلات. `Since 1.10`
`definingEntity`	ReferenceProperty	الكيان المرتبط به التفاعل (`{dataEntity:'DataEntity', dataEntry:<entityId>}`)
`eventType`	`EventTypeEnum`	الحدث الذي يُطلق هذا التفاعل (انظر أدناه)
`eventConfig`	`DataReactionEventConfig` union	إعداد خاص بكل حدث + شروط
`executorType`	`EventExecutorTypeEnum` (`SYSTEM` / `MANAGER` / `EXECUTOR`)	الهوية/الصلاحيات التي تُشغِّل الإجراءات بموجبها
`sensitiveDataAccessPolicy`	`SensitiveDataAccessPolicyEnum`	ما يُسمح لإجراءات التفاعل برؤيته من **الحقل الحساس** في المُدخلات التي تتعامل معها؛ وغيابه يعني `Privileged`. انظر سياسة الوصول إلى البيانات الحساسة. `Since 1.10`
`query`	ReferenceProperty | null	`DataQuery` محفوظ اختياري تُدمَج معاييره بعامل AND ضمن الشرط
`criteria`	`FilterExpression`	مرشّح شرط اختياري يُقيَّم في سياق التشغيل
`actions`	`DataReactionAction[]`	الإجراءات المراد تنفيذها، مرتّبةً حسب تسلسل التنفيذ
`type`	`DataReactionTriggerTypeEnum` (`SLATrigger=1` . `NormalTrigger=0`)	مشغّل عادي أو مشغّل SLA
`sourceEntity`	null | `DataReactionSourceEntity`	يُطلَق من كيان مرتبط/مطلق بدلًا من الكيان نفسه (انظر تفاعلات الكيانات المرتبطة)
`relatedEntities`	`DataReactionRelatedEntities[]`	الكيانات المرتبطة المُدرَجة في السياق
`spaceScope`	`DataReactionSpaceScope`	المساحات التي يُنفَّذ فيها التفاعل (`rootSpace` / `currentSpace` / `otherSpaces`)
`loopThreshold`	number (default 0)	الحد الأقصى لحلقات إعادة التنفيذ لكل إجراء؛ `0` = يُشغَّل مرة واحدة (انظر الحماية من الحلقات)
`isImplicit`	boolean	تفاعل يولّده النظام (محميّ من عمليات CRUD الخاصة بالمستخدم)
`disabled`	boolean	يتخطّاه المُرسِل عندما يكون `true`
`validations`	null | `DataReactionValidation`	حالة الصلاحية؛ تُتخطّى المشغّلات غير الصالحة عند الإرسال
`updateIdentifier`	string (UUID)	رمز التزامن التفاؤلي (انظر PATCH)
`creationInfo` / `updateInfo`	ActionInfo	أختام التدقيق

ويستخدم نموذج `DataReaction` تحديدًا مرنًا للأنواع بأسلوب `settings.strict:false` في مواضع معيّنة، لأنّ `options` و `eventConfig` اتّحادات من النوع `any` تُحسَم أنواعها وقت التشغيل.

سياسة الوصول إلى البيانات الحساسة `Since 1.10`

يمكن للتفاعل تحديد ما يُسمح لإجراءاته **برؤيته** من الحقل الحساس [بناء التطبيقات › حماية حقل] في المُدخلات التي تتعامل معها. ويُضبَط ذلك عبر `sensitiveDataAccessPolicy`، الذي يُختار في محرّر التفاعل من قائمة **سياسة الوصول إلى البيانات الحساسة** – ولا تظهر القائمة إلا إذا كان الكيان يحتوي فعلًا على حقل حساس، ولذلك لا تظهر في أغلب التفاعلات. وله أربعة خيارات:

السياسة	ما تراه الأتمتة
Privileged – الافتراضية عند غياب أي قيمة مخزّنة	**القيمة الحقيقية** دائمًا، بصرف النظر عمّن أطلق التشغيل أو عمّا أطلقه، وعن الأذونات الممنوحة لذلك الشخص. وهذا هو سلوك كل تفاعل كُتب قبل إتاحة هذا الإعداد، لذا لم يتغيّر سلوكه.
Prohibited	**الحجب** (`****`) دائمًا، بصرف النظر عمّن أطلق التشغيل أو عمّا أطلقه. ويظلّ بوسع التفاعل إجراء عمليات على المُدخل؛ غير أنه لا يرى القيمة الحقيقية لأي حقل حساس أبدًا.
User-Based	يُحدَّد ذلك **لكل تشغيل، من منح الشخص الذي أطلقه فعلًا**: فيرى ما كان سيراه لو قرأ المُدخل بنفسه، ومن لا يملك الإذن يرى الحجب.
Auto	**التشغيل من النظام يتصرّف كـ Privileged، والتشغيل من مستخدم يتصرّف كـ User-Based.** استخدمها لتفاعل يُطلَق بالطريقتين، وينبغي أن يتّبع أذونات الشخص كلما وُجد.

ولا يمكن الجمع بين `User-Based` ومشغّل آلي. فالتفاعل الذي يُطلَق دون تدخّل شخص – أي يكون منفّذه **النظام**، أو حدثه **مؤقّت** أو **تجاوز وقت الـ SLA** – ليس له «منفّذ فعلي» يُرجَع إلى أذوناته. ولذلك يُرفَض حفظ `User-Based` عليه، وتظهر أسفل القائمة نفسها رسالة تفيد بأن سياسة «User-Based» لا يمكن اختيارها مع نوع مشغّل آلي. ويُستخدَم إعداد **Auto** في هذه الحالة، إذ يُعامَل بوصفه Privileged في تلك التشغيلات تحديدًا. أمّا `Privileged` و`Prohibited` فيبقيان متاحين على المشغّل الآلي.

وماذا تعني سياسة الحجب بالنسبة لعملية كتابة. عند استخدام `Prohibited` – أو `Auto` / `User-Based` حين تُطبَّق سياسة الحجب على التشغيل – لا يجوز للإجراء أن يستبدل قيمةً حقيقيةً بقيمة محجوبة عند التخزين، ولذلك تفشل أي عملية كتابة تفعل ذلك. و`Since 1.10` صار هذا الفحص مقصورًا على **الكتابة نفسها**. ففي السابق، كان يتعذّر تشغيل إجراء **تحديث مدخل** (`UpdateDataEntry`) تحت سياسة حجب لمجرّد أن الكيان يحوي حقلًا حساسًا، **أيًّا كان ما يكتبه فعلًا**: إذ كان يكفي أن يضبط حقلًا واحدًا غير حساس ولا صلة له بالأمر على قيمة ثابتة حتى يفشل، ولم يكن ثمة حلّ سوى تحويل التفاعل إلى `Privileged`. أمّا الآن فلا يفشل الإجراء إلا إذا كانت **تلك الكتابة بعينها** ستخزّن قيمة محجوبة؛ أمّا الحجب الموجود أصلًا في المُدخل لأن السياسة حجبته عند الدخول، فيُترك على حاله ولا يُمرَّر إلى التخزين أبدًا. ولم يتأثّر **إنشاء مدخل** (`CreateDataEntry`) يومًا – إذ يبدأ من قيمة فارغة، فأي حجب فيه يكون قد أدخله الإجراء – بينما يظلّ **نداء API** (`CallApi`) و**إرسال بريد إلكتروني** (`SendEmail`) خاضعين للفحص الأشدّ على القيمة كاملةً، لأنهما يُخرجان القيمة من المنصة.

نموذج الإجراء (`DataReactionAction`)

المصدر: `models/data-reaction/data-reaction-action.model.ts` (والنسخة المطابقة في `data-service` عند `models/data-reaction-action.model.ts`).

الحقل	النوع	المعنى
`normalizedName`	string	الاسم المعياري للإجراء (يُستخدَم مفتاحًا في سياق التشغيل وتتبّع الحلقات)
`name`	LocalizedValue	الاسم المعروض
`description`	NullableLocalizedValue	اختياري
`actionType`	`AutomatedActionTypeEnum`	المعالِج الذي يُشغَّل (انظر أنواع الإجراءات)
`options`	union `DataReactionActionOptions`	إعداد خاص بكل نوع من الإجراءات
`resultField`	string	يُضبَط وقت التشغيل على معرّف النتيجة، لتمكين الإجراءات اللاحقة من الإشارة إلى مُخرَج هذا الإجراء
`overridableOptions`	`ActionOverridableOptions[]`	مسارات الخيارات التي يجوز تجاوزها لكل عملية تنفيذ

إجراء واحد لكل تفاعل. يخزّن النموذج `actions` في مصفوفة، لكن الباني ينشئ إجراءً واحدًا بالضبط لكل تفاعل ويتيح تحريره، كما أن التحقّق عند الحفظ مصمّم لإجراء واحد. لذا صمّم أي تسلسل من الخطوات على هيئة تفاعلات منفصلة يتسلسل

بعضها إثر بعض عبر `ReactionSucceeded` (راجع تسلسل التفاعلات).

التكرار في الباني `Since 1.8` . يُفتح التفاعل المكرَّر بوصفه تفاعلًا جديدًا يمكن تغيير حدثه (ولا يبقى الحدث مقفلًا إلا في التفاعل المضاف من زر **+** في مجموعة حدث، أو في تفاعل قائم قيد التحرير). وإذا غُيّر الحدث من *Event Announced* إلى حدث آخر، وكان المصدر **كيان مطلق**، وهو مصدر لا يستخدمه إلا الحدث المُعلَن، عاد المصدر إلى **Self**، بدلًا من استمرار حفظ الكيان المطلق المخفي.

أحداث التشغيل

`EventTypeEnum` (`packages/osos-shared/src/models/enums.ts`) – القائمة الكاملة للأحداث التي يمكن للتفاعل الارتباط بها عبر `eventType` :

#	`EventTypeEnum`	يُطلق عند	نوع إعداد الحدث
0	`DataEntryCreated`	إنشاء مُدخل	`DataEntryCreatedEventConfig`
1	`DataEntryUpdated`	تحديث مُدخل	`DataEntryUpdatedEventConfig` (مطابقة على مستوى الحقل)
2	`DataEntryArchived`	أرشفة مُدخل	`DataEntryArchivedEventConfig`
3	`WorkflowActionExecuted`	تنفيذ إجراء سير عمل	`WorkflowActionExecutedEventConfig` (مرشّح خطوة/إجراء)
4	`WorkflowStepReached`	بلوغ خطوة سير عمل	`WorkflowStepReachedEventConfig` (مرشّح خطوة)
5	`WorkflowStepExited`	الخروج من خطوة سير عمل	`WorkflowStepExitedEventConfig` (مرشّح خطوة/إجراء)
6	`Timer`	حلول وقت مجدوّل / cron	`TimerEventConfig`
7	`CustomEvent`	إطلاق الإجراء المخصّص من زرّ، أو استدعائه من إجراء **تنفيذ إجراء مخصص** في تفاعل آخر (من نوع **Local** أو **External** فقط)	`CustomEventConfig` (`customActionType` + `contextual` `Since 1.8` : قد يطلب مُدخلًا من المستخدم)
8	`ReactionSucceeded`	اكتمال تفاعل آخر بنجاح	`ReactionSucceededEventConfig` (`entityId` + `triggerId`)
9	`FileGenerated`	توليد ملف/قالب	`FileGeneratedEventConfig` (`template`)
10	`ReactionFailed`	إخفاق تفاعل آخر	`ReactionFailedEventConfig` (`entityId` + `triggerId`) `Since 1.8`
11	`EventAnnounced`	نشر إجراء `AnnounceEvent` إعلانًا	`EventAnnouncedEventConfig` (`entityId` + `announcementId`) `Since 1.8`
12	`DataEntryRestored`	استعادة مُدخل مؤرشف	`BaseEventConfig` (`Since 1.8`)
14	`WorkflowStepOwnerChanged`	تغيّر مالك خطوة سير عمل	`WorkflowStepOwnerChangedEventConfig` (مرشّح نوع تغيّر المالك + مرشّح الخطوة) `Since 1.8`
20	`ZapierEvent`	ورود حدث تكامل من Zapier	`BaseEventConfig` ( )
100	`SlaTimeExceeded`	تجاوز وقت SLA لسير عمل	`SLATimeExceededEventConfig` (التكرارات: `repetitions` و `repeatAfter` / `startingAfter` بالثواني، عدد)

تتيح الأحداث 8 و10 و11 **تسلسل التفاعلات** – انظر تسلسل التفاعلات. ويُطلَق `SlaTimeExceeded` عبر فحص يعمل في الخلفية كل دقيقة، ويضع علامة تجاوز الوقت على خطوات سير العمل المفتوحة التي تجاوزت موعدها، مما يتيح لك الاستجابة تلقائيًا لتجاوزات SLA (مثلًا بالتصعيد إلى مدير أو إعادة إسناد ملكية الخطوة).

أيّ الأحداث تُطلَق – مع سير عمل وبدونه

تنشأ أحداث التفاعلات في مسارات الكتابة في `data-service`، وتعتمد مجموعة الأحداث التي تُطلَق على العملية وحدها – **فإرفاق سير عمل لا يمنع أيّ حدث أبدًا**. وعلى وجه الخصوص، يُنشَر `DataEntryCreated` دون شروط عند كلّ إنشاء لمُدخل؛ وعندما يكون للكيان سير عمل مرفق، تنشر عملية الإنشاء *إضافةً إلى ذلك* حدث `WorkflowStepReached` للخطوة الأولى من سير العمل النشط. وينطبق الأمر نفسه على مسار إنشاء المُدخلات المرتبطة: يُطلَق `DataEntryCreated` للمُدخل الرئيسي ولكلّ مُدخل مرتبط، إضافةً إلى حدث `WorkflowStepReached` واحد لكلّ كيان له سير عمل.

العملية	يُطلَق لكلّ كيان	يُطلَق فقط مع سير عمل مرفق
إنشاء مُدخل	`DataEntryCreated`	`WorkflowStepReached` (الخطوة الأولى)
تحديث مُدخل	`DataEntryUpdated` (فوري، أو مؤجَّل حتى تُثبَّت المعاملة التي يجري ضمنها)	–
أرشفة مُدخل	`DataEntryArchived`	–
استعادة مُدخل	`DataEntryRestored`	–
نقل خطوة سير عمل	–	`WorkflowStepExited` + `WorkflowStepReached`
تنفيذ إجراء سير عمل	–	`WorkflowStepExited` + `WorkflowActionExecuted`
تغيير مالك خطوة	–	`WorkflowStepOwnerChanged`
إدراج خطوة إضافية (ad-hoc)	–	`WorkflowStepReached`
تجاوز وقت SLA لسير عمل	–	`SlaTimeExceeded` (يُطلِقه فحص SLA الذي يعمل كل دقيقة)

تترتّب على ذلك نتيجتان ينبغي مراعاتهما عند التصميم:

- **انتقالات خطوات سير العمل لا تنشر `DataEntryUpdated`.** فمعالِجات تغيير الخطوة تنشر أحداث سير العمل فقط؛ لذا يجب ربط التفاعل المطلوب تشغيله عند انتقال مُدخل عبر سير عمله بـ `WorkflowStepReached` / `WorkflowStepExited` (أو غيرهما من أحداث سير العمل) – إذ لن يرصد تفاعل `DataEntryUpdated` تغيير الخطوة.
- **فشل نشر الحدث على مسار الإنشاء يُبتلَع.** تُحاط كتلة النشر بـ try/catch يسجّل `Error publishing event` ويواصل التنفيذ، فتُبلَّغ الجهة المستدعية للواجهة البرمجية بنجاح الإنشاء حتى إن لم يُنشَر أيّ حدث. فإذا أُنشئ مُدخل بنجاح ولم يُشغَّل تفاعل `DataEntryCreated` على خلاف المتوقَّع، فراجِع سجلّ `data-service` بحثًا عن تلك الرسالة قبل تصحيح التفاعل نفسه.

وفي عمليات التثبيت الأقدم من 1.8، لا توجد أنواع الأحداث `ReactionFailed` (10) و`EventAnnounced` (11) و`DataEntryRestored` (12) و`WorkflowStepOwnerChanged` (14) في `EventTypeEnum` أصلًا – ولذلك لا يمكن للتفاعلات الارتباط بها، ولا تُطلَق فيها أبدًا.

تفاعلات المؤقّت

يدعم `TimerEventConfig.timerOptions` إمّا تشغيلًا لمرة واحدة (`oneTime`، وهو طابع زمني بصيغة ISO) أو جدول تكرار `repeat`. أمّا `repeat.expression` فهو «سلسلة نصية بصيغة مقروءة للبشر، أو سلسلة نصية بصيغة cron»، ويقبل `repeat.options` كلًا من `timezone` (moment-timezone) و`startDate` و`endDate`. تمرّ عمليات تنفيذ المؤقّت عبر

`EventsHandler.handleTimerTrigger`، الذي يبني السياق، ويشغّل الإجراءات، ويكتب `TriggerHistory`، وينشر `TriggerExecutedEvent` – وهي مراحل التنفيذ الختامية نفسها المتّبعة مع الأحداث العادية.

التوزيع غير المتزامن على دفعات. لا يعالج المؤقّت مجموعته المستهدفة كاملةً بصورة متزامنة. فعند إطلاقه، يحمّل معالِج المجدول (`trigger-timer-job-triggerd.handler.ts`) تفاعل المؤقّت النشط، ويطبّق معاييره لتحديد المُدخلات المطابِقة، ثم **يقسّم معرّفات تلك المدخلات إلى دفعات (chunks) ويُدرِجها كوظائف أب/ابن في BullMQ** عبر `FlowProducer`، بحيث تُشغَّل الإجراءات بصورة غير متزامنة على دفعات. ويحدّد `TIMER_EVENT_CHUNK_SIZE` حجم الدفعة (الافتراضي **100**). لذا يعالج العمّال المجموعات المستهدفة الكبيرة دفعةً بعد أخرى بدلًا من معالجتها كلّها دفعة واحدة.

الإجراءات المخصّصة و"اسأل المستخدم"

يُطلَق `CustomEvent` – الذي يظهر في الباني باسم **الإجراء المخصّص (Custom Action)** `Since 1.8`، وما يزال يُسمّى *Custom Event* في الرماز وواجهة برمجة التطبيقات – من زرّ في الواجهة مرتبط بخيارات المُدخل (انظر دليل الأزرار)، مما يتيح استدعاء التفاعل عند الطلب بدلًا من استدعائه استجابةً لتغيّر في البيانات. ويوفّر الباني إمكانيتين إضافيتين له:

- **مؤشر التنفيذ** – في تفاعل الواجهة، يمكنك ضبط مؤشّر التحميل المعروض أثناء تشغيل التفاعل: **منع** (Blocking؛ يعرض المؤشّر ويمنع المستخدم من المتابعة حتى الانتهاء)، أو **إعلام** (Inform؛ يعرض المؤشّر مع السماح للمستخدم بالانتقال)، أو **بدون** (None). وعند اختيار **منع**، يمكنك أيضًا تحرير رسالتَي **عند بدء التنفيذ** و**عند نجاح التنفيذ** المعروضتين للمستخدم النهائي.
- **اسأل المستخدم** – فعِّل خيار **اسأل المستخدم عن** لجمع مُدخلات قبل تشغيل الإجراءات. أدخل **عنوان مربع الحوار**، واختر **مصدر الحقول**: **كيان ضمنيّ** (Inline Entity؛ بيانات JSON مُدخَلة يدويًا) أو **معرض** (لا تظهر إلا المعارض التي فُعِّل فيها خيار «اسأل المستخدم»، وتُستخدَم حقولها لملء مربّع الحوار). ويمكن بعدئذٍ ربط القيم التي يُدخلها المستخدم بحقول الإجراء المختار. وفي الرماز، يُمرَّر هذا الإدخال إلى السياق بوصفه كيان AskUser.

كما هو الحال مع المؤقّتات، **توزّع الأحداث المخصّصة مدخلاتها المطابِقة على وظائف أب/ابن دفعية في BullMQ** (`custom-event-triggered.handler.ts`) بدلًا من معالجتها بصورة متزامنة – ويحدّد `CUSTOM_EVENT_CHUNK_SIZE` حجم الدفعة (الافتراضي **100**). ويُصدِر المعالِج، إضافةً إلى ذلك، حدث مقبس (socket) باسم `JobStarted` (`JobStartedMessage`) عند إدراج الدفعة، مما يتيح للواجهة الإبلاغ عن التقدّم أثناء معالجة مجموعة مستهدفة كبيرة بصورة غير متزامنة على دفعات.

نوع الإجراء المخصّص وخيار Contextual `Since 1.8`

يحمل المشغّل (trigger – ويرد في بقية هذه الصفحة باسم «التشغيل» أيضًا) اسم **الإجراء المخصّص (Custom Action)** في الباني، ويُهيَّأ بإعدادين يحلّان معًا محلّ مفتاح **UI Event** القديم:

- **مستوى الإجراء المخصص** – قائمة منسدلة (`customActionType`) بثلاث قيم، تظهر باسم **إجراء واجهة** و**إجراء محلي** و**إجراء خارجي**:
  - **UI** – يُستدعى الإجراء من واجهة مستخدم، أي من زرّ نظام. وهذا يكافئ تفعيل مفتاح *UI Event* القديم.
  - **Local** – إجراء على مستوى الوحدة، لا يمكن استدعاؤه من إجراء تنفيذ إجراء مخصّص في تفاعل آخر إلا إذا كان كيان المشغّل المستدعي ينتمي إلى **الوحدة نفسها**.
  - **External** – يمكن استدعاؤه من إجراء **تنفيذ إجراء مخصص** من أيّ وحدة.
- **Contextual** – مفتاح تبديل (`contextual`) **مفعَّل افتراضيًا**. يحدّد ما إذا كان للإجراء **مدخل هدف** أصلًا: فالإجراء السياقي يحتاج إلى مُدخل هدف (يُحدَّد من حدثه الخاص، أو من نطاق (Scope) إجراء **تنفيذ إجراء مخصص** المستدعي – انظر أدناه)؛ أمّا الإجراء غير السياقي فلا هدف له، ويُشغَّل مرة واحدة فقط عند استدعائه.

> **تصحيح مصطلحي – Since 1.8**
>
> طُرح نوع الإجراء المخصّص أولًا بالقيم **Internal** / **Module** / **UI**. ثم أعادت مراجعة لاحقة ضمن دورة 1.8 نفسها **تسمية** القيمتين غير الـ UI إلى **Local** و**External** – وبقيت **UI** بلا تغيير. وأعاد ترحيل بيانات أحادي الاتجاه كتابة القيمة المحفوظة لكل تفاعل قائم (`Module` ← `Local`، `Internal` ← `External`)؛ ولا يمكن التراجع عنه تلقائيًا. **فإن رأيت `Module` أو `Internal` في أي مكان آخر – لقطة شاشة قديمة، ملاحظة، سكربت – فاقرأها بوصفها `Local` / `External` على التوالي؛ فالمنتج وواجهة البرمجة لم يعودا يستخدمان التسميتين القديمتين.**

ويترتّب على النوع الذي تختاره حكمان:

- خيارات **إظهار للمستخدم** (Show to user)، أي مؤشّر التنفيذ ورسالتا **عند بدء التنفيذ** و**عند نجاح التنفيذ** الموصوفتان أعلاه، لا تنطبق **إلّا** حين يكون النوع **UI**. أمّا مع **Local** و**External** فتكون معطَّلة.
- يحمل حقل سؤال المستخدم التسمية **اسأل المستخدم عن** حين يكون النوع **UI**، والتسمية **المعاملات** حين يكون **Local** أو **External**؛ لأنّ الحقول نفسها، في الإجراء من نوع غير الـ UI، تصبح أيضًا الحقول المستهدفة بربط المعاملات (Parameters) في إجراء **تنفيذ إجراء مخصص** المستدعي (انظر أدناه). ولم يعد طلب المُدخل مرتبطًا بالنوع؛ إذ يمكن لإجراء من نوع Local أو External أن يجمع مُدخلًا أيضًا.
- **لا تستطيع أزرار النظام إطلاق سوى إجراءات من مستوى الـ UI.** قد يشير زرّ من حيث البنية، إلى إجراء مخصّص من مستوى `Local` / `External`، لكنّ تشغيله بهذه الطريقة يُرفَض بخطأ 403 – انظر الأزرار: لا تستطيع أزرار النظام إطلاق سوى إجراءات من مستوى الـ UI [بناء التطبيقات › الأزرار] لمعرفة السبب وصيغة الخطأ.

التفاعلات القائمة. رُحِّل كلّ مشغّل إجراء مخصّص كان موجودًا قبل 1.8 إلى النوع **UI** مع تفعيل **Contextual**، وأُزيلت الراية `uiEvent` التي أُوقِف استخدامها – راجِع أيّ مشغّل ينبغي أن يصبح الآن **Local** أو **External** وغيّر نوعه يدويًا؛ إذ لا يمكن استرجاع قيمة الراية الأصلية. أمّا المشغّلات التي كانت مُهيَّأة مسبقًا بالقيمتين القديمتين **Module/Internal** فقد انتقلت تلقائيًا إلى **Local/External** عبر ترحيل إعادة التسمية المذكور أعلاه – ولا يلزم أي إجراء يدوي لهذا الجزء.

تفاعلات "عند تغيّر المالك" `Since 1.8`

يُطلَق `WorkflowStepOwnerChanged` (14) كلّما تغيّر مالك خطوة سير عمل على مُدخل. وينشر `data-service` الحدث مباشرةً بعد حفظ المالك الجديد، فيطّلع التفاعل المرتبط به دائمًا على الحالة المثبَّتة.

ويغطّي الحدث ثلاثة أنواع من التغيير، ويحدّد أيّها وقع، مما يتيح للتفاعل الاستجابة لنقل الملكية دون أن يُطلَق أيضًا حين يتولّى أحدهم ملكية خطوة بلا مالك:

`OwnerChangeTypeEnum`	يُطلَق عند
(0) `ChangeOwner`	تغيير المالك إلى شخص آخر. وهو أيضًا القيمة الاحتياطية عندما لا تحدّد الجهة المستدعية نوع التغيير.
(1) `TakeOwnership`	استلام مستخدم ملكية الخطوة. ويُحدَّد هذا النوع من جهة الخادم كلّما جرى التغيير عبر وضع استلام الملكية، أيًّا كان النوع الذي طلبته الجهة المستدعية.
(2) `Reassign`	إعادة إسناد الخطوة.

وإلى جانب نوع التغيير، يحمل الحدث معرّفَي سير العمل والخطوة، واسم الخطوة المُوطَّن، والمالك السابق والجديد، والمالك الأصلي، والملاحظة وعلامة التغيير الدائم المرفقتين بالتغيير، وما إذا كانت الخطوة **خطوة إضافية** (ad-hoc).

الإعداد في أداة البناء. يوفّر إعداد الحدث نفسه (`WorkflowStepOwnerChangedEventConfig`) مرشّحين اختياريين، وتعرض أداة بناء التفاعلات لوحة مخصّصة لهما:

- **نوع تغيّر المالك** – قائمة اختيار متعدّد مرتبطة بـ `ownerChangeTypes` وتضمّ القيم الثلاث أعلاه. اتركها فارغة ليُطلَق التفاعل عند أي نوع من تغيّر المالك.
- **اختيار الخطوة** – أداة اختيار الخطوات المشتركة، وتُخزَّن اختياراتها في `steps` . اتركها فارغة ليُطلَق التفاعل عند أي خطوة، بما فيها الخطوات المُضافة إلى سير العمل لاحقًا. وتُطابَق الخطوات باستخدام معرّف سير العمل **و**معرّف الخطوة معًا بوصفهما زوجًا، فلا يمكن لمعرّف خطوة مُعاد استخدامه في سيرَي عمل أن يُطلِق تفاعلًا في غير موضعه. ويطابق إدخال `ADHOC_STEP` في الأداة تغيّر المالك الذي يقع على خطوة إضافية في سير العمل المُختار.

وتستطيع إجراءات التفاعل قراءة تفاصيل التغيير عبر مخطّط التفاعل الخاص بالحدث، الذي يتيح نوع التغيير في صورة تسمية مُوطَّنة، إضافةً إلى كتلة معلومات الخطوة: معرّف الخطوة واسمها، والمالك السابق والجديد (يمكن تحديد كلّ منهما بوصفه مستخدمًا أو مجموعة أو تجميعة بحث)، ومَن نفّذ التغيير.

الشروط

للتفاعل **مرحلتَي ترشيح مستقلّتين**، وكلتاهما اختياريتان. وإذا لم تُهيَّأ أيّ منهما، فإنّ كلّ حدث مطابق يشغّل الإجراءات.

مطابِقات الحدث (على مستوى إعداد الحدث)

قبل بناء السياق، يبحث `EventsHandler.checkEventMatcher` عن `IEventMatcher` لنوع الحدث `eventType` (عبر `EventMatchersFactoryService`). فإن وُجد، تحدّد دالّته `isEventMatchTrigger(eventConfig, event)` ما إذا كان التنفيذ سيستمرّ. وتتوفّر مطابِقات لمجموعة فرعية من الأحداث:

- **`DataEntryUpdated`** – تتحقّق المطابقة عبر `DataEntryUpdatedEventMatcher` فقط إذا كان أحد الحقول المتغيّرة (`event.valueChanges[].fullNormalizedName`) موجودًا في `eventConfig.fields` . **إذا كان كلٌّ من `fields` و `relatedFields` فارغًا، فإنه يطابق أيّ تغيير.**
- **`WorkflowStepReached` / `WorkflowStepExited` / `WorkflowActionExecuted`** – تتحقّق المطابقة وفق القيم المهيّأة في `steps` / `actions` ؛ وإذا تُرك الاختيار فارغًا، شملت المطابقة الكلّ.
- **`WorkflowStepOwnerChanged`** – تتحقّق المطابقة وفق القيم المهيّأة في `ownerChangeTypes` و `steps` ؛ وإذا تُرك أيٌّ منهما فارغًا، شملت المطابقة الكلّ بالنسبة إليه. **Since 1.8.**
- **`SlaTimeExceeded`** و** `FileGenerated` ** – لكلّ منهما مطابِقه الخاص (اختيار الخطوة / القالب).

أمّا أنواع الأحداث التي **لا** مطابِق لها (created، archived، restored، timer، custom، reaction-succeeded/failed، event-announced، zapier)، فتجتاز هذه المرحلة دائمًا وتعتمد كلّيًا على مرشّح المعايير.

مرشّح المعايير والاستعلام (على مستوى المدخل)

`criteria` هو تعبير من نوع `FilterExpression` يُقيَّم على المُدخل المُحلَّل بالآلية نفسها المستخدمة في الاستعلامات، مما يتيح لمن يبني التفاعل إعادة استخدام استعلام قائم في الكيان ضمن خيار **مصدر المدخلات**. وإذا ضُبط مرجع `query` أيضًا، تُدمَج معاييره **بعامل AND**:

```
trigger.criteria = {
  operator: ExpressionTypeEnum.And,
  args: [trigger.criteria, query.result.criteria],
};
```

تُقيَّم المعايير المُحلَّلة بتنفيذها كاستعلام MongoDB فعلي على المُدخل المرشّح (`QueryEngineService.parseCriteria` ← `DataEntryService.findWithResolvesFactory`)، فتعمل كما يعمل المرشّح نفسه في استعلام محفوظ. و `Since 1.10`

يُمرَّر المُدخل إلى MongoDB بوصفه مصدرًا للتجميع في الذاكرة من نوع **`$documents`**، بدلًا من نسخه إلى مجموعة مؤقّتة `trigger_<random>`؛ وبذلك لا تُجرى عمليات كتابة أو إسقاط في كلّ تشغيل، ولا تترك الحاوية (pod) أثرًا إذا انهارت. ويترتّب على هذا التصميم أنّ **المعاملات التي تحتاج إلى مجموعة حقيقية لا يمكن استخدامها** في معايير التفاعل، وهي: `$text` و`$geoNear` / `$near` / `$nearSphere` و`$indexStats` و`$collStats` و`$out` و`$merge`. في المسار القياسي، يدمج `EventsHandler.getResolvedEntry` المعايير مع مرشّح للمعرّف، ولا يتابع التنفيذ إلا إذا ظلّ المُدخل مطابقًا؛ أمّا في مسار reaction-succeeded، فيستخدم `findMatchedCriteriaInTriggerContext`. وإذا لم تتحقّق المطابقة، يُتخطّى التفاعل مع تسجيل الرسالة "criteria is not matched with event data".

وهذا يعني أنّ المعايير يمكن أن تشير إلى **أيّ حقل يمكن بلوغه في سياق التشغيل**، بما في ذلك الكيانات المرتبطة عبر مسارات اجتياز العلاقات (`value.requester.:relation.value.first_name`) التي يحلّها `TriggerContext`.

مؤشرات التاريخ في المعايير. تحلّ وسيطة المعايير من النوع Pointer دوالَّ التاريخ المسبوقة بـ `@` قبل تشغيل الاستعلام. وتدعم كلًّا من `@today()` و`@now()` (إلى جانب `@StartOfDay()` و`@StartOfWeek()` و`@StartOfMonth()` و`@StartOfYear()`)، لكنّ الدالّتين تعيدان قيمًا بصيغتين مختلفتين، وتتوقّف المطابقة على الصيغة:

- يعيد `@now()` **تاريخًا ووقتًا كاملين بصيغة ISO**، لذا لا يناسب إلا المقارنة بقيم من نمط التاريخ والوقت. أمّا إذا قورن بقيمة مخزَّنة بصيغة تاريخ فقط، فلن تتحقّق المطابقة مطلقًا في فحص المساواة أو المدى؛ وقد يبدو عندئذٍ أنّ "`@now()` لا يعمل"، رغم نجاح حلّ المؤشِّر.
- يعيد `@today('±Nd')` **سلسلة تاريخ فقط**؛ فاستخدمه لحقول التاريخ فقط. (داخليًا، يُنسِّق بداية اليوم بعد إضافة إزاحة قدرها ساعتان مثبَّتة في الرماز.)
- مقدار التعديل **يجب أن يكون مُوقَّعًا**؛ يعمل `@today('+1d')` و`@today('-3d')`، أمّا المقدار غير المُوقَّع `@today('1d')` فيُتجاهَل دون إشعار، ويُعاد التاريخ الأساسي. الوحدات: `d` أيام، `w` أسابيع، `M` أشهر، `y` سنوات، `h` ساعات، `m` دقائق، `s` ثوانٍ.
- أمّا فشل تقييم المؤشِّر فلا يمرّ دون إشعار، إذ ترمي خدمة التعبيرات الاستثناء `ExpressionExecutionException`.

أنواع الإجراءات

تُوجَّه الإجراءات وفق المطابقة `AutomatedActionTypeEnum` ← `IActionHandler`. وتُسجَّل المعالِجات تلقائيًا بواسطة `ActionHandlersBooter`، الذي يفحص `action-handlers/**/*.action-handler.js` ويربط كل فئة بالمفتاح `entity-action-handlers:<actionType>` باستخدام `static actionType` الخاص بها. وعند التشغيل، يستخرج `EventsHandler.actionHandler` المعالِج عبر `ActionHandlerFactoryService.getActionHandler(action.actionType)`؛ وإذا كان النوع غير معروف، يُرمى الاستثناء `DataNotSupportedException`.

يعرّف التعداد enum عددًا من القيم يبلغ 26؛ **24 منها لها معالِج فعلي في `reaction-service`** – منها ثلاث أُضيفت في 1.10: `SetPermission` (31) و`RemovePermission` (32) (انظر إجراءا منح الإذن وإزالته) و`SendWhatsApp` (107). وترد البنية الدقيقة لـ `options` لكل نوع في مرجع خيارات الإجراءات. أمّا `CreateDataEntity` (0) و`UpdateDataEntity` (1) فمُعرَّفان في التعداد enum، ولهما واجهات خيارات (`CreateDataEntityActionOptions` و`UpdateDataEntityActionOptions`)، لكن لم يكن لهما معالِج قطّ. و`Since 1.10` أُحيلا رسميًا إلى التقاعد: لم يعد الباني يعرضهما، ويُرفَض إنشاء تفاعل أو تحديثه إذا تضمّن أيًّا من النوعين بالرمز **422**. أمّا الإجراء المخزَّن من أيّ النوعين في تفاعل أقدم فـ**يُتخطّى** عند إطلاق التفاعل، ويُسجَّل بوصفه إجراءً فاشلًا بنصّ الخطأ `action type N (<name>) was removed (…); action skipped`، مع استمرار تنفيذ الإجراءات التالية له. وتبقى القيمتان في التعداد enum لأنّ التفاعلات المخزَّنة قد تتضمّنهما.

#	AutomatedActionTypeEnum	ملف المعالِج	ما يفعله
0	CreateDataEntity	*(لا يوجد)*	أُوقف – يُرفَض عند الحفظ (422) ويُتخطّى عند التشغيل Since 1.10
1	UpdateDataEntity	*(لا يوجد)*	أُوقف – يُرفَض عند الحفظ (422) ويُتخطّى عند التشغيل Since 1.10
2	ArchiveDataEntity	archive-data-entity	أرشفة كيان بيانات كامل (على مستوى الكيان، وتختلف عن أرشفة مُدخل)
10	CreateDataEntry	create-data-entry	إنشاء مُدخل في كيان مستهدف، مع تعيين القيم من السياق
11	UpdateDataEntry	update-data-entry	تحديث المُدخلات الذاتية / المُشار إليها / المرتبطة / المُختارة بالاستعلام
12	ArchiveDataEntry	archive-data-entry	أرشفة المُدخلات (الذات / العلاقة / مصدر الاستعلام)
13	CreateBlockItem	create-block-item	إلحاق عنصر بحقل كتلة (جدول فرعي)
14	UpdateBlockItem	update-block-item	تحديث عنصر أو عناصر من الكتلة وفق معايير الفهرس/الحقل/التعبير
15	DeleteBlockItem	delete-block-item	حذف عنصر أو عناصر من الكتلة وفق المعايير
16	DeleteArchivedEntry	delete-archived-data-entry	حذف المُدخلات المؤرشفة نهائيًا
20	MoveWorkflowToStep	move-workflow-to-step	نقل سير عمل إلى خطوة مستهدفة
21	ExecuteWorkflowAction	execute-workflow-step	تنفيذ إجراء أو أكثر من إجراءات سير العمل
22	ChangeCurrentStepOwner	change-current-step-owner	إعادة إسناد ملكية خطوة سير العمل الحالية
30	UserActivationToggle	user-activation-toggle	تفعيل المستخدمين أو إلغاء تفعيلهم جماعيًا
31	SetPermission	set-permission	منح مستخدمين أو مجموعات مستوى وصول إلى مورد Since 1.10
32	RemovePermission	remove-permission	سحب أذونات الوصول الممنوحة لمستخدمين أو مجموعات على مورد Since 1.10
90	CallApi	call-api	استدعاء واجهة برمجة تطبيقات HTTP خارجية
100	SendNotification	send-notification	إرسال إشعار داخل التطبيق من قالب
101	SendEmail	send-email	إرسال بريد إلكتروني من قالب
102	SendSms	send-sms	إرسال رسالة نصية من قالب
103	GenerateFileFromTemplate	generate-file-from-template	تصيير ملف من قالب
104	ExecuteCustomFunction	execute-custom-function	تشغيل دالّة مخصّصة (Knative)
105	AnnounceEvent	announce-event	نشر إعلان تستهلكه تفاعلات EventAnnounced
106	ExecuteCustomAction	execute-custom-action	تشغيل تفاعل إجراء مخصّص من مستوى **Local** أو **External** على المُدخل أو المُدخلات التي يحدّدها نطاقه (Scope) – أو مرة واحدة بلا مُدخل عندما يكون الهدف غير سياقي Since 1.8

#	AutomatedActionTypeEnum	ملف المعالِج	ما يفعله
107	SendWhatsApp	send-whatsapp	إرسال قالب WhatsApp معتمد من Meta إلى المستلمين الذين جرى تحديدهم Since 1.10
200	NotifyZapier	notify-zapier	إرسال حدث صادر إلى Zapier

تعيين القيم والخيارات المختارة

يمثّل DataReactionActionOptions نوع الاتحاد لخيارات action.options (packages/osos-shared/src/models/trigger-actions.ts). وفيما يلي أبرز ما يحتاج إليه المطوّر؛ أمّا نصّ الطلب الكامل لكل نوع فيرد في مرجع خيارات الإجراءات:

- **تعيين القيم (EntryActionOptionsValue[])** – تتضمّن معظم إجراءات البيانات values: {fieldName, valueType, value}[]. وقد تكون value قيمة ثابتة (يدوية)، أو **حقلًا** من كيان المصدر، أو **مؤشرًا**. ويشير @thisReference() (المعروض في الباني بوصفه **هذا المدخل**) إلى المُدخل الذي أطلق الحدث؛ ويجب كتابته مع القوسين، انظر مرجع خيارات الإجراءات. أمّا **المستخدم الحالي** فيشير إلى المستخدم الذي أطلق التفاعل. وعند تعيين القيم لحقول متكرّرة/متعدّدة القيم أو حقول كتلة، لا تُعرَض إلا حقول المصدر التي يتوافق نمط بياناتها وحدود تكرارها مع الهدف.
- **CreateDataEntry** – entitySource: {entityId?} أو {fieldName?} (قد يُحدَّد الكيان الوجهة ديناميكيًا من حقل) + values . يُعاد معرّف المُدخل المُنشأ في resultId ، وتُتاح قيمته إلى جانب مرجع thisEntry بوصفها مُخرَج الإجراء.
- **UpdateDataEntry / عناصر الكتلة** – updateEntryMode = Self (0) / Reference (1) / Relation (2)؛ ويحدّد entryToUpdate الهدف (mode : Fixed / Dynamic ، إضافة إلى relationId / targetEntityId لوضع العلاقة). وتضيف صيغ إجراءات الكتلة blockField و(للتحديث/الحذف) criteria ، حيث تكون قيمة type إحدى القيم FixedIndex / Field / Expression ، مع سياسة onFailure .
- **CallApi** – apiUrl ، و verb (GET / POST / PATCH / PUT / DELETE)، و headers اختيارية، و body ({custom,) {customBody}، و tokensMap لاستبدال الرموز. وتُعدّ استجابات 2xx نجاحًا؛ ويُعيد المعالِج الطلب والاستجابة كاملين (مع تسلسل الترويسات/الجسم) بوصفهما مُخرَج الإجراء.
- **SendEmail** – toRecipients / ccRecipients / bccRecipients (كلّ منها {sourceType, source:{type, value}})، و template (معرّف UUID لقالب أو EmailTemplateValue مُضمَّن يتضمّن subject / body / tokens)، و from (System / ReactionExecutor / Custom عبر SendEmailFromModeEnum)، و link ({enabled,) ، و tokensMap و attachments ({viewId}).
- **SendNotification** – channels (حاليًا InApp=0 فقط في NotificationChannelTypeEnum)، و recipients ، و template (UUID أو NotificationTemplateValue مُضمَّن)، و module ، و tokensMap ، و customRedirection ({enabled, viewId}).
- **SendSms** – recipients (المصادر: DataCollection / Entity / Static)، و template ({content,) مُضمَّن ({tokens})، و tokensMap ، و link .
- **AnnounceEvent** – announcementId إضافة إلى schemaMapping ، الذي يعيّن قيم سياق التشغيل لحقول الكيان الضمنيّ/الخارجي في الإعلان.
- **ChangeCurrentStepOwner** – type / source / ownerOptions ، و fallbackOwner ، و permanentChange (الافتراضي false).
- **ArchiveDataEntry / DeleteArchivedEntry** – يحدّد sourceMode المصدر: الذات أو العلاقة أو مصدر الاستعلام، مع relation ({relationId?, referenceFieldId?}) و querySource .

- **`ExecuteCustomAction`** – `Since 1.8` – `customActionId` (الإجراء المخصّص المستهدف، ويجب أن يكون تفاعل `CustomEvent` من مستوى **Local** أو **External**)، و`scope` (`Self` / `Reference` / `Relation` / `Expression` / `Pointer` ، ويظهر فقط عندما يكون الهدف سياقيًا)، و`parameters` (ربط القيم بمُعامِلات (Parameters) الهدف، وهو مستقلٌّ عن `scope`). انظر تنفيذ إجراء مخصّص للتفاصيل الكاملة.
- **`RemovePermission` / `SetPermission`** – `Since 1.10` – `resource` (`entitlementId` + `entitlementType`) و`principal` (`kind` + `source`)، و`level` لإجراء المنح وحده. انظر إجراءا منح الإذن وإزالته.
- **`NotifyZapier`** – لا خيارات (`NotifyZapierActionOptions` فارغ).

تعيين توافق الحقول (إنشاء مدخل)

في `CreateDataEntry` ، لا يعرض الباني إلا حقول المصدر التي يتوافق نمط بياناتها **و**حدود تكرارها مع الهدف. وعند التشغيل، تُحوَّل كل صيغة للمصدر إلى قيمة الهدف وفق قاعدة اختزال ثابتة. وتعتمد الحقول المعروضة وطريقة تحويل القيم في كلّ سيناريو على صيغة الحقل الهدف:

الهدف حقل متكرّر متعدّد القيم (مستقلّ أو في كتلة مفردة) – لا تظهر إلا الحقول المطابِقة لنمط بيانات الهدف وحدود تكراره:

نوع حقل المصدر	الحلّ
مفرد القيمة (مستقلّ / كتلة مفردة)	يُملأ الهدف مباشرة بقيمة المصدر
مفرد القيمة في كتلة متكرّرة	تُستخدَم قيمة أوّل تكرار للكتلة
متكرّر (مستقلّ / كتلة مفردة)	يُستخدَم أوّل تكرار للمصدر
متكرّر في كتلة متكرّرة	أوّل كتلة ← أوّل تكرار للحقل
متعدّد القيم (مستقلّ / كتلة مفردة)	تُستخدَم أوّل قيمة للمصدر
متعدّد القيم في كتلة متكرّرة	أوّل كتلة ← أوّل قيمة
متكرّر متعدّد القيم (مستقلّ / كتلة)	أوّل تكرار ← أوّل قيمة
متكرّر متعدّد القيم في كتلة متكرّرة	أوّل كتلة ← أوّل تكرار ← أوّل قيمة

الهدف حقل متكرّر مفرد القيمة (مستقلّ أو في كتلة مفردة) – لا تُعرَض إلا حقول المصدر متعدّدة القيم التي يكون حدّها الأدنى أكبر من أو يساوي الحدّ الأدنى للهدف:

نوع حقل المصدر	الحلّ
مفرد القيمة (مستقلّ / كتلة مفردة)	يُملأ التكرار الوحيد بقيمة المصدر
مفرد القيمة في كتلة متكرّرة	تُستخدَم قيمة أوّل تكرار للكتلة للتكرار الوحيد
متكرّر مفرد القيمة (مستقلّ / كتلة)	تُستخدَم جميع قيم المصدر
متكرّر مفرد القيمة في كتلة متكرّرة	تُستخدَم قيم أوّل كتلة
متعدّد القيم (مستقلّ / كتلة)	تُستخدَم أوّل قيمة للتكرار الوحيد
متعدّد القيم في كتلة متكرّرة	أوّل كتلة ← أوّل قيمة
متكرّر متعدّد القيم (مستقلّ / كتلة)	تُستخدَم أوّل قيمة من كلّ تكرار
متكرّر متعدّد القيم في كتلة متكرّرة	أوّل كتلة ← أوّل قيمة من كلّ تكرار

الهدف حقل مفرد القيمة إجباري (مستقلّ أو في كتلة مفردة) – لا تُعرَض إلا حقول المصدر الإجبارية ذات نمط البيانات المطابِق تمامًا، ويجب أن تكون مستقلّة أو في كتلة مفردة (لا في كتلة متكرّرة حدّها الأدنى للتكرار صفر):

نوع حقل المصدر	الحلّ
مفرد القيمة إجباري (مستقلّ / كتلة مفردة)	تُنسَخ قيمة المصدر مباشرة إلى الهدف
مفرد القيمة إجباري في كتلة متكرّرة	تُستخدَم قيمة أوّل كتلة
متكرّر (مستقلّ / كتلة مفردة)	تُستخدَم قيمة أوّل تكرار
متكرّر في كتلة متكرّرة	أوّل كتلة ← قيمة أوّل تكرار
متعدّد القيم (مستقلّ / كتلة مفردة)	تُستخدَم أوّل قيمة
متعدّد القيم في كتلة متكرّرة	أوّل كتلة ← أوّل قيمة
متكرّر متعدّد القيم (مستقلّ / كتلة مفردة)	أوّل تكرار ← أوّل قيمة
متكرّر متعدّد القيم في كتلة متكرّرة	أوّل كتلة ← أوّل تكرار ← أوّل قيمة

إجراءات جديرة بالملاحظة

إلى جانب البنية الخام للخيارات، تتّسم بعض الإجراءات بسلوك في الباني يستحقّ الإبراز:

- **إرسال بريد إلكتروني** يوفّر ثلاثة أوضاع للمرسِل: **النظام** (المرسِل الافتراضي)، أو **منفّذ التفاعل** (المستخدم الذي شغّل التفاعل)، أو **مخصّص** (مستخدم ثابت، أو حقل مستخدم متغيّر، أو عنوان يُدخَل يدويًا). ويمكن جلب **المرفقات** من حقول الملفات في المُدخل (متغيّرة)، أو رفعها في صورة ملف ثابت.
- **إنشاء ملف** (`GenerateFileFromTemplate`) يصيّر ملفًا من قالب كيان، مع **فترة احتفاظ** قابلة للضبط (للأبد، أو عدد من الأيام) واسم ملف مخصّص اختياري (باستخدام متغيّرات موضعية أو بإدخاله يدويًا). وعند الاكتمال، يتلقّى المستخدم إشعارًا يستطيع تنزيل الملف منه.
- **تنفيذ إجراء خطوة سير العمل** (`ExecuteWorkflowAction`) ينفّذ إجراء الخطوة الحالية آليًا، ثم يتابع التنفيذ عبر الخطوات اللاحقة القابلة للأتمتة؛ و**نقل سير العمل إلى الخطوة** ينقل مثيل سير العمل إلى خطوة مختارة.
- **تغيير مالك الخطوة الحالي** يعيد إسناد ملكية خطوة سير العمل الحالية للمُدخل، ويُقرَن عادةً بـ `SlaTimeExceeded` لتصعيد خطوة تجاوزت الوقت المحدّد.
- **تفعيل / تعطيل مستخدم** (`UserActivationToggle`) يفعّل مُدخلات المستخدمين أو يعطّلها جماعيًا (مثل إنهاء الخدمة عند أرشفة مُدخل).
- **تنفيذ وظيفة** (`ExecuteCustomFunction`) يستدعي دالّة عديمة الخادم يكتبها المستخدم لتنفيذ منطق يتجاوز الكتالوج المدمج؛ ويجب أن تكون الدالّة مُهيّأة وفعّالة، وتُلتقط نتيجتها في السياق لتستخدمها الإجراءات اللاحقة. و `Since 1.10` تستطيع الإجراءات اللاحقة والتفاعلات المتسلسلة ربط `success` و `message` وأيّ مفتاح ضمن `result` – انظر مخرجات «تنفيذ وظيفة» في التفاعلات المتسلسلة.
- **الرموز المرتبطة بالمالك الأصلي للخطوة** `Since 1.10` – كان أيّ رمز في بريد إلكتروني أو إشعار أو رسالة نصية (أو متغيّر مخصّص) يقرأ قيمة عرض **المالك الأصلي** للخطوة الحالية – `_metaCurrentSteps[n].originalOwner._resolved.mainTitle` أو `mainPicture` – يُصيَّر **نصًّا فارغًا** كلما كان ذلك المالك **مجموعة** (أو تجميعة بيانات)، بينما كان التعبير نفسه يعمل في المعارض والبطاقات. أمّا الآن فيُصيَّر العنوان الفعلي للمجموعة، وذلك لكل مثيل من الخطوة الحالية، لا للمثيل النشط وحده.

إصلاحات كتابة الكتلة القابلة للتكرار `Since 1.10`

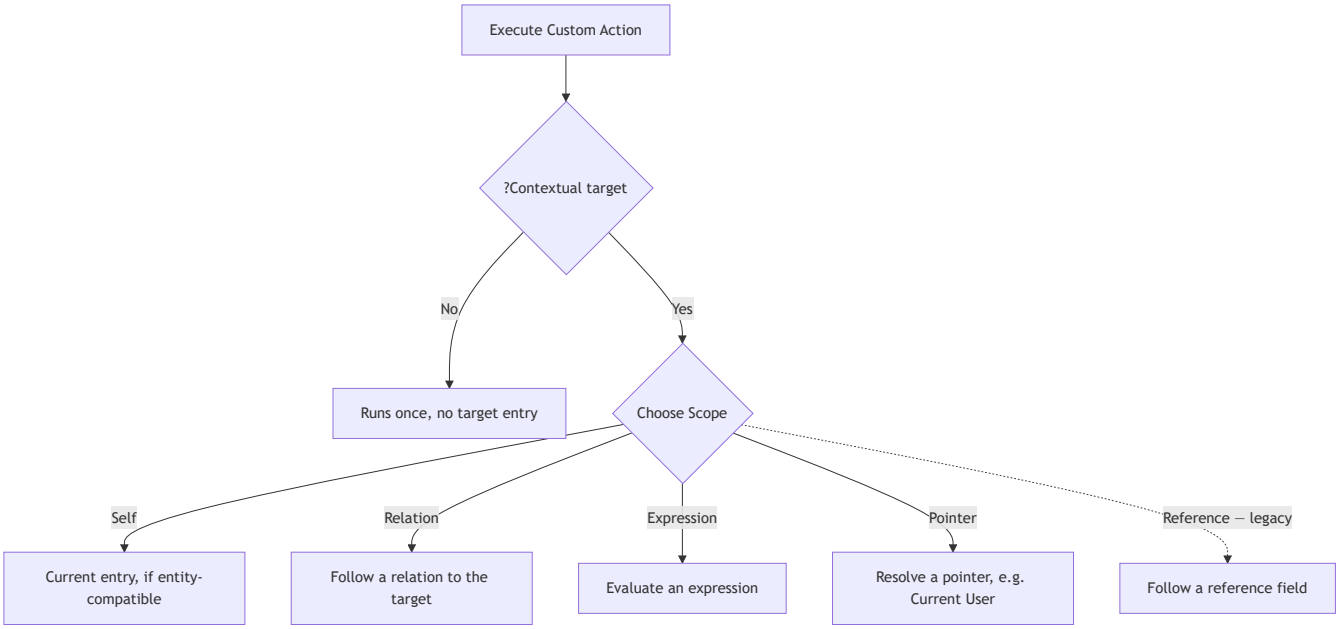
أُصلح في هذه الدورة خطآن صامتان يؤثّران في صحّة الكتابة في الكتل القابلة للتكرار. فقد كانت كلتا الحالتين تُعيد استجابة نجاح عادية، وتظهر في السجل التاريخي للمُدخل بوصفها عملية كتابة مكتملة، لذا لم يكن للخلل أيّ مظهر سوى البيانات الخاطئة في الكتلة:

- **إجراء ثانٍ من "إنشاء عنصر كتلة" يستهدف الكتلة نفسها لم يعد يستبدل الأول.** كان التفاعل الذي يتضمّن إجراءَي `CreateBlockItem` يُلحِقان بحقل الكتلة **نفسه** في تشغيل واحد يُبقي على الصفّ الثاني فقط، إذ كان صفّ الإجراء الأول يُهمَل بصمت. أمّا الآن فيُلحَق صفّا الإجراءين بصورة صحيحة، بحيث يحتفظ التشغيل بجميع الصفوف التي يضيفها إلى الكتلة نفسها.
- **إجراء "تحديث مدخل" يكتب حقلًا فرعيًا واحدًا على كتلة قابلة للتكرار لم يعد يطوي بقية صفوف الكتلة.** كانت كتابة حقل فرعي واحد فقط عبر جميع صفوف كتلة قائمة تستبدل الكتلة كاملةً بصفّ جديد وحيد (وإذا كان للكتلة حدّ أدنى لعدد الصفوف، كانت تُكمَل حشوة ذلك الصفّ الوحيد حتى الحدّ الأدنى؛ لذا قد يبدو عدد الصفوف ثابتًا، بينما يكون محتوى الصفوف جميعًا متطابقًا وخاطئًا). أمّا الآن فتُطبَّق الكتابة بصورة صحيحة على كلّ صفّ قائم، مع تحديث الحقل الفرعي المستهدَف وحده في كلّ صفّ.

تنفيذ إجراء مخصّص `Since 1.8`

يتيح إجراء **تنفيذ إجراء مخصص** (`ExecuteCustomAction`، 106) للتفاعل تشغيل إجراء مخصّص من مستوى **Local** أو **External** على مُدخل آخر (أو عدّة مُدخلات، أو بلا مُدخل على الإطلاق). وهو النظير الآلي لضغط المستخدم زرًّا، إذ يبدأه مشغّل بدلًا من نقرة. ولا يستطيع أبدًا استهداف إجراء مخصّص من مستوى **UI**؛ فهذه الإجراءات لا تُشغَّل إلا من زرّ (انظر نوع الإجراء المخصّص).

هيّئه في ثلاث خطوات كحدّ أقصى؛ ويوضّح المخطّط كيفية اختيار **النطاق (Scope)** عندما يكون الهدف سياقيًا:



1. **الإجراء المخصّص** – اختر إجراءً من قائمة منسدلة قابلة للبحث تضمّ الإجراءات المخصّصة القابلة للاستدعاء في الوحدة الحالية، مُجمَّعة تحت **Local Actions** (المملوكة لكيان في هذه الوحدة) و**External Actions** (المملوكة لكيان في وحدة أخرى، والموسومة باسم تلك الوحدة). ولا تُعرَض الإجراءات المخصّصة من نوع **UI** أبدًا.
2. **النطاق (Scope)** – يظهر فقط عندما يكون الإجراء المخصّص المختار **سياقيًا** (انظر نوع الإجراء المخصّص وخيار Contextual)؛ فالإجراء غير السياقي ليس له مُدخل هدف، لذا تُتخطّى هذه الخطوة ويُشغَّل الإجراء مرة واحدة فقط. وفيما يلي طرق تحديد المُدخل الهدف (أو المُدخلات):
   - **Self** – شغّل الإجراء على **المدخل الحالي**. لا يُعرَض إلا عندما يطابق الكيان الحالي كيان الإجراء المخصّص نفسه، بما في ذلك عندما يكون الكيان الحالي **كيانًا وهميًا (Virtual Proxy)** ينوب عنه؛ وعندها يُشغَّل الإجراء على المُدخل الذي ينوب عنه الوكيل.

- **Relation** – اتبِع علاقة من المُدخل الحالي إلى المُدخل الهدف (أو المُدخلات). يعمل في الاتجاهين: تُفضي العلاقة الصادرة إلى هدف واحد، والواردة إلى كل مُدخل يشير إلى المُدخل الحالي.
- **Expression** – قيّم تعبيرًا برمجيًا باستخدام سياق التشغيل؛ ويجب أن ينتمي كل مُدخل ينتج عنه إلى كيان متوافق، وإلّا فشل الإجراء بسبب خطأ في الإعداد. لذا وجّه التعبير بدقّة إلى الكيان المقصود.
- **Pointer** – استخدم مؤشِّرًا سياقيًا (المستخدم الحالي مثلًا) لتحديد المُدخل الهدف.
- **Reference** *(تراثي)* – اتبِع حقل المرجع في المُدخل الحالي وصولًا إلى الهدف. لا يزال يعمل بالكامل للتفاعلات القائمة وعند استخدام واجهة برمجة التطبيقات مباشرةً، لكنّ الباني لم يعد يعرضه للإعدادات الجديدة؛ استخدم **Relation** بدلًا منه.

3. **Parameters** – تظهر كلّما كان للإجراء المخصّص المختار **مُعامِلات (Parameters)** مفعَّلة (مفتاح اسأل المستخدم الخاص به – انظر نوع الإجراء المخصّص)، بصرف النظر عمّا إذا كان النطاق يظهر أيضًا. اربط كل مُعامِل بالطريقة نفسها التي تُسنِد بها أي قيمة إجراء: **Manual** (قيمة ثابتة)، أو **Dynamic** (حقل من المُدخل الذي أطلق هذا التشغيل)، أو **Expression**، أو **Pointer**. ويُضبَط الربط لكل استدعاء على حدة، بحيث يمكن استدعاء الإجراء المخصّص نفسه من إجراءَي تنفيذ إجراء مخصّص مختلفين بقيم مُعامِلات مختلفة. ولا يُعدّ ترك مُعامِل دون ربط خطأً، بل يُمرَّر فارغًا.

ما الذي يحدث عند التشغيل. يحدّد الإجراء المُدخلات الهدف وفق النطاق المختار (أو لا يحدّد أيّ مُدخل للإجراء غير السياقي)، ويحسب قيم أيّ مُعامِلات مربوطة، ثم يُطلق حدث الإجراء المخصّص نفسه الذي يُطلقه زرّ النظام. ويلتقطه مستهلِك الإجراء المخصّص القائم، فيشغّل الإجراء المخصّص الهدف **مرّة واحدة لكلّ مدخل هدف، كلٌّ منها في سياق معزول خاصّ به**؛ إذ لا يرى التفاعل الهدف إلّا بيانات ذلك المُدخل، دون سياق التفاعل المُستدعي. ويسجّل مُخرَج الإجراء عدد المُدخلات التي أُرسِل إليها (`1` للإرسال غير السياقي).

أنماط فشل الإعداد التي ينبغي معرفتها:

- يُرفَض عند الحفظ استهداف إجراء مخصّص **غير موجود**، أو **ليس من مستوى Local أو External**، أو **معطَّل**؛ إذ لا يستطيع إجراء تنفيذ إجراء مخصّص أن يستهدف إجراءً مخصّصًا من مستوى **UI**.
- **يُرفَض نطاق Self عبر كيانين مختلفين** وقت الحفظ؛ فإن لم يكن الكيان الحالي وكيلًا (proxy) عن كيان الإجراء المخصّص، فشل الحفظ.
- **نطاقا Reference/Relation دون هدف خطأُ إعداد.** يُرفَض الحفظ إن لم يُضبَط حقل المرجع (أو العلاقة).
- **تُرفَض Parameters كلّيًا إن لم تكن للإجراء المخصّص الهدف مُعامِلات مفعَّلة**؛ فلا يمكن ربط قيم بإجراء لم تُفعَّل مُعامِلاته، إذ لا يملك موضعًا لاستقبالها.
- **عدم حلّ أيّ مدخلات هدف *ليس* خطأً.** إذا لم تُفضِ Relation إلى أيّ مُدخل، ينجح الإجراء ولا يُرسِل شيئًا، ولا يتسبّب في فشل التفاعل. فإن كنت تتوقّع تنفيذ عمل، فراجِع عدد المُدخلات المُرسَل إليها في نتائج التنفيذ والسجلّ بدلًا من البحث عن خطأ. أمّا Expression وPointer فأكثر صرامة: إذا لم تُفضِ نتيجتهما إلى كيان متوافق، فشل الإجراء كلّيًا بدلًا من تخطّيه بصمت.

إجراءا منح الإذن وإزالته `Since 1.10`

يتيح نوعان جديدان من الإجراءات للتفاعل إدارة **أذونات الوصول**، أي المنح نفسها التي يضبطها المدير في شاشات الأذونات. فإجراء **منح إذن** (`SetPermission`، 31) يمنح مستخدمين أو مجموعات مستوى وصول إلى مورد، وإجراء **إزالة إذن** (`RemovePermission`، 32) يسحب ما مُنح لهم. ومن الاستخدامات المعتادة منح المكلَّف بطلب وصولًا إلى تقرير عند بلوغ خطوة، أو إبطال وصول عضو إلى وحدة عند أرشفة مُدخله. ويكتب الإجراءان مباشرةً عبر مسار الأذونات في خدمة البيانات، دون المرور بأمر؛ ولذلك لا يتركان التشغيل في حالة انتظار أبدًا، بخلاف إجراءات البيانات (انظر ما الذي تتوقّعه من نموذج التنفيذ).

الخيار	المعنى
`resource.entitlementType`	`System` أو `Module` أو `DataEntity` أو `DataView` أو `GenericView` (الصفحات) أو `Report` .
`resource.entitlementId`	معرّف المورد – مطلوب لكل نوع ما عدا `System` ، حيث يجب ألا يكون موجودًا.
`principal.kind`	`USER` أو `GROUP` .
`principal.source`	مصدر الجهات الممنوحة، ويُهيّأ بالطريقة نفسها التي يُهيّأ بها مصدر إجراء تفعيل / تعطيل مستخدم: مرجع ثابت، أو مسار حقل في سياق التشغيل. وقد يُفضي المصدر الواحد إلى عدّة جهات.
`level`	**للمنح وحده** – مستوى الوصول المراد منحه.

لا يجوز للتفاعل أن يمنح إلا المستويات التالية؛ ولا يُمنح `Restricted` ولا `None` أبدًا:

المورد	المستويات القابلة للمنح
`System`	Admin، Editor، Developer، Viewer
`Module` و `DataEntity`	Editor، Developer، Viewer
`DataView` و `GenericView` و `Report`	Viewer

يُعدّ الخيار المفقود أو غير السليم خطأ تحقّق عند الحفظ (`missingRequired` / `invalidData`)، مع تحديد مسار الخيار المخالف. ويشمل ذلك غياب المورد، أو وجود نوع غير معروف، أو وجود `entitlementId` على `System` أو غيابه في سواه، أو غياب الجهة، أو وجود نوع غير `USER` / `GROUP` ، أو مصدر غير صالح، أو مستوى مفقود أو غير قابل للمنح في إجراء المنح.

لا يستطيع التفاعل أن يمنح أكثر ممّا يملكه من يحفظه. فالتفاعل يعمل بهوية النظام، ولولا هذا القيد لكان حفظه وسيلة لتصعيد الصلاحيات. لذلك، عند إنشاء تفاعل أو تحديثه، يُفحص كل إجراء إذن جديد أو معدَّل مقابل **الوصول الفعلي للمستخدم الذي يحفظه** إلى مورد ذلك الإجراء: يتطلّب المنح أن يملك المستخدم المستوى المراد منحه على الأقل، وتتطلّب الإزالة مستوى Admin. ويُعبَّر عن الرفض بخطأ تحقّق يحمل الرمز `PERMISSION_ACTION_EXCEEDS_SAVER_ACCESS` وتفاصيله `{entitlementType, entitlementId, requiredLevel}` (ويعرض الباني رسالة مترجمة)؛ وإذا تعذّرت قراءة مستوى وصول المستخدم الذي يحفظ التفاعل، فشل الحفظ احتياطًا. ولا تُستثنى إجراءات الإذن غير المعدَّلة من الفحص إلا حين يكون التحديث وصفيًا بحتًا (الاسم أو الوصف)؛ أمّا أيّ تغيير آخر – المشغّلات أو المعايير أو إجراءات أخرى أو إعادة تفعيل التفاعل – فيؤدّي إلى إعادة فحصها جميعًا، لأنّ المنح سيجري في ظروف جديدة. ويجري الفحص عند الحفظ فقط: فالتفاعل المحفوظ سابقًا يواصل منح الأذونات التي حُفظ بها.

أثناء التشغيل تُعالَج كل جهة جرى تحديدها على حدة:

- **المنح لا يُخفّض أبدًا.** فالجهة التي تملك مباشرةً على المورد مستوى أعلى قابلًا للمنح تُتخطّى بالرمز `ALREADY_HIGHER` . أمّا **الإزالة** فتكتب المستوى `None` .
- **المستخدم** الذي يُرفَض منحه على `Module` أو `DataEntity` لأنه لا يملك وصولًا إلى النظام بعد (`USER_MUST_HAVE_VIEW_ACCESS_FIRST`) يُضاف أولًا إلى النظام بمستوى **Restricted**، ثم تُعاد محاولة الكتابة مرة واحدة. ولا تُضاف المجموعات بهذه الطريقة.
- تفشل معالجة الجهة المحدَّدة إذا كان نوعها غير مطابق، بالرمز `PRINCIPAL_KIND_MISMATCH` ، ويُسجَّل أيّ خطأ آخر من الخدمة بالرمز `UPSTREAM_ERROR` ؛ أمّا المصدر الذي لا يُفضي إلى أيّ جهة فيُفشل الإجراء كله بالرمز `NO_PRINCIPALS` .
- مُخرَج الإجراء هو `{applied[], failed[{principalId, code}], skipped[{principalId, code}]}` . ويُعدّ الإجراء فاشلًا متى لم تكن `failed` فارغة، مع استمرار التشغيل وإتاحة المُخرَج للإجراءات اللاحقة كأيّ مُخرَج آخر.

مخرجات «تنفيذ وظيفة» في التفاعلات المتسلسلة `Since 1.10`

أصبح من الممكن استخدام استجابة الوظيفة لتوجيه التنفيذ اللاحق. فإجراء **تنفيذ وظيفة** يعرض في مخطّط التفاعل ثلاثة حقول مُخرَجات – `success` (منطقي) و`message` (نص) و`result` (كائن JSON مفتوح) – بحيث تستطيع الإجراءات اللاحقة في التفاعل نفسه، والتفاعلات المتسلسلة المترتّبة على نجاحه أو فشله، ربطها كأيّ مُخرَج إجراء آخر. ولأنّ `result` بلا بنية مُعلَنة، تتيح أدوات الاختيار في الباني ربط **مفتاح تحته** (مثل `result.customer.id`)، بدلًا من الاقتصار على ربط الكائن كله، ويستخرج المحرّك قيمة ذلك المسار من الكائن المُعاد عند تشغيل التفاعل. فحافظ على ثبات بنية ما تُعيده الدالّة، إذ تعتمد الروابط على مسارات داخلها.

معايير نجاح الإجراء

ترث المعالِجات المؤثّرة في المُدخلات من `BaseEntryActionHandler`، الذي يقرأ `options.actionSuccessCriteria` (`ActionSuccessCriteriaEnum`، الافتراضي `Any=0`؛ `All=1`). عند استخدام `Any`، لا يُعدّ الإجراء فاشلًا إلا عند وجود أخطاء **و**عدم نجاح أيّ مُدخل؛ أمّا مع `All`، فيؤدّي أيّ خطأ إلى فشل الإجراء. وعند النجاح، تُسجَّل المُدخلات المتأثّرة في `ActionResult.effectedEntities` ويُكتَب المُخرَج مجدّدًا في سياق التشغيل عبر `setActionOutput`، بحيث تستطيع الإجراءات اللاحقة الإشارة إليه.

مرجع خيارات الإجراءات

لكلّ إجراء في `actions[]` الخاصة بالتفاعل الغلافُ نفسه؛ ولا يتغيّر إلا `options` بحسب `actionType`:

```
{
  "id": "<uuid>",
  "normalizedName": "notify_requester",
  "name": {"en": "Notify requester"},
  "actionType": 100,
  "resultField": "",
  "options": { }
}
```

يُضبَط `resultField` وقت التشغيل. أرسِله سلسلةً فارغة (`""`)؛ فقيمة `null` المقروءة من الواجهة البرمجية تُرفَض عند الكتابة.

إجراء واحد لكل تفاعل. تخزّن الواجهة البرمجية مصفوفة، لكنّ التحقّق عند الحفظ مكتوب لإجراء واحد في التفاعل، والباني يحرّر إجراءً واحدًا. صمِّم أيّ تسلسل على هيئة تفاعلات منفصلة مربوطة بـ `ReactionSucceeded` (انظر تسلسل التفاعلات).

اللبنات المشتركة

تعيين القيم – `EntryActionOptionsValue`

يُستخدَم في `values` (إنشاء المدخل وتحديثه، وعناصر الكتلة)، و`parameters` (تنفيذ إجراء مخصّص)، و`schemaMapping` (إعلان حدث):

```
{"fieldName": "status", "valueType": 2, "value": {"id": "<category option id>"}}
```

`valueType` (`FieldValueTypeEnum`)	ما تحمله `value`	مثال
`0` Default · `1` NotSet	لا شيء / القيمة الافتراضية للحقل	–
`2` Manual	قيمة حرفية بالشكل المخزَّن في الحقل **الهدف** (الفئة `{"id": "<option id>"}`، والارتباط `{"dataEntity": "...", "dataEntry": "..."}`)	`true`، `"Open"`، `{"id": "…"}`

valueType (FieldValueTypeEnum)	ما تحمله value	مثال
3 Dynamic	مسار في سياق التشغيل: value.<field> على المدخل الذي أطلق الحدث، أو مسار كامل scope.entries.<entity>.value.<field>	"value.requester"
4 Expression	شيفرة JavaScript تُقيَّم مع value (قيمة المدخل **المُطلِق** – حتى عندما يحدّث الإجراء مدخلًا آخر) و scope	"(value.reopen_count || 0) + 1"
5 Pointer	مؤشّر يبدأ بـ @ ، انظر أدناه	"@now()"

يُشار إلى الحقل الفرعي في الكتلة داخل إجراء الكتلة باسمه **المجرّد** ("level" ، لا "escalation_event.level").

المؤشرات

تبدأ قيمة المؤشّر بـ @ . المؤشّر المكتوب **مع** القوسين يُشغَّل على نطاق يوفّر أيضًا thisReference ؛ أمّا المكتوب **بدونهما** فيُشغَّل على هيئة scope.<name>() على النطاق الافتراضي وحده.

المؤشّر	يُحَلّ إلى
@thisReference()	{dataEntity, dataEntry} للمدخل الذي أطلق الحدث. **القوسان إلزاميّان** – فـ @thisReference المجرّد لا يُحَلّ إلى شيء.
@me / @me() · @current_user()	المستخدم المنفِّذ (مع المنفِّذ SYSTEM : مستخدم النظام)
@now()	التاريخ والوقت الحاليّان بصيغة ISO
@today('+7d')	سلسلة تاريخ فقط؛ ويجب أن تكون الإزاحة بإشارة (+ / -)، والوحدات d w M y h m s

خرائط الرموز – tokensMap و TokenSource

تملأ إجراءات البريد والإشعار والرسائل النصية واستدعاء الواجهة البرمجية والدالّة المخصّصة العناصرَ النائبة {{token}} من tokensMap ، وهو **كائن مفاتيحه أسماء الرموز**:

```
"tokensMap": {
  "ticket_no": {"type": 1, "source": "scope.entries.tm_ticket.value.ticket_no"},
  "greeting":  {"type": 0, "source": "Hello"}
}
```

type (TokenSourceType)	ما يحمله source
0 Fixed	النص الحرفي
1 Field	مسار في السياق – scope.entries.<entity normalizedName>.value.<field> ؛ وفي التفاعل المتسلسل scope.entries._<triggerId>_event.value.<field> أو scope.entries._<triggerId>__<actionId>_action.value.<field> (مع استبدال الشَّرطات في المعرّفات بـ _)
2 Expression	شيفرة JavaScript تُقيَّم على السياق
3 Pointer	مؤشّر يبدأ بـ @

قواعد تُفرَض عند الحفظ حين يكون template **مضمَّنًا** (كائنًا): يجب أن تطابق مفاتيح الخريطة رموزَ القالب تمامًا (INVALID_TOKENS_MAP)، ويحتاج كل إدخال إلى type (Required parameter tokensMap.<i>.type is missing). أمّا حين يكون template **معرّف قالب مخزَّن** فلا يُتحقَّق من شيء عند الحفظ: والرمز غير المعيَّن يُرسَل نصًّا حرفيًا {{name}} .

المستلمون

```
{"sourceType": 0, "source": {"type": 1, "value": "value.requester"}}
{"sourceType": 1, "source": {"type": 0, "value": {"dataEntity": "GROUP", "dataEntry": "<group id>"}}}
{"sourceType": 4, "source": {"type": 0, "value": "ops@example.com"}}
```

الحقل	القيم
`sourceType` – البريد (`EmailSourceType`)	`0` Manual · `3` OrgUnit · `4` DataCollection · `1` Group · `2` User (عنوان بريد)
`sourceType` – الإشعار (`StepOwnerType`)	`0` OrgUnit · `3` DataCollection · `2` Group · `1` User
`source.type` (`StepOwnerSourceType`)	`0` Fixed (`ReferenceProperty`، أو عنوان في حالة Manual) · `1` Dynamic (مسار حقل) · `2` PreviousOwner · `3` Expression

يستخدم البريد `toRecipients` / `ccRecipients` / `bccRecipients`؛ ويستخدم الإشعار `recipients`. واسم المفتاح الخاطئ لا يوجِّه الرسالة إلى أحد، بصمت.

اختيار المدخل المراد الكتابة فيه – `updateEntryMode` و`entryToUpdate`

`updateEntryMode`	`entryToUpdate`
`0` Self – المدخل الذي أطلق الحدث	**احذفه.** فـ `{"mode": 0}` تعني *Fixed* وتتطلّب عندئذٍ `value` (`Required parameter "options.entryToUpdate.value" is missing`).
`1` Reference – مدخل يشير إليه حقل ارتباط	مطلوب. `{"mode": 1, "fieldName": "<reference field>"}` (Dynamic) أو `{"mode": 0, "value": {"dataEntity": "…", "dataEntry": "…"}}` (Fixed)
`2` Relation – مدخلات يُوصَل إليها عبر علاقة	مطلوب، مع `relationId` (و`targetEntityId` حين تكون للعلاقة عدّة مصادر)

`actionSuccessCriteria`: `0` Any (الافتراضي – يفشل فقط إن لم ينجح أيّ مدخل) · `1` All.

الخيارات بحسب نوع الإجراء

إجراءات البيانات

```
// 10 CreateDataEntry — entitySource: {"entityId"} or {"fieldName"} (entity taken from a field)
{"entitySource": {"entityId": "<entity id>"},
 "values": [{"fieldName": "request", "valueType": 5, "value": "@thisReference()"},
            {"fieldName": "message", "valueType": 2, "value": "Auto-acknowledged"}]}

// 11 UpdateDataEntry — self
{"updateEntryMode": 0,
 "values": [{"fieldName": "closed_at", "valueType": 5, "value": "@now()"}]}

// 11 UpdateDataEntry — through a reference field
{"updateEntryMode": 1, "entryToUpdate": {"mode": 1, "fieldName": "parent_contract"},
 "values": [{"fieldName": "last_request", "valueType": 5, "value": "@thisReference()"}],
 "actionSuccessCriteria": 0}

// 12 ArchiveDataEntry — sourceMode 0 CurrentEntry · 1 RelatedEntries · 2 DataQuery
{"sourceMode": 1, "relation": {"relationId": "<relation id>"}, "querySource": null}

// 16 DeleteArchivedEntry — sourceMode 0 CurrentEntry · 1 Reference · 2 Relation · 3 DataQuery
{"sourceMode": 3, "relation": null, "querySource": {"dataEntity": "<entity id>", "dataEntry": "<saved query id>"}}

// 2 ArchiveDataEntity
{"entitySource": {"entityId": "<entity id>"}}
```

إجراءات الكتلة القابلة للتكرار

تأخذ إجراءات الكتلة خيارات تحديث المدخل إضافةً إلى `blockField`؛ ويضيف التحديث والحذف معيارًا للصفوف `criteria`.

```
// 13 CreateBlockItem — sub-fields by BARE name
{"updateEntryMode": 0, "blockField": "sla_event",
 "values": [{"fieldName": "level", "valueType": 2, "value": 1},
            {"fieldName": "breached_at", "valueType": 5, "value": "@now()"}]}

// 14 UpdateBlockItem / 15 DeleteBlockItem (Delete has no "values")
{"updateEntryMode": 0, "blockField": "sla_event",
 "criteria": {"type": 3, "onFailure": {"type": 0}},
 "values": [{"fieldName": "acknowledged", "valueType": 2, "value": true}]}
```

`criteria.type`	`criteria.value`
FixedIndex `0`	مواضع و/أو قيم `__id` للصفوف (رقم، أو سلسلة، أو مصفوفة منها)
Field `1`	مسار حقل يُحَلّ إلى قيمة كهذه
Expression `2`	تعبير يُقيَّم إلى قيمة كهذه
LastRecord `3` · AllRecords `4`	–

`onFailure.type` : `0` FallbackToLastOrSkip · `1` FallbackToLastOrFail · `2` Fail · `3` Skip.

إجراءات سير العمل

```
// 20 MoveWorkflowToStep
{"workflowId": "<workflow id>", "targetStepId": "<step id>"}

// 21 ExecuteWorkflowAction — one entry per step the reaction may automate;
//    only the entry matching the entry's CURRENT step runs, then the engine
//    keeps automating while the next step also has a matching entry.
{"actions": [{"workflowId": "<workflow id>", "stepId": "<step id>", "actionId": "<action id>"}]}

// 22 ChangeCurrentStepOwner — owner shape as a workflow step owner
{"type": 1,
 "source": {"type": 0, "value": {"dataEntity": "GROUP", "dataEntry": "<group id>"}},
 "ownerOptions": {"allowChangeOwner": true, "mode": 0, "accessPermission": 1, "selectionCriteria": 1},
 "fallbackOwner": {"allowChangeOwner": true},
 "permanentChange": false}
```

بإجراء «تنفيذ إجراء سير العمل» «يضغط» التفاعلُ زرَّ سير العمل – مثل تفاعل اتفاقية مستوى الخدمة الذي يغلق مدخلًا تُرك في خطوة *Resolved*. وإدراج `targetStepId` (النقل) أو `actions` (التنفيذ) في `overridableOptions` للإجراء يتيح للمستخدم اختيارهما وقت التشغيل بدلًا من ذلك.

إجراءات المراسلة

```
// 100 SendNotification — channels: only 0 (InApp)
{"channels": [{"type": 0}],
 "recipients": [{"sourceType": 0, "source": {"type": 1, "value": "value.assignee"}}],
 "template": "<NOTIFICATION_TEMPLATE id>",
 "module": {"dataEntity": "DataModule", "dataEntry": "<module id>"},
 "tokensMap": {"ticket_no": {"type": 1, "source": "scope.entries.tm_ticket.value.ticket_no"}},
 "customRedirection": {"enabled": false}}

// 101 SendEmail — from.mode 0 System · 1 ReactionExecutor · 2 Custom (+ customFrom recipient)
{"toRecipients": [{"sourceType": 0, "source": {"type": 1, "value": "value.requester"}}],
 "ccRecipients": [], "bccRecipients": [],
 "template": "<EMAIL_TEMPLATE id>",
 "tokensMap": {"ticket_no": {"type": 1, "source": "scope.entries.tm_ticket.value.ticket_no"}},
 "from": {"mode": 0},
 "link": {"enabled": true, "viewId": "<view id>"},
 "attachments": [{"type": 1, "value": "value.contract_file"}]}

// 102 SendSms — recipients sourceType 0 Entity · 1 DataCollection · 2 Static; template is inline
{"recipients": [{"sourceType": 2, "source": {"type": 1, "value": "value.mobile_number"}}],
 "template": {"content": "Ticket {{ticket_no}} was resolved", "tokens": [{"name": "ticket_no", "location": ["conten
t"]}]},
 "tokensMap": {"ticket_no": {"type": 1, "source": "scope.entries.tm_ticket.value.ticket_no"}},
 "link": {"enabled": false}}

// 107 SendWhatsApp — an approved Meta template; parameters[i] fills {{i+1}}; recipients as Send SMS
{"recipients": [{"sourceType": 2, "source": {"type": 1, "value": "value.mobile_number"}}],
 "templateName": "ticket_resolved", "language": "en",
 "parameters": [{"type": 1, "source": "scope.entries.tm_ticket.value.ticket_no"}]}
```

يمكن أن يكون `template` مضمَّنًا أيضًا بدلًا من معرّف قالب مخزَّن — للبريد `{"subject", "body", "tokens"}`، وللإشعار `{"name", "title", "content", "icon", "tokens"}`، وكل رمز `{"name", "location": [...]}` (مثل `["subject"` أو `"body"]` أو `["title.en"]`). و `attachments[].type` في البريد: `0` Fixed (ملف مخزَّن) · `1` Dynamic (مسار حقل ملف). والمرفق Fixed ملف مخزَّن يحمل التفاعل الآن مرجعًا إليه: فإن انتهت صلاحية رفعه المؤقّت قبل الحفظ، فشل الحفظ بـ `FILE_EXPIRED` (انظر الملفات في خيارات التفاعل).

خيارات إرسال WhatsApp `Since 1.10`

الخيار	يحمل
`recipients`	أشكال المستلمين نفسها في إرسال SMS
`templateName`	اسم قالب Meta **معتمد** في حساب WhatsApp Business الخاص بالمساحة
`language`	رمز لغة القالب، مثل `en` أو `ar`
`parameters`	مصادر رموز (Fixed أو Field أو Expression أو Pointer، كما في خرائط الرموز)؛ العنصر `i` يملأ `{{i+1}}` في متن القالب. ويجوز أن تكون فارغة.

- **اختر القالب من القائمة.** يسرد الباني القوالب المعتمدة للمساحة حيّةً من `GET /api/reaction/v1/whatsapp/templates` — `[{name, language, category, bodyParameterCount}]`، ودون أيّ قالب من فئة `AUTHENTICATION`، لأن الإجراء يُرسل دائمًا بالفئة `other`. ويتطلّب استدعاؤها وصول عرض على مستوى النظام فما فوق (ويُرفض المستخدم النهائي الذي لا يملك إلا الحدّ الأدنى `Restricted`). وتعيد `409`

`whatsapp_not_configured` حين لا يكون للمساحة معرّف حساب أعمال أو رمز وصول، و`502` `whatsapp_provider_error` برمز خطأ Meta (لا رسالة Meta ولا الرمز أبدًا) حين يفشل Meta. ويُبيّن النموذج أيّ هذه منع تحميل القائمة، ويُعلِّم القالب المحفوظ الذي لم يعد معتمدًا، ويُسقط المعاملات المحفوظة الزائدة على عدد متغيّرات القالب المختار. وإعدادات الحساب هي الكتلة `whatsapp` في إعدادات النظام.

- **الحفظ لا يتحقّق من القالب مقابل Meta.** لا يُكتشف عند حفظ التفاعل الاسم أو اللغة أو عدد المعاملات الخاطئ؛ فاجعل عناصر `parameters` بعدد `bodyParameterCount` تمامًا.
- **ما يُرسَل.** تُرسَل القيمة الفارغة `''` (تُحفظ المواضع لأن Meta ترفض المعامل الناقص)؛ وتُرسَل القيمة المحلية باللغة الافتراضية للقاطن لا بـ `language` القالب؛ وتُرسَل القيمة الحساسة المحجوبة `****`؛ وتُطوى الأسطر الجديدة والجدولات وتتابعات الفراغات الخمسة فأكثر إلى فراغ واحد وتُقصّ القيمة (ترفضها Meta، الخطأ 132018). ويخرج طلب رسالة واحد لكل المستلمين؛ وإن لم يُحلّ أيّ رقم هاتف لم يُرسَل شيء ونجح الإجراء مع ذلك.
- **المُخرَج:** `{templateName, recipientsPhoneNumbers}`، وتقرؤه الإجراءات اللاحقة والتفاعلات المتسلسلة.

إجراءات التكامل والأتمتة

```
// 90 CallApi — verb "GET" | "POST" | "PATCH" | "PUT" | "DELETE"; success = any 2xx
{"apiUrl": {"value": "https://api.example.com/tickets/{{ticket_no}}"},
 "verb": "POST",
 "headers": {"value": "{\"Content-Type\": \"application/json\"}"},
 "body": {"custom": true, "customBody": {"value": "{\"ref\": \"{{ticket_no}}\"}"}},
 "tokensMap": {"ticket_no": {"type": 1, "source": "scope.entries.tm_ticket.value.ticket_no"}}}

// 103 GenerateFileFromTemplate — retentionPeriod in days (omit = keep)
{"templateId": "<template id>", "retentionPeriod": 30,
 "fileOutputName": {"type": 2, "source": "'Contract-' + scope.entries.contract.value.ref"},
 "downloadFile": false}

// 104 ExecuteCustomFunction — resolved tokens reach the function as payload.parameters.<key>
{"functionId": "<function id>",
 "tokensMap": {"customer": {"type": 1, "source": "scope.entries.order.value.customer"}}}

// 105 AnnounceEvent
{"announcementId": "<announcement id>",
 "schemaMapping": [{"fieldName": "order_ref", "valueType": 3, "value": "value.ref"}]}

// 106 ExecuteCustomAction — scope 0 Self · 1 Reference · 2 Expression · 3 Relation · 4 Pointer
//     Reference: entryToRun {mode, fieldName | value} · Relation: entryToRun {relationId}
//     Expression: "expression" · Pointer: "pointer". Send only the chosen scope's setting,
//     and no scope at all when the custom action is not contextual.
{"customActionId": "<CustomEvent reaction id>", "scope": 3,
 "entryToRun": {"relationId": "<relation id>"},
 "parameters": [{"fieldName": "reason", "valueType": 2, "value": "Escalated"}]}

// 200 NotifyZapier
{}
```

استدعاء API: ما تسمح به المنصة `Since 1.8`

إجراء استدعاء API طلب HTTP يُبنى في الخادم من إعدادك، فتضع له المنصة حدودًا (الثغرة الأمنية N-14):

القاعدة	ما تراه
HTTP(S) فقط	أيّ مخطّط آخر، أو عنوان لا يُحلَّل، يُفشل الإجراء بـ `CALL_API_DESTINATION_BLOCKED`.

القاعدة	ما تراه
لا وجهات خاصة أو داخلية	العنوان الواقع في نطاق خاص أو loopback أو link-local (ومنه عنوان بيانات السحابة الوصفية) أو CGNAT أو multicast أو محجوز، في IPv4 أو IPv6، يُفشل الإجراء. وتُفحص أسماء المضيفات عند حلّها، فيُرفض الاسم العام الذي يشير إلى عنوان خاص، ويُعاد الفحص عند كل إعادة توجيه (5 على الأكثر). وتُتجاهل إعدادات الوكيل (proxy). ولا يفتح وجهةً خاصة إلا المشغِّل، عبر قائمة السماح `CALL_API_ALLOWED_PRIVATE_DESTINATIONS`.
حدّ 30 ثانية	يجب أن ينتهي التبادل **كلّه** خلال `CALL_API_TIMEOUT_MS` (الافتراضي 30 ثانية)؛ وتنتهي مهلة الاستجابة المُقصَّرة ببطء أيضًا.
سقف 10 MiB	يُحدّ جسما الطلب والاستجابة، كما يُسجَّلان في سجلّ التشغيل، بـ `CALL_API_MAX_RESPONSE_BYTES` (الافتراضي 10 MiB)؛ والاستجابة الأكبر تُفشل الإجراء.
الرموز لا تضيف ترويسات	يُقرأ قالب `headers` بوصفه JSON أولًا، ثم تُملأ الرموز في اسم كل ترويسة وقيمتها كلٌّ على حدة، فلا تستطيع قيمة رمز أن تضيف ترويسة أو تستبدلها.
ترويسات المنصة تغلب	تضبط المنصة `x-space-id` و`Content-Type: application/json` وتتقدّمان على ترويساتك.
القيم المحجوبة `Since 1.10`	الاستدعاء الذي يحمل عنوانه أو ترويساته أو جسمه قيمة حسّاسة محجوبة يفشل بدل إرسال `****` (انظر سياسة وصول البيانات الحسّاسة).

أسماء الترويسات المحجوزة `Since 1.10`. `x-shared-key` و`x-app-id` و`x-space-id` و`x-service` (بأيّ حالة أحرف) ترويسات المنصة نفسها. وقالب `headers` الذي يسمّي إحداها يُرفض عند الحفظ بـ `400 RESERVED_HEADER_NAME`. أما التفاعل المحفوظ قبل هذا الفحص فيظلّ يعمل: تُسقَط الترويسة المحجوزة وقت التشغيل.

المستخدمون والصلاحيات

`source.type` (`UserActivationToggleSourceTypeEnum`): `0` Fixed (`ReferenceProperty`) · `1` Field · `2` Expression · `3` Pointer (و`value` هي المسار أو التعبير أو المؤشّر).

```
// 30 UserActivationToggle — type 0 Activate · 1 Deactivate
{"source": {"type": 1, "value": "value.employee"}, "type": 1, "actionsSuccessCriteria": 0}

// 31 SetPermission (32 RemovePermission: the same without "level")
{"resource": {"entitlementType": "DataView", "entitlementId": "<view id>"},
 "principal": {"kind": "USER", "source": {"type": 1, "value": "value.assignee"}},
 "level": 0}
```

`entitlementType`: `System` (ويجب حينها أن يكون `entitlementId` قيمة `null`) · `DataModule` · `DataEntity` · `DataView` · `GenericView` · `Report`. و`level`: `0` Admin · `1` Edit · `2` Developer · `0.5` View، ضمن المستويات المذكورة في إجراءا منح الصلاحية وإزالتها.

أخطاء التحقّق عند الحفظ

يتحقّق إنشاء التفاعل أو تحديثه من الخيارات ويرفض الحفظ مع تسمية المسار المخالف:

الخطأ	الحالة	السبب
`MUST_AUTOMATE_AT_LEAST_ONE_STEP`	400	«تنفيذ إجراء سير العمل» مع `actions` فارغة أو مفقودة (ودون تجاوز لـ `actions`)
`TRIGGERS_CAN_NOT_AUTOMATE_LAST_STEP`	400	إدخال في `actions[]` على الخطوة الأخيرة من سير العمل
`WORKFLOW_NOT_EXIST` · `WORKFLOW_STEP_NOT_EXIST` · `WORKFLOW_ACTION_NOT_EXIST`	404	معرّف في خيارات النقل / التنفيذ غير موجود على الكيان
`Required parameter options.entryToUpdate[.value | .fieldName | .relationId] is missing`	400	انظر اختيار المدخل المراد الكتابة فيه

الخطأ	الحالة	السبب
`Required parameter tokensMap.<i>.type is missing` · `INVALID_TOKENS_MAP`	400	انظر خرائط الرموز
`Required parameter options.customRedirection.viewId is missing`	400	«إرسال إشعار» مع `customRedirection.enabled: true` ودون `viewId`
`Required parameter options.entryToRun[.relationId | .fieldName، options.expression | .value] options.pointer`	400	نطاق «تنفيذ إجراء مخصّص» دون إعداده
`missingRequired` / `invalidData` على `options.resource`، `options.principal`، `options.level`، `options.source`	400	خيارات منح الصلاحية / إزالتها وتبديل تفعيل المستخدم
–	422	نوع الإجراء `0` أو `1` (متقاعد)
`missingRequired` / `invalidData` على `eventConfig.entityId` و `eventConfig.triggerId` و `spaceScope`	400 / 422	تفاعل `ReactionSucceeded` / `ReactionFailed` لا يمكن أن يُطلَق أبدًا، أو بنطاق مساحة غير مساحة التشغيل. `Since dev` – انظر أدناه
`RESERVED_HEADER_NAME`	400	قالب `headers` في استدعاء API يسمّي ترويسة محجوزة للمنصة. `Since 1.10` – انظر استدعاء API
`FILE_EXPIRED`	400	تشير الخيارات إلى رفع مؤقّت (مرفق بريد مثلًا) انتهت صلاحيته قبل الحفظ. `Since 1.10` – انظر الملفات في خيارات التفاعل
`STORAGE_SERVICE_UNAVAILABLE`	503	تعذّر بلوغ تخزين الملفات بينما يضيف الحفظ ملفًا؛ أعد المحاولة. `Since 1.10`
`REACTION_AUTHORING_FORBIDDEN`	403	لا تملك وصولًا بمستوى مطوّر فما فوق على كيان التفاعل (إنشاء وتحديث وحذف). `Since 1.8` – انظر من يحقّ له تأليف تفاعل

أشكال إعداد الحدث المستخدمة مع هذه الإجراءات

`eventType`	`eventConfig`
`1` DataEntryUpdated	`{"fields": [{"fullNormalizedName": "sla_breached"}]}` – القائمة الفارغة تطابق أيّ تغيير
`3` WorkflowActionExecuted · `5` WorkflowStepExited	`[{"steps": [{"workflowId": "…", "stepId": "…", "actions": ["<action id>"]}]}]`
`4` WorkflowStepReached	`[{"steps": [{"workflowId": "…", "stepId": "…"}]}]`
`6` Timer	`{"timerOptions": {"repeat": {"expression": "*/5 * * * *", "options": {"timezone": "Asia/Dubai"}}}}`
`8` ReactionSucceeded · `10` ReactionFailed	`{"triggerId": "<reaction id>", "entityId": "<its entity id>"}` – **كلاهما** مطلوب للمطابقة، ويجب أن يكون `triggerId` تفاعلًا مُعرَّفًا على `entityId`. في 1.10 يُخزَّن التفاعل الذي ينقصه أحدهما ولا يُطلَق أبدًا؛ و `Since dev` يُرفض الحفظ (انظر أدناه). واضبط `spaceScope` على **مساحة التشغيل** وحدها، انظر أدناه
`100` SlaTimeExceeded (تفاعل `type: 1`)	`[{"occurrencesConfig": {"startingAfter": 0, "repeatAfter": 0, "repetitions": 0}, "steps": [{"workflowId": "…", "stepId": "…"}]}]`

اجعل نطاق التفاعل المتسلسل مساحة التشغيل وحدها. يُشغَّل تفاعل `ReactionSucceeded` / `ReactionFailed` على السياق والمدخل المخزَّنين من التشغيل الأصلي، وهما يخصّان المساحة التي جرى فيها ذلك التشغيل. فاضبط له `"spaceScope": {"rootSpace": false, "currentSpace": true, "otherSpaces": false}`.

- **في 1.10 لا تفرض المنصّة ذلك.** أيّ نطاق أوسع يُوسَّع كما في التفاعلات الأخرى، فيُشغَّل التفاعل المتسلسل مرة لكل مساحة محلولة – وفي كل مرة بمدخل المساحة الأصلية. ومع *النطاق الأساسي ونطاق التشغيل*، يُطلَق التفاعل المتسلسل مرّتين إذا بدأ التشغيل في مساحة غير الجذر؛ ومع *كل النطاقات* يُشغَّل في كل مساحة.
- **و `Since dev` صار مفروضًا.** يُشغَّل التفاعل المتسلسل مرة واحدة، في المساحة التي جرى فيها التشغيل المراقَب، أيًّا كان النطاق المخزَّن (لا تحتاج التفاعلات القائمة إلى تغيير)، ويُرفض حفظه بأيّ `spaceScope` آخر بـ `422` يسمّي `spaceScope`؛ ويُقبل غياب `spaceScope`.

فحوص أخرى للتفاعلات المتسلسلة عند الحفظ `Since dev`:

الفحص	النتيجة
غياب `eventConfig.entityId` أو `eventConfig.triggerId`	`400`، يسمّي الغائب منهما
ليس `triggerId` تفاعلًا موجودًا، أو ليس مُعرَّفًا على `entityId`	`422`، يسمّي `eventConfig.triggerId`
فشل قراءة التفاعل المراقَب نفسها (خطأ قاعدة بيانات)	يفشل الحفظ بذلك الخطأ، لا بوصفه `triggerId` غير صالح
عدّة تفاعلات في حفظ واحد مصدرها العلاقة نفسها	يُتحقَّق من كل منها (كان الأول وحده يُتحقَّق منه)
«تنفيذ إجراء سير العمل» تُطلب `actions` فيه من المستخدم، تليه إجراءات أو تفاعلات أخرى في الحفظ نفسه	يتخطّى ذلك الإجراء وحده التحقّق (وكان بقية الحفظ يتخطّاه أيضًا)

و`Since dev` أيضًا: حين تراقب تفاعلاتٌ على كيانين التفاعلَ نفسه، لا يشغّل كل كيان إلا التفاعلات المتسلسلة التي **يعرّفها هو**؛ وكان كل منها يُشغَّل قبلًا مرّتين، مرة مع الكيان الآخر.

في `SlaTimeExceeded` تُقاس `startingAfter` و`repeatAfter` بالثواني، وتعدّ `repetitions` مرّات الحدوث **الإضافية** بعد الأولى: `0` يُطلَق مرة واحدة؛ و`1` مع `repeatAfter: 0` يُطلَق مرّتين في اللحظة نفسها. وفي الإصدارات التي لا تتضمّن إصلاح المُجدوِل لا يُسلَّم الحدوث المستحَق أصلًا لحظة جدولته (`startingAfter: 0`) أبدًا – فاضبط `startingAfter` على `120` هناك.

مسار التنفيذ

في الحدث العادي (غير المؤقّت)، يجري التنفيذ على النحو الآتي:

1. يستقبل **معالِج حدث الناقل** (`handlers/events/**`، الذي يمدّد `BaseTriggerEventHandler`) حدثًا من المجال، ويستخرج `DataEntity` المعرِّف من `data-service`، ويحمّل عمليات التشغيل المرشّحة: `dataReactionRepository.find({ 'definingEntity.dataEntry': entityId, disabled: false })`.
2. يحدّد **عمليات التشغيل المباشرة** (`sourceEntity.type !== Relation`) و**عمليات التشغيل المرتبطة** (انظر تفاعلات الكيانات المرتبطة)، ويتخطّى عمليات التشغيل غير الصالحة (`validations.isValid === false`).
3. تُهيّأ المُدخلات المتأثّرة مرة واحدة بوصفها مصادر للاستعلام في الذاكرة، يتشاركها كل مشغّل قابل للتشغيل (وكانت مجموعة مؤقّتة قبل 1.10 – انظر مرشّح المعايير والاستعلام)؛ ثم يستدعي المعالِج `EventsHandler.handle(...)` لكل تشغيل ولكل مُدخل. ويُوسَّع `spaceScope` عبر `SpaceService.getDataReactionSpaceScope` ليُتاح تشغيل التفاعل عبر المساحات المهيّأة. وتُتخطّى المُدخلات التي تختلف قيمة `store` لديها عن مخزن الكتابة للكيان.
4. يتحقّق `EventsHandler.handle` من مطابقة نوع الحدث، ويشغّل مطابِق الحدث، ويدمج أيّ `query` في `criteria`، ويبني `TriggerContext`، ويطبّق مرشّح المعايير، ثم ينفّذ `trigger.actions` بالترتيب.
5. يمرّ كل إجراء عبر `EventsHandler.actionHandler`، الذي يطبّق الحماية من الحلقات، ويسجّل مقاييس Prometheus (`osos_reaction_duration_seconds` و`osos_reaction_executed_total` – `Since 1.8`)، ويستدعي المعالِج.
6. تُجمَّع النتائج في سجلّ `TriggerHistory` (مجموعة `ActionResult` لكل إجراء)، ويُخزَّن السجلّ عبر `TriggerHistoryRepository`. ويُخزَّن السياق المُتسلسَل الكامل مؤقّتًا في Redis تحت `trigger:<triggerId>:context:<historyId>` لمدة **600 ثانية (10 دقائق)** – و`Since 1.10` لا يُخزَّن إلا عند وجود تفاعل يستمع إلى نتيجة هذا التفاعل، ويُخزَّن بصيغة مضغوطة؛ انظر تخزين سياق التشغيل – ويُنشَر `TriggerExecutedEvent` على الناقل.

تُسجَّل الحالة الإجمالية للتشغيل على أنها ناجحة كلّيًا، أو ناجحة جزئيًا، أو فاشلة، استنادًا إلى عدد نتائج `ActionResult` الفاشلة.

سياق التشغيل

يمثّل `TriggerContext` (`handlers/trigger.context.ts`) الحالة المشتركة التي تُمرَّر إلى كل إجراء. ويحتفظ بـ `entity` / `entry` الذي أطلق الحدث، وبجميع `entities` / `entries` (بما في ذلك كيانات AskUser وكيانات المصدر المرتبطة)، وبخريطة `actions` التي تضمّ `input` / `output` لكل إجراء. ومن أبرز سلوكياته:

- **حلّ المسار** – يحلّ `getValueByPath` / `getFieldByPath` المسارات المنقّطة، ويطبّع المسارات القديمة (`value.name` ، أو اسم حقل مجرّد) إلى الصيغة الأساسية `scope.entries.<entity>.value.<field>` ، ويدعم **اجتياز العلاقات** باستخدام العلامة `.:relation.` / `.:relation_<entity>.` للانتقال عبر حقول المراجع.
- **تسلسل الإجراءات ضمن التشغيل الواحد** – بعد نجاح إجراء، يُخزَّن مُخرَجه عبر `setActionOutput` ، مما يتيح لإجراء لاحق في التفاعل نفسه قراءة مُخرَج الإجراء السابق (مثل معرّف مُدخل أُنشئ).
- **هوية المنفّذ** – يضبط `calculateIsAdminModification` القيمة `isAdminModification` عندما تكون قيمة `executorType` هي `SYSTEM` / `MANAGER` (أو تكون غائبة، للتوافق مع الإصدارات السابقة). وفي حالة `SYSTEM` ، يُضبَط معرّف المستخدم على معرّف مستخدم النظام ويُمحى المفوَّض إليه، لتُنفَّذ عمليات الكتابة بامتيازات النظام.
- `triggerContextId` – يحدّد هذا الخيار في إعداد الحدث (`BaseEventConfig.triggerContextId`) كيانًا معيّنًا للسياق لتعمل عليه الإجراءات بدلًا من الكيان الذي أطلق الحدث. *(في الإصدار 1.8، انتقل هذا الخيار من خيارات الإجراء الفردية إلى إعداد الحدث – فالسياق يتبع الحدث، لا الإجراء.)* **Since 1.8.**

مساحات التنفيذ (المساحات)

يتحكّم `spaceScope` الخاص بالتفاعل في المساحات التي يُنفَّذ فيها، ويُوسَّع عند الإرسال بواسطة `SpaceService.getDataReactionSpaceScope` . وفي الباني، تتاح ضمن **مساحات التنفيذ** الخيارات **المساحة الأساسية**، و**المساحة المسببة للتفاعل**، و**المساحة الأساسية و المساحة المسببة للتفاعل**، و**كل المساحات ماعدا المساحة الأساسية**، و**جميع المساحات** (تراكيب `rootSpace` / `currentSpace` / `otherSpaces`). ويتيح ذلك لتفاعل واحد توزيع إجراءاته عبر المساحات المهيّأة، بدلًا من اقتصارها على المساحة التي أُطلق فيها الحدث. ويُستثنى من ذلك التفاعل المتسلسل عند نجاح تفاعل آخر أو إخفاقه: اجعل نطاقه المساحة المسبِّبة وحدها؛ ففي 1.10 يؤدّي النطاق الأوسع إلى تشغيله مرة لكل مساحة على مُدخل المساحة الأصلية، و `Since dev` يُتجاهل هذا النطاق وقت التشغيل ويُرفض عند الحفظ (انظر مرجع خيارات الإجراءات › أشكال إعداد الحدث).

ما الذي تتوقّعه من نموذج التنفيذ `Since 1.10`

`Since 1.10` لم يعد المحرّك يُبقي عاملًا (worker) مشغولًا بانتظار أن تُنهي `data-service` إجراءً؛ إذ يُسجَّل تقدّم التشغيل أولًا بأول، ثم يُستأنف من حيث توقّف. ولا تتطلّب هذه الآلية تهيئة منك، لكن لها ثلاث نتائج تؤثّر في تصميمك.

- **لا تعتمد على الترتيب بين المدخلات في دفعة واحدة.** فعندما يُطبَّق إجراء واحد على عدة مُدخلات، أصبح من الممكن إرسال أوامره **معًا**، وإكمالها **بأي ترتيب**. أما الترتيب *بين الإجراءات* داخل التفاعل فلم يتغيّر – فالإجراء الثاني ما زال يعمل بعد الأول – لكن لم يكن هناك أي ضمان بأن «يُكتب المُدخل أ قبل المُدخل ب»، وأصبح عدم الالتزام بهذا الترتيب واضحًا الآن. وإذا استهدف أمران في الدفعة نفسها **المدخل ذاته**، فإنهما يُرفضان قبل الإرسال بدلًا من تنفيذهما على نحو متسابق.
- **يصمد التشغيل أمام إعادة تشغيل الخدمة ويسجّل سجلّه – لكنه لا يُكمل.** فإذا أُعيد تشغيل خدمة الأتمتة في أثناء تشغيلٍ ما (بسبب نشر أو انهيار)، استُردّ ذلك التشغيل وكُتب صفّه في السجل التاريخي للتنفيذ للإجراءات التي اكتملت، بدلًا من ضياع أثره. غير أنّ الاسترداد **يكتب الذيل ثم يتوقّف**: فالإجراءات التي لم يصل إليها التنفيذ **لا تعمل**، لأن إعادة

تنفيذ إجراءات التفاعلات دون التحقّق من آثارها ليست آمنة. لذا صمّم الإجراءات المتأخّرة بحيث تتعامل مع احتمال عدم تنفيذ إجراء سابق أصلًا – وهو الاحتمال نفسه الذي يفرضه إخفاق في منتصف السلسلة.

- **وقد يُطلَق مشغّل `ReactionSucceeded` / `ReactionFailed` مرتين لتشغيل واحد مُستردّ.** فالاسترداد يعيد نشر حدث «انتهى هذا التفاعل»، ولذلك قد يُطلَق تفاعل مُتسلسِل عند نجاح تفاعل آخر أو إخفاقه أكثر من مرة للتشغيل المنطقي نفسه. لذا اجعل تشغيل التفاعل المتسلسل مرتين آمنًا – بأن يتحقّق من المُدخل الذي سيُنشئه، أو بأن تصمّم عملية الكتابة فيه بحيث لا يترتّب على تكرارها أثر إضافي – بدلًا من افتراض إطلاقه مرة واحدة بالضبط.

ولم يتغيّر شيء في **طريقة تهيئتك** للتفاعل: الأحداث نفسها، وخيارات الإجراءات نفسها، ووثيقة السجل التاريخي للتنفيذ نفسها، وأسماء المقاييس نفسها.

تسلسل التفاعلات

تترابط التفاعلات عبر الأحداث بدلًا من الاستدعاءات المباشرة:

- ينشر كلّ تنفيذ مكتمل `TriggerExecutedEvent`. وتُطابَق التفاعلات المرتبطة بـ `ReactionSucceeded` **(8)** / `ReactionFailed` **(10)** بواسطة `ReactionSucceededEventConfig` / `ReactionFailedEventConfig` (`triggerId` + `entityId`)، مما يتيح ربطها وفق المنطق الآتي: «عندما ينجح التفاعل X على الكيان Y، افعل Z» (أو ربط مسار لمعالجة الفشل). ويستطيع التفاعل اللاحق قراءة مُدخلات التفاعل المُطلِق ومُخرَجاته ضمن إعدادات إجراءاته.
- ينشر إجراء `AnnounceEvent` **(105)** إعلانًا مُسمّى؛ وتُشغَّل التفاعلات المرتبطة بـ `EventAnnounced` **(11)** (التي تُطابَق باستخدام `entityId` + `announcementId`) استجابةً له. وينقل `schemaMapping` بيانات من السياق المُعلِن إلى مخطّط الإعلان، الذي تستطيع التفاعلات اللاحقة قراءته. ويفصل ذلك الناشر عن المشتركين، حتى عبر كيانات مختلفة. **Since 1.8.**
- تُصدِر إجراءات تعديل البيانات (إنشاء المُدخلات/تحديثها/أرشفتها) أحداث `DataEntryCreated` / `Updated` / `Archived` الاعتيادية، التي تستطيع بدورها إطلاق تفاعلات أخرى – وهذا أكثر مسارات التسلسل شيوعًا (وأسهلها تسبّبًا في حلقة).

يُحفَظ سياق التفاعل المُطلِق لمدة محدودة (انظر خبء Redis أعلاه) ليتمكّن التفاعل المتسلسل من إعادة استخدامه.

يجب أن يحدّد تفاعل `ReactionSucceeded` / `ReactionFailed` **كلًّا من** التفاعل المراقَب (`triggerId`) والكيان الذي يعرّفه (`entityId`)، وأن يقتصر نطاقه على **المساحة المسببة للتفاعل**. وفي 1.10 لا يُتحقَّق من أيّ من القاعدتين: يُخزَّن التفاعل الذي يفتقر إلى أحد المعرّفين ولا يُطلَق أبدًا، ويؤدّي النطاق الأوسع إلى تشغيل التفاعل المتسلسل مرة لكل مساحة. و `Since dev` تُرفض المخالفتان عند الحفظ، ويُشغَّل التفاعل المتسلسل دائمًا مرة واحدة في المساحة التي جرى فيها التشغيل المراقَب. وعندما تراقب تفاعلات على كيانين التفاعلَ نفسه، لا يشغّل كل كيان إلا تفاعلاته – انظر مرجع خيارات الإجراءات › أشكال إعداد الحدث.

ولا ينشر التشغيل الذي يرفضه حدّ التشغيل الشهري للقاطن حدث «انتهى هذا التفاعل»، ولذلك **لا** يُطلَق تفاعل `ReactionFailed` عند الرفض `Since 1.10`.

تخزين سياق التشغيل `Since 1.10`

يقرأ التفاعل المتسلسل من نوع `ReactionSucceeded` / `ReactionFailed` سياق التشغيل الأصلي بصيغته المتسلسلة – وكثيرًا ما يبلغ حجمه بضع مئات من الكيلوبايتات – ويُحفَظ هذا السياق عشر دقائق. ومنذ 1.10 لا يكتبه المحرّك **إلا حين يوجد تفاعل مفعَّل يستمع إلى نتيجة ذلك التفاعل**، ويخزّنه مضغوطًا باستخدام gzip. وإذا تعذّر على المحرّك تحديد ما إذا كان هناك مستمع، خزّن السياق كما كان يفعل سابقًا. ويترتّب على ذلك نتيجتان تؤثّران في التصميم:

- يجب أن يوجد المستمع **وأن يكون مفعَّلًا قبل أن ينتهي التشغيل الأصل**. فتفاعل `ReactionSucceeded` / `ReactionFailed` الذي يُنشأ أو يُعاد تفعيله بعد ذلك لا يجد سياقًا للتشغيلات المنتهية – تمامًا كما لو انقضت مهلة الدقائق العشر.
- يُعامَل السياق المخزَّن الذي يتعذّر فكّ ترميزه على أنه *لا سياق*: فيُتخطّى التفاعل المتسلسل بدلًا من تشغيله على حقول مفقودة.

وتُحفَظ القيم التي يحجبها السجل التاريخي وفق الشرط نفسه، ليحجبها التفاعل المتسلسل على نحو متّسق مع التفاعل الأصلي. ويستطيع القائمون على التشغيل تعطيل السلوكين باستخدام `OSOS_TRIGGER_CONTEXT_LISTENER_GATE` (للعودة إلى تخزين سياق كل تشغيل) و`OSOS_TRIGGER_CONTEXT_GZIP` (للتخزين دون ضغط)، وكلاهما مفعَّل افتراضيًا. ولا تستطيع خدمة تفاعلات أقدم من 1.10 قراءة الصيغة المضغوطة، ولذلك يبقى الضغط معطّلًا أثناء الترقية المتدرّجة حتى تعمل جميع حاويات reaction-service بالإصدار 1.10.

الحماية من الحلقات

لأنّ إجراءات البيانات تُعيد إصدار أحداث البيانات، يمكن للتفاعل أن يُعيد إطلاق نفسه. ويتولّى `EventsHandler.actionHandler` الحماية من ذلك: إذ يعدّ صفوف `TriggerHistory` الموجودة للمجموعة نفسها من `traceId` + `spaceId` + `triggerId` النشط + `action.normalizedName`، وإذا تجاوز العدد **`trigger.loopThreshold`** أوقف ذلك الإجراء بنتيجة `ActionResult` مفادها "preventing a loop" (`success:false`).

القيمة الافتراضية لـ `loopThreshold` هي `0`، مما يعني أنّ **الإجراء يُشغَّل مرة واحدة لكل أثر (trace)**. ارفعها عن قصد للسماح بعدد محدود من عمليات إعادة التنفيذ الذاتية. ويُمرَّر معرّف الترابط/الأثر عبر سلسلة التنفيذ (وتُضاف إليه لاحقة لكل مُدخل عندما يؤثّر حدث واحد في عدّة مُدخلات)، ليربط أجزاء السلسلة بعضها ببعض لأغراض هذا العدّ. وإذا لم يُنفَّذ إجراء متوقَّع دون سبب ظاهر، فتحقّق مما إذا كان حارس الحلقات قد تخطّاه.

أحداث خطوات سير العمل تُعدّ لكل خطوة `Since 1.4`. ففي تفاعلات `WorkflowStepReached` و`WorkflowStepExited` يُجرى العدّ لكل خطوة من خطوات سير العمل (إذ يسجّل كل صفّ في السجل التاريخي الخطوة المعنيّة): فمن المشروع أن يُشغَّل هذا التفاعل مرة لكل خطوة يمرّ بها الأثر، أمّا الحلقة الفعلية فلا بدّ أن تعود إلى خطوة سابقة، وبذلك تظلّ خاضعة للحارس. أمّا أنواع الأحداث الأخرى فيُجرى عدّها على مستوى الأثر كله كما في السابق.

تفاعلات الكيانات المرتبطة

يُطلَق التفاعل الذي يُضبَط له `sourceEntity` من كيان **مختلف** عن `definingEntity` الخاص به. ويضمّ `DataReactionSourceEntity` الحقل `type` (`TriggerSourceEntityTypeEnum` : `Self=0`، `Relation=1`، `AbsoluteEntity=2`) والحقل `sourceId`. ويحلّ `BaseTriggerEventHandler.findRelatedTriggersUsedAsSource` هذه الارتباطات في اتّجاهين:

- **العلاقات الواردة** – بالنسبة إلى الكيان الذي أطلق الحدث، ابحث عن العلاقات التي تستهدفه، ثم ابحث عن التفاعلات التي تطابق فيها قيمة `sourceEntity.sourceId` معرّف علاقة؛ واجلب المُدخلات المُشار إليها وشغّل إجراءات كل تفاعل عليها، مع إتاحة مُدخل/كيان المصدر في السياق.
- **العلاقات الصادرة** – ابحث عن التفاعلات التي تطابق فيها قيمة `sourceEntity.sourceId` التعبير النمطي : `<entityId>` (علاقة في الذاكرة تشير إلى هذا الكيان)، وحدّد المُدخلات في الكيان المعرِّف التي يشير حقل المرجع فيها إلى المُدخل المُطلِق، ثم شغّل التفاعل لكل منها. *(مثال في الرماز: يشير كيان `User` إلى `OrgUnit`؛ وعندما يُطلِق `OrgUnit` الحدث، يُشغَّل تفاعل `User` لكل مستخدم في تلك الوحدة التنظيمية.)*

أمّا مخطّط التفاعل الخاص بواجهة الباني (أي الكيانات/الحقول التي يمكن الوصول إليها في سياق التشغيل)، فيوفّره `SystemReactionController` (انظر أدناه).

عزل الكيانات `Since 1.8`

يُشغَّل التفاعل **بوصفه كيانه الخاص**: إذ تحلّ إجراءاته مسارات الحقول وتكتب البيانات وفق تعريف ذلك الكيان. ومنذ Since 1.8 يفرض المحرّك هذا صراحةً – **فلن يُشغَّل تفاعل على مدخل ينتمي إلى كيان مختلف**.

- المُدخل الذي يختلف كيانه عن كيان التفاعل نفسه **يُتخطّى** مع تسجيل تحذير في سجلّ الخدمة، بدلًا من تشغيل التفاعل عليه وفق تعريفات حقول كيان آخر.
- في تفاعلات الكيانات المرتبطة، تُرشَّح المُدخلات المسترجعة عبر علاقة **لكلّ مشغّل على حدة**، لتقتصر على المُدخلات التابعة لكيان المشغّل نفسه *قبل* جلبها. وتبرز أهمية ذلك في حالة **حقل مرجع يقبل أكثر من كيان**: إذ كان كلّ تفاعل يشترك في تلك العلاقة يتلقّى سابقًا كلّ مُدخل مُشار إليه، بصرف النظر عن الكيان الذي ينتمي إليه فعلًا.
- والمشغّل الذي يتعذّر حلّ كيانه المعرِّف – كأن يكون الكيان محذوفًا – **يُتخطّى مع تحذير** بدلًا من إفشال العملية بأكملها، فتستمرّ بقية عمليات المشغّلات المستمعة للحدث نفسه في التنفيذ.

ولا يُرمى أيّ خطأ جديد لأيٍّ من هذا. فالمُدخل التابع لكيان آخر أو الكيان الذي يتعذّر حلّه يؤدّي إلى تخطٍّ مُسجَّل، لا إلى إخفاق؛ ولذلك لا يظهر شيء في نتائج التنفيذ والسجل التاريخي، إذ لا يُشغَّل التفاعل لذلك المُدخل أصلًا. فإذا بدا أنّ تفاعلًا لم يُطلَق لبعض المُدخلات، فابحث عن تحذيرات التخطّي في سجلّ reaction-service بدلًا من البحث عن نتيجة إجراء فاشلة.

واجهة برمجة REST

تتطلّب جميع النقاط الطرفية للتفاعلات استراتيجية المصادقة `Auth2`، ويُحدَّد نطاقها بمعرّف `spaceId` المحقون.

`DataReactionController` – عمليات CRUD

المصدر: `controllers/data-reaction.controller.ts`.

الطريقة والمسار	الغرض
`POST /v1/reaction/{entity}`	إنشاء تفاعل على `{entity}`. يستثني جسم الطلب `updateIdentifier` و`creationInfo` و`updateInfo` (يضبط الخادم `definingEntity` من المسار). ويُرفَض الطلب بالاستثناء `ImplicitReactionCreateException` إذا كان سيُنشئ تفاعلًا ضمنيًا.
`GET /v1/reaction/{entity}`	سرد تفاعلات كيان (`filter` بأسلوب LoopBack)، ويتضمّن كلّ صفّ أيضًا عدّادَي التنفيذ/الفشل الخاصَّين به – انظر عدادات التنفيذ/الفشل والسجل التاريخي لعمليات التشغيل. `Since 1.10`
`GET /v1/reaction/{entity}/count`	عدّ تفاعلات كيان (`where` بأسلوب LoopBack).
`GET /v1/reaction/{entity}/{id}`	جلب تفاعل واحد، مع التحقّق من انتمائه إلى `{entity}`.
`GET /v1/reaction/{entity}/{id}/history`	جلب صفحة واحدة من عمليات التشغيل المُسجَّلة لتفاعل، الأحدث أولًا، بصفوف مختصرة – **لمسؤولي النظام فقط** – انظر عدادات التنفيذ/الفشل والسجل التاريخي لعمليات التشغيل. `Since 1.10`
`PATCH /v1/reaction/{entity}/{id}`	تحديث. **التزامن التفاؤلي**: يجب أن تساوي قيمة `updateIdentifier` في جسم الطلب القيمة المُخزَّنة، وإلا أُطلِق الاستثناء `UpdateIdentifierConflictException`؛ ثم يستبدل الخادم قيمة `updateIdentifier` بمعرّف UUID جديد.

الطريقة والمسار	الغرض
`DELETE /v1/reaction/{entity}/{id}`	حذف؛ ويُمنَع للتفاعلات الضمنية بالاستثناء `ImplicitReactionDeleteException`.
`GET /v1/reactions`	سرد جميع التفاعلات عبر الكيانات.
`GET /v1/reactions/module/{moduleId}`	سرد التفاعلات المُعرَّفة على أيّ كيان ضمن وحدة، مع إمكانية تضييق النتائج بمُرشِّح `filter` بأسلوب LoopBack. `Since 1.8`
`GET /v1/reactions/custom-actions?moduleId={moduleId}`	النقطة الطرفية لـ**اكتشاف الإجراءات القابلة للاستدعاء** التي يستند إليها مُنتقي تنفيذ إجراء مخصّص: تسرد إجراءات الوحدة المخصّصة القابلة للاستدعاء من مستوى **Local/External**، مُجمَّعة ضمن `external` / `local`. `moduleId` إلزامي. `Since 1.8` – انظر تنفيذ إجراء مخصّص.
`GET /v1/reactions/{id}`	جلب أيّ تفاعل بمعرّفه.
`GET /v1/reactions/count`	عدّ جميع التفاعلات.

إذا لم يكن الكيان موجودًا، تُعاد `NotFoundException('ENTITY_NOT_FOUND')`؛ وإذا لم ينتمِ التفاعل إلى الكيان المحدَّد في المسار، تُعاد `REACTION_NOT_FOUND`. وتمرّ عمليات الكتابة (إنشاء/تحديث/حذف) عبر `ReactionWriterService`، ولا تمرّ مباشرةً عبر المستودع.

من يحقّ له تأليف تفاعل `Since 1.8`

تتطلّب كلّ عملية كتابة على التفاعل (إنشاء، تحديث، حذف) شرطين: أن تشمل رخصة القاطن الإدارة، وأن **تملك أنت وصولًا بمستوى مطوّر فما فوق على كيان التفاعل**، وهو المستوى الذي يتيح لك تحرير الكيان.

- **الرفض:** `403 REACTION_AUTHORING_FORBIDDEN`، مع `details: {entityId}`. وتُرفَض الكتابة أيضًا إذا تعذّرت قراءة مستوى وصولك.
- **الاستيراد يتبع القاعدة نفسها.** يُفحَص إذن الشخص الذي يؤكّد الاستيراد لتأليف كلّ تفاعل في الملف المستورَد؛ ويُتخطّى أيّ تفاعل لا يحقّ له تأليفه، ثم يُستكمَل باقي الاستيراد. وعند التحديث، يستند التحقّق إلى الكيان الذي ينتمي إليه التفاعل المخزَّن، لا إلى الكيان الوارد في الطلب.
- **التفويض:** يُقيَّم التفاعل الذي يُنشئه مفوَّض، بما في ذلك عبر الاستيراد، وفق الأذونات المفوَّضة.
- **لا يُفحص** ما تكتبه المنصة نفسها: الاستدعاءات بين الخدمات، والتفاعلات الضمنية، والترحيلات، وتجهيز القاطن، وتثبيت الحزم.

الملفات في خيارات التفاعل `Since 1.10`

يحتفظ التفاعل الذي تتضمّن خياراته معرّفات ملفات (مرفق بريد ثابت مثلًا) الآن بمرجع تخزين إليها، كما يفعل المُدخل: يصبح الملف المرفوع مؤقّتًا دائمًا عند حفظ التفاعل، ويُحرَّر المرجع عند حذف التفاعل (أو كيانه). ويترتّب على ذلك خطآن عند الحفظ:

الخطأ	متى	ما العمل
`400 FILE_EXPIRED`	يشير طلب الحفظ إلى ملف مرفوع مؤقّتًا انتهت صلاحيته قبل أن يجعله الحفظ دائمًا. وتسرد `details` `{fileId, path, reason}` لكلّ ملف – وتكون `path` مثل `actions[0].options.attachments[1].value`، وتكون `reason` رسالة مترجَمة «ارفعه مرة أخرى».	ارفع الملف مجددًا واحفظ.
`503 STORAGE_SERVICE_UNAVAILABLE`	يتعذّر الوصول إلى خدمة تخزين الملفات بينما الحفظ **يضيف** ملفًا. أمّا الحفظ الذي لا يضيف ملفًا جديدًا فيُستكمَل.	أعد المحاولة.

ولا تترك عملية الحفظ المرفوضة على هذا النحو أيّ أثر – فلا يُجدوَل لها مؤقِّت.

عدادات التنفيذ/الفشل والسجل التاريخي لعمليات التشغيل `Since 1.10`

يعرض تبويب التفاعلات، لكلّ تفاعل، عدد التشغيلات الناجحة وعدد التشغيلات الفاشلة **منذ آخر تعديل على التفاعل**. وعند استخدام مسؤول النظام لأيّ من العدّادين، تُفتَح لوحة جانبية تسرد التشغيلات الفردية التي يحتسبها؛ أمّا لغير مسؤولي النظام، فيظهر العدّادان كنصّ عادي.

يُعيد `GET /v1/reaction/{entity}` جميع الحقول الاعتيادية في `DataReaction`، إضافةً إلى خمسة حقول محسوبة للقراءة فقط:

الحقل	النوع	المعنى
`executedCount`	number	عدد جميع التشغيلات المُسجَّلة لهذا التفاعل منذ البداية، أيًّا كانت نتائجها – باستثناء التشغيلات التي رفضها حدّ التشغيل الشهري.
`succeededCount`	number	عدد التشغيلات التي نُفِّذ فيها إجراء واحد على الأقلّ ولم يفشل أيّ إجراء، منذ البداية – فلا يُحتسَب هنا التشغيل الذي فشل جزئيًا.
`failedSinceLastChangeCount`	number	عدد التشغيلات الفاشلة المُسجَّلة منذ آخر تعديل على التفاعل (`statsSince` أدناه). يُحتسَب التشغيل مرة واحدة، أيًّا كان عدد إجراءاته الفاشلة.
`statsSince`	string (ISO-8601)	بداية الفترة التي يُحتسَب خلالها الفشل – الطابع الزمني لآخر تعديل على التفاعل (أو الطابع الزمني لإنشائه إذا لم يُعدَّل منذ إنشائه).
`refusedCount`	number	عدد التشغيلات التي رفضها حدّ التشغيل الشهري منذ البداية؛ تُسجَّل، لكن لا يُنفَّذ فيها أيّ إجراء، ولا تدخل ضمن `executedCount`. `Since 1.10`

تُحسَب هذه العدّادات مباشرةً عند كلّ قراءة بدلًا من تخزينها، ولذلك تظلّ مطابقةً للسجل التاريخي للتشغيل الفعلي دائمًا.

يُعيد `GET /v1/reaction/{entity}/{id}/history` عمليات التشغيل التي يحتسبها أحد العدّادين في صفحات، الأحدث أولًا، ويقبل المعاملات التالية:

المعامِل	المعنى
`outcome`	`failed`، أو `succeeded`، أو `refused` (`Since 1.10`)، أو `all` (الافتراضي). وتؤدّي أيّ قيمة أخرى إلى خطأ تحقّق.
`from` / `to`	حدّ أدنى شامل / حدّ أعلى غير شامل للطابع الزمني للتشغيل (ISO-8601). لا تُقيَّد الفترة تلقائيًا بقيمة `statsSince` – يملأ مربّع الحوار `from` مسبقًا بهذه القيمة عند اختيار نتيجة الفشل، لكن يمكن للمستدعي توسيع الفترة عمدًا.
`before`	مؤشّر ترقيم صفحات مأخوذ من `nextBefore` في الصفحة السابقة.
`limit`	حجم الصفحة؛ الافتراضي 10، والحدّ الأقصى 100.

لا تتضمّن الاستجابة (`items[]`، `hasMore`، `nextBefore?`) عددًا إجماليًا، وذلك مقصود لأنّ شارة التفاعل تعرض الإجمالي أصلًا. وتقتصر النقطة الطرفية على **مسؤولي النظام** (وصول System Admin دون منح `None`)، ويُختصَر كلّ عنصر عمدًا إلى `{id, timestamp, status, relatedTo}`؛ وتكون قيمة `status` هي `succeeded` أو `failed` أو `none` وفق قاعدة الاحتساب نفسها التي تستخدمها العدّادات، أو `refused` لتشغيل أوقفه حدّ التشغيل الشهري، بينما يتضمّن `relatedTo` المُدخل وعنوانه بعد حلّه. وتقتصر التشغيلات المعروضة على المساحة الحالية للمستدعي، في حين تشمل العدّادات كلّ المساحات. أمّا السجلّ الكامل المخزَّن لتشغيل – مدخلات كلّ إجراء ومخرجاته – فلا يقدّمه إلا مسار التفاصيل في واجهة سجل التفاعلات؛ ويحلّ هذا محلّ التصميم السابق، الذي كان فيه كلّ صفّ سجلًّا كاملًا يستطيع قراءته أيّ مستدعٍ يملك وصولًا إلى التفاعلات.

واجهة سجل التفاعلات على مستوى النظام `Since 1.10`

يُتيح `ReactionLogController` (`controllers/reaction-log.controller.ts`) الوصول إلى كلّ تشغيل تفاعل مُسجَّل في مساحة المستدعي، وهو الواجهة التي تستند إليها **الإدارة › المراقبة › سجل التفاعلات** واللوحة الجانبية في تبويب التفاعلات. ويقتصر المساران كلاهما على **لمسؤولي النظام فقط**، لأنّ مسار التفاصيل يعيد الحمولة المخزَّنة لكلّ إجراء.

الطريقة والمسار	الغرض
`GET /v1/reaction-logs`	جلب صفحة واحدة من التشغيلات، الأحدث أولًا. ويقبل `skip` (بأيّ عمق) و`limit` (الافتراضي 20 والحدّ الأقصى 100).
`GET /v1/reaction-logs/{id}`	جلب تشغيل واحد كاملًا، مع التشغيل السابق والتشغيل التالي له ضمن المرشّح نفسه.

ويقبل المساران المرشّحات نفسها: `reactionId`؛ و`eventName` (مفتاح من `EventTypeEnum`)؛ و`actionType` (مفتاح من `AutomatedActionTypeEnum` – يطابق التشغيلَ متى كان *أيّ* من إجراءاته من ذلك النوع)؛ و`status` = `failed` أو `succeeded` أو `refused` أو `all` (الافتراضي)؛ و`from` (شاملة) / `to` (غير شاملة)، وهما لحظتان بصيغة ISO-8601؛ و`doneBy` (معرّف مستخدم أو `SYSTEM`)؛ و`search`؛ و`order` = `desc` (الافتراضي) أو `asc`. ولا يميّز `search` بين حالات الأحرف في أيّ لغة من لغات القاطن، ويطابق الاسم الحالي للتفاعل، أو اسم كيانه أو وحدته، أو الاسم الكامل للفاعل، أو يطابق معرّف تشغيل أو مُدخل مطابقةً تامّة؛ ويُتجاهَل إذا قلّ طوله عن حرفين. وتؤدّي المعاملات غير السليمة إلى أخطاء تحقّق.

وتقرأ هذه الواجهة سجلّات `TriggerHistory` القائمة دون مخزن ثانٍ، ولذلك تتّفق صفحة السجل والعدّادات واللوحة الجانبية دائمًا على التشغيلات الفاشلة والناجحة. ويُحسَب الإجمالي ضمن مهلة ثلاث ثوانٍ، وتُعاد `null` إذا انقضت المهلة؛ أمّا `hasMore` فموثوق دائمًا. ويُعاد المُدخل الذي لا يملك صفّ فهرسة دون `_resolved`، مما يتيح للعميل التمييز بين *بلا عنوان* و*محذوف أو مجهول*.

حدّ التشغيل الشهري والتشغيلات المرفوضة `Since 1.10`

يمكن لخطة القاطن أن تضع حدًّا لعدد تشغيلات التفاعلات التي يبدأها في **الشهر الميلادي بتوقيت UTC** (`maxReactionRunsPerMonth`). ويُفرَض الحدّ داخل خدمة الأتمتة، فهو **ليس خطأ HTTP**: لا يفشل أيّ إجراء أجراه المستخدم، وإنما لا يبدأ التشغيل.

- **التشغيل المرفوض** يُسجَّل مع ذلك، دون نتائج إجراءات، ومع `refusal` قيمته `{code: "REACTION_RUN_CAP_EXCEEDED", limit, used, period}`. فلا يُنفَّذ أيّ إجراء، ولا يُنشر حدث «انتهى هذا التفاعل»، ولذلك لا يُشغَّل بسببه تفاعل `ReactionFailed`.
- **أين تراه.** تظهر حالته بالقيمة `refused` في صفوف السجل التاريخي للتشغيلات وفي واجهة سجل التفاعلات؛ ويمكن ترشيح النتائج باستخدام `outcome=refused` / `status=refused`؛ ويُعيد `GET /v1/reaction/{entity}` القيمة `refusedCount` بصورة منفصلة عن `executedCount`. ولا يعرض عميل الويب في 1.10 بعدُ عدّادًا أو مرشّحًا للتشغيلات المرفوضة.
- **ما يُحتسب.** كلّ تشغيل يبدأه حدث أو مؤقِّت أو إجراء مخصّص. **تفاعلات المنصة (الضمنية) لا تُحتسب أبدًا**، وتُحتسَب الرسالة المُعاد تسليمها مرة واحدة. ويرفض الحدّ `0` كلّ تشغيل. وإذا تعذّرت قراءة العدّاد، يُستكمَل التشغيل: فالحدّ مخصّص للقياس، لا للحماية.

الحدّ جزء من خطة القاطن. راجع حدود الخطة.

`SystemReactionController` – مخطّطات الباني

يستخدمه باني التفاعلات لمعرفة البيانات المتاحة في سياق التشغيل. المصدر: `controllers/system-reaction.controller.ts`.

الطريقة والمسار	الغرض
`GET /v1/schema/{entity}/{trigger}`	مخطّط التفاعل الكامل: مخطّط كيان الحدث، ومخطّطات كيانات كلّ إجراء، و`updateContextIds`، والكيانات المصدر.

الطريقة والمسار	الغرض
`GET /v1/schema/{entity}/event/{eventType}/{newTriggerId}`	مخطّط كيان الحدث لقيمة `eventType` معطاة.
`GET /v1/schema/{entity}/event/{sourceType}/{sourceId}/{newTriggerId}/source`	مخطّط الكيان المصدر لمصدر علاقة.
`GET /v1/schema/{entity}/announcement/{announcementId}/source/{sourceType}`	مخطّط مصدر الإعلان (+ علاقة اختيارية)؛ ومعاملا الاستعلام `newTriggerId` و `relationId` مطلوبان عندما تكون قيمة `sourceType` هي `Relation` .
`GET /v1/schema/announcement/{announcementId}`	مخطّط الإعلان.

`AnnouncementController` — الإعلانات

تدعم الإعلانات زوج التسلسل `AnnounceEvent` / `EventAnnounced` . المصدر: `controllers/announcement.controller.ts` .

الطريقة والمسار	الغرض
`POST /v1/announcements/{entityId}`	إنشاء إعلان لكيان.
`PATCH /v1/announcements/{entityId}/{announcementId}`	تحديث.
`GET /v1/announcements`	سرد جميع الإعلانات.
`GET /v1/announcements/{entityId}`	سرد إعلانات كيان.
`GET /v1/announcements/{entityId}/{announcementId}`	جلب إعلان واحد ضمن نطاق الكيان.
`GET /v1/announcements/by-id/{announcementId}`	جلب إعلان واحد بمعرّفه.

نتائج التنفيذ والسجل التاريخي

يُنتِج كل إجراء `ActionResult` (`models/data-entity-action/action-result.ts`) يتضمّن `action` و `actionType` و `actionName` و `success` ، والحقول الاختيارية `resultId` / `error` / `output` ، إلى جانب `traceId` و `spaceId` و `input` و `effectedEntities` . وتُطبَّع الأخطاء باستخدام `serializeOutputError` (ولا تُرفَق آثار المكدّس إلا عندما يكون `NODE_ENV === 'development'`). وتُحفَظ جميع نتائج التشغيل الواحد في مستند `TriggerHistory` يحمل `eventName` و `triggerId` و `triggerName` و `relatedTo` و `doneBy` و `delegatee` و `timestamp` ومصفوفة `result[]` ، وهو ما يشكّل مسار تدقيق الأتمتة.

حجب الأسرار في السجل التاريخي للتفاعلات `Since 1.6`

يخزّن السجل التاريخي `input` و `output` و `error` لكل إجراء، وقد تتضمّن هذه الحقول رموزًا مميّزة (tokens) وبيانات اعتماد لواجهات API. وتُحجب هذه القيم إلى `****` قبل كتابة الصفّ (وقد أُتيح الحجب أيضًا كتحديث لخطّي الإصدار 1.6 و1.8، بدءًا من 1.6):

- **القيم التي تعلّمها أنت.** يتضمّن كل سطر في تعيين الرموز خانة **محجوب في السجل التاريخي** (`redactInHistory` على مصدر الرمز)، وهي متاحة في إجراءات استدعاء API وتنفيذ وظيفة وإرسال البريد الإلكتروني وإرسال إشعار وإرسال رسالة نصية. وتُحجب القيمة الناتجة عن تقييم ذلك الرمز أينما ظهرت في السجل التاريخي للتشغيل.
- **القيم المقروءة من الحقول الحسّاسة** تُحجب حتى دون تحديد تلك العلامة.
- **الحقول الحسّاسة في لقطة المدخل المُطلِق.** في أحداث الإنشاء / التحديث / الأرشفة / الاستعادة، تُحجب الحقول الحسّاسة (بما فيها الحقول الفرعية للكتل) بحسب مسارها في قيمة المُدخل وفي فرق القيم، ويُحجب فهرس البحث

كاملًا.

- **مفاتيح الاعتماد المعروفة.** يشمل الحجب أيّ مفتاح يطابق، مع تجاهل حالة الأحرف و `-` و `_`، أحد الأسماء `authorization` أو `proxy-authorization` أو `cookie` أو `set-cookie` أو `x-api-key` أو `x-app-token`، أو يحتوي على `token` أو `secret` أو `password` أو `apikey`؛ إضافةً إلى بيانات اعتماد `Bearer` / `Basic` ومعلومات المستخدم في عناوين URL داخل النص الحرّ.
- **القيم المتعلّمة.** متى حُجبت قيمة، تُحجب أينما ظهرت مجددًا لاحقًا في التشغيل، بما في ذلك صيغها المرمّزة لعناوين URL والمُهرَّبة لـ JSON (حتى 200 قيمة في التشغيل؛ ولا تُحجب القيمة الأقصر من 4 أحرف إلا حين تطابق القيمة كاملةً).

وإذا فشل الحجب نفسه، يُخزَّن الجزء المتأثّر بصيغة `{redacted: "REDACTION_FAILED"}`، ولا يُخزَّن بصيغته الخام أبدًا. ويُطبَّق الحجب في التفاعل المتسلسل من نوع `ReactionFailed` / `ReactionSucceeded` باستخدام أسرار التشغيل الأصلي. والحجب مستقلّ عن سياسة الوصول إلى البيانات الحساسة: إذ تظلّ إجراءات التفاعل `Privileged` تتلقّى القيم الحقيقية، ويقتصر الحجب على السجل التاريخي. ولا تُعاد كتابة الصفوف المكتوبة قبل الترقية.

فهارس السجل التاريخي للتفاعلات والتنظيف `Since 1.10`

تُقلّل ثلاث ترحيلات في خدمة البيانات كلفة تصفّح السجل التاريخي وتزيل المخلّفات القديمة. يضيف `1.10.0-a19` ستة فهارس على `TriggerHistory` لدعم مرشّحات سجل التفاعلات، يبدأ كلّ منها بمفتاح الترشيح ثم يرتّب النتائج من الأحدث إلى الأقدم. ويضيف `1.10.0-a32` فهرسًا جزئيًا فريدًا على `DataEntryHistory.sourceMessageId`، بحيث لا تؤدّي إعادة تسليم رسالة إنشاء / تحديث مُدخل إلى كتابة صفّ في السجل التاريخي للمُدخل أكثر من مرة واحدة (ولا تحمل الصفوف المكتوبة قبل هذا الترحيل معرّف رسالة، لذا تقع خارج الفهرس). أمّا `1.10.0-a33` فيُسقط مجموعات `trigger_<10 letters>` اليتيمة التي كان تقييم المعايير قبل 1.10 قد يخلّفها، ولا يُسقط إلا المجموعات التي تطابق أسماؤها ذلك النمط تمامًا ولا يستخدمها أيّ كيان. ويُخزّن التشغيل الذي لم يُسجّل فيه أيّ إجراء `result: []`.

استكشاف أخطاء التفاعلات

أين يظهر فشل التفاعل فعلًا، وأين لا يظهر:

- **سجل التفاعلات** `Since 1.10`. يستطيع مسؤولو النظام تصفّح كل تشغيل عبر **الإدارة › المراقبة › سجل التفاعلات** (المدعومة بواجهة سجل التفاعلات) ومن خلال عدّادات تبويب التفاعلات، والاطّلاع على المدخلات والمخرجات والخطأ المخزَّنة لكل إجراء.
- **لا يظهر شيء للمستخدم الذي أطلقه.** لا يُنتج فشل التفاعل أيّ رسالة خطأ ولا استجابة API فاشلة للمستخدم الذي أدّى فعله إلى إطلاق التفاعل. وخارج سجل التفاعلات، تظهر معلومات التنفيذ للمستخدم في السجل التاريخي للمُدخل: إذ تُضمَّن عمليات تنفيذ التشغيل (مع `success` لكلّ إجراء) في `GET /v1/entities/{entity}/entries/{id}/history` ونظيرتها `/v2` عندما يكون المستدعي عارضَ نظامٍ يرسل الطلب دون معرض، **أو** عندما يكون الإعداد `entriesConfiguration.showTriggerHistory.enabled: true` مضبوطًا في معرض البيانات؛ ويتطلّب إظهار تفاصيل `error` / `input` / `output` لكلّ إجراء تفعيل `showTriggerHistory.showTriggerDetails` أيضًا. وكلا الإعدادين معطَّل افتراضيًا.
- **كلّ تنفيذ يُحفَظ.** سواء نجح التشغيل أم فشل، يُكتب لكلّ تشغيل مستند `TriggerHistory` (انظر أعلاه) يتضمّن أخطاء كلّ إجراء بصيغتها المتسلسلة؛ فتُحفَظ الإخفاقات فيه، ولا تُحفَظ على مستند `DataReaction` أو على قيمة المُدخل. وهذا هو المستند الذي تقرأه النقاط الطرفية للسجل التاريخي للمُدخل.
- **سجلّات الخدمة.** يُسجَّل عند فشل الإجراء `Execution action '<name>' has failed`، وعند فشل التشغيل بكامله `Execution trigger '<event>' on '<entry>' entry has failed`، في سجلّ `reaction-service`.

- **تشغيل لم يحدث قطّ** `Since 1.10` . حين يستنفد القاطن حصّته الشهرية من التشغيلات، تُسجَّل التشغيلات بالحالة `refused` ولا يُنفَّذ فيها أيّ إجراء؛ انظر حدّ التشغيل الشهري والتشغيلات المرفوضة.
- **تسلسل الفشل كلّي أو لا شيء.** يُنشَر `TriggerExecutedEvent` بعد كلّ تنفيذ، لكنّ التفاعل المرتبط بـ `ReactionFailed` لا يُطلَق إلا عندما تفشل **جميع** إجراءات التنفيذ؛ أمّا الفشل الجزئي فيؤدّي إلى إطلاق `ReactionSucceeded` بدلًا منه.
- **المقاييس** `Since 1.8` – يحصي Prometheus كلّ تنفيذ في `osos_reaction_executed_total` (بالوسوم `reaction_type` و `reaction_id` و `entity_id` و `space_id` و `status`) ويقيس مدّته في `osos_reaction_duration_seconds` .
- **ما لا يُسجَّل:** تقتصر خدمة التدقيق على تسجيل تغييرات **تعريف** التفاعل (إنشاء/تحديث/حذف)، ولا تسجّل عمليات التنفيذ أبدًا؛ أمّا عدم مطابقة المعايير فيؤدّي إلى تخطٍّ يُسجَّل على مستوى info **دون** إنشاء أيّ سجلّ في التاريخ. وفي المقابل، يؤدّي مسار قيمة **Dynamic** خاطئ في إجراء إلى فشل صريح *داخل السجل التاريخي* (`INVALID_ENTRY_FIELD` / `Path "…" does not exist in the context.`)، ولا يظهر ذلك الخطأ إلا في سجلّ `TriggerHistory` .

تصحيحات وقت التشغيل

تغيّر هذه التصحيحات سلوك التفاعل عند التشغيل. ويُوسَم ما أُتيح منها أيضًا كتحديث لخطوط إصدار منشورة بأقدم خطّ إصدار تلقّاه.

- **حارس الحلقات محصور في خطوة سير العمل** `Since 1.4` – انظر الحماية من الحلقات.
- **يمكن تشغيل إجراء خطوة الدخول في تدفق فرعي من مشغّل لاحق** `Since 1.4` . كانت كل خطوة دخول في تدفق فرعي تُتخطّى، فلم يكن بوسع أيّ تفاعل تشغيل الخطوة الأولى لتدفق فرعي. ولا يزال الانتقال الذي *يبدأ* التدفق الفرعي لا يدفع خطوة دخوله إلى الأمام تلقائيًا، لكنّ التشغيل الذي يطلقه حدث لاحق غير ذي صلة (عند التحديث، أو إجراء مخصّص، أو جدولة) صار يشغّلها، كما كانت خطوة الدخول في سير العمل الرئيسي تُشغَّل دائمًا.
- **يمكن تعيين بيانات سير العمل الوصفية** `Since 1.4` . حين يكون لكيان المشغّل سير عمل، يعرض مخطّط التفاعل `_metaCurrentSteps.*` و `_metaWorkflowHistory.*` و `_metaAdhocSteps.*` ؛ لذا لم يعد تعيين مثل *مقدّم الطلب* ← *مالك الخطوة الحالية* يفشل بالرمز `INVALID_ENTRY_FIELD` أو يُظهر منتقيًا فارغًا، ويشمل ذلك الإعدادات المحفوظة سابقًا.
- **إعادة إسناد خطوة إلى مستخدم واحد تُبقي مجموعة الخطوة** `Since 1.4` .
- **لم تعد إجراءات عناصر الكتلة تنجح صامتةً** `Since 1.4` (أُتيح أيضًا كتحديث للخطّ 1.6). فإذا لم يُفضِ حلّ الهدف في إجراء إنشاء / تعديل / حذف تكرار في كتلة متكررة في وضع المرجع أو العلاقة إلى أيّ مُدخل، صار الإجراء يفشل بالرمز `NO_ENTRIES_TO_UPDATE_FOUND` بدلًا من الإبلاغ عن نجاح بمُخرَج فارغ.
- **يقبل In / Not In مرجعًا أو تصنيفًا مفردًا** `Since 1.10` . كان المعيار مثل *المكلَّف* *In* `scope.me()` ، أو المعيار الذي تكون قيمته كائن مرجع أو تصنيف مفرد، لا يطابق شيئًا؛ وأصبحت القيمة تُعامَل بوصفها قائمةً من عنصر واحد، كما لم يعد العنصر الفارغ (null) في قائمة مراجع يتسبّب في طرح استثناء.
- **تُسجَّل إجراءات الرسائل النصية بالنوع** `SendSms` `Since 1.10` . كان كل إجراء رسالة نصية يُخزَّن ويُعرض في سجل التفاعلات بالنوع `SendNotification` ، فلم يكن المرشّح `actionType=SendSms` يطابق شيئًا. وتحتفظ التشغيلات المخزَّنة سابقًا بالنوع القديم.
- **الاستعلام المحفوظ المتعذّر حلّه يفشل احتياطًا** `Since 1.10` . كان التفاعل المقيَّد باستعلام محفوظ يعمل **دون أيّ مرشّح** حين يتعذّر تحميل الاستعلام أو لا تكون له معايير قابلة للتطبيق، فيتفرّع تفاعل المؤقّت عندئذ ليشمل جميع مُدخلات الكيان. والآن يؤدّي الاستعلام المحذوف (4xx غير قابل لإعادة المحاولة) أو الاستجابة الفاشلة أو الاستعلام الذي

لا يتضمّن معايير قابلة للتطبيق إلى **تخطّي** التفاعل (مع التسجيل بمستوى خطأ)، أمّا أيّ فشل آخر (الشبكة، 5xx، 408 / 429) فتُعاد المحاولة عند حدوثه.

ملاحظات الإصدار

تحمل الميزات التالية وسم إصدار معلَم:

- **إجراء `AnnounceEvent` + تشغيل `EventAnnounced` – Since 1.8.**
- **إجراء `DeleteArchivedEntry` – Since 1.8.**
- **تفاعلات العلاقات المرتبطة الصادرة – Since 1.8.**
- `triggerContextId` **على مستوى الحدث** (نُقِل من خيارات الإجراء إلى إعداد الحدث) **– Since 1.8.**
- **حدث التشغيل** `WorkflowStepOwnerChanged` **(14)** («عند تغيّر المالك»)، بمرشّحَي نوع تغيّر المالك والخطوة **– Since 1.8.**
- **نوع الإجراء المخصّص وخيار Contextual** على مشغّل الإجراء المخصّص (بديلًا عن مفتاح *UI Event*) **– Since 1.8.**
- **إعادة تسمية قيم نوع الإجراء المخصّص** من `Internal` / `Module` إلى `External` / `Local` (بقيت `UI` كما هي)، بترحيل أحادي الاتّجاه **– Since 1.8.**
- **إجراء** `ExecuteCustomAction` **(106)** («تنفيذ إجراء مخصّص»)، بدأ بنطاقَي Self/Reference فقط، ثم توسّع ليشمل نطاقات Relation/Expression/Pointer، وربط المُعامِلات (Parameters)، وحلّ المراجع مع مراعاة الكيانات الوهمية (proxy) **– Since 1.8.**
- `GET /v1/reactions/custom-actions?moduleId=` – نقطة اكتشاف الإجراءات القابلة للاستدعاء التي يعتمد عليها مُنتقي تنفيذ إجراء مخصّص **– Since 1.8.**
- **لا تستطيع أزرار النظام إطلاق سوى إجراءات من مستوى** `UI` (يُرفَض ربط زرّ بإجراء من مستوى `External` / `Local` بخطأ 403 وقت التنفيذ) **– Since 1.8.**
- **عزل الكيانات** – لا يُشغَّل التفاعل أبدًا على مُدخل من كيان آخر **– Since 1.8.**
- `GET /v1/reactions/module/{moduleId}` **– Since 1.8.**
- **حدثا التشغيل** `ReactionFailed` **(10)** و`DataEntryRestored` **(12)** – غائبان عن `EventTypeEnum` في خطّي الإصدار 1.4 و1.6g **– Since 1.8.**
- **مقاييس تنفيذ التفاعلات** (`osos_reaction_executed_total` و`osos_reaction_duration_seconds`) **– Since 1.8.**
- `sensitiveDataAccessPolicy` **على التفاعل** (Prohibited / Privileged / User-Based / Auto)، مع رفض `User-Based` على مشغّل آلي **– Since 1.10.** انظر سياسة الوصول إلى البيانات الحساسة.
- **عاد إجراء تحديث مدخل يعمل تحت سياسة حاجبة** – إذ صار الحارس مقصورًا على القيمة التي أدخلتها عملية الكتابة هذه بدلًا من المُدخل كاملًا **– Since 1.10.**
- **صارت عناوين المالك المجموعة تُحلّ في رموز التفاعلات** (`_metaCurrentSteps[n].originalOwner`) بعد أن كانت تُصيَّر فارغة **– Since 1.10.**
- **تنفيذ دائم قابل للاستئناف** – قد تكتمل مُدخلات الدفعة بأي ترتيب، والتشغيل الذي تقطعه إعادة تشغيل يكتب سجلّه ثم يتوقّف، ويُنشر حدث انتهاء التفاعل مرة واحدة على الأقل **– Since 1.10.** انظر ما الذي تتوقّعه من نموذج التنفيذ.
- `DataReaction.description` **– Since 1.10.** – حقل وصف اختياري مُوطَّن على التفاعل

- **عدّادات التنفيذ/الفشل على قائمة التفاعلات، ونقطة طرفية جديدة لسجلّ عمليات التشغيل** (`GET /v1/reaction/{entity}/{id}/history`) – **Since 1.10.** انظر عدادات التنفيذ/الفشل وسجلّ عمليات التشغيل.
- **إعادة تصميم تبويب التفاعلات** – بحث، ومرشّح بيانات وصفية، وفرز، وتجميع، ومرشّح حالة مُنفَّذ/غير مُنفَّذ، ووصف/نطاق في الصفّ الموسَّع – **Since 1.10.** انظر تبويب التفاعلات [بناء التطبيقات › التفاعلات والأتمتة].
- **إصلاحات كتابة الكتلة القابلة للتكرار** – بات الإجراء الثاني من «إنشاء عنصر كتلة» على الكتلة نفسها يُلحِق العناصر بدلًا من استبدالها، ولم تعد كتابة حقل فرعي واحد عبر «تحديث مدخل» على كتلة قابلة للتكرار تطوي بقية صفوفها – **Since 1.10.** انظر إصلاحات كتابة الكتلة القابلة للتكرار.
- **يمكن لكتابات إجراء "تحديث مدخل" في التفاعلات الانضمام إلى حماية إعادة المحاولة عند التعارض نفسها المتاحة لكتابة مدخل مباشرة**، من خلال راية ميزة – **Since 1.10.** انظر الكتابات المدفوعة بالتفاعلات تكتسب الحماية نفسها [بناء التطبيقات › إنشاء كيان وتصميمه] في نمذجة البيانات: الكيانات.
- **إجراءا** `SetPermission` **(31) /** `RemovePermission` **(32)**، مع حارس الحفظ الذي يمنع التفاعل من منح أذونات تتجاوز مستوى وصول من يحفظه – **Since 1.10.** انظر إجراءا منح الإذن وإزالته.
- **مخرجات «تنفيذ وظيفة»** (`success` و`message` والمفاتيح تحت `result`) قابلة للربط في الإجراءات اللاحقة والتفاعلات المتسلسلة – **Since 1.10.** انظر مخرجات «تنفيذ وظيفة» في التفاعلات المتسلسلة.
- **تقاعد** `CreateDataEntity` **/** `UpdateDataEntity` – يُرفضان بالرمز 422 عند الحفظ ويُتخطّيان عند التشغيل – **Since 1.10.**
- **تقييم المعايير على مصدر** `$documents` **في الذاكرة** بدلًا من مجموعة مؤقّتة – **Since 1.10.** انظر مرشّح المعايير والاستعلام.
- **تخزين سياق التشغيل لتفاعل مستمع فقط، ومضغوطًا** – **Since 1.10.** انظر تخزين سياق التشغيل.
- **قصر نقطة سجلّ التشغيلات على مسؤولي النظام بصفوف مختصرة، وواجهة سجل التفاعلات على مستوى النظام** (`GET /v1/reaction-logs` و`GET /v1/reaction-logs/{id}`) – **Since 1.10.** انظر واجهة سجل التفاعلات على مستوى النظام.
- **حجب الأسرار في سجلّ التفاعلات** – **Since 1.6.** انظر حجب الأسرار في سجلّ التفاعلات.
- **فهارس سجلّ التفاعلات وتنظيف المجموعات المؤقّتة** – **Since 1.10.** انظر فهارس سجلّ التفاعلات والتنظيف.
- **تصحيحات وقت التشغيل** – حارس الحلقات لكل خطوة سير عمل، وخطوة الدخول في التدفق الفرعي، وتعيين بيانات سير العمل الوصفية، وإبقاء المجموعة عند إعادة الإسناد، وفشل إجراءات عناصر الكتلة بدل نجاحها الصامت (**Since 1.4**)؛ وقيم In / Not In المفردة، وتسجيل الرسائل النصية بالنوع `SendSms`، وفشل الاستعلام المحفوظ احتياطًا (**Since 1.10**). انظر تصحيحات وقت التشغيل.
- **تأليف التفاعل يتطلّب وصولًا بمستوى مطوّر فما فوق على كيانه** (`403 REACTION_AUTHORING_FORBIDDEN`)، ويشمل ذلك الاستيراد – **Since 1.8.** انظر من يحقّ له تأليف تفاعل.
- **حدود استدعاء API** – رفض الوجهات الخاصة والداخلية، وحدّ 30 ثانية لعملية التبادل كاملةً، وسقف 10 MiB للأجسام، ومنع الرموز من إضافة ترويسات – **Since 1.8**؛ وأسماء ترويسات المنصة المحجوزة (`400 RESERVED_HEADER_NAME`) – **Since 1.10.** انظر مرجع خيارات الإجراءات › استدعاء API: ما تسمح به المنصة.
- **إجراء إرسال WhatsApp (107)** مع منتقي القوالب المعتمدة – **Since 1.10.** انظر مرجع خيارات الإجراءات › خيارات إرسال WhatsApp.
- **مراجع الملفات في خيارات التفاعل** (`400 FILE_EXPIRED` و`503 STORAGE_SERVICE_UNAVAILABLE`) – **Since 1.10.** انظر الملفات في خيارات التفاعل.

- **حدّ التشغيل الشهري** مع التشغيلات `refused` و`refusedCount` — **Since 1.10.** انظر حدّ التشغيل الشهري والتشغيلات المرفوضة.
- **يمكن للتفاعل المكرَّر تغيير حدثه**، ويعود مصدر الكيان المطلق الذي يستبعده الحدث الجديد إلى **Since 1.8** — Self. انظر تعريف تفاعل [بناء التطبيقات › التفاعلات والأتمتة].
- **التفاعلات المتسلسلة: فرض المساحة المسبَّبة، ورفض السلاسل المستحيلة عند الحفظ، وتشغيل كل كيان تفاعلاته المتسلسلة وحدها** — **Since dev.** انظر مرجع خيارات الإجراءات › أشكال إعداد الحدث.

أمّا بقية وظائف المحرّك فهي **Long-standing** وتسبق المعلَم 1.2، ولذلك لا تحمل وسم "Since"، بما في ذلك `loopThreshold` و`SlaTimeExceeded` وتفاعلات مصدر العلاقة الواردة و`NotifyZapier`.

مرجع واجهة الأزرار البرمجية

هذه الصفحة موجّهة إلى **المُكاملين والمطوّرين** الذين يقرؤون أزرار الكيان أو ينشئونها أو ينفّذونها عبر الواجهة البرمجية، أو يحتاجون إلى معرفة ما يحدث تحديدًا عند النقر على زر. وهي توثّق النموذج المخزَّن، وتعداداته الرقمية، وآليات التحقق، ونقاط النهاية، وتدفّق التنفيذ.

> **تصمّم الأزرار من الواجهة؟**
>
> على بُناة التطبيقات الذين يضيفون أزرارًا من تبويب **الأزرار** في الكيان أن يبدؤوا بدليل الأزرار [بناء التطبيقات]؛ فهو يغطي الميزات نفسها بالتسميات التي يعرضها المصمّم.

كيف يُخزَّن الزر

داخليًا، لا يُعدّ زر النظام **مجموعة مخصّصة**؛ بل هو `DataEntry` عادي يشير فيه `dataEntity` إلى الكيان المدمج `SYSTEM_BUTTON` (`StringConstants.SYSTEM_BUTTON_ENTITY_ID = 'SYSTEM_BUTTON'`)، ومفتاح المستودع (`SYSTEM_BUTTON_NAME = 'system_button'`). ويترتّب على ذلك ما يلي:

- **الإنشاء / التحديث / الحذف تمرّ عبر عمليات CRUD العامة للمُدخلات** الخاصة بكيان `system_button`، وليس عبر `SystemButtonController`؛ فالمتحكّم المخصّص لا يوفّر إلا نقاط نهاية *القراءة* و*الحالة* و*الخيارات* و*التنفيذ*.
- ترث الأزرار آليات `DataEntry`: `state` و `isInvalid` و `store` وإدارة الإصدارات والسجل التاريخي وتحديد نطاق الوصول بحسب المساحة/المخزن.

المرجع المعتمد: `SystemButtonController` (`controllers/system-button.controller.ts`)، و `SystemButtonService` (`services/system-button.service.ts`)، والنماذج الموجودة ضمن `models/system-buttons/` .

ملاحظات سلوكية وراء خيارات المصمّم

- **أنواع الإجراء.** تجمع قائمة **الإجراء** في المصمّم الإجراءاتِ المخصّصة للكيان ذات المستوى **UI** تحت **إجراء مخصص** (`RelevantActionTypeEnum.CustomEvent`)، وتعرض **الإنتقال** (`Redirect`) و**سكريبت** تحت **حدث واجهة**. ولا تمرّ عبر معالج التنفيذ إلا أزرار `CustomEvent`؛ أما أزرار `Redirect` و `Script` فيعالجها العميل. ويُستدلّ على وجود واجهة مستخدم للتفاعل المرتبط من كون نوع الإجراء المخصّص [بناء التطبيقات › التفاعلات والأتمتة] فيه **UI** `Since 1.8`، لا من علَم منفصل على التفاعل. ولتنفيذ منطق خارجي أكثر تعمّقًا، يستطيع التفاعل الذي يُطلقه الزر بدوره استدعاء وظيفة.
- **النطاق.** نوع `actionTypes` هو `ButtonActionTypeEnum[]`، ويشمل: `SingleEntry` و `MultipleEntries` و `DataCollection`. ويعرض المصمّم **مدخل واحد** و**عدة مدخلات** (تحديد متعدد)؛ أما `DataCollection` (قائمة) فتقبلها الواجهة البرمجية ومعالج التنفيذ.
- **رموز الرابط المخصّص** `Since 1.8`. تُخزَّن روابط رموز عنوان URL في زر **الإنتقال** في `redirection.urlTokensMap`، وتُستبدل الرموز بقيمها في `redirection.url` وقت القراءة عند تمرير `entryId`. ويحدّد `TokenSourceType` مصدر قيمة كل رمز (`Fixed` ، `Field` ، `Expression` ، `Pointer`). ويجري حلّ الرموز وفق مبدأ بذل أفضل جهد: فإذا تعذّر حلّ رمز (كحقل لم يعد موجودًا)، سُجِّل تحذير، وعاد الزر إلى نص الرابط الأصلي دون حلّ الرموز، بدلًا من منع التنفيذ؛ ولا يُعرَض أي خطأ على المستخدم النهائي.

- **متغيّرات نافذة التأكيد** `Since 1.8` لا يُسمح بها إلا حين تكون قيمة `actionTypes` هي `[SingleEntry]` تمامًا (يفرض ذلك التحقق).
- **ربط قيم الحقول.** تُخزَّن قيم التعبئة المسبقة في `relevantAction.fieldsMapping`. وتقابل أنواع القيم في المصمّم `FieldValueTypeEnum`: **حقل** = `Dynamic` (مسار المصدر في المُدخل الحالي)، و**يدوي**، و**بدون قيمة** (null)، و**مؤشر** (يقيّم رمز سياق مثل **@me** ويُدرج القيمة الناتجة مع عنوان عرضها)، و**تعبير برمجي** (JavaScript مخزّن). ويمكن أيضًا تحويل الربط نفسه إلى مرشّح: فرمز السياق متعدد القيم، مثل *مجموعاتي*، يُنشئ شرط *In*، مما يتيح للزر تعبئة نموذج مسبقًا وتحديد المُدخلات التي يستهدفها في آن واحد. وعند ضبط `relevantAction.overrideFieldsMappingLockedFields`، تُعامل أسماء الحقول المربوطة على أنها مقفلة؛ وتُعيد `getSystemButtonOverridableLockedFields(id)` تلك الأسماء (وإلا `[]`) لتقفلها واجهة التجاوز.
- **مرشّح الإتاحة.** قد يتضمّن الزر استعلامًا محفوظًا (`query`) ومرشّحًا مضمّنًا (`criteria`)؛ وعند وجودهما معًا، يُدمجان باستخدام **AND**. وتُعرض الاستعلامات المحفوظة ضمن منتقي مصدر المُدخلات في المصمّم. ويحدّد `nonMatchEntriesMode` السلوك مع المُدخلات غير المطابقة: `Disable` (الافتراضي) أو `Hidden`. ويرد وصف التقييم لكل مُدخل في الإتاحة: حالة المُدخلات.
- **أزرار السكربت** `Since 1.8` تخزّن متن السكربت في `relevantAction.customScript`. ويستدعيه العميل مع وسيط `context` الذي يتيح `entity` و`view` و`entry` و`entries` و`popupManager`؛ وفي الزر متعدد المُدخلات، يتضمّن `context.entries` جميع الصفوف المحدّدة.

نموذج البيانات

`SystemButtonDataEntry`

يوسّع `DataEntry`؛ ويضبط مُنشئه `dataEntity = 'SYSTEM_BUTTON'` على نحو ثابت. وتُخزَّن الحمولة الخاصة بالزر بأكملها ضمن `value` (`SystemButtonDataEntryValue`).

`SystemButtonDataEntryValue`

الحقل	النوع	مطلوب	ملاحظات
`name`	`LocalizedValue`	نعم	تسمية الزر. يُتحقَّق من أنها مُترجمة و**فريدة لكل كيان مرتبط**.
`normalizedName`	`string`	لا	معرّف برمجي يُولَّد تلقائيًا من `name`، ويكون فريدًا لكل `DataEntity` مرتبط.
`description`	`NullableLocalizedValue`	لا	وصف اختياري.
`icon`	`Icon`	نعم	أيقونة الزر.
`relatedTo`	`DataEntityReferenceProperty`	نعم	الكيان الذي يعمل عليه الزر. `relatedTo.dataEntry` هو معرّف الكيان.
`relevantAction`	`SystemButtonRelevantAction`	نعم	ما ينفّذه الزر (حدث / إعادة توجيه / سكربت).
`actionTypes`	`ButtonActionTypeEnum[]`	نعم	نطاقات الاختيار التي يدعمها الزر (مفرد / مجموعة / متعدّد).
`isPublished`	`boolean`	لا (الافتراضي `false`)	لا تعيد نقاط النهاية «النشطة» إلا الأزرار المنشورة، وهي وحدها القابلة للتنفيذ.
`query`	`ReferenceProperty | null`	لا	مرجع إلى استعلام بيانات مخزَّن تتحكّم `criteria` الخاصة به في إتاحة الزر.
`criteria`	`FilterExpression`	لا	تعبير ترشيح مضمّن يتحكّم في إتاحة الزر. يُدمَج باستخدام AND مع `query.criteria` عند وجود كليهما.

الحقل	النوع	مطلوب	ملاحظات
`nonMatchEntriesMode`	`SystemButtonNonMatchEntriesMode`	لا	السلوك عند عدم مطابقة المُدخلات المختارة للمعايير: التعطيل أو الإخفاء.
`confirmationDialog`	`SystemButtonConfirmationDialog`	لا	حوار تأكيد اختياري يُعرض قبل التنفيذ.

`SystemButtonRelevantAction`

الحقل	النوع	ملاحظات
`type`	`RelevantActionTypeEnum`	`CustomEvent = 0` ، `Redirect = 1` ، `Script = 2` .
`triggerId`	`string`	لأزرار `CustomEvent` : معرّف التفاعل/المشغّل الذي يُحلّ بالرجوع إلى reaction-service.
`redirection`	`SystemButtonRedirection`	هدف إعادة التوجيه الأساسي.
`successRedirection`	`SystemButtonRedirection`	إعادة التوجيه عند النجاح.
`failureRedirection`	`SystemButtonRedirection`	إعادة التوجيه عند الفشل.
`fieldsMapping`	`EntryActionOptionsValue[]`	تجاوزات الحقول المُمرَّرة إلى الإجراء.
`overrideFieldsMappingLockedFields`	`boolean`	عند ضبط هذه القيمة، تُعامَل أسماء الحقول المربوطة على أنها مقفلة.
`customScript`	`string`	متن السكربت للأزرار من نوع `Script` . `Since 1.8`

`SystemButtonRedirection`

الحقل	النوع	ملاحظات
`type`	`ButtonRedirectionTypeEnum`	`RelatedEntries = 0` ، `SameView = 1` ، `Url = 2` .
`relationId`	`string`	العلاقة التي يجري التنقّل عبرها (لعمليات إعادة التوجيه المعتمدة على العلاقات).
`relationRedirectionTarget`	`ButtonRelationRedirectionTargetEnum`	`EntriesList = 0` ، `ShowEntryDetails = 1` ، `CreateEntry = 2` ، `UpdateEntry = 3` .
`url`	`string`	عنوان URL المستهدف لعمليات إعادة التوجيه من نوع `Url` .
`urlTokensMap`	`{ [key]: TokenSource }`	روابط الرموز التي تُدرَج قيمها في `url` وقت القراءة عند توفير `entryId` (تستخدم المفاتيح نمط البادئة `str-`).

`SystemButtonConfirmationDialog`

`Since 1.8`

يُعرض قبل تنفيذ الزر.

الحقل	النوع	ملاحظات
`enabled`	`boolean` (الافتراضي `false`)	المفتاح الرئيسي.
`title`	`NullableLocalizedValue`	عنوان الحوار (يدعم الرموز النائبة).
`content`	`NullableLocalizedValue`	نص المتن المطوّل (يدعم الرموز النائبة).

الحقل	النوع	ملاحظات
`proceedButtonLabel`	`NullableLocalizedValue`	تسمية زر التأكيد.
`cancelButtonLabel`	`NullableLocalizedValue`	تسمية زر الإلغاء.
`variables`	`SystemButtonDialogVariable[]`	روابط الرموز المستخدمة لإدراج القيم في `content` / `title` . **مسموح بها فقط عندما يكون** `actionTypes` **مساويًا تمامًا لـ** `[SingleEntry]` .

`SystemButtonDialogVariable` : `{ name: string, valueType: TokenSourceType, value: any }` . و `TokenSourceType` هو `Fixed = 0` ، `Field = 1` ، `Expression = 2` ، `Pointer = 3` .

`ButtonClickedHistory`

يُسجَّل لكل مُدخل مستهدف بعد تنفيذ زر من نوع `CustomEvent` (عبر `DataEntryHistoryService.createHistoryForButtonClicked` ، مع `historyType = ButtonClickedHistory`). ويخزّن `buttonId` والاسم المُترجم `buttonName` .

التعدادات

```
RelevantActionTypeEnum          CustomEvent = 0, Redirect = 1, Script = 2
ButtonActionTypeEnum            SingleEntry = 0, DataCollection = 1, MultipleEntries = 2
SystemButtonNonMatchEntriesMode  Disable = 0, Hidden = 1
ButtonRedirectionTypeEnum       RelatedEntries = 0, SameView = 1, Url = 2
ButtonRelationRedirectionTargetEnum  EntriesList = 0, ShowEntryDetails = 1, CreateEntry = 2, UpdateEntry = 3
EventExecutorTypeEnum           SYSTEM, MANAGER, EXECUTOR   // EXECUTOR = the user who initiated the action
AskUserForEntityTypeEnum        InlineEntity = 0, View = 1
TokenSourceType                 Fixed = 0, Field = 1, Expression = 2, Pointer = 3
```

قيم التعدادات رقمية (تُخزَّن كأرقام)، باستثناء `EventExecutorTypeEnum` ؛ فهو تعداد نصّي.

إنشاء زر وتحديثه (API)

لا توجد نقطة نهاية لإنشاء زر عبر `POST /v1/system-buttons` . تُدار الأزرار عبر عمليات CRUD العامة للمُدخلات، باستخدام كيان `system_button` :

العملية	المسار (من `data-entries.controller.ts`)
الإنشاء	`POST /v1/entities/{entity}/entries`
التحديث	`PATCH /v1/entities/{entity}/entries/{id}` (v1) / `PATCH /v2/entities/{entity}/entries/{id}`
الحذف	`DELETE /v1/entities/{entity}/entries/{id}`

هنا يُحلّ `{entity}` إلى كيان `SYSTEM_BUTTON` / `system_button` . ومتن الطلب هو `DataEntry` يحتوي في `value` على `SystemButtonDataEntryValue` ، على النحو الموصوف أعلاه.

التحقّق

يعمل `SystemButtonValidator` في مرحلتَي `CREATE` و `UPDATE` لكيان `system_button` ، ويفرض ما يلي:

1. يجب أن يكون `value.relatedTo.dataEntry` موجودًا، وأن يكون الكيان المرتبط موجودًا. إذا كان الكيان المرتبط ضمنيًا ولم يكن `implicitOptions.allowEdit` مفعّلًا، يُطلَق `ImplicitEntityUpdateException` .
2. **متغيّرات حوار التأكيد خاصة بالمُدخل الواحد فقط.** إذا كانت `confirmationDialog.variables` موجودة، فيجب أن يكون `actionTypes` مساويًا تمامًا لـ `[SingleEntry]` ؛ وإلّا يُطلَق `ConfirmationDialogVariablesNotAllowedForMultiScopeException` (`SYSTEM_BUTTON_CONFIRMATION_DIALOG_VARIABLES_NOT_ALLOWED_FOR_MULTI_SCOPE`).
3. لأزرار `CustomEvent` التي لها `triggerId` ، يُحلّ المشغّل بالرجوع إلى reaction-service. وإذا لم يكن المشغّل موجودًا، يُطلَق `TriggerNotExistException` ؛ أما إذا كان غير صالح (`validations.isValid === false`) أو كانت قيمة `disabled` مفعّلة، فيُطلَق `InvalidOrDisabledTriggerException` .
4. **فرادة الاسم:** يجب أن يكون `value.name` مُترجمًا (`LanguagesService.validateName`) وفريدًا لكل كيان مرتبط؛ فأي تطابق في قيمة أي لغة مع زر آخر يتبع `relatedTo.dataEntry` نفسه يؤدّي إلى إطلاق `UniqueValueException` .

قراءة الأزرار (API)

تتطلّب جميع مسارات `SystemButtonController` مصادقة `Auth2` وتمرّ عبر مُصوِّت الكيان (entity voter). وتُحدَّد نطاقات القراءات ضمن المخازن القابلة للقراءة في المساحة (`StoreService.getStoresForRead`) وضمن `state = Active` و `isInvalid = false` .

الطريقة والمسار	الغرض
`GET /v1/system-buttons`	جميع الأزرار النشطة في المساحة، **مُثراة** لكل زر عبر reaction-service (`SystemButtonDataEntryDto[]`). يحدث N+1 عبر حدّ الخدمة – بطيء للمساحات الكبيرة.
`GET /v1/system-buttons/raw`	جميع الأزرار النشطة كـ **مُدخلات مخزَّنة خام** (`SystemButtonDataEntry[]`)، دون إثراء بالتفاعل؛ ولا تُحذف الأزرار ذات المُشغِّل المعطَّل. مخصّصة للمستهلكين الداخليين (مثل كشف تعارضات الاستيراد أثناء التحزيم). `Since` `1.8`
`GET /v1/system-buttons/count`	عدد الأزرار النشطة (يدعم مُرشِّح `Where`).
`GET /v1/system-buttons/by-id/{id}`	زر واحد نشط وصالح بحسب المعرّف (مُدخل خام). يعيد 404 `SystemButtonNotFound` إن لم يُعثَر عليه.
`GET /v1/system-buttons/{relatedToId}`	أزرار كيان مرتبط (DTOs مُثراة).
`GET /v1/system-buttons/{relatedToId}/active?entryId=`	الأزرار النشطة (المنشورة) التي يمكن للمستخدم رؤيتها لكيان مرتبط. يحلّ `entryId` الاختياري رموز عنوان URL لإعادة التوجيه ومتغيّرات حوار التأكيد مقابل ذلك المُدخل.
`POST /v1/system-buttons/{relatedToId}/status/entries`	حالة التفعيل/الإخفاء لكل زر عبر مجموعة من المُدخلات (انظر الإتاحة).
`GET /v1/system-buttons/{id}/options`	إجراءات التفاعل لزر `CustomEvent` (`DataReactionAction[]`).
`GET /v2/system-buttons/{id}/options`	خيارات v2: الإجراءات **بالإضافة إلى** بيانات وصفية لكيان/معرض/حوار سؤال المستخدم (`SystemButtonOptionsDto`).
`POST /v1/system-buttons/{id}`	تنفيذ الزر (انظر التنفيذ).

إثراء الـ DTO

يجلب `getAllActiveSystemButtonsDto` / `getSystemButtonsByRelatedToDto` التفاعل المرتبط من reaction-service ويطوي البيانات الوصفية للمُشغِّل داخل `relevantAction` كـ `SystemButtonRelevantActionDto` : `triggerName` و `uiEvent` و `askUser` و `isTriggerDisabled` و `executionIndicator` و `messages` و `executorType` (بالقيمة الافتراضية `EventExecutorTypeEnum.EXECUTOR` عندما لا يكون للمُشغِّل مُنفِّذ صريح). في قوائم DTO «النشطة»، تُتخطّى الأزرار التي يكون مُشغِّلها المرتبط `disabled` ؛ أما نقطة `raw` فتبقيها.

`uiEvent` مُشتقّة لا مُخزَّنة `Since 1.8` – لا يزال الـ DTO يُخبِر العميل بما إذا كان للمشغِّل المرتبط واجهة مستخدم، لكن المشغِّل لم يعد يحمل علَم `uiEvent` منفصلًا؛ لذا يحسب كلا بانيا الـ DTO هذا الحقل الآن من نوع الإجراء المخصّص [بناء التطبيقات ‹ التفاعلات والأتمتة] للمشغِّل: تكون `uiEvent` صحيحة تمامًا حين يكون نوع الإجراء المخصّص **UI**. ولم يتغيّر شكل البيانات المتبادلة – فهي ما تزال قيمة منطقية عادية – فلا يتأثّر مستهلكو الأزرار الحاليون؛ إذ تغيّر مصدر القيمة فقط.

`askUser` علَم ثانية مستقلّة `Since 1.8` . بما أنّ الزرّ لا يمكن ربطه الآن إلا بإجراء مخصّص من مستوى **UI** (انظر لا تستطيع أزرار النظام إطلاق سوى إجراءات من مستوى UI أدناه)، صارت قيمة `uiEvent` هي `true` لكلّ زرّ – فلم تعد قادرة على إخبار العميل بما إذا كان يجب فتح حوار «اسأل المستخدم» قبل التنفيذ. لذا تتيح `SystemButtonRelevantActionDto` علَمًا منطقيًا ثانيًا، `askUser` ، تُحسَب قيمته `true` بالضبط حين يكون `eventConfig.askUser.enabled` الخاص بالمشغِّل المرتبط مُفعَّلاً. **يجب على العملاء ربط إظهار حوار سؤال المستخدم بـ `askUser` ، لا بـ `uiEvent`** – إذ يُنفَّذ الآن زرٌّ مرتبط بإجراء من نوع `UI` بلا إعداد سؤال مستخدم فوراً، دون ظهور حوار فارغ أولاً.

عند توفير `entryId` إلى `getSystemButtonsByRelatedToDto` ، تقوم الخدمة أيضًا بما يلي:

- تحلّ `relevantAction.redirection.urlTokensMap` مقابل المُدخل وتدرجها في `redirection.url` ، و
- تحلّ `confirmationDialog.variables` إلى قيم رموز وتدرجها في `content` / `title` الخاصة بالحوار (لكل لغة).

تُجرى كلتا العمليتين بأفضل جهد ممكن: تُسجَّل الإخفاقات كتحذيرات ويُعاد الزر رغم ذلك بنصّ غير مُحلّل.

خيارات سؤال المستخدم (`GET /v2/system-buttons/{id}/options`)

لزر `CustomEvent` يكون لتفاعله `eventConfig.askUser.enabled` ، تعيد هذه النقطة واجهة سؤال المستخدم التي يجب أن تعرضها الواجهة:

- `mode = InlineEntity` ← تعيد كيان التفاعل المضمّن `entity` بوصفه `askUserEntity` .
- `mode = View` ← تحمّل الكيان والمعرض المهيّأين وتعيد كيانًا **مُسقَطًا** عبر `getEntityBasedOnAskUserViewConfigurations` ، الذي يبقي فقط الحقول المدرجة تحت العملية المدعومة `AllowAskUserFor` للمعرض، ويعيد تسمية الكيان إلى `askUser` ، ويعيد ترقيمه كـ `TRASK_{entityId}_{triggerId}` .

إذا كان المُشغِّل مفقودًا أو غير صالح أو معطَّلًا، تنطبق الأخطاء نفسها: `TriggerNotExistException` / `InvalidOrDisabledTriggerException` .

تُستبقى الحقول الواقعة داخل كتلة مشتركة `Since 1.8` – تحمل قائمة السماح `AllowAskUserFor` الخاصة بالمعرض قيم `fullNormalizedName` مثل `cc.email` أو `attachment` . ويُستبقى الحقل ذو المستوى الأعلى عندما يطابق `normalizedName` الخاص به عنصرًا مسموحًا في القائمة؛ ويُستبقى الحقل الفرعي التابع لكيان داخلي (كتلة) عندما يكون **أيّ** من أسمائه المؤهَّلة بالكتلة مسموحًا. وقاعدة «أيّ» هذه هي ما يجعل الكتل المشتركة تعمل: فحين يقوم تعريف كتلة واحد خلف عدّة حقول كتلة ذات مستوى أعلى – مثل كتل مستلمي To / CC / BCC – يُصدِر الكيان المُسقَط الحقول الفرعية لتلك الكتلة مرّة لكل بادئة (`to.email` ، و `cc.email` ، و `bcc.email` ، …)، وكلها تحمل معرّف الحقل الفرعي نفسه. لذا فالمعرض الذي يتيح `cc.*` و `bcc.*`

فقط يظلّ يستبقي تلك الحقول الفرعية المشتركة، لأن المطابقة تُختبر مقابل كل ظهور مؤهَّل بالكتلة للمعرّف لا مقابل أول ظهور فقط. ومن ثمّ صار بإمكانك أن تسأل المستخدم عن حقل فرعي يقيم داخل كتلة مشتركة.

الإتاحة: الحالة لكل مُدخلات

يجيب `POST /v1/system-buttons/{relatedToId}/status/entries` عن السؤال «بالنسبة لهذه المُدخلات المختارة، هل كل زر مُفعَّل أم مُخفى؟».

متن الطلب (`GetSystemButtonsStatusForEntriesRequestDtoModel`):

```
{
  "dataEntriesIds": ["<entryId>", "..."],
  "selectAll": { "enabled": true, "...": "SelectAllDto criteria" }
}
```

يجب أن تُوفِّر **إمّا** `dataEntriesIds` **أو** `selectAll.enabled = true` ؛ وإلّا يُطلَق `GetSystemButtonsStatusForEntriesMustProvideIdsOrSelectAllException` (`GET_SYSTEM_BUTTONS_STATUS_FOR_ENTRIES_MUST_PROVIDE_IDS_OR_SELECT_ALL`). مع `selectAll` ، تُحلّ معرّفات المُدخلات من محرّك الاستعلام (`getEntriesIdsBySelectAllDto`).

الاستجابة هي `SystemButtonStatusForEntriesResponseDtoModel[]` : `{ systemButtonId, isDisabled, isHidden }` .

المنطق (`getSystemButtonsStatusForEntries` + `generateGetButtonsStatusForEntriesAggregationStages`):

- الأزرار التي **لا** تملك `query` و**لا** `criteria` تكون دائمًا `isDisabled: false` .
- بالنسبة للأزرار التي **تملك** مُرشِّحًا، يشغّل تجميع MongoDB عملية `$facet` – فرع واحد لكل زر – تحصي كم من المُدخلات المختارة يطابق معايير الزر (`query.criteria` مدموجة مع `criteria` المضمّنة عبر `AND`). ويكون الزر **مُفعَّلًا فقط إذا طابق كل مُدخل مختار** (`matchingEntriesCount === dataEntriesIds.length`)؛ وإلّا يكون مُعطَّلًا.
- عندما **لا** يطابق أي مُدخل مختار على الإطلاق، يعتمد سلوك عدم التطابق على `nonMatchEntriesMode` : الافتراضي (`Disable` ← `{ isDisabled: true, isHidden: false }` ؛ و `Hidden` ← `{ isDisabled: true, isHidden: true }` .
- تُحلّ الحقول النظامية مثل `@current_user` / `@myGroups` في المعايير عبر `getSystemFieldsStages` قبل عملية الـ facet.

تنفيذ زر

`POST /v1/system-buttons/{id}` مع متن `ExecuteSystemButtonDto` :

الحقل	النوع	ملاحظات
`mode`	`ButtonActionTypeEnum`	نطاق الاختيار لهذا التنفيذ.
`source`	`string | string[]`	معرّف مُدخل واحد، أو معرّف تجميعة بيانات، أو مصفوفة من معرّفات المُدخلات – بحسب `mode` .
`askUserValue`	`IEntryValue`	حمولة سؤال المستخدم المجمَّعة من الحوار المُمرَّرة إلى التفاعل.
`selectAll`	`SelectAllDto`	لـ `MultipleEntries` مع تحديد الكل: يحلّ مجموعة المعرّفات الكاملة على جانب الخادم.

تدفّق التنفيذ (ذو معنى فقط لـ `RelevantActionTypeEnum.CustomEvent` ؛ أزرار `Script` / `Redirect` تُحلّ على جانب العميل ولا تمرّ عبر هذا المعالِج):

1. **حارس المعرض** – إذا كان الطلب محدود النطاق بـ `DataViews`، فيجب أن يكون الزر مدرجًا في `view.systemButtons`، وإلّا يُطلَق `SystemButtonNotFound`.
2. **تحميل الزر** – يجب أن يكون `state = Active`، و`isInvalid = false`، و`value.isPublished = true`، وإلّا يُطلَق `SystemButtonNotFound`.
3. **حلّ المُدخلات الهدف** من `mode` + `source`:
   - `SingleEntry` ← `ReferenceProperty` واحد لـ `source`.
   - `DataCollection` ← عناصر `items` الخاصة بالتجميعة (يجب أن تنتمي التجميعة إلى الكيان المرتبط بالزر، وإلّا `DataCollectionNotFoundException`).
   - `MultipleEntries` ← مصفوفة المعرّفات؛ وإذا كان `selectAll.enabled`، تُوسَّع المعرّفات عبر `getEntriesIdsBySelectAllDto`.
   - `source` مفقود ← `MissingPropertyException`.
4. **إعادة فحص الإتاحة** للمُدخلات المُحلّة؛ فإذا كان الزر معطَّلًا لتلك المجموعة، يُطلَق `SystemButtonIsDisabled` (`SYSTEM_BUTTON_IS_DISABLED`، 403).
5. **حارس مستوى** `UI` `Since 1.8` – إذا كانت `relevantAction.triggerId` مضبوطة، يُفحَص نوع الإجراء المخصّص للمشغّل المرتبط؛ ويُرفَض الربط بمستوى `Local` أو `External` بخطأ `SystemButtonNotUiCustomAction` (403). انظر لا تستطيع أزرار النظام إطلاق سوى إجراءات من مستوى UI أدناه.
6. **إصدار تغذية راجعة + نشر حدث** – يُصدَر `JobScheduledMessage` عبر المقبس فورًا لتغذية الواجهة الراجعة، ثم يُنشَر `CustomEventTriggeredEvent` (معرّف الكيان، والمُدخلات الهدف، ومعرّف المستخدم، و`triggerId`، والطابع الزمني، و`askUserValue`) على ناقل الخدمة، حاملًا `messageId` و`instanceId` و`correlationId` (معرّف التتبّع) والسمات اللاصقة (`userId` و`delegateeId` و`delegationId` و`spaceId`). ويلتقط هذا معالِج reaction-service.
7. **السجل** – يسجّل `createHistoryForButtonClicked` مُدخل `ButtonClickedHistory` لكل هدف.

تعيد النقطة `messageId` المُولَّد (وهو UUID) كي يستطيع العميل ربط نتيجة التنفيذ غير المتزامن.

لا تستطيع أزرار النظام إطلاق سوى إجراءات من مستوى UI `Since 1.8`

قد تشير `relevantAction.triggerId` الخاصة بزرّ بنيويًا، إلى أيّ تفاعل `CustomEvent` – إذ لا شيء يمنعها من تسمية إجراء مخصّص [بناء التطبيقات › التفاعلات والأتمتة] من مستوى `Local` أو `External`، وهذه مُعدّة لتُستدعى فقط من إجراء تنفيذ إجراء مخصّص [بناء التطبيقات › التفاعلات والأتمتة] في تفاعل آخر، لا من نقرة زرّ. وتسدّ الخطوة 5 من تدفّق التنفيذ أعلاه هذه الثغرة: فهي تُحمِّل نوع الإجراء المخصّص للمشغّل المرتبط قبل نشر حدث التنفيذ، وترفض النقرة بخطأ `SystemButtonNotUiCustomAction` (403، "*this button is bound to a custom action that is not a UI action*") عندما يكون `Local` أو `External`.

أما الاتجاه المعاكس – فلا يمكن لإجراء تنفيذ إجراء مخصّص أن يستهدف أبدًا إجراءً مخصّصًا من مستوى `UI` – فيُفرَض ذلك وقت حفظ *التفاعل*؛ انظر تنفيذ إجراء مخصّص [بناء التطبيقات › التفاعلات والأتمتة].

مثال تنفيذ

```
POST /v1/system-buttons/{buttonId}
Content-Type: application/json

{
  "mode": 0,
  "source": "8f3c...entryId"
}
```

لتنفيذ متعدّد المُدخلات مع تحديد الكل:

```
{
  "mode": 2,
  "selectAll": { "enabled": true }
}
```

وضع المعاينة

`GET /v1/system-buttons/{relatedToId}/active` في وضع المعاينة يعيد أزرار الكيان **الحقيقية** (الاسم والأيقونة والتهيئة الحقيقية) لكن بـ **معرّفات وهمية** (`FakeEntryGeneratorService.withFakeIds`) بحيث لا يمكن تنفيذ أيّ منها على مُدخل وهمي؛ ولا يُمرَّر `entryId` عمدًا إلى الاستعلام الحقيقي في وضع المعاينة. وإذا كان المعرض الحالي يملك قائمة `systemButtons` فارغة، تعيد النقطة `[]` .

ملاحظات التنفيذ

- **تحديد نطاق المخزن/المساحة** يُطبَّق في `buildActiveSystemButtonsWhere` وفي `getSystemButtonsByRelatedToDto` : تُرشَّح الأزرار بحسب `store.dataEntry ∈ readableStores` ، و`state = Active`، و `isInvalid = false` ، و(للقوائم النشطة) `value.isPublished = true` .
- **تنبيه N+1:** تستدعي قوائم DTO المُثراة reaction-service مرة واحدة لكل زر. وُجد `getAllActiveSystemButtonsRaw` / `GET /v1/system-buttons/raw` تحديدًا لتفادي ذلك لدى المستهلكين الذين يحتاجون البيانات المخزَّنة فقط.
- **`getSystemButtonOverridableLockedFields(id)`** يعيد أسماء الحقول المُطابَقة عندما تكون `relevantAction.overrideFieldsMappingLockedFields` مُعيَّنة (وإلّا `[]`) – تُستخدم لقفل الحقول المُطابَقة في واجهة التجاوز.
- يشير `ExecuteSystemActionOverrideValuesDto` (`{ actionId, options }`) و `getTriggerToExecuteForSystemButton` / `validateSystemButtonTrigger` المُعلَّقان في الخدمة إلى مسار تجاوز إجراء أقدم غير نشط حاليًا؛ فمعالجة التجاوز تعيش الآن في reaction-service. لا تعتمد على تلك الكتل المُعلَّقة.

ملاحظات الإصدار

الميزة	المرحلة
إجراء السكربت المخصّص (`relevantAction.customScript`)	Since 1.8
حوار التأكيد (`confirmationDialog` ، ومتغيّرات الحوار)	Since 1.8
رموز الرابط المخصّص / حوار التأكيد	Since 1.8
نقطة القائمة الخام (`GET /v1/system-buttons/raw`)	Since 1.8
اشتقاق `uiEvent` من نوع الإجراء المخصّص للمشغّل (بديلًا عن العلَم المُخزَّن)	Since 1.8

الميزة	المرحلة
`askUser` على DTO الإجراء المرتبط بالزرّ (مُنظِّم حوار سؤال المستخدم، مستقلّ عن `uiEvent`)	Since 1.8
رفض تنفيذ أزرار النظام عند ارتباطها بإجراء مخصّص من مستوى `External` / `Local` (`SystemButtonNotUiCustomAction`، 403)	Since 1.8
`normalizedName` على الأزرار	Since 1.6

يسبق وضع عدم تطابق المُدخلات (`nonMatchEntriesMode`) ونقطة خيارات v2 مرحلة 1.2، لذا لا تحملان شارة توفّر (تُعامَلان كـ **Long-standing**).

تسجيل الدخول إلى البوابة والمساحات (التفاصيل الداخلية)

هذه الصفحة موجّهة إلى المُكامِلين والمطوّرين الذين يُحيلون المستخدمين إلى البوابة عبر روابط، أو يبنون عميلًا خاصًا بهم يعتمد على خدمة الهوية نفسها، أو يحتاجون إلى فهم دقيق للأسباب التي تحدّد وجهة المستخدم. وتتناول تدفّق OIDC في البوابة، والرابط العميق `/signin`، ورموز الأخطاء الكامنة خلف الرسائل التي يراها المستخدمون، وكيف تُحدَّد المساحة ويُفرض نطاقها.

للاطّلاع على هذه الميزات من منظور المستخدم النهائي، راجع تسجيل الدخول واختيار مساحة [استخدام أسس]. ويشرح دليل خادم الهوية للمشغّلين خدمة الهوية نفسها (المزوّدين، والرموز لمرة واحدة، وحدود المعدّل، ومخازن التدقيق)، فيما يشرح دليل المصادقة OAuthg نقاط نهاية OIDC ومِنَحه.

المساحات والنطاقات

المساحة هي الحدّ الفاصل لبيانات القاطن: فهي تحدّد نطاق الكيانات والمجموعات والمعارض والمستخدمين الذين يستطيع الطلب الوصول إليهم. ويحمل سجل المساحة بيانات عرضها (`space_information` : الاسم والوصف والأيقونة)، وعنوانها (`url` ، و `domain_identifier` ، و `full_url` المشتق منهما)، وحالتها من حيث النشاط وعدم النشاط، وحالة صيانة المستخدم النهائي.

تُقدَّم المساحة **الجذر** على النطاق الأساسي المجرّد (وتكون قيمة `domain_identifier` فيها `root`). وتُقدَّم كل مساحة أخرى على نطاق فرعي من النطاق الأساسي نفسه: فإذا كان العنوان الأساسي `example.com` والمعرّف `sales` ، أمكن الوصول إلى المساحة على `sales.example.com` . وبذلك يحدّد مضيف المتصفح المساحة، ويعني التبديل بين المساحات الانتقال إلى مضيف آخر.

وثمة مفهومان يُشتقّ كلاهما من المضيف، وينبغي ألّا يُخلط بينهما:

- **المساحة**: حدّ **البيانات**، أي البيانات التي يراها الطلب.
- **النطاق**: حدّ **تسجيل الدخول والهوية البصرية**، أي طرق تسجيل الدخول وسياسة المصادقة متعددة العوامل والمزوّدون الخارجيون المعتمَدون. وهذه العناصر هي التي تشكّل صفحة تسجيل الدخول.

للتعرّف إلى إدارة المساحات والنطاقات، راجع المساحات والنطاقات [الإدارة].

تدفّق تسجيل الدخول في البوابة

لا تجمع البوابة كلمة المرور أبدًا، بل تعيد توجيه المتصفح إلى الصفحات التي تستضيفها خدمة الهوية، ثم تستقبل المستخدم عند عودته، مستخدمةً تدفّق **رمز التفويض** (authorization-code) في OIDC. وتُحدَّد نقاط النهاية التالية نسبةً إلى عنوان الجهة المخوّلة (authority) الذي تستخدمه البوابة:

الغرض	نقطة النهاية
التفويض (تسجيل الدخول)	`{AUTHORITY}/auth`
الرمز (Token)	`{AUTHORITY}/token`
معلومات المستخدم	`{AUTHORITY}/me`
إنهاء الجلسة (تسجيل الخروج)	`{AUTHORITY}/session/end`

الغرض	نقطة النهاية
المفاتيح (JWKS)	`{AUTHORITY}/jwks`
صفحة العودة بعد تسجيل الدخول	`{APP_URL}/auth-callback`

التسلسل:

1. **فحص البدء.** يتحقّق التطبيق من وجود جلسة صالحة، بما في ذلك إجراء فحص صامت عبر iframe لدى خدمة الهوية. فإن وُجدت جلسة صالحة، دخل المستخدم مباشرةً.
2. **إعادة التوجيه إلى تسجيل الدخول.** إذا لم يكن المستخدم مسجّلًا دخوله وكان النشر مهيّأً لإعادة التوجيه تلقائيًا، أو اختار المستخدم **تسجيل الدخول**، ينتقل المتصفح إلى خدمة الهوية. وتُمرَّر لغة الواجهة في المعامل `lang` .
3. **المصادقة** بطريقة يسمح بها النطاق (انظر طرق تسجيل الدخول).
4. **العامل الثاني**، إن تطلّبه سياق المصادقة متعددة العوامل (انظر العامل الثاني).
5. **العودة** إلى `{APP_URL}/auth-callback` ، حيث يستكمل التطبيق الجلسة وينقل المستخدم إلى وجهته.

وإن تعذّر الوصول إلى خدمة تسجيل الدخول حين يختار المستخدم **تسجيل الدخول**، تعرض البوابة رسالة خطأ بدلًا من زر لا يستجيب `Since 1.10` .

أول تسجيل دخول

عند تسجيل دخول المستخدم للمرة الأولى، توجّهه البوابة إلى صفحة التعريف الأولي على `/welcome` بدلًا من الوجهة المقصودة. ويضع زر البدء فيها علامة اكتمال أول تسجيل دخول، ويبدأ الجولة التعريفية بالمنتج إن كانت لدى القاطن جولة مهيّأة ومفعّلة. وتبقى الصفحة متاحة من عنصر **المساعدة** في قائمة الملف الشخصي.

الرابط العميق `/signin` `Since 1.6`

يُحيل `https://<portal>/signin?returnUrl=<path>` المستخدم إلى تسجيل الدخول، ثم يعيده إلى `<path>` بعد تسجيل دخوله. ولا تعرض الصفحة نفسها إلا مؤشّر تحميل.

- **لا تُعتمد إلا المسارات الآمنة ضمن الموقع نفسه.** إذ تُتجاهل قيم `returnUrl` الخارجية أو المشوَّهة، فلا يمكن استغلال تدفّق تسجيل الدخول لإجراء إعادة توجيه مفتوحة (open redirect).
- **إن كان مسجّلًا دخوله أصلًا:** ينتقل المستخدم مباشرةً إلى الوجهة (أو إلى الصفحة الرئيسية)، ولا يعيده زر **الرجوع** عبر `/signin` مجددًا.
- **إن لم يكن مسجّلًا دخوله:** يُنقل المستخدم إلى صفحة تسجيل الدخول. ويعيده **الرجوع** منها إلى الصفحة الرئيسية للبوابة بدلًا من مؤشّر التحميل، ولا يبدأ تسجيل الدخول مجددًا، مما يمنع الوقوع في حلقة مفرغة.
- **`&provider=<name>` `Since 1.10` .** يبدأ تسجيل الدخول مباشرةً عبر أحد المزوّدين الخارجيين للنطاق. وتمرّر البوابة قيمته في معامل التفويض `idp_hint` . وهذه القيمة هي الاسم الفريد للمزوّد `name` ، وليست تسميته المعروضة. ويُتجاهل الاسم الفارغ أو المجهول، فتظهر صفحة تسجيل الدخول المعتادة. للاطّلاع على السلوك في خدمة الهوية، راجع `idp_hint` .

أُضيف الرابط العميق إلى سلسلة الإصدارات 1.6، وهو جزء من 1.8 وما بعده.

سلوكيات صفحة تسجيل الدخول التي تعتمد عليها البوابة

- **تخطّي المزوّد الوحيد** `Since 1.10`: إذا كان النطاق لا يتيح الدخول إلا عبر مزوّد خارجي واحد، تُتخطّى صفحة تسجيل الدخول. وهذا السلوك مفعّل افتراضيًا، ويستطيع المشغّل إيقافه. انظر تخطّي المزوّد الوحيد.
- **الإلغاء** `Since 1.4`: يعيد رابط **إلغاء** في صفحة تسجيل الدخول (وكذلك مؤقّت الخمول فيها المضبوط على عشر دقائق) `access_denied` إلى العميل؛ وتتعامل البوابة مع ذلك على أنه طلب للانتقال إلى الصفحة الرئيسية، فيصل المستخدم إلى صفحتها الرئيسية بدلًا من صفحة عدم الوصول (في 1.6 وما بعده). انظر إلغاء صفحة تسجيل الدخول.

حين يتعذّر إقلاع البوابة: رموز الأسباب `Since 1.10`

قبل أن يبدأ Angular، تجلب البوابة إعدادات تسجيل الدخول الخاصة بالمضيف (انظر إقلاع المتصفح دون سرّ العميل). وعند الفشل، تحلّ صفحة ثابتة ثنائية اللغة محلّ شاشة البدء. وتعرض الصفحة الرسالة الإنجليزية، والرسالة العربية، وكتلة تفاصيل (`Reason` ، `Status` ، `Request` ، `Correlation ID` ، `Time`)، وزر **إعادة المحاولة** الذي يعيد تحميل الصفحة. ويَرِد السبب أيضًا في سمة الصفحة `data-reason`.

رمز السبب	الرسالة العربية في الصفحة
`TENANT_NOT_RESOLVED`	هذا العنوان غير مرتبط بأي جهة.
`BOOTSTRAP_ENDPOINT_MISSING`	خدمة تسجيل الدخول غير متاحة على هذا الخادم.
`TENANT_NOT_READY`	لا يزال إعداد هذه الجهة قيد التنفيذ. يرجى المحاولة بعد قليل.
`SERVER_ERROR`	أعادت خدمة تسجيل الدخول خطأً.
`NETWORK`	تعذّر الوصول إلى خدمة تسجيل الدخول.
`TIMEOUT`	استغرقت خدمة تسجيل الدخول وقتًا طويلًا للاستجابة.
`BAD_CONFIG`	إعدادات تسجيل الدخول غير مدعومة في هذا الإصدار من التطبيق.
`DEPLOYMENT_CONFIG`	تعذّر تحميل إعدادات التطبيق.

القاطن قيد الصيانة `Since 1.10`

أثناء استعادة قاطن من نسخة احتياطية أو نقله إلى خلية أخرى، يُرفض تسجيل الدخول بالاستجابة `503` `TENANT_IN_MAINTENANCE` مع الترويسة `Retry-After`. ويكون نصّ الرسالة "*This tenant is being restored from a backup. Try again shortly.*" (أو "*…is being moved…*"). وأثناء الإقلاع، تظهر الحالة نفسها بالرمز `TENANT_NOT_READY`. للتفاصيل، راجع الأخطاء وإعادة المحاولة.

طرق تسجيل الدخول

يتوقّف ما تعرضه صفحة تسجيل الدخول المستضافة على النطاق:

الطريقة	تظهر حين	ملاحظات
اسم المستخدم + كلمة المرور	يكون `supportLocalSignIn` مفعّلًا في النطاق	تسجيل الدخول المحلي القياسي.
مزوّد خارجي (SSO)	يكون مزوّد واحد أو أكثر مرتبطًا بالنطاق	زر لكل مزوّد OAuth2/OIDC خارجي مسموح به.
مفتاح المرور (بلا كلمة مرور)	يكون `supportPasskey` مفعّلًا في النطاق	WebAuthn بتدفّق بيانات الاعتماد القابلة للاكتشاف؛ يمكن لأي مفتاح مرور مسجّل للموقع أن يستجيب.

وبعد نجاح تسجيل الدخول بكلمة المرور، قد تظهر للمستخدم الذي ليس لديه مفتاح مرور صفحة **تسجيل مفتاح المرور** (مع **تخطي**).

وعند فشل تسجيل الدخول المحلي، تُعرض صفحة تسجيل الدخول مجددًا مع الرسالة المطابقة:

الرمز	الرسالة
`INVALID_USERNAME_OR_PASSWORD`	اسم المستخدم أو كلمة المرور غير صحيحة
`ACCOUNT_LOCKED`	الحساب مقفل
`ACCOUNT_DEACTIVATED`	الحساب معطل، يرجى التواصل مع مسؤول النظام

وتخضع طلبات الهوية الحسّاسة (غير `GET`) لحدّ معدّل لكل عنوان عميل (10 طلبات كل 15 دقيقة افتراضيًا). انظر حدّ المعدّل على طلبات الهوية.

وتُسجَّل كل محاولة لأغراض التدقيق الأمني، سواء نجحت أم فشلت، وأيًا كانت الطريقة المستخدمة (`Since 1.4`)، بما في ذلك كلمات المرور الخاطئة، والحسابات المقفلة أو المعطّلة، وإخفاقات مفاتيح المرور، ونتائج المصادقة متعددة العوامل. ويرى المستخدمون سجلّهم الخاص في **تسجيلات الدخول › المحاولات**. انظر مسارات تدقيق المصادقة.

ربط الحسابات ومزامنة الملف الشخصي

- إذا كانت هوية المزوّد مرتبطة أصلًا بمستخدم، سُجّل دخوله مباشرةً.
- إذا وُجد حساب قائم بالبريد الإلكتروني نفسه دون ربط، فقد يُعرض **الربط اليدوي** `Since 1.8`؛ وفيه يثبت المستخدم ملكيته للحساب القائم بكلمة المرور. انظر الربط اليدوي للحسابات.
- إن لم يوجد حساب مطابق وكان المزوّد مهيّأً لذلك (`supportSignup`)، يُنشأ مستخدم تلقائيًا. انظر الإنشاء التلقائي.
- يحدّث `syncProfileOnSignIn` `Since 1.8` الملف الشخصي ببيانات المزوّد عند كل تسجيل دخول. ولا يمنع فشل المزامنة تسجيل الدخول أبدًا.

صفحات كلمة المرور

تستضيف خدمة الهوية صفحات الخدمة الذاتية لكلمة المرور:

- **نسيت كلمة المرور** على `/identity/reset-password`: يطلب المستخدم رابط إعادة تعيين بالبريد الإلكتروني، ثم يعيّن كلمة مرور جديدة عبر الرابط المرسَل. وتعرض الصفحة دائمًا رسالة النجاح نفسها، سواء طابق البريد حسابًا أم لا.
- **تغيير كلمة المرور** على `/identity/change-password`: للمستخدمين المحليين المسجِّلين دخولهم. أما الحسابات الخارجية (SSO)، فيُعاد توجيهها إلى خارج الصفحة، لأن المزوّد هو الذي يدير كلمة مرورها.

للاطّلاع على مدة صلاحية رابط إعادة التعيين واستبداله، راجع إعادة تعيين كلمة المرور.

العامل الثاني

بعد العامل الأول (كلمة المرور أو المزوّد الخارجي)، تحدّد خدمة الهوية **سياق المصادقة متعددة العوامل** للمستخدم بناءً على:

- ما إذا كان المستخدم قد سجّل **TOTP** (تطبيق المصادقة) `Since 1.8`؛
- ما إذا كان النطاق يتطلّب رموز **SMS** أو **البريد الإلكتروني** لمرة واحدة (`sms2FA` / `email2FA`)، وما إذا كان WhatsApp متاحًا `Since 1.10`؛
- ما إذا كانت لدى المستخدم **مفاتيح مرور** تصلح عاملًا ثانيًا.

إن لزم عامل ثانٍ، ينتقل المتصفح إلى `/interaction/{uid}/mfa`. وهناك يُدخل المستخدم رمزًا (أو يستخدم مفتاح مرور أو رمزًا احتياطيًا)، ويمكنه تبديل الطريقة وطلب إعادة الإرسال. ورموز الأخطاء هي: `MFA_INVALID_CODE`، و`MFA_CODE_EXPIRED`، و`MFA_ATTEMPTS_EXCEEDED` (ويستلزم الأخير طلب رمز جديد). وإن لم ينطبق أي عامل، تُتخطّى المصادقة متعددة العوامل.

- تُحفظ آخر طريقة مختارة في **الحساب** لا في المتصفح، فتظل محفوظة بعد إعادة التحميل، وتكون لأي مستخدم آخر على الحاسوب نفسه طريقته الخاصة `Since 1.8`.
- يعتمد حدّ إعادة الإرسال على نافذة زمنية متدحرجة ثابتة، وليس على عدّاد يشمل محاولة تسجيل الدخول بأكملها، لذا لا يُحرم المستخدم أبدًا من طلب رمز جديد بعد انتهاء صلاحية الرمز السابق `Since 1.8`. انظر حدّ معدّل الرموز لمرة واحدة.

ويمكن للتحقق البشري (`Since 1.8`) أن يحمي تسجيل الدخول والمصادقة متعددة العوامل وإعادة تعيين كلمة المرور. وحين تكون خدمته غير متاحة، تُرفض هذه التدفّقات، ولا يمكن تجاوزه. انظر التحقق البشري [الإدارة › التحقق البشري (الحماية من الروبوتات)].

تحديد المساحة النشطة

مساحات المستخدم

تحمّل البوابة **المساحات المتاحة** من `GET /api/v1/spaces/my`. ولا تُعاد إلا المساحات **النشطة** و**غير المعطّلة** التي يكون المستخدم عضوًا فيها.

المساحة النشطة

تختار البوابة المساحة النشطة بمطابقة مضيف المتصفح مع `` `${domainIdentifier}.${url}` `` لكل مساحة. وإن لم تجد أي تطابق، اعتمدت المساحة **الجذر**.

مبدّل المساحات

لا تُعرض قائمة المساحات في الشريط الجانبي إلا إذا كان للمستخدم أكثر من مساحة. ويُظهر كل مُدخل أيقونة المساحة واسمها. وعند اختيار مساحة، ينتقل المتصفح إلى `full_url` الخاص بها: وهو انتقال كامل إلى مضيف آخر، وليس تبديلًا داخل الصفحة، مما يستدعي إعادة فحص الجلسة على المضيف الجديد. وتُخفى القائمة حين يتصرّف المستخدم بموجب تفويض، كما تُخفى عن المستخدمين المقيَّدين.

وحين تتغيّر العضويات، يُحدَّث خبء بوّابة الـ API الخاص بالمضيفات والعضويات تلقائيًا. وإذا أُضيف المستخدم للتوّ إلى مساحة لم تظهر بعد، فالسبب في الغالب عدم تحديث الخبء بعد.

كيف تربط بوّابة الـ API الطلب بمساحة

1. **`space-resolver`** يطابق مضيف الطلب مع قائمة مضيفات المساحات المخبوءة، ويضبط `x-space-id` (وعلَم `x-space-maintenance`). وتُهمَل أولًا أي ترويسة من هذا النوع أرسلها العميل، إذ يُعتمَد المضيف وحده في تحديد المساحة. ويُرفض المضيف المجهول بالرمز `SPACE_ERROR` («Space ID not found»؛ برمز HTTP `404` منذ 1.10، و`400` قبله).
2. **`domain-resolver`** يحلّ نطاق القاطن الخاص بالمضيف.
3. **`user-space-checker`** يتأكّد من أن المستخدم المسجِّل دخوله عضو في المساحة التي جرى حلّها. وإن لم يكن عضوًا فيها، يعيد استجابة **HTTP 403** بالرمز `SPACE_FORBIDDEN` وقائمة `available_spaces`.

وتلتقط البوابة `SPACE_FORBIDDEN` وتوجّه المستخدم إلى صفحة عدم الوصول (**لا يوجد لديك إذن للوصول إلى المساحات المتاحة**)، ممرِّرةً `available_spaces` ليتمكّن من الانتقال إلى مساحة يحقّ له فتحها. وإن تبيّن عندئذ أن المستخدم غير مصادَق عليه، يُعاد توجيهه عبر تسجيل الدخول بدلًا من ذلك.

تفاصيل الإضافات: تحديد المساحة والنطاق · القاطنون والهوية.

المساحات غير المتاحة

- **معطّلة:** المساحة غير نشطة كليًا. وتُستبعد من `spaces/my` ومن عملية تحديد المساحة بحسب المضيف، فلا يمكن فتحها.
- **صيانة المستخدم النهائي:** تظلّ عملية حلّ المساحة ممكنة، لكن بوّابة الـ API تَسِم الطلب (`x-space-maintenance`)، فتظهر للمستخدمين النهائيين حالة صيانة.

ولا يمكن تعطيل المساحة **الجذر**. ويضبط المديرون كلتا الحالتين.

نموذج الاستعلام ومرجع واجهته البرمجية

هذه الصفحة موجّهة إلى **مطوّري التكامل والمطوّرين** الذين ينشئون الاستعلامات المحفوظة أو يشغّلونها عبر الواجهة البرمجية، أو يستدعون مسارات القوائم مع المرشّحات السريعة والمعاملات، أو يحتاجون إلى فهم الدلالة الدقيقة لشرطٍ ما. وتوثّق الصفحة العوامل بأسمائها في المحرّك، ومعاملات الطلب، وقواعد مجموعات الاستعلامات السريعة والوسوم، ورموز الأخطاء.

> **تبني استعلامًا من الواجهة؟**
>
> ينبغي لبُناة التطبيقات البدء من الاستعلامات والمرشّحات [بناء التطبيقات]؛ فهي تغطّي الميزات نفسها بالتسميات التي يعرضها المصمّم (فالعامل **مملوء** مثلًا هو `IsNotEmpty` هنا، والمؤشِّر **myGroups@** هو `myGroups`). أمّا صياغة تعبير الترشيح نفسه، فترد في صيغ المعايير.

الاستعلامات في التكاملات (Grafana)

يمثّل الاستعلام المحفوظ أيضًا وحدة التنفيذ في لوحة **Grafana** `Since 1.6` : إذ يستخدم مصدر بيانات Apptomator استعلامات البيانات المحفوظة لترشيح الصفوف، والاستعلامات الإحصائية المحفوظة لاستخراج القيم المجمَّعة، وتتحوّل أي معاملات يعلنها الاستعلام إلى حقول إدخال في اللوحة تقبل متغيّرات لوحة المعلومات في Grafana. راجع لوحات معلومات Grafana.

`Since 1.8` ، أصبح هذا التكامل أكثر موثوقية في جميع مراحله؛ إذ صار **المرشّح السريع** المبنيّ من استعلام ذي معاملات يستبدل معاملاته بقيمها على نحو صحيح حين تطبّقه اللوحة، بعدما كانت المعاملات تبقى دون حلّ فلا يطابق المرشّح شيئًا. كذلك يخضع كل طلب لجلب مُدخلات معرض، بما في ذلك طلبات اللوحة، لسقفٍ يفرضه الخادم (10,000 مُدخل افتراضيًا)، فلا يمكن لطلب واحد أن يبثّ مجموعة غير محدودة؛ ويعتمد مصدر بيانات Grafana بالفعل على هذا السقف، ولهذا يرسل دائمًا حدًّا صريحًا خاصًّا به لعدد الصفوف. وبالمثل، صار المبدأ نفسه، «لا تبثّ كل شيء أبدًا»، يحدّ من عدد تعريفات الكيانات التي يستطيع تكاملٌ ما جلبها دفعة واحدة عبر الواجهة البرمجية، فتُقسَّم النتائج إلى صفحات بدل إرجاع مخطّط كل كيان في استجابة واحدة.

كيف يُخزَّن الشرط

يتألّف كل شرط من ثلاثة أجزاء: **الحقل** (حقل من بنية الكيان، أو حقل من البيانات الوصفية للمُدخل مثل *أنشأه / تاريخ الإنشاء*)، و**العامل** (العلاقة بين الطرفين)، و**القيمة** (الطرف الثاني من المقارنة). وتُجمَّع الشروط وتُربط بعلاقة منطقية، **كل الشروط محققة (AND)** أو **بعض الشروط محققة (OR)**، ويمكن تداخل المجموعات لبناء منطق مركّب.

العوامل

يعبّر العامل عن العلاقة بين طرفي الشرط. وتتوقّف العوامل المتاحة على نوع الحقل: فعوامل المساواة تنطبق على كل الأنواع؛ وعوامل النص (يحتوي، ويبدأ بـ، ويشبه، ونفيها) تنطبق على الحقول النصية؛ وعوامل المقارنة (أصغر/أكبر من، أو يساوي) تنطبق على التواريخ والأرقام؛ والعاملان **في / ليس في** ينطبقان على حقول الكيانات (المرجع والتصنيف والمستخدم)؛ والعاملان **فارغ / مملوء** ينطبقان على أي حقل (راجع فارغ ومملوء يشترطان وجود الحقل). ويطابق العامل **تحت (Under)** المُدخلات التي تقع تحت سلفٍ معيّن في كيان شجري (هرمي) (**Since 1.2**)، ويطابق العامل **النوع (Type)** المُدخلات بحسب نوع القيمة المخزّنة.

وفيما يلي المرجع الكامل للعوامل، بما في ذلك أنواع الحقول التي ينطبق عليها كل عامل، وعوامل التجميع:

تدعم محرّكة الاستعلام عوامل الشروط التالية:

العامل	المعنى
`Equal` / `NotEqual`	الحقل يساوي / لا يساوي قيمة.
`LessThan` / `LessThanOrEqual`	مقارنة رقمية أو زمنية.
`GreaterThan` / `GreaterThanOrEqual`	مقارنة رقمية أو زمنية.
`In` / `NotIn`	الحقل واحد (ليس واحدًا) من قائمة قيم.
`Contains` / `DoesNotContain`	النص يحتوي (لا يحتوي) على سلسلة فرعية – غير حساس لحالة الأحرف.
`StartsWith` / `DoesNotStartWith`	النص يبدأ (لا يبدأ) بقيمة – غير حساس لحالة الأحرف.
`Like` / `IsNotLike`	مطابقة نمط على النص – غير حساسة لحالة الأحرف.
`IsEmpty` / `IsNotEmpty`	الحقل بلا قيمة – غير موجود أصلًا، أو null، أو قائمة فارغة / الحقل موجود **وله** قيمة. أما اشتراط وجود الحقل فهو `Since 1.8` .
`ElemMatch`	يطابق حقول القوائم التي يستوفي فيها عنصر واحد على الأقل الشروط المتداخلة.
`Under`	مطابقة هرمية – السجل يقع **تحت** (منحدر من) السجل المحدَّد في شجرة.
`Type`	يطابق نوع البيانات المخزَّن للحقل.
`And` / `Or`	يجمع الشروط الفرعية.

بالنسبة إلى الحقول التي تحمل **قائمة** من القيم، يمكن للشرط أن يستخدم إضافةً إلى ذلك *عامل حقل* يختزل القائمة أولًا إلى قيمة واحدة ثم يقارنها:

عامل الحقل	يختزل القائمة إلى…
`Size`	عدد العناصر.
`Sum`	مجموع العناصر.
`Avg`	متوسط العناصر.
`Max` / `Min`	أصغر / أكبر عنصر.
`First` / `Last`	العنصر الأول / الأخير.
`Any`	(الافتراضي) يطابق إذا استوفى أي عنصر الشرط.

فارغ ومملوء يشترطان وجود الحقل

`Since 1.8` – لم يعد العاملان **فارغ** (`IsEmpty`) و**مملوء** (`IsNotEmpty`) يكتفيان باختبار ما إذا كانت القيمة المخزّنة null أو قائمة فارغة؛ بل صار المُسنَد يختبر أيضًا **وجود** الحقل: فلا يطابق *مملوء* إلّا حين يكون الحقل موجودًا في المُدخل *ويحمل* قيمة فعلية، في حين يطابق *فارغ* الحقلَ الغائب كما يطابق قيمة null أو القائمة الفارغة. ويبني كلٌّ من محرّك استعلام البيانات ومحرّك استعلام التفاعلات هذين العاملين باستخدام الدالة البانية المشتركة نفسها، فيتّفق الاستعلام المحفوظ وشرط التفاعل دائمًا على معنى «فارغ».

ويعالج هذا التغيير حالة **حقل مرجع داخل كتلة قابلة للتكرار**. فعندما لا تكون للكتلة أي نسخة، يُخزَّن المرجع في صورة مجموعة فارغة؛ ولأنّ الشرط يُصاغ باستخدام المسار النقطي للحقل، كان اجتياز تلك المجموعة الفارغة يُنجح اختبار «لا يساوي null»، فيجتاز المُدخل شرط *مملوء* خطأً رغم خلوّ كتلته القابلة للتكرار من أي صف. واشتراط وجود الحقل يستبعد تلك المُدخلات.

وقد سدّ إصلاحٌ لاحق الثغرة نفسها في موضعين يُحلّ فيهما الاستعلام عبر مسار داخلي مختلف: شرط **تفاعل** يختبر *مملوء* على النوع نفسه من الحقول، و**حقل مرجع** يستهدف استعلامُ مصدره كيانًا فيه كتلة قابلة للتكرار. وقبل الإصلاح، كان المُسنَد على ذلك المسار لا يطابق شيئًا بصمت، فلا يُطلَق التفاعل أبدًا، وتظهر قائمة الاختيار في حقل المرجع فارغة، رغم صحّة الشرط نفسه وتطبيق الإصلاح السابق بالفعل.

قيد مُعلَن: لا يمكن قصر الشرط على نسخة واحدة من الكتلة القابلة للتكرار. يتلقّى هذان العاملان مسار حقل، لا نسخة من كتلة، فلا سبيل إلى التعبير عن «نسخة واحدة على الأقل من الكتلة يكون فيها هذا الحقل مملوءًا». ولذلك تُعدّ الكتلة التي تجمع نسخة فارغة ونسخة مملوءة **فارغةً** في مجملها. وهذا **سلوك قائم من قبل، لم يستحدثه تغيير 1.8**؛ فراعِه عند تصميم شرط على كتلة قابلة للتكرار، واستخدم حقل علامة منفصلًا في المُدخل إن احتجت إلى دلالة على مستوى النسخة.

مصادر القيم

القيمة هي الطرف الثاني من الشرط، أي ما يُبحث عنه. ويمكن أن تكون:

- **ثابتة**: قيمة محدّدة تُكتب يدويًّا وتبقى دون تغيير.
- **متغيّرة (ديناميكية)**: تُؤخذ من حقل آخر في المُدخل نفسه أو في كيان مرتبط، فتصبح المقارنة بين حقلين (مثل *مُسند إلى* يساوي *البيانات الوصفية ← عدّله*).
- **مؤشّر**: قيمة سياقية تجعل الاستعلام مرتبطًا بالسياق: يُحلّ `@me` إلى المستخدم الحالي؛ وتجعل رموز العضوية الاستعلامَ مرتبطًا بعضويات المستخدم الحالي: `myGroups` (المجموعات التي ينتمي إليها المستخدم) و`myRoles` (أدوار المستخدم) و`myOrgUnit` (الوحدة التنظيمية للمستخدم)؛ وتقارن مؤشّرات التاريخ مثل `@today` حقلَ تاريخ بالتاريخ الحالي وقت تشغيل الاستعلام.
- **تعبير رمازي**: تعبير JavaScript يُستخدم للشروط التي لا تستطيع أنواع القيم الأخرى التعبير عنها (ويمنح المديرين مرونة إضافية).
- **معامل**: عنصر نائب تُقدَّم قيمته لاحقًا عند استخدام الاستعلام (راجع الاستعلامات ذات المعاملات).

وتحمل هذه المصادر في المنشئ التسميات **ثابت** و**حقل** و**مؤشر** و**تعبير برمجي** و**معامل**، وتُسرد المؤشِّرات بإدائتها `@` (`@me` و`@myGroups` و`@myRoles` و`@myOrgUnit` و`@today` و`@now` و`@StartOfDay` و`@StartOfWeek` و`@StartOfMonth` و`@StartOfYear`، إضافةً إلى مؤشِّرات أخرى لعضوية المجموعات).

العلاقات والحقول الافتراضية والبيانات الوصفية لسير العمل

في كيان `Task` يضمّ مرجع `department` إلى كيان `Department`، مثلًا، يمكنك ترشيح المهام التي يتحقّق فيها الشرط **"department → region = West"** دون مغادرة استعلام `Task`. إذ يتتبّع المحرّك العلاقة داخليًّا (علامة المسار `:relation`) ويضمّ المُدخل المرتبط ليُقيَّم الشرط عليه. ويمكن اجتياز العلاقات لأكثر من مستوى واحد، لكنّ كل مستوى يضيف عملية ضمّ، لذا أبقِ سلاسل العلاقات قصيرة قدر الإمكان.

وإذا عرّف الكيان **حقولًا افتراضية (محسوبة)**، أي حقولًا تُستمدّ قيمها من تعبير بدل أن تُخزَّن، فيمكنك استخدامها في الترشيح والترتيب كأي حقل آخر؛ إذ يحسب المحرّك الحقل الافتراضي أولًا، ثم يُطبَّق الشرط على النتيجة المحسوبة.

وإذا تعذّر تقييم تعبير حقل افتراضي، يواصل الاستعلام إرجاع النتائج مع عرض تحذير (ويُتخطّى ذلك الحقل فحسب)، فلا يعطّل حقلٌ محسوب معطوب القائمةَ كلها.

وفي الكيانات التي لها سير عمل، يمكنك إضافة شروط على **البيانات الوصفية لسير العمل**: الخطوة الحالية، أو مالك الخطوة الحالية، أو الخطوات السابقة، أو مالكي الخطوات السابقة، بما في ذلك رصد الخطوات العارضة. وتظهر هذه الخيارات حين

يسمح المعرض أو الاستعلام بالترشيح على الخطوة الحالية / الخطوات السابقة.

عوامل الحقل (التجميعية)

في الحقول التي تحمل قيمًا متعددة (الحقول متعددة القيم، أو الحقول داخل كتلة قابلة للتكرار)، يوفّر المنشئ **عوامل تجميعية** تختزل مجموعة القيم *قبل* المقارنة: **Size** (عدد القيم)، و**Sum**، و**Average**، و**Max**، و**Min**، و**First**، و**Last**. فالشرط «**Size** الكتلة أكبر من 3» مثلًا يُعيد المُدخلات التي تضمّ كتلتها القابلة للتكرار أكثر من ثلاثة صفوف. وتظهر هذه العوامل في مرجع العوامل أعلاه (وفي المنشئ: **عدد القيم**، و**جمع القيم**، و**المتوسط**، و**القيمة العليا**، و**القيمة الدنيا**، و**القيمة الأولى**، و**القيمة الأخيرة**، و**أي قيمة** للسلوك الافتراضي).

حين لا يُختار عامل حقل

يُقيَّم الشرط الذي لا يستخدم عاملًا تجميعيًّا وفق السلوك الافتراضي **Any**، أي يتحقّق الشرط حين تستوفيه أيٌّ من قيم الحقل. و`Since 1.8` صار المحرّك يتعرّف على نحو موثوق إلى الشرط المحفوظ دون تحديد عامل تجميعي بوصفه حالة «لم يُختر عامل»، فيُحيله إلى ذلك السلوك الافتراضي بدل أن يرفضه بوصفه عاملًا غير مدعوم. أمّا اسم العامل الذي لا يعرفه المحرّك فعلًا، فما زال يُرفض بخطأ.

ويؤثّر ذلك أيضًا في **شروط التفاعلات**، إذ يقيّمها المحرّك نفسه: فقد كان الشرط الخالي من عامل تجميعي يُعامَل على أنّه «غير متحقّق»، فيظلّ التفاعل صامتًا لا يُطلَق أبدًا.

الحفظ والتسمية

بعد بناء الشروط، يمكنك **حفظ الاستعلام** لإعادة استخدامه في المعارض والتقارير وحقول المرجع ولوحات المعلومات ومواضع أخرى. و`Since 1.10` صارت نافذة الإنشاء / التعديل في تبويب الاستعلامات **تحفظ دائمًا**: فقد أُزيل منها مفتاح «حفظ الاستعلام» ونمط التطبيق دون حفظ، وصار زر التأكيد دائمًا *حفظ*. ولتجربة شروط دون حفظها، ابنِها في لوحة **الترشيح** في قائمة المُدخلات؛ وما زالت الاستعلامات المحفوظة تُطبَّق من قائمة الاستعلامات.

يتراوح طول الاسم متعدد اللغات للاستعلام المحفوظ بين 3 و128 محرفًا. و`Since 1.10` يولّد الخادمُ `normalizedName` للاستعلام، ويردّ عند التعارض بالرمز **409** `DATA_QUERY_NORMALIZED_NAME_DUPLICATE` بدل إرجاع خطأ في الخادم.

الاستعلامات ذات المعاملات

- عند إنشاء استعلام أو تعديله، اختر **معامل** مصدرًا للقيمة، وامنح المعامل اسمًا.
- يمكن إعادة استخدام اسم المعامل نفسه في شروط على حقول من نوع البيانات **نفسه**، ولا يمكن إعادة استخدامه في حقول من أنواع بيانات **مختلفة**.
- عند حفظ استعلام يحتوي على معاملات، يُصنَّف **استعلامًا ذا معاملات**، ويظهر بوسم مخصّص *استعلام بمعامل* في قائمة الاستعلامات. و`Since 1.10` تميّز شارةٌ مشتركة **معامل** هذه الاستعلامات في كل موضع يختار فيه المدير استعلامًا: تبويب الاستعلامات، ومنتقيات مصدر المُدخلات للمرجع / العلاقة / المستخدمين، ومحدِّدات استعلام مصدر المعرض واستعلام الخلاصة / لوحة المعلومات، وقائمة الاستعلامات المحفوظة في المعرض العام، وقوائم الاستعلامات السريعة في الإدارة. أمّا في قائمة الحالة المنسدلة وعلى مفاتيح المرشّحات السريعة، فلا تظهر إلّا في تبويب المُدخلات في الإدارة، ولا تظهر أبدًا في قائمة الاستعلامات التي يطبّقها المستخدم النهائي.
- يمكن استخدام الاستعلام ذي المعاملات مصدرًا في مواضع أخرى، مثل **حقول المرجع** أو **المعارض**، حيث تُقدَّم قيم المعاملات في إعدادات تلك الميزة.

وإذا أُزيلت كل الشروط ذات المعاملات (أو حُوّلت إلى نوع قيمة آخر)، يعود الاستعلام استعلامًا عاديًّا. أمّا العكس فممنوع: **لا يمكن** تحويل استعلام عادي (بلا معاملات) إلى استعلام ذي معاملات عند التحديث.

الاستعلامات السريعة ذات المعاملات: قيم لكل استعلام `Since 1.10`

حتى الإصدار 1.8، كان المرشّح السريع ذو المعاملات يأخذ قيمه من كائن `parameters` مسطّح واحد في الطلب (مسار Grafana)، فلا يستطيع استعلامان سريعان ذوا معاملات مطبَّقان معًا استخدام قيمتين مختلفتين لمعامل يرد في كليهما. ويمنح الإصدار 1.10 كل استعلام سريع مطبَّق قيمه الخاصة، ويتيح للمستخدمين النهائيين طريقة لتقديمها.

- **ما يراه المستخدمون النهائيون.** عند تطبيق استعلام ذي معاملات من قائمة الحالة أو مفتاح مرشّح سريع مستقل أو قائمة الاستعلامات المحفوظة أو لوحة الترشيح، تُفتح نافذة **معاملات الاستعلام** التي تطلب قيمة كل معامل معلَن. ثم يُشغَّل الاستعلام **نسخةً** تحوّلت فيها شروط المعامل إلى شروط ثابتة؛ وتعرضها لوحة الترشيح بوصفها شروطًا قابلة للتعديل، ويؤدّي حفظ تلك النسخة إلى إنشاء استعلام جديد بدل الكتابة فوق الاستعلام ذي المعاملات.
- **معامل الطلب.** `quickFilterParameters` كائن مفاتيحه معرّفات الاستعلامات: `{"<queryId>": {"<parameterName>": value}}`. وتقبله مسارات المُدخلات والشجرة والخلاصة والعدّ (على مستوى الكيان وعلى مستوى المعرض)، ومسارا لوحة المهام (Kanban) وقيم أعمدتها، وكلا مساري التصدير السريع (معارض البيانات والواجهات [بناء التطبيقات ‹ معارض البيانات وأوجهها]).
- **الحلّ.** يُحدَّد ما إذا كان الاستعلام السريع ذا معاملات **بعدّ شروط المعامل في معاييره**، لا براية `isParameterized` المخزّنة ولا بمصفوفة `parameters`، إذ قد تكون أيٌّ منهما غير محدَّثة. وفي هذا النوع من الاستعلامات، تكون `quickFilterParameters[queryId]` الإجابةَ **الكاملة** متى وُجدت (ولا تُدمج مع أي شيء آخر)؛ وإلّا استُخدم كائن `parameters` المسطّح، فتظلّ لوحات Grafana القائمة تعمل. وتُطابَق الأسماء دون تمييز حالة الأحرف (`Dept` = `dept`)؛ وتُحوَّل المعاملات المعلَنة إلى نوعها المعلَن، أمّا الاسم المستخدم في المعايير دون إعلان، فيأخذ القيمة الخام.
- **العزل.** تُستبدل معاملات كل استعلام سريع بقيمها في نسخة من معاييره الخاصة، فلا يمكن أن تتسرّب قيم مرشّح سريع إلى مرشّح آخر ولا إلى المعاملات المحلولة لمصدر المعرض.
- **القيم الناقصة** تُفشل الطلب بالرمز HTTP **422** `QUICK_FILTER_PARAMETER_VALUE_MISSING` (`details: {query,parameter}`)؛ وتُعدّ `null` قيمة ناقصة.
- **الحقول الحسّاسة.** تُفحص المعايير بعد إدراج قيم المعاملات فيها مقابل منح الحقول الحسّاسة الخاصة بالمستدعي، كأي معايير أخرى، فلا يستطيع مستدعٍ لا يملك المنحة أن يغيّر قيمة معامل على حقل حسّاس ليتحرّى قيمته المخفية.
- **لا تطبيق افتراضيًّا أبدًا.** لا يستطيع المعرض أن يجعل استعلامًا سريعًا ذا معاملات مطبَّقًا افتراضيًّا؛ راجع المرشّحات السريعة ومجموعاتها وترتيبها [بناء التطبيقات ‹ معارض البيانات وأوجهها].

ولا يقبل التصدير عبر القالب قيمَ المعاملات، لذا يستبعد المرشّحات السريعة ذات المعاملات ويُبلغ المستخدم بذلك؛ أمّا التصدير السريع فيطبّقها بقيمها.

الاستعلامات السريعة

يُدرج الاستعلام المحفوظ الذي حُدّد له خيار **استعلام سريع** (`useInQuickFilter`) ضمن **استعلام سريع** في تبويب الاستعلامات. ولتطبيقه، افتح قائمة المُدخلات، وانتقل إلى القائمة المنسدلة **الحالة** (التي تضمّ خيارات الترشيح الإضافية)، وفعّل الاستعلام السريع؛ فتُضاف شروطه إلى المرشّحات المطبّقة حاليًّا وتُربط بها بـ **و**. ثم تظهر شروط الترشيح معطّلةً في قائمة المرشّحات المخصّصة.

و`Since 1.10` يضمّ قسم الاستعلام السريع في النموذج أيضًا:

- **الاسم** (اسم الاستعلام السريع): التسمية المترجَمة التي يراها المستخدمون النهائيون لهذا الاستعلام بوصفه مرشّحًا سريعًا. وإن تُرك فارغًا، يُملأ من اسم الاستعلام عند الحفظ؛ وبعد ذلك يستقلّ كلٌّ منهما عن الآخر.
- **المجموعة**: مجموعة الاستعلامات السريعة التي ينتمي إليها الاستعلام (أو لا شيء). ويمكنك إنشاء مجموعة في الموضع نفسه عبر **مجموعة جديدة**.

أمّا موضع ظهور الاستعلام السريع في المعرض، سواء داخل قائمة الحالة أو بوصفه مفتاحًا مستقلًّا، وما إذا كان يُطبَّق افتراضيًّا، فيُحدَّد **لكل معرض** لا على مستوى الاستعلام؛ راجع المرشّحات السريعة ومجموعاتها وترتيبها [بناء التطبيقات › معارض البيانات وأوجهها]. ولا يصل المستخدمون النهائيون إلى استعلام سريع إلّا عبر معرض يسرده (أو يفعّل مجموعته).

مجموعات الاستعلامات السريعة `Since 1.10`

تصنّف **مجموعة الاستعلامات السريعة** الاستعلاماتِ السريعة الإدارية لكيانٍ ما، مثل *حسب المنطقة* التي تضمّ *الشمال* و*الجنوب* و*الغرب*. وتُدار المجموعات في تبويب **الاستعلامات** للكيان، ضمن التبويب الفرعي **مجموعات الاستعلامات السريعة**: أنشئ مجموعة، وسمّها بكل لغة، ثم استخدم **إضافة استعلام سريع موجود** أو **إنشاء استعلام سريع** داخلها. وبعد ذلك يفعّل المعرض المجموعة كلها دفعة واحدة.

- **العضوية مخزّنة على الاستعلام** (`quickFilterGroup`، وهو معرّف مجموعة أو `null`)، لذا ينتمي الاستعلام إلى مجموعة **واحدة على الأكثر**، ولا يتطلّب حذفه أي تنظيف في المجموعة. وإضافة استعلام ينتمي إلى مجموعة أخرى تنقله إلى المجموعة الجديدة (وتنبّه النافذة بعبارة *سيُنقل من …*).
- لا يمكن تجميع إلّا **الاستعلامات السريعة الإدارية** للكيان. أمّا الاستعلام غير السريع، أو الاستعلام الذي حفظه مستخدم نهائي لنفسه، فيُخزَّن دائمًا بلا مجموعة.
- **الأسماء فريدة داخل الكيان الواحد**، وتُقارَن لكل لغة بعد حذف المسافات الطرفية ودون تمييز حالة الأحرف (`G1` و`g1` تتعارض)؛ ورمز الخطأ هو `QUICK_FILTER_GROUP_NAME_DUPLICATE`.
- **حذف المجموعة يُبقي استعلاماتها السريعة**؛ فتصبح بلا مجموعة.

المسار	الغرض
`GET` / `POST /v1/entities/{entity}/quick-filter-groups`	السرد / الإنشاء.
`PATCH` / `DELETE /v1/entities/{entity}/quick-filter-groups/{id}`	التحديث / الحذف (يُخرج الحذفُ الأعضاءَ من المجموعة ثم يحذفها).
`GET /v1/quick-filter-groups[/{id}]`	المجموعات الخام، للتحزيم وتتبّع الاعتماديات.

وتُعيد عمليتا الإنشاء والتحديث المجموعةَ مع معرّفات الاستعلامات الأعضاء فيها. وتضبط مصفوفة `queries` في طلب الكتابة العضوية **بدقّة**: تُضاف الاستعلامات المسرودة، ويُزال الأعضاء الحاليون غير المسرودين، ولا يجوز أن تتضمّن المصفوفة إلّا الاستعلامات السريعة الإدارية للكيان (`QUICK_FILTER_GROUP_INVALID_QUERY`). وتحمل استجابة الاستعلام الحقل `quickFilterGroupName` للقراءة فقط؛ ويُرفض جسم الكتابة الذي يتضمّنه بالرمز 422. وعند تحديد مجموعة تابعة لكيان آخر، يُرجَع الرمز 404 (`QUICK_FILTER_GROUP_NOT_FOUND`). ولا يغيّر نقل استعلام بين المجموعات `updateIdentifier` الخاص به. وثمّة قيد معروف: المجموعة المُنشأة بعد لقطة تصدير حزمة لا تُدرَج في رسم الاعتماديات إلّا عند فحص إعادة البناء / المزامنة التالي.

الوسوم

اتّبع الخطوات نفسها، وقبل الحفظ فعّل مفتاح **حفظ الوسم**. أدخل اسم الوسم متعدد اللغات (من 3 إلى 128 محرفًا)، واختر لونه، ثم اختر **مجموعة الوسوم** من المجموعات الموجودة، أو فعّل **مجموعة وسوم جديدة** وسمّها. عندها يُحفظ الاستعلام

باسم وسمه إلى جانب وسم *استعلام عادي*. وعند تطبيقه من قائمة المُدخلات (عبر قائمة **الوسم**، بوصفه **إطارًا ملوّنًا** أو **خلفية ملوّنة**)، يُطبَّق لون الوسم على المُدخلات المطابقة للاستعلام.

تغيير وسم الاستعلام عبر الواجهة البرمجية

`Since 1.10` — صار `PATCH /api/v1/entities/{entity}/queries/{id}` يقبل بنية الوسم نفسها التي كان يقبلها إنشاء الاستعلام، ويُعيد الاستعلام بعد تحديثه.

ويظهر الأثر العملي لهذا التغيير في `tag.tagGroupName`؛ فقد كان إرساله في التحديث **يُسقَط بصمت**، بلا خطأ وبلا أثر، لأنّ مخطّط طلب التحديث لم يكن يتضمّن هذه الخاصية. أمّا الآن، فيُعامَل كما في الإنشاء: فتحديد مجموعة غير موجودة **ينشئ مجموعة الوسوم في الحال** ويربط الوسم بها. ولأنّ الاستجابة صارت تُعيد الاستعلام المحدَّث بعد حلّ مجموعة الوسوم، فإنّ العميل الذي أنشأ مجموعة في الحال **يعرف معرّفها من الاستجابة نفسها** بدل إجراء طلب قراءة لاحق. ولم يتغيّر رمز الحالة؛ وإنّما صارت الاستجابة `200` تتضمّن جسمًا.

وفيما يلي قواعد الوسم عند التحديث كاملةً:

ما ترسله في `tag`	ما يحدث
تُغفله	يبقى الوسم المخزّن كما هو، فيظلّ التحديث جزئيًّا فعلًا.
`null`، أو وسم فيه `enabled: false`	يُحذف الوسم كليًّا، ومعه إشارته إلى المجموعة.
وسم يحمل `tagGroupName` جديدًا	تُنشأ مجموعة الوسوم في الحال ويُربط الوسم بمعرّفها.
وسم يحمل معرّف `tagGroup` غير موجود	يفشل الطلب: فالمجموعة غير المعروفة تُعدّ خطأ، ولا يُتجاهل الطلب بصمت.

و`tagGroupName` خاصية لتسهيل الطلب فقط: لا تُخزَّن أبدًا على الاستعلام نفسه، الذي يحتفظ بـ **معرّف** المجموعة. ويُحسب سجلّ التدقيق للتحديث بناءً على الحالة النهائية للوسم، فلم يعد إغفال `tag` يُسجَّل حذفًا لم يقع.

حذف الاستعلام

افتح تبويب **الاستعلامات**، واختر الاستعلام، وانقر أيقونة الحذف بجانب اسمه، ثم أكّد؛ والحذف نهائي. ولا يمكن حذف بعض الاستعلامات: الاستعلامات **الضمنية** (التي يولّدها النظام)، وأي استعلام يستخدمه حاليًّا كيان وهمي أو كيان موصِّل (Joiner) أو كيان مجمِّع (Aggregator).

قائمة الاستعلامات `Since 1.10`

لتبويب **الاستعلامات** في الكيان تبويبان فرعيان: *الاستعلامات* و*مجموعات الاستعلامات السريعة*. وتضمّ قائمة الاستعلامات شريط الأدوات نفسه الموجود في تبويب التفاعلات:

- **مرشحات**: مرشّحات بمربعات اختيار حسب **نوع** الاستعلام و**وسمه** و**حقله**، إضافةً إلى مرشّح للبيانات الوصفية (أنشأه / حدّثه، ونطاقا تاريخ الإنشاء / التحديث).
- **الترتيب**: حتى ثلاثة مفاتيح من بين *آخر تعديل* (الافتراضي، الأحدث أولًا) و*تاريخ الإنشاء* و*الاسم*.
- **التجميع حسب**: *النوع* (الافتراضي) أو *الوسم* أو *بلا*.
- **البحث**: في اسم الاستعلام **ونصّ شروطه**.

تحتفظ القائمة بهذه الحالة لدى العميل؛ ولم يتغيّر `GET /v1/entities/{entity}/queries`. وبمعزل عن ذلك، صارت عملية قراءة الاستعلام تُلحق تسمية `_resolved` بكل **قيمة مرجع ثابتة** في معاييره، فتعرض لوحة الترشيح اسم المُدخل المشار إليه بدل معرّفه؛ ثم يُزال `_resolved` مجدّدًا عند الكتابة، فلا حاجة إلى إرساله أبدًا.

سلوكيات قائمة المُدخلات التي يعتمد عليها الاستعلام

في قائمة المُدخلات، يمكنك كتابة قيمة في **مربع البحث** لعرض كل مُدخل مطابق عبر جميع الحقول القابلة للبحث (النص والرقم والبريد الإلكتروني وغيرها)، دون شروط صريحة. ويجب السماح بالبحث في الحقل ليُشمل في البحث: إمّا بتفعيل البحث عند إنشاء **معرض** (فيقتصر البحث حينئذٍ على الحقول المضافة إلى المعرض)، وإمّا بتفعيل خيار البحث على الحقل نفسه من إعداداته.

تُعرض مُدخلات الكيان افتراضيًّا في جدول مرتّبة **تنازليًّا حسب تاريخ التعديل**. ويمكنك تغيير ذلك بطريقتين:

- **القائمة المنسدلة للترتيب** (بجانب مربع البحث): اختر **الاتجاه** و**حقل الترتيب**. يرتّب الاتجاه التصاعدي النصوص و*أنشأه* / *حدّثه* أبجديًّا (0-9، ثم A-Z، ثم أ-ي)، والتواريخ من الأقدم إلى الأحدث؛ ويعكس الاتجاه التنازلي كلا الترتيبين. ويمكنك الترتيب حسب *أنشأه* أو *تاريخ الإنشاء* أو *حدّثه* أو *تاريخ التعديل* أو أي حقل من حقول الكيان.
- **أيقونة الترتيب في رأس الحقل**: انقر أيقونة الترتيب في رأس عمود الحقل لتبديل ترتيب ذلك الحقل بين التصاعدي والتنازلي.

وتحصر المرشّحات الإضافية المُدخلات حسب **حالتها**: نشط، أو مؤرشف، أو بالٍ، أو غير بالٍ. ويمكن الوصول إليها من القائمة المنسدلة بجانب المرشّح، وهي متاحة للمديرين، وللمستخدمين النهائيين عبر المعارض عند السماح بها وتفعيلها. فعّل الحالات التي تريدها وعطّل البقية؛ ويمكنك الجمع بين الحالات (مثل *مؤرشف* و*بالٍ* معًا). وتشمل حالات المُدخل: **بالٍ** (لم يعد مطابقًا لبنية الكيان أو سير عمله الحاليين)، و**صالح** (مطابق للبنية وسير العمل الحاليين)، و**حالي** (موجود بأي حالة)، و**مؤرشف** (نُقل إلى الأرشيف).

الاستعلامات الإحصائية

بينما يُعيد الاستعلام العادي **قائمة** من المُدخلات، يجمّع **الاستعلام الإحصائي** المُدخلات في **رقم أو مخطّط أو قائمة**، مثل «عدد الطلبات حسب القسم». ويستخدم منشئ الترشيح نفسه، مضافًا إليه **دالة التجميع** (العدّ، والمجموع، والأصغر، والأكبر، والمتوسط)، وحقل **التجميع حسب** (ويمكن تقسيم التواريخ إلى فئات حسب اليوم أو الأسبوع أو الشهر أو الربع أو السنة وغيرها؛ وتُحلّ قيم التجميع لحقول المرجع والتصنيف إلى تسمياتها المعروضة تلقائيًّا)، وخيار **العرض بشكل** (رقم أو مخطّط أو قائمة)، واتجاه **الترتيب** (تصاعدي أو تنازلي على النتيجة المجمَّعة). وتستبعد الاستعلامات الإحصائية دائمًا المُدخلات المؤرشفة والبالية؛ أمّا التجميع على حقول ذات عدد كبير جدًّا من القيم المتمايزة، فيستخدم حدًّا لعدد الشرائح مع فئة «قيم أخرى». وتشكّل هذه الاستعلامات أساس لوحات المعلومات وقطع الخلاصات. والاستعلام الإحصائي موردٌ مختلف عن **تجميعة الكيان**، التي تجمع عمودًا من قائمة المُدخلات في موضعه بدل أن تُنتج قطعة؛ راجع تجميعات الكيان وتجميع المُدخلات [بناء التطبيقات › المجاميع والتجميع].

ويرد توثيق نموذج الاستعلام الإحصائي ومساراته في مرجع واجهة الخلاصات ولوحات المعلومات.

العلاقات والمراجع وحقول الربط: النموذج والواجهة البرمجية

هذه الصفحة موجّهة إلى **مطوّري التكامل والمطوّرين** الذين يقرؤون تعريفات العلاقات أو يكتبونها، أو يتنقّلون بين المُدخلات المرتبطة عبر الواجهة البرمجية، أو يحتاجون إلى معرفة القواعد الدقيقة التي تحكم حقول المرجع وحقول الربط وروابط التنقل.

تربط الكيانات من الواجهة؟

ينبغي لبُناة التطبيقات البدء بصفحة ربط الكيانات [بناء التطبيقات]؛ فهي تغطّي حقول المرجع، وتبويب العلاقات، وقوائم الاختيار المتتالية، وحقول الربط، مستخدمةً التسميات التي يعرضها المصمّم.

تغطّي هذه الصفحة الآليات المختلفة التي يستخدمها أسس لربط البيانات بين الكيانات. ومع أنها آليات متمايزة، فقد يسهُل الخلط بينها، وتتولّاها جميعًا خدمة **data-service**:

1. **علاقات البيانات** (`DataRelation`) – *تعريفات* قائمة بذاتها وقابلة للاستعلام، تصف كيفية ارتباط كيان بآخر. وتستند إليها تبويبات «المُدخلات المرتبطة» والقوائم الفرعية وكيان الموصِّل (**Joiner**) والتفاعلات التي مصدرها علاقة.
2. **حقول المرجع** – حقول من النوع `Entity` في الكيان، تحمل الإشارة الفعلية إلى المُدخل المُشار إليه (إعداد `source` / `sources`).
3. **حقول الربط** – حقول تُنسَخ قيمتها أو تُزامَن من مُدخل مُشار إليه، وفق سياسة وضع التحديث.

كما توثّق الصفحة **روابط التنقل** (`LinksService`) – أي شجرة روابط الشريط الجانبي وروابط النظام القابلة للمشاركة – لاشتراكها مع هذه الآليات في مصطلحات «الروابط والمراجع».

تُعرَّف جميع نقاط النهاية الواردة أدناه، والمقيّدة بكيان أو مُدخل، على `DataRelationController` ، الذي يشترط استراتيجية المصادقة `Auth2` ويشغّل مُصوِّت التخويل `entityVoter` في كل عملية.

خريطة المفاهيم

المفهوم	موضعه	ما الذي يخزّنه	الاتجاه
علاقة بيانات (واردة)	مستند `DataRelation`	تعريف صريح يملكه `dataEntity` ، يشير إلى `targetField` على `targetEntity`	واردة (`relationType` = `1`)
علاقة بيانات (صادرة)	*تُشتقّ وقت القراءة* من حقول المرجع في الكيان نفسه	اصطناعية، ولا تُحفَظ أبدًا	صادرة (`relationType` = `0`)
حقل مرجع	`DataEntityField` بقيمة `dataType = Entity` ومعه `source` / `sources`	الكيان المُشار إليه + الاستعلام المستخدَم لاختيار مُدخل	لا ينطبق (على مستوى الحقل)
حقل ربط	`DataEntity.dataSources[]` + `mapping` على مستوى الحقل	نسخ قيمة `fromFieldId → toFieldId` مع `updateMode`	لا ينطبق (مزامنة قيمة)
رابط تنقل	`LinkDataEntry` (كيان الروابط)	عقدة في الشريط الجانبي تشير إلى وحدة/معرض/صفحة/تقرير	لا ينطبق

كيف تُربط الكيانات

قبل الخوض في تفاصيل النموذج، من المفيد التعرّف إلى الطريقتين المعتادتين اللتين يستخدمهما المدير لربط الكيانات في المنتج:

1. **حقل المرجع** – حقل من نوع البيانات **Entity** في الكيان (أ) يشير إلى مُدخلات الكيان (ب). ويُخزَّن المرجع داخليًا في صورة إشارة موحّدة من النوع `ReferenceProperty`، تتكوّن من الزوج `{ dataEntity, dataEntry }` (معرّف الكيان + معرّف المُدخل)، وغالبًا ما ترافقها كتلة `_resolved` تحمل عنوان عرض المُدخل المرتبط. ويُبقي تخزين الإشارة بدلًا من النص الصريح الرابط ثابتًا، حتى لو تغيّر عنوان عرض المُدخل الهدف لاحقًا.
2. **العلاقة (Data Relation)** – علاقة مسمّاة قائمة بذاتها بين كيانين (واردة/صادرة)، تُدار من تبويب **العلاقات** داخل الكيان. تتيح العلاقات تصفّح المُدخلات المرتبطة، وعرضها في قوائم فرعية، وبناء الكيانات الوهمية من نوع **الموصِّل (Joiner)**، وتشغيل تفاعلات مصدرها علاقة.

وتتيح علاقات البيانات أيضًا بناء **بنى تنظيمية متداخلة وهرمية**: فيمكن لكيان يمثّل مؤسسة أن يكون أبًا لكيانات فرعية تمثّل أقسامها أو مشاريعها، بحيث تُنظَّم البيانات وفق التبعية الهرمية (أب-ابن) بينها.

نموذج `DataRelation`

`DataRelation` مستند في data-service، يمتدّ من `Identified<UUID>` ويُنفّذ الواجهة المشتركة `IDataRelation`.

الحقل	النوع	مطلوب	ملاحظات
`id`	UUID	(الخادم)	المفتاح الأساسي.
`specificationVersion`	number	نعم	القيمة الافتراضية `1`.
`dataEntity`	UUID	نعم	الكيان الذي **يملك** تعريف هذه العلاقة.
`name`	LocalizedValue	نعم	الاسم المعروض؛ يُتحقَّق منه عبر `LanguagesService.validateName`.
`description`	NullableLocalizedValue	لا	
`icon`	Icon	نعم	
`targetEntity`	UUID	نعم	الكيان في الطرف الآخر.
`targetField`	string	نعم	`fullNormalizedName` لحقل المرجع الذي يُنشئ الرابط (راجع ما يلي لمعرفة الطرف الذي يقع فيه).
`relationType`	`DataRelationTypeEnum`	نعم	`Outgoing = 0`، `Incoming = 1`.
`normalizedName`	string	(تلقائي)	الاسم النموذجي، وهو فريد ضمن كل `dataEntity`. يُتحقَّق من نمطه (`NormalizedNameValidation`). **Since 1.6**
`isImplicit`	boolean	لا	القيمة الافتراضية `false`. العلاقات الضمنية للقراءة فقط (انظر أدناه).
`updateIdentifier`	string	نعم	رمز التزامن المتفائل (UUID). **Since 1.6**

جسم طلب الإنشاء `NewDataRelation` **يستثني** `id` و`updateIdentifier` (يُسنِد الخادم قيمتيهما)؛ أما جسم طلب PATCH فيستثني `id` فقط.

الواردة مقابل الصادرة

هذا هو التمييز الأهم الذي ينبغي فهمه:

- **العلاقات الواردة هي مستندات `DataRelation` المخزَّنة.** فهي تصف المُدخلات في `targetEntity` التي تشير *إلى* `dataEntity` عبر حقل مرجع يحدّده `targetField` (ويقع ذلك الحقل على `targetEntity`). وفي المنتج، يكون الكيان الحالي هو الكيان «الأب» في العلاقة الواردة، ويعرّفها المدير ويعدّلها من تبويب العلاقات.
- **العلاقات الصادرة تُحسَب لحظياً** من حقول المرجع في `dataEntity` نفسه (`getEntityRelationFieldsWithoutADT()`)، ولا تُحفَظ أبدًا. فكل حقل مرجع في الكيان يُنتِج علاقة صادرة؛ ويكون معرّفها الاصطناعي `id` مفتاحًا مركّبًا (انظر أدناه)، وتكون قيمة `targetField` فيها هي `fullNormalizedName` لحقل على `dataEntity` نفسه. وفي المنتج، يكون الكيان الحالي هو «الابن» في العلاقة الصادرة.

يعكس `transformToRelationDto` هذا الاختلاف: ففي حالة العلاقة **الواردة**، يحلّ مرجع الحقل على الكيان *المرتبط* (الهدف)؛ وفي حالة العلاقة **الصادرة**، يحلّ مرجع الحقل على الكيان *المالك*.

وتُرجِع نقاط نهاية السرد والعدّ في V2 كلتا المجموعتين؛ فمثلًا، يبني `findV2` مجموعة `outgoingRelations` بالمرور على `sources[]` لكل حقل علاقة (أو `source` المفرد القديم)، ويُنتِج `DataRelation` اصطناعيًا واحدًا لكل مصدر، بمعرّف `id = "{dataEntity.id}:{field.id}:{source.entity}"`.

إدارة العلاقات من تبويب العلاقات

في المنتج، تُعرَض العلاقات الواردة (inbound) والصادرة (outbound) معًا في تبويب **العلاقات** الخاص بالكيان، وتظهر كلّ علاقة في بطاقة تحمل اسمها وأيقونتها والكيان المرتبط وحقل العلاقة. وعند النقر على البطاقة، يُفتح نموذج يعرض تفاصيل العلاقة؛ فأدخل القيم الجديدة وانقر **إنهاء** لحفظ التغيير. ولإزالة علاقة، انقر أيقونة الحذف بجوارها؛ فتظهر رسالة تأكيد، وبعد التأكيد تُحذف العلاقة نهائيًا دون إمكانية التراجع.

العلاقات الضمنية

يدير النظام العلاقة التي تحمل `isImplicit = true` ويقفلها:

- `PATCH` يُطلِق الاستثناء `ImplicitRelationUpdateException` (`IMPLICIT_UPDATE_RESTRICTION`).
- `DELETE` يُطلِق الاستثناء `ImplicitRelationDeleteException` (`IMPLICIT_DELETE_RESTRICTION`).

وتُخفي نقاط نهاية السرد والعدّ على مستوى النظام العلاقات الضمنية، ما لم تطلب تضمينها صراحةً. ويضيف `applyImplicitFilter` / `applyImplicitWhere` القيد `isImplicit: false` باستخدام اقتران صريح `{and: [...]}`، بحيث لا يسمح أي شرط `or` متداخل يمرّره المستدعي بإظهار صفوف ضمنية، وذلك كلما كانت قيمة راية الاستعلام `includeImplicit` هي `false` (القيمة الافتراضية). **Since 1.8**

مثال: العلاقة الواردة «Themes» على هيكلية التطبيق `Since 1.10`

يجسّد الكيان الأساسي **هيكلية التطبيق** (Application Shell، المعرّف `SHELL`) نمط العلاقة الواردة الموضّح أعلاه في علاقة يديرها النظام: إذ ينشئ ترحيل data-service `1.10.0-a28` على هذا الكيان العلاقة الضمنية `SHELL_THEMES_RELATION`:

الخاصية	القيمة
`dataEntity`	`SHELL`
`targetField` / `targetEntity`	`shell` / `THEME`
`relationType`	`Incoming`
`normalizedName`	`themes`
`name`	"Themes" / "السمات" – "Themes that use this application shell"

الخاصية	القيمة
`isImplicit`	`true`

لا يملك كيان هيكلية التطبيق حقلًا صادرًا يشير إلى السمات؛ فالعلاقة تسرد مُدخلات `THEME` التي يشير حقل المرجع `shell` فيها إلى مُدخل الهيكلية. وبما أن العلاقة ضمنية، فلا يمكن تعديلها أو حذفها عبر واجهة Relations API، وتُخفى من القوائم على مستوى النظام ما لم تمرّر `includeImplicit=true` . ويكتب الترحيل العلاقة مباشرةً، فلا يُنشر لها `DataRelationCreatedEvent` .

مُدخل هيكلية التطبيق هو بيانات تخطيط مخزَّنة يمكن لسمة أن تشير إليها. ولا يعرض عميل الويب هذا المُدخل في 1.10، ولا يتضمّن الإصدار 1.10 أي أداة لبناء الهياكل.

`normalizedName` والتزامن

عند **الإنشاء**، إذا قدّم العميل `normalizedName` ، فيجب أن يكون فريدًا ضمن الكيان المالك (`DataRelationNormalizedNameDuplicateException` / `DATA_RELATION_NORMALIZED_NAME_DUPLICATE`)؛ وإذا لم يقدّمه، فيُشتَقّ تلقائيًا من `name` عبر `generateUniqueNormalizedName(name, existingNames, 'relation')` . ويشمل فحص التفرّد جميع العلاقات الشقيقة التابعة إلى `dataEntity` نفسه.

ويُستخدَم `updateIdentifier` رمزًا للقفل المتفائل: فعند الإنشاء، يُضبط على قيمة `v4()` جديدة. وعند التحديث، إذا أرسل العميل `updateIdentifier` مختلفًا عن المخزَّن، تُرفَض الكتابة بالاستثناء `UpdateIdentifierConflictException` ؛ وإلا استبدل الخادم قيمته بمعرّف UUID جديد. **Since 1.6**

واجهة برمجة العلاقات

تتبع جميع المسارات أدناه `DataRelationController` . ويقبل `{entity}` **معرّف الكيان أو اسمه النموذجي** ضمن مساحته (`getEntityByIdOrNormalizedNameInSpace`).

الطريقة والمسار	المُعالِج	الغرض
`POST /v1/entities/{entity}/relation`	`create`	إنشاء علاقة لـ `{entity}` . ينشر `DataRelationCreatedEvent` .
`GET /v1/entities/{entity}/relation`	`find`	سرد العلاقات المخزَّنة (الواردة) لـ `{entity}` بصيغة `DataRelationDto[]` .
`GET /v2/entities/{entity}/relation`	`findV2`	إرجاع `{incoming, outgoing}` (`DataRelationListDto`).
`GET /v1/entities/{entity}/relation/count`	`count`	عدّ العلاقات المخزَّنة (مع `where` اختياري).
`GET /v2/entities/{entity}/relation/count`	`countV2`	إرجاع عددَي العلاقات في `{incoming, outgoing}` (`DataRelationCountDto`).
`GET /v1/entities/{entity}/relation/{id}`	`findById`	إرجاع علاقة واحدة لـ `{entity}` بصيغة `DataRelationDto` .
`GET /v1/relations`	`findAll`	إرجاع قائمة على مستوى النظام. `includeImplicit` (الافتراضي `false`)؛ `limit` مقيَّد بالنطاق `[1, 500]` ، وقيمته الافتراضية `100` .

الطريقة والمسار	المُعالِج	الغرض
`GET /v1/relations/{id}`	`findOneById`	إرجاع علاقة واحدة بحسب المعرّف (يدعم المعرّفات المركّبة).
`GET /v1/relations/count`	`countAll`	العدّ على مستوى النظام. `includeImplicit` (الافتراضي `false`).
`POST /v1/entities/joiner/relations`	`findEntitiesRelations`	يتلقّى `{dataEntities: string[]}` ويُرجِع اتحاد علاقاتها الواردة.
`PATCH /v1/entities/{entity}/relation/{id}`	`updateById`	تحديث العلاقة (غير متاح للعلاقات الضمنية). ينشر `DataRelationUpdatedEvent` مع فرق خام.
`DELETE /v1/entities/{entity}/relation/{id}`	`deleteById`	حذف العلاقة (غير متاح للعلاقات الضمنية). ينشر `DataRelationDeletedEvent`.

نقاط المدخلات المرتبطة

عند تزويد نقاط النهاية هذه بكيان ومُدخل وعلاقة، فإنها تُرجِع المُدخلات على *الطرف الآخر* أو تعدّها. وداخليًا، تحلّ كلٌّ منها العلاقة، وتحمّل `targetEntity`، ثم تفوّض العمل إلى مصانع `DataEntryService`، مع تمرير سياق `relation` المتمثّل في `{targetField, entityId, entryId, relationType}`.

الطريقة والمسار	المُعالِج	مدخلات إضافية
`GET /v1/entities/{entity}/entries/{entryId}/relations/{relationId}`	`findRelatedEntries`	`filter`، `include_virtual`
`GET /v2/.../relations/{relationId}`	`findRelatedEntriesV2`	`filterCriteria`، `search`، `include_virtual`
`GET /v3/.../relations/{relationId}`	`findRelatedEntriesV3`	`quickFilters`، `tagsFilters` → `{entries, tags}` + `appliedQuickFilters`
`GET /v1/.../relations/{relationId}/kanban/{categoryFieldId}`	`findRelatedEntriesKanbanView`	لوحة مجمّعة + `categoryId`، الوسوم/المرشحات السريعة
`GET /v1/.../relations/{relationId}/count`	`findRelatedEntriesCount`	`where`، `include_virtual`
`GET /v2/.../relations/{relationId}/count`	`findRelatedEntriesCountV2`	`whereCriteria`، `search`

يقبل مسار V3 قيمة `appliedTagsFilters`، وتُحلّ معايير الوسوم الواردة فيها عبر `TagsService.getTagsFilterCriteria(targetEntity.getId(), appliedTagsFilters)` قبل تطبيقها.

معرّفات العلاقات المركّبة (على مستوى الحقل)

يأتي `relationId` بصيغ متعدّدة. ويقبل `DataRelationRepository.findRelationById` ما يلي:

- **UUID بسيط** – يُحلّ عبر `findById`.
- **`{entityId}:{fieldId}:{sourceEntityId}`** (3 أجزاء) – يحلّ حقل العلاقة `fieldId` في `entityId`، ويتحقّق من وجود `sourceEntityId` ضمن `sources[]` للحقل (أو من مطابقته لقيمة `source` المفردة القديمة)، ثم يُنشئ علاقة **صادرة** اصطناعية يكون فيها `targetEntity = sourceEntityId` و`targetField = field.fullNormalizedName`. وإذا كان الحقل أو المصدر غير معروف، ينتج عن ذلك `RelationNotExistException` (`RELATION_NOT_EXIST`).
- **`{entityId}:{fieldId}`** (جزءان) – ينفّذ الأمر نفسه، باستخدام `source` القديم للحقل بوصفه `targetEntity`.

لذلك تعمل مسارات لوحة المهام (Kanban) والمُدخلات المرتبطة مع العلاقات الصادرة التي لا يوجد لها مستند `DataRelation` مخزَّن.

مثال إنشاء

```
POST /v1/entities/orders/relation
Content-Type: application/json

{
  "specificationVersion": 1,
  "dataEntity": "…orders-entity-id…",
  "name": { "en": "Line items", "ar": "…" },
  "icon": { … },
  "targetEntity": "…order-lines-entity-id…",
  "targetField": "order",          // field on order-lines that points back to orders
  "relationType": 1                 // Incoming
  // normalizedName optional; updateIdentifier must be omitted
}
```

الأحداث

تنشر عمليات تعديل العلاقات أحداث المجال (domain events) على ناقل الخدمة (`correlationId = traceId` ، إضافةً إلى سمات ثابتة تُلحَق بالرسالة لكلّ من `userId` و `delegateeId` و `delegationId` و `spaceId`):

- `DataRelationCreatedEvent(relation, userId)`
- `DataRelationUpdatedEvent(updatedRelation, userId, rawDiff)` – يُحسَب `rawDiff` بواسطة `just-diff` بمقارنة المستند قبل التحديث وبعده.
- `DataRelationDeletedEvent(relation, userId)`

حقول المرجع

حقل المرجع هو `DataEntityField` يحمل مصدرًا وتكون فيه `dataType = Entity` . وتختار `getEntityRelationFieldsWithoutADT()` هذه الحقول وفق الشروط التالية: أن تكون `dataType === Entity` ، وأن يكون للحقل `source` أو `sources[]` غير فارغ، وألّا يكون حقلًا افتراضيًا، وألّا يكون مصدره كيانًا من نوع ADT (تدقيق).

- `source?: UUID` – معرّف كيان مصدر مفرد **مُهمَل**. لا يزال معتمَدًا في مواضع من الرماز (صيغة الجزأين للمعرّف المركّب وفرع «المصدر المفرد (القديم)» في `findV2`).
- `sources?: FieldSources[]` – النموذج الحالي. يتضمّن كل مُدخل `FieldSources` ما يلي:

الخاصية	المعنى
`entity`	معرّف الكيان المصدر الذي يشير إليه الحقل.
`entriesSource`	معرّف استعلام اختياري يُستخدَم لملء قائمة الاختيار (يقبل القيمة null).
`entriesSourceParameters`	المعاملات المرتبطة بالاستعلام ذي المعاملات `entriesSource` .
`entriesSourceDependencies`	الأسماء النموذجية الكاملة للحقول التي تعتمد عليها تعابير الاستعلام (تحسبها الواجهة الخلفية لتحديد وقت تحديث القائمة المنسدلة).
`relatedTo`	إن لم يكن فارغًا، فهو `fullNormalizedName` للحقل الأب الذي يتفرّع منه هذا الحقل تتابعيًا.

الخاصية	المعنى
`relation`	معرّف `DataRelation` المستخدَم للمُدخلات المتتالية (لا يكون ذا معنى إلا عند ضبط `relatedTo`).

إعداد حقل المرجع

عندما يُعدّ المدير حقل مرجع، يحدّد ما يلي:

- **المصدر** الذي يوفّر الخيارات المتاحة للاختيار: مخزن، أو تجميعة، أو استعلام، أو كيان/معرض/وحدة.
- ما إذا كان الاختيار **مفرد أم متعدد**، وما إذا كان العرض **شجري أم مسطّح**.
- **مؤشّرات الترشيح السياقية** (مثل: أنا، مجموعاتي، أدواري، وحدتي التنظيمية) لتضييق نطاق الخيارات بحسب المستخدم الحالي.

الحالات الخاصة

- **الترشيح بحسب الأب في القوائم المتتالية** – يمكن تقييد خيارات حقل مرجع بناءً على قيمة حقل مرجع آخر في المُدخل نفسه (وهذا ما يعبّر عنه `relatedTo` و`relation` أعلاه).
- **منع العلاقات الدائرية** – يمنع النظام تكوين علاقات دائرية قد تؤدي إلى مراجع لا نهائية.
- **إعادة المزامنة (Resync)** – يمكن لحقل المرجع تحديد حقول يُعاد حسابه عند تغيّرها (`resyncFields`)، مع خيار **الحفاظ على القيمة عند التعديل** (`preserveValueOnEdit`) للإبقاء على القيمة المحسوبة بدل الكتابة فوقها.

سلوك العرض والاختيار `Since 1.8`

- **المرجع إلى تجميعة يعرض دائمًا عنوانه الرئيسي المحلول في وضع العرض.** فالمرجع الذي يشير إلى تجميعة بيانات بدلًا من كيان لا يحمل سوى معرّف وعنوان رئيسي محلول، ولا يوجد `mainTitleField` يُشتقّ منه العنوان. ومع ذلك، يعرض وضع العرض العنوان الرئيسي، فتظهر القيمة بدلًا من أن تبقى فارغة.
- **القيم المختارة عبر البحث تبقى بعد إعادة تحميل القائمة المنسدلة.** تعتمد منتقيات المرجع على بحث من جهة الخادم، لذا لا تحمل قائمة الخيارات سوى الصفحة الحالية من النتائج. والحقل الذي يحمل قيمًا **متعدّدة** يستبقي كل معرّف سبق أن اختاره المستخدم، حتى بعد أن تعيد القائمة المنسدلة تحميل خياراتها لمصطلح بحث مختلف – فالاختيار هو المرجع لا الصفحة الظاهرة، ومع ذلك يظلّ إلغاء اختيار معرّف يزيله.
- **القيمة المفردة غير المحلولة تصير فارغة.** ففي حقل مرجع أحادي القيمة، يؤول المعرّف الذي لا يُحلّ إلى أي حمولة إلى قيمة فارغة صريحة، بدلًا من أن يبقى مرجعًا معطّلًا أو ينشر قيمة غير معرَّفة.

حقول الربط

تنسخ حقول الربط القيم من مُدخل مُشار إليه إلى الكيان الحالي عند إنشاء المُدخل أو تحديثه. ويتولّى `MappingFieldService` تنفيذ ذلك.

تظهر أوضاع التحديث في إعدادات الحقل باسم **الكيان الحالي** (`Local`)، و**عدم التحديث** (`Never`)، و**من الكيان الأب** (`ParentOnly`)، و**مزامنة** (`TwoWay`). وفي النموذج، يكون حقل الربط في وضع **عدم التحديث** أو **من الكيان الأب** للقراءة فقط. ولا يُعرض وضع **مزامنة** إلا إذا تطابق إعداد *الإلزام* في حقل الربط مع إعداده في الحقل المصدر. أمّا حقل الربط الواقع داخل كتلة قابلة للتكرار تختلف عن كتلة حقل المرجع الخاص به، فلا يمكنه أن يستخدم إلا **عدم التحديث** أو **من الكيان الأب**.

الإعداد

تتحكّم في الربط ثلاثة أجزاء من الإعداد:

1. `dataSources: DataEntityFieldSources[]` **على مستوى الكيان** – كل مصدر هو `{ referenceEntity, referenceField, mappedFields: MappingFields[] }`. `referenceField` هو معرّف حقل المرجع في *هذا* الكيان، وتأتي القيم من المُدخل المختار فيه.
2. `MappingFields` **لكل عملية ربط** – `{ fromFieldId, toFieldId, updateMode }`، حيث `fromFieldId` حقل في الكيان المُشار إليه، و `toFieldId` حقل في هذا الكيان.
3. `mapping: DataEntityFieldMapping` **على مستوى الحقل** – `{ enabled, referenceFieldId, sourceFieldId, updateMode, searchMode }`. لا تُكتب قيمة في الحقل الهدف إلا إذا كانت قيمة `mapping.enabled` هي true. ويتحكّم `searchMode` (`MappedFieldSearchModeEnum` : `None` ، `Full` ، `StartsWith` ، `Partial`) في كيفية استخدام القيمة المنسوخة في البحث.

أوضاع التحديث

يحدّد `MappedFieldUpdateModeEnum` أيّ القيمتين تُعتمد عند تعارض قيمة المصدر مع القيمة المحلية. ويُنفَّذ هذا السلوك في `handleMappingFields` :

الوضع	القيمة	السلوك الفعلي
`Local`	0	يُبقي القيمة المحلية **ما لم يتغيّر المرجع** (`isReferenceChanged`)؛ وعند تغيّر المرجع، يجلب قيمة المصدر. ويحافظ على التعديلات المحلية بين تغييرات المرجع.
`Never`	1	يُبقي القيمة المحلية متى كانت معرَّفة (أو حين لا يوجد حقل مصدر)؛ وإلا يأخذ القيمة الأولية من المصدر. ولا تُستبدَل القيم المحلية الموجودة أبدًا.
`ParentOnly`	2	ينتقل إلى الفرع الافتراضي: يأخذ قيمة المصدر دائمًا (`fromFieldValue`). فالمصدر هو المرجع الموثوق.
`TwoWay`	3	إذا تغيّر المرجع و**لم** يضبط المستدعي `childToParentHighPriority` ، يأخذ قيمة المصدر؛ وإلا يُبقي القيمة المحلية (الابن). ويتيح انتقال تعديل الابن إلى الأب أيضًا. **Since 1.6**

يقارن `isReferenceChanged` معرّف المُدخل المُشار إليه، المخزَّن حاليًا في المُدخل، مع `sourceEntryId` الوارد. وفي المعالجة على مستوى الكيان كله (`getValuesByEntityDataSources`)، يُستدعى `handleMappingFields` مع `childToParentHighPriority = true` ، فتكون الأسبقية في وضع `TwoWay` لقيمة الابن.

تدفّق الحلّ

يُرجِع `getValuesByEntityDataSources(spaceId, dataEntity, dataEntry, isUpdateMode)` قيمة `Map<toFieldId, value>` (قد تكون القيم مصفوفات للكتل القابلة للتكرار):

1. لكل مُدخل في `dataSources` ، حمّل `referenceEntity` (المحفوظ في خبء ضمن خريطة محلية)، وحدّد `referenceField` بالمعرّف.
2. اقرأ القيمة المرجعية عند `referenceField.fullNormalizedName` . إذا كانت فارغة وكان خيار `options.autoSelectFirstValue` مفعّلًا في الحقل، فارجع إلى `DataEntryService.getDefaultFirstValue` . وإذا بقيت فارغة وكان الحقل محسوبًا أو له مشغّل on-form-opened، فشغّل `calculateField` .
3. إذا كان حقل المرجع داخل **كتلة قابلة للتكرار** (`isParentBlockRepeatable` + `blockNormalizedName`)، فمُرّ على كل نسخة من الكتلة؛ إذ تُضاف القيمة `null` لكل هدف من أهداف الربط في الكتلة التي لا تحمل مرجع `dataEntry` . وإلا فحُلّ المُدخل المُشار إليه المفرد.
4. لكل حقل ربط، حمّل `handleMappingFields` مُدخل المصدر (`state = Active` ، `isInvalid = false`)، ويطبّق قواعد وضع التحديث أعلاه. وتُطابَق الكتل الهدف القابلة للتكرار بالكتلة التي تكون قيمة حقل مرجعها فيها `dataEntry`

`=== sourceEntryId`.

ولا تُملأ إلا الحقول التي تكون قيمة `mapping.enabled` فيها true؛ وتُتخطّى الأهداف المعطَّلة.

الانتشار بعد تغيّر المصدر

بمعزل عن مرحلة الإنشاء/التحديث أعلاه، ينقل معالج أحداث التغييرات التي تُجرى على مُدخل **أب** إلى المُدخلات التي تأخذ قيمها منه، وذلك لحقول الربط في الوضع `ParentOnly` أو `TwoWay`. وفي حقل الربط ذي الوضع `TwoWay`، يُكتب التغيير الذي يُجرى على الابن في المُدخل الأب أيضًا (`UpdateMapFieldValueEntryCreatedHandler` / `UpdateMapFieldValueEntryUpdatedHandler` في data-service). أمّا الأهداف في الوضعين `Local` و`Never`، فلا تُحدَّث أبدًا نتيجة تغيير لاحق في الأب.

روابط التنقل

يبني `LinksService` شجرة الشريط الجانبي المُصادَق عليها، وهي نوع مختلف من «المرجع»: عقدة رابط تشير إلى مورد. والروابط هي مُدخلات كيان الروابط (`LINK_ENTITY_NORMALIZED_NAME`).

قيم `LinkType`: `Module = '0'`، `View = '1'`، `Page = '2'`، `Report = '3'`، `Untyped = '4'`.

> **التقارير على الويب فقط.** يُفتح رابط التقرير في تطبيق الويب فقط؛ أمّا على الجوّال، فيُحلّ إلى `NotSupportedLink` ولا يمكن الانتقال إليه.

روابط الشريط الجانبي (كيان الروابط)

تُدار مُدخلات الشريط الجانبي عبر **كيان الروابط**، ويمكن أن تكون ملكيتها يدوية أو تابعة للنظام. وتُحدَّث روابط النظام تلقائيًا: فعند إنشاء وحدة أو معرض أو تقرير، أو تعديل أيّ منها أو حذفه، يُنشئ النظام المُدخل المقابل في كيان الروابط أو يحدّثه أو يحذفه. أمّا المُدخلات المضافة يدويًا، فتتيح للمدير وضع عقد مخصّصة في الشريط الجانبي، مع اختيار النوع (وحدة، معرض، صفحة، تقرير، غير مصنّف)، والأيقونة، والاسم، والوصف، والأب (للتبعية الهرمية)، وحالة النشر، والأذونات (موثق، مجهول، مقيد، الجميع). وتُرتَّب المُدخلات بحسب حقل **الترتيب** من الأصغر إلى الأكبر. ولكل مُدخل يدوي أيضًا قسم **الهدف (Target)**، وتتوقّف حقوله الفرعية على النوع: **الرابط (Link)** (وهو الرابط، ويُستخدَم عندما يكون النوع غير مصنّف)، و**مُعرّف الهدف** (Target ID) (يُملأ تلقائيًا للمُدخلات ذات ملكية النظام)، و**إجراء المعرض الهدف** (Target View Action) (رابط المعرض، ويُستخدَم عندما يكون النوع معرضًا). وتَرِث المُدخلات ذات **ملكية النظام** كل إعداداتها من مصدرها؛ فمثلًا، يستمدّ المُدخل من نوع معرض إعداداته من إعدادات المعرض.

يجمّع `getLinksForSidebar(domain, spaceId, userId, delegationId?)` الشجرة:

- **مجهول** (بلا `userId`): يشمل الروابط التي تكون قيمة `authentication.type.id` فيها `0` (Anonymous) أو `2` (Everyone)، إضافةً إلى روابط `1` (Authenticated) التي تكون قيمة `value.type.id` فيها `'0'`.
- **مُصادَق عليه**: يشمل النوعين `1` و`2`، إضافةً إلى روابط `3` (Restricted) التي يظهر فيها المستخدم (أو إحدى مجموعاته) ضمن `value.authentication.users` / `groups`.
- يجب أن تكون كل الروابط `published: true`، و`state: 1`، وأن تطابق النطاق (النطاق `domain` الفارغ = الكل)، وأن تستوفي شروط المشاركة (`ApplicationSharingLevelEnum` Public، أو Private/Shared المقيَّدة بـ `spaceId`). وتُرتَّب بحسب `value.order ASC`، ثم `value.title.en ASC`.
- عند وجود `delegationId`، يستبعد `filterLinksByDelegatedPermissions` روابط المعرض/التقرير/الصفحة التي لا يندرج معرّف هدفها ضمن الموارد المسموح بها للمُفوَّض إليه.

- تُجمَّع الشجرة بحسب معرّف الأب؛ وتُحذَف منها عقدة `Module` التي لا أبناء لها. وتُتخطّى روابط المعرض التي يكون أبوها مخفيًا.

ومن الدوال المساعدة ذات الصلة: يبني `createLinkForModule` قيمة `LinkDataEntryValue` لوحدة مملوكة للنظام؛ ويرسل `archiveExistingLinks(targetId, userId, correlationId)` أمر `ArchiveDataEntriesCommand` لكل الروابط النشطة التي تشير إلى هدف (ويُستخدَم للتنظيف عند زوال المورد الأساسي).

روابط النظام (الروابط القابلة للمشاركة)

إلى جانب روابط الشريط الجانبي، يوفّر أسس **روابط نظام** – وهي روابط قابلة للمشاركة، تستند إلى رمز مميّز وتُحيل إلى هدف محدّد: معرض بيانات، أو صفحة/عرض عام، أو تقرير، أو وحدة. وتتضمّن:

- **الرمز (Token)** – رمز فريد مرتبط بالهدف؛ يُنشئ النظام رابطًا موقّعًا ويحلّ الرمز عند فتح الرابط، فيُحيل الزائر إلى الهدف المطلوب.
- **الهدف** – المعرض أو الصفحة أو التقرير أو الوحدة التي يُحيل إليها الرابط.
- **نوع المصادقة – مجهول** (وصول عام دون تسجيل دخول) أو **موثّق** (يتطلب تسجيل الدخول).
- **الصلاحية / الانتهاء** – فترة صلاحية وتاريخ انتهاء؛ ويمكن تفعيل الرابط أو تعطيله أو تحديث صلاحيته أو إزالته.

تتيح الروابط المجهولة عرض المعارض العامة للزوّار غير المسجّلين. ولحماية هذه المعارض، يضيف النظام خطوة للتحقّق البشري (CAPTCHA)، ويمكن معاينة الهدف قبل المشاركة عبر وضع المعاينة.

القيود والمزالق

- **جانب `targetField` يتوقّف على الاتجاه.** في العلاقات الواردة، يقع `targetField` في `targetEntity`؛ وفي العلاقات الصادرة، يقع في الكيان المالك `dataEntity`. وإذا أخطأت في تحديد ذلك، فسيفشل `transformToRelationDto` في حلّ الحقل.
- **العلاقات الصادرة عابرة.** لها معرّفات مركّبة ولا مستند لها، لذا لا يمكنك تطبيق PATCH/DELETE عليها؛ فهذه المسارات تعمل على المستندات المخزَّنة عبر `findById`.
- **العلاقات الضمنية غير قابلة للتغيير** عبر الواجهة.
- `updateIdentifier` **إلزامي للتحديثات الآمنة.** يؤدّي إغفاله إلى تجاوز فحص التعارض (لا يعمل الحارس إلا عند تزويده بقيمة مختلفة)، لذا ينبغي أن يعيد العميل الرمز الذي تلقّاه مع كل تحديث.
- `findAll` **يحدّ النتائج** بحدّ أقصى قدره 500 وقيمة افتراضية قدرها 100، لذا قسّم النتائج إلى صفحات لدى القاطنين الكبار.
- **الربط لا يكتب إلا أهداف** `mapping.enabled`. ويحمي `Local` / `Never` التعديلات المحلية عمدًا؛ لذا فإن بقاء القيمة «لا تُحدَّث» كثيرًا ما يعكس السلوك المقصود لوضع التحديث، ولا يدلّ على خلل.

سجل السلوك

- **كانت إضافة حقل ربط أو حقل مرجع تطبيق تتطلّب الحفظ مرّتين.** تعرض قائمة الحقول في مصمّم الكيان كل عنصر مربوطًا بعنصر تحكّم نموذج بسيط، لكن مربّعَي حوار حقل الربط وحقل مرجع التطبيق كانا يُدرجان عنصر تحكّم مختلف البنية (متداخلًا). لذلك كان الحقل الجديد يظهر في القائمة، لكن الحفظ لا يلتقط قيمته حتى يُعيد شيء آخر ربطه. وأُصلح ذلك بإدراج عنصر التحكّم البسيط نفسه الذي تستخدمه بقية قائمة الحقول.

واجهة المستخدمين البرمجية وحسابات الهوية

هذه الصفحة موجّهة إلى مطوّري التكامل والمهندسين الذين ينشئون المستخدمين أو يديرونهم برمجيًا، أو يحتاجون إلى فهم دقيق للعمليات التي تنفّذها شاشات إدارة المستخدمين في الخلفية. وللاطّلاع على المهام نفسها من منظور المسؤول، مع الشاشات، راجع إدارة المستخدمين [الإدارة]. أما تفاصيل تسجيل الدخول (خادم الهوية، والمزوّدون، والعوامل الثانية)، فيجدها المشغّلون في خادم الهوية ومزوّدو الهوية الخارجيون.

نموذج المستخدم ثنائي المستوى

يُمثَّل «المستخدم» في **موضعين** تُزامَن بياناتهما تلقائيًا:

المستوى	الخدمة	التخزين	يحتوي على
المُدخل التجاري	`data-service`	كيان البيانات `USER` (مستندات المُدخلات)	حقول الملف التعريفي (`basic_info` و `account_information` و `phone_number`)، وإسنادات الوحدات التنظيمية/المجموعات، والعَلَم `is_deactivated`
حساب الهوية	`users-service`	مجموعة `user` في Mongoose (`UserSchema`)	بيانات الاعتماد (`username` و `password` المُجزَّأة)، و `email` ، وإعداد MFA، وعدّادات قفل تسجيل الدخول، و `isDeactivated`

تدير أنت (أو أي تكامل خارجي) **المُدخل التجاري**، أي المُدخل `USER` ، عبر التدفقات المعتادة لإنشاء المُدخلات وتحديثها وأرشفتها. وتستمع `users-service` إلى أحداث النطاق الناتجة عن ذلك، فتُنشئ **حساب الهوية** المطابق أو تُحدّثه أو تزيله. ولا تُنشئ حساب الهوية مباشرةً أبدًا؛ إذ ينشأ تبعًا لدورة حياة المُدخل.

ولا تتحوّل إلى حسابات لتسجيل الدخول إلا المُدخلات التي نوع مستخدمها **طبيعي** (`basic_info.user_type.id === 'NORMAL_USER'`). أما المستخدمون من النوع **تطبيق** (`APPLICATION_USER`) فيحصلون على رمز API بدلًا من ذلك (انظر مستخدمو التطبيقات).

التهيئة: ما الذي يحدث عند إنشاء مُدخل USER أو تغييره

يُطلق كل حدث لإنشاء مُدخل `USER` أو تحديثه أو أرشفته أو استعادته عملية **مطابقة** واحدة: تُوائَم خلالها سجلات الهوية مع المُدخل *كما هو الآن*، بإعادة قراءته وقت المعالجة. وبذلك لا يهمّ ترتيب وصول الأحداث: فلم يعد التحديث يضيع إذا وصل قبل حدث الإنشاء. `Since 1.10`

عندما يكون المُدخل **الطبيعي النشط**، تُنشئ المطابقة حساب الهوية أو تُحدّثه (upsert):

1. تبحث عن حساب قائم بمعرّف المُدخل، ثم بالبريد الإلكتروني (بشرط أن يحمل ذلك الحساب المعرّف نفسه).
2. **حساب قائم** ← تنسخ `username` و `email` (فقط حين لا يكونان فارغين، فلا تستبدل قيمةً موجودة بقيمة فارغة أبدًا)، وتنسخ دائمًا `isExternal` و `externalUserId` و `isDeactivated` و `phoneNumber` و `phoneVerified` . ولا تُمَسّ كلمة المرور أبدًا.
3. **لا حساب، لكن حسابًا آخر يملك البريد الإلكتروني أصلًا** ← لا يُنشأ شيء، ويُسجَّل التعارض. وبهذه الآلية يُمنع التكرار بحسب البريد الإلكتروني.
4. **لا حساب** ← يُنشأ حساب يُستخدَم فيه **معرّف UUID عشوائي كلمةَ مرور أولية**. وهذه القيمة النائبة ليست بيانات اعتماد قابلة للاستخدام.

5. وللمستخدم **الداخلي** المُدرَج حديثًا (حين لا تكون قيمة `isExternal` هي true)، تُنشئ المطابقة رمز إعادة تعيين، وتنشر رسالة «عيّن كلمة مرورك» إذا أمكن الحصول على قالب رسالة إعادة تعيين كلمة المرور. وتُرسَل الرسالة مرة واحدة بالضبط، من الاستدعاء الذي أدرج الحساب. أما المستخدمون **الخارجيون** (الموحَّدون أو المستوردون من الدليل) فلا تصلهم رسالة؛ إذ تتم المصادقة عليهم عبر مزوّدهم الخارجي.

أما إذا كان المُدخل **المؤرشف** أو كان مفقودًا، فيُزال حساب الهوية المرتبط به، في حين يُترك المُدخل **المسوّدة** على حاله.

حقول الهوية المطلوبة. الحقول `username` و`password` و`email` مطلوبة في حساب الهوية. فإذا غاب `account_information.username` أو `account_information.email_address` عن المُدخل، فشل التحقق ولم يُنشأ الحساب (ويُسجَّل ذلك دون إعادة محاولة)؛ لذا احرص دائمًا على تعبئة الحقلين كليهما.

التفرّد وعدم الحساسية لحالة الأحرف. لا يميّز البحث بالبريد الإلكتروني واسم المستخدم بين حالة الأحرف الكبيرة والصغيرة؛ لذا يشير `Jane@x.com` و`jane@x.com` إلى الحساب نفسه. ويُفرض تفرّد البريد الإلكتروني عند الإنشاء بفحص على مستوى التطبيق، لا بقيد في قاعدة البيانات؛ إذ لا يوجد فهرس فريد فعلي إلا لـ `externalUserId`، فلا يمكن لحسابَين أن يتشاركا معرّف الهوية الخارجية نفسه.

موثوقية المعالِجات. تُعيد معالِجات التهيئة رمي الخطأ عند الفشل كي يعيد ناقل الأحداث محاولة معالجة الرسالة، باستثناء واحد: يُسجَّل فشل التحقق `validateSync` في Mongoose أثناء إنشاء الحساب أو استعادته، وتُسقَط الرسالة بدلًا من إعادة المحاولة، حتى لا يتسبّب مُدخل غير سليم في حلقة متكرّرة من محاولات معالجة رسالة سامّة.

الحساب المؤسسي والإسنادات. يرفض مدقّق الإسنادات أي `ASSIGNMENT` لمستخدم لا تكون قيمة `account_information.organization_account.id` لديه `ORGANIZATION_ACCOUNT_TRUE` ("User must be an organization account").

التعديل

يُجرى التعديل عبر مُدخل `USER`، ويُحدَّث حساب الهوية تبعًا له تلقائيًا (انظر الخطوة 2 أعلاه). ولا يغيّر تعديل المُدخل كلمة المرور **أبدًا**؛ فلا تتغيّر كلمات المرور إلا عبر تدفقَي إعادة التعيين والتغيير.

وتنعكس حالة التعطيل في حساب الهوية عند كل تعديل؛ إذ تُضبَط `isDeactivated` فيه وفق قيمة `is_deactivated` في المُدخل كلما حُدِّث مُدخل `USER`، وليس عبر نقطتَي نهاية التفعيل والتعطيل المخصّصتين فحسب.

الأرشفة والاستعادة

تؤدي أرشفة مُدخل `USER` إلى **حذف قاطع** لحساب الهوية المطابق، فتُزال بيانات اعتماده وإعداد MFA الخاص به.

أما **استعادة** مستخدم مؤرشف فتعيد تهيئة حساب الهوية بكلمة مرور عشوائية جديدة، وبرابط إعادة تعيين جديد للمستخدمين الداخليين، مع نقل حالة التعطيل. ولأن الحساب يُزوَّد بكلمة مرور جديدة، يجب على المستخدم الداخلي المُستعاد أن يعيّن كلمة مرور جديدة عبر رسالة إعادة التعيين قبل تسجيل الدخول. `Since 1.6`

كلمات المرور

إعادة التعيين بمبادرة المسؤول

تتطلّب إعادة التعيين بمبادرة المسؤول الدور `SystemAdmin`، وتؤدي إلى نشر حدث إعادة تعيين؛ ثم تُنشئ `users-service` رمز إعادة تعيين وترسل رسالة إلى المستخدم. ولا يرى المسؤول كلمة المرور ولا يعيّنها أبدًا.

سياسة كلمات المرور

القاعدة	المتطلّب
الطول الأدنى	8 أحرف
أحرف كبيرة + صغيرة	حرف واحد على الأقل من كل نوع
أحرف + أرقام	واحد على الأقل من كل نوع
رمز خاص	رمز واحد على الأقل

تُجزَّأ كلمات المرور قبل تخزينها، ولا يُحفظ نصّها الصريح أبدًا.

تدفقات الخدمة الذاتية

يدير المستخدمون كلمات مرورهم بأنفسهم عبر صفحات توفّرها خدمة الهوية:

- **تغيير كلمة المرور**: يتطلّب جلسة مُصادَقًا عليها، ولا يتاح للحسابات الخارجية. ويتحقّق هذا التدفق من كلمة المرور القديمة، ويمنع إعادة استخدامها، ويفرض سياسة كلمات المرور.
- **طلب إعادة التعيين**: نموذج عام. لا تُرسَل رسالة إعادة التعيين إلا إذا كان الحساب موجودًا و**غير خارجي** و**غير معطَّل**. ولا يميّز البحث بين حالة الأحرف الكبيرة والصغيرة، ويعرض النموذج دائمًا رسالة نجاح سواء وُجد حساب مطابق أم لا (دون أي إشارة تكشف الحسابات الموجودة).
- **إكمال إعادة التعيين**: يتحقّق من رمز إعادة التعيين، ويطبّق سياسة كلمات المرور، ويحدّث كلمة المرور، ويُبطل الرمز.

تجد مدة صلاحية رمز إعادة التعيين وطريقة تجزئته في خادم الهوية: إعادة تعيين كلمة المرور.

التعطيل والتفعيل

تتطلّب نقطتا نهاية التفعيل والتعطيل الدور `SystemAdmin`، وتطبّقان قاعدتَي حماية:

- **لا يمكنك تعطيل نفسك أو تفعيلها**: يُرفض الطلب بالخطأ `400 Bad Request`. (وتُخفي شاشات الإدارة أيضًا هذين الإجراءين في الصفّ الخاص بك.)
- **حارس عدم تكرار العملية**: تؤدي محاولة تعطيل مستخدم معطَّل أصلًا، أو تفعيل مستخدم مفعَّل أصلًا، إلى إرجاع `409 Conflict`.

ويُفرض التعطيل على مستويين: **يُحظر تسجيل الدخول** (فالحساب المعطَّل لا يستطيع تسجيل الدخول، ولا تستطيع جلسة قائمة التعرّف عليه)، و**تُبطَل الرموز النشطة**، فتُنهى الجلسات القائمة بدلًا من بقائها حتى انتهاء صلاحيتها. وتعرض صفحة تسجيل الدخول الرسالة `ACCOUNT_DEACTIVATED` («الحساب معطّل، يُرجى التواصل مع مسؤول النظام»). أما إعادة التفعيل فتُزيل علامة التعطيل، فيتمكّن المستخدم من تسجيل الدخول مجددًا ببيانات اعتماده الحالية.

التعطيل يزيل كل سجلات الجلسة `Since 1.10`

تحتفظ خدمة الهوية بفهرس لكل سجل يخصّ الحساب، وتحذف عملية التعطيل هذه السجلات تحديدًا: المنح والجلسات ورموز الوصول ورموز التحديث، فلا يبقى أيّ منها عالقًا. وقبل 1.10، لم تكن آلية المسح القديمة تعثر على رموز التحديث، ولم تكن سجلات المنح والجلسات تُحذف قط. ولم يتغيّر السلوك الظاهر (فالحساب المعطَّل لم يكن يستطيع أصلًا تسجيل الدخول أو تحديث الرموز)؛ وإنما يكمن الفرق في عدم بقاء أي سجل قديم. أما السجلات المكتوبة قبل الترقية فليست مفهرسة، وتنتهي بانقضاء مدة صلاحيتها.

قفل الحساب

يُطبَّق القفل تلقائيًا للحماية من هجمات تخمين بيانات تسجيل الدخول:

- تزيد كل محاولة دخول فاشلة عدّادات المحاولات الفاشلة، وتُسجّل وقت آخر محاولة فاشلة.
- عندما يبلغ عدد المحاولات الفاشلة **5**، يُقفَل الحساب. وتُحدَّد نهاية مدة القفل بالوقت الحالي مضافًا إليه عدد المحاولات الفاشلة منذ آخر نجاح (بالدقائق)، فتطول المدة مع تكرار القفل.
- يُرفض تسجيل الدخول للحساب المقفل حتى تنقضي مدة القفل («الحساب مقفل»).
- يمسح تسجيل الدخول الناجح جميع العدّادات والقفل، ويسجّل وقت آخر دخول.

لا توجد نقطة نهاية للمسؤول لـ «إلغاء القفل»؛ إذ يزول القفل بانقضاء مدته أو عند نجاح المستخدم في المصادقة في المرة التالية.

المصادقة متعددة العوامل `Since 1.8`

يُخزَّن إعداد MFA الخاص بكل مستخدم في حساب الهوية:

- إعداد **TOTP** (تطبيق المصادقة)، مع تخزين السرّ المشترك مشفّرًا، وعلامة تأكيد تُضبَط حين يُكمل المستخدم التسجيل.
- **طريقة مفضّلة**: واحدة من `totp` أو `sms` أو `email` أو `passkey` .

أما التسجيل، وفرض المصادقة متعددة العوامل عند تسجيل الدخول، والإدارة، فتتمّ جميعها عبر تدفقات تسجيل دخول موجَّهة للمستخدم، ولا تُعدّ عمليات CRUD للمسؤول. ومن جانب المسؤول، يتبع إعداد MFA دورة حياة حساب الهوية: فيُهيَّأ ويُحذَف ويُستعاد معه. ويُرفض تسجيل TOTP ومفاتيح المرور والتحقق الإضافي بكلمة المرور للحسابات الخارجية.

الملف التعريفي للمستخدم

الملف التعريفي للمستخدم الحالي نموذج قراءة محفوظ في الذاكرة المخبئية يجمع:

- معرّف الموضوع، والاسم، والبريد الإلكتروني.
- مجموعات المستخدم، مقسّمةً إلى مجموعات المستخدمين، والأدوار، ومجموعات الوحدات التنظيمية، ومجموعات الاستعلام (وتُستبعد المجموعات الوهمية).
- تجميعات المستخدم ووحدته التنظيمية الرئيسية.
- مجموعة الأذونات بعد استكمال تحديدها (أو أذونات المفوَّض إليه عند التصرّف بموجب تفويض).

وتُرجع نقاط نهاية القراءة ذات الصلة مجموعات المستخدم غير الوهمية، ومديره المباشر (بتتبّع تسلسل الوحدات التنظيمية الأم صعودًا)، وأعضاء وحدته التنظيمية مع وحدتها الأم ووحداتها الفرعية. انظر المجموعات والوحدات التنظيمية والعضوية.

ويمثّل المستخدم `SYSTEM` حالةً خاصة: إذ يُكتفى له بملف تعريفي اصطناعي بمجموعات وأذونات فارغة، بدلًا من تجميع ملفه التعريفي من البيانات المخزَّنة.

فحوص عضوية المجموعات والتجميعات

إلى جانب الملف التعريفي، تتيح المنصة فحوص عضوية تُستخدم في التخويل وسير العمل للإجابة عن سؤال «هل ينتمي هذا المستخدم إلى X؟»:

- **عضوية المجموعة الثابتة**: هل ينتمي المستخدم إلى `GROUP` معيّنة؟
- **عضوية مجموعة الاستعلام الديناميكية**: هل ينتمي المستخدم إلى مجموعة استعلام؟ وتُسجَّل عضوية مجموعات الاستعلام في مُدخلات بحث `USER_QUERY_GROUP_ASSIGNMENT` عبر معالِجات تعمل في الخلفية، فيقتصر الفحص على التحقّق السريع من وجود إسناد نشط، بدلًا من إعادة تقييم الاستعلام مباشرةً. `Since 1.8`
- **عضوية التجميعة**: هل ينتمي المستخدم إلى تجميعة معيّنة؟

تفضيلات المستخدم (حالة المستخدم)

تُخزَّن تفضيلات الواجهة الخاصة بكل مستخدم في كيان البيانات `USER_STATE`. وهذه التفضيلات خاصة بكل مستخدم، ولا يديرها المسؤول. وتُنشأ حالة افتراضية عند أول وصول، ويمكن تحديثها أو إعادتها إلى القيم الافتراضية.

الحقل	المعنى
`firstLogin`	`true` للحالة المُنشأة للتو؛ يوجّه تجربة أول تسجيل دخول
`language`	اللغة المفضّلة
`theme`	السمة المفضّلة
`fontSize`	حجم خط الواجهة
`layoutMode` / `menuMode`	وضع عرض القائمة ووضع التخطيط
`timeZone`	المنطقة الزمنية المفضّلة
`dateTimeFormat` / `timeFormat` / `dateFormat`	تنسيق التاريخ والوقت المفضّل

تضبط الحالة الافتراضية `firstLogin: true` وتترك كل تفضيل فارغًا، فيعتمد العميل القيم الافتراضية للمنظمة أو النظام حتى يحدّد المستخدم تفضيلاته.

مستخدمو التطبيقات

يمثّل مستخدمو التطبيقات أنظمةً خارجية، وهم مُدخلات `USER` تحمل `basic_info.user_type.id === 'APPLICATION_USER'` وكتلة `application_information`؛ وتُجرى مصادقتهم باستخدام **رمز API** يُرسَل في ترويسة الطلب، وتتعامل المنصة مع كل طلب على أنه صادر عن مستخدم التطبيق. وترد تفاصيل آلية الرمز وترويساته وإبطاله في رموز الـ API.

عندما يكون مُدخل التطبيق نشطًا، تُنشئ عملية المطابقة رمز الـ API أو تحدّثه استنادًا إلى `application_information` (`name` و`expiry_date` و`unlimited_expiry`)، وتتبع قيمةُ علامة التنشيط في الرمز قيمةَ `is_deactivated`. ولا يُصدَر رمز مستخدم التطبيق الجديد ولا يُكشَف عنه إلا مرة واحدة، ولا يُكشَف عنه إلا للشخص الذي **أنشأ** المُدخل؛ فإن تعذّر العثور على سجل هوية ذلك الشخص، فلا يُنشأ أي رمز.

إعادة توليد الرمز `Since 1.8`

قبل 1.8، كان رمز التطبيق يُصدَر مرة واحدة فقط عند إنشاء مُدخله، وكان تدويره يتطلّب حذف التطبيق وإعادة إنشائه، مما يؤدي إلى فقدان هويته وحقوق وصوله وسجلّه التدقيقي. أما الآن، فيستطيع صاحب الدور **SystemAdmin** تدوير الرمز مع الإبقاء على التطبيق نفسه:

- يُرسَل الرمز الجديد **بالبريد الإلكتروني إلى المسؤول الذي طلبه**، ولا يُعرَض أبدًا على الشاشة ولا تُعيده الواجهة البرمجية.
- يظل الرمز المُستبدَل صالحًا للمصادقة طوال فترة سماح قابلة للضبط، حتى لا ينقطع التكامل الجاري لحظة النقر على الزر.
- لا يمكن إعادة توليد رموز التطبيقات **المؤرشفة** (فإعادة التوليد قد تُنشئ مستند رمز من الصفر، فتُعيد تفعيل بيانات اعتماد لتطبيق أُخرِج من الخدمة)؛ أما التطبيقات المعطَّلة فيمكن إعادة توليد رموزها.

يظهر الإجراء **إعادة إنشاء الرمز** في كل موضع تتوافر فيه قائمة صف مستخدم التطبيق: قائمة الإجراءات في ترويسة مُدخل التطبيق نفسه، وقائمة صف المستخدم في **إدارة الكيانات**، وقائمة صف المستخدم في **الإعدادات**. وتستخدم القوائم الثلاث مربع حوار التأكيد نفسه، وتُرسل الطلب نفسه. وترد التفاصيل الكاملة لآلية التدوير مقارنةً بالإصلاح الذاتي، وضبط فترة السماح، في رموز الـ API: إعادة توليد رمز.

تاريخ الانتهاء والجهة المُصدِرة `Since 1.10`

- **تاريخ الانتهاء.** كان حقل **تاريخ الانتهاء** في نموذج مستخدم التطبيق (`application_information.expiry_date`) مُدرَجًا بحالة إخفاء دائمة، فلم يكن من الممكن تحديد تاريخ انتهاء لأي مستخدم تطبيق، وكان كل رمز API يُنشأ بلا تاريخ انتهاء. ويحوّله الترحيل `1.10.0-a42` إلى حقل من نوع **تاريخ** لا يُخفى إلا للمستخدمين الطبيعيين أو عند تفعيل الصلاحية غير المحدودة (`unlimited_expiry`)، مع الاحتفاظ بأي شرط إخفاء سبق أن ضبطه المسؤول. والحقل غير إلزامي. ولا يُتحقّق من الرمز إلا بمقارنته بتاريخ بالصيغة البسيطة `yyyy-MM-dd`: فإذا كان تاريخ انتهاء المُدخل قد خُزِّن سابقًا على هيئة طابع زمني، أدرجه الترحيل بمعرّفه، دون إعادة كتابة التاريخ، ويُعامَل رمزه على أنه **منتهي الصلاحية** إلى أن يُعاد إدخال التاريخ.
- **الجهة المُصدِرة.** يضيف الترحيل `1.10.0-a34` إلى الكتلة نفسها حقلًا مخفيًا للقراءة فقط باسم `issuer`. ويستخدمه مسار المتجر لتمييز مستخدمي التطبيقات الذين ينشئهم، وتُترك معالجة رموز هؤلاء المستخدمين للمتجر عند مطابقة الرموز. ولا يرى المسؤولون هذا الحقل ولا يضبطونه.
- **المسؤول الأول لقاطن جديد** (في بيئة نشر متعدّدة القاطنين) يتلقّى حساب هوية ورسالة لتعيين كلمة المرور حين يصبح القاطن نشطًا. انظر الهوية لكل قاطن.

نقاط نهاية المسؤول والقراءة

تتوافر نقاط النهاية التالية في `UserController` ضمن `data-service` (وتخضع جميعها لاستراتيجية `Auth2`):

الطريقة والمسار	المصادقة	ملاحظات
`POST /v1/users/{userId}/reset-password`	`SystemAdmin`	إرسال رسالة إعادة تعيين كلمة المرور بمبادرة المسؤول
`POST /v1/users/{userId}/regenerate-app-token`	`SystemAdmin`	تدوير رمز الـ API لمستخدم تطبيق؛ يرفض المُدخلات المؤرشفة
`POST /v1/users/{userId}/deactivate`	`SystemAdmin`	400 عند استهداف المستخدم نفسه؛ 409 إن كان معطَّلًا بالفعل
`POST /v1/users/{userId}/activate`	`SystemAdmin`	400 عند استهداف المستخدم نفسه؛ 409 إن كان مفعَّلًا بالفعل
`GET /v1/users/profile`	مُصادَق عليه	الملف التعريفي للمستخدم الحالي

الطريقة والمسار	المصادقة	ملاحظات
`GET /v1/users/my-groups`	مُصادَق عليه	المجموعات غير الوهمية
`GET /v1/users/direct-manager`	مُصادَق عليه	المدير المباشر
`GET /v1/users/my-orgunit`	مُصادَق عليه	أقران الوحدة التنظيمية + الوحدات الأم + الوحدات الفرعية
`GET /v1/users/state`	مُصادَق عليه	جلب حالة المستخدم أو إنشاؤها
`PUT /v1/users/state`	مُصادَق عليه	تحديث حالة المستخدم
`PUT /v1/users/state/reset`	مُصادَق عليه	إعادة ضبط حالة المستخدم

يُنشأ المستخدمون ويُعدَّلون ويُؤرشَفون ويُستعادون عبر نقاط نهاية المُدخلات العامة على الكيان `USER`. انظر كتابة البيانات.

ملاحظات الإصدارات

- **استعادة مستخدم مؤرشف** (إعادة تهيئة حساب الهوية): `Since 1.6`.
- **المصادقة متعددة العوامل** (إعداد MFA/TOTP لكل مستخدم): `Since 1.8`.
- **عضوية مجموعة الاستعلام الديناميكية** (مُدخلات بحث `USER_QUERY_GROUP_ASSIGNMENT` المُجسَّدة): `Since 1.8`.
- **إعادة توليد رمز التطبيق**: `Since 1.8`.
- **تاريخ انتهاء مستخدم التطبيق (قابل للضبط)، وحقل الجهة المُصدِرة، ومطابقة الهوية المستقلة عن ترتيب الأحداث، والإزالة الكاملة لسجلات الجلسة عند التعطيل**: `Since 1.10`.
- التفعيل والتعطيل، وإعادة تعيين كلمة المرور بمبادرة المسؤول، وتفضيلات المستخدم (حالة المستخدم) متاحة منذ ما قبل الإصدار 1.2، وهي ميزات **long-standing**.

مرجع المحصّلات والتجميع

Availability

.Since 1.10

هذه الصفحة موجّهة إلى **المُكاملين والمطوّرين** الذين يقرؤون الإجماليات أو القوائم المجمَّعة عبر الواجهة البرمجية، أو يبنون عميلًا يعرضها، أو يحتاجون إلى معرفة القواعد الدقيقة التي تحكم الأرقام. وتبيّن كل قاعدة يفرضها الخادم وسبب تطبيقها، ثم تعرض المسارات.

تُعِدّ الإجماليات من الواجهة؟

على بُناة التطبيقات أن يبدؤوا من الإجماليات والتجميع [بناء التطبيقات › المجاميع والتجميع]. ويحمل تبويب الكيان الذي يضمّ التعريفات في المصمّم التسمية **المحصلات**؛ وتسمّي هذه الصفحة التعريف الواحد *محصّلة*. والمقصود في الحالتين واحد. أمّا لمعرفة ما يراه القارئ وما يمكنه فعله بقائمة مجمَّعة، فراجع المُدخلات [استخدام أسس › إنشاء المُدخلات وتعديلها].

تعمل ميزتان معًا، ويحسبهما أسس في مرور واحد على الخادم:

- **المحصّلة** تعريف محفوظ بصيغة «حقل + دالّة» – *مجموع المبلغ*، *عدد المُدخلات* – تُنشأ مرّة واحدة على الكيان، ثم تُفعَّل على المعارض التي ينبغي أن تُظهرها.
- **التجميع** يقسّم قائمة المُدخلات إلى مجموعات حسب قيمة حقل معيّن، بحيث تحمل كل مجموعة مجاميعها الفرعية الخاصة، وتحمل القائمة ككلّ إجماليًا كليًا.

دورة الحياة

تسري المحصّلة **فورًا**. فلا تتطلّب خطوة نشر، لأنها مورد قائم بذاته وليست جزءًا من بنية الكيان، ومن ثمّ لا تنتظر دورة تحديث الكيان نفسه. ومع ذلك، ترافق المحصّلة الكيان عند التحزيم: فتُصدَّر محصّلاته معه، وتُحذف عند حذفه. ويؤدي حذف محصّلة واحدة إلى إزالتها من كل معرض كان قد فعّلها، بصمت ودون خطأ – وتستمر تلك المعارض في العمل، وتكتفي بالتوقّف عن إظهار ذلك الرقم.

الدوال وأنواع الحقول التي تقبلها

تتوافر خمس دوال لا غير: **Count** (العدد) و**Sum** (المجموع) و**Min** (الأصغر) و**Max** (الأكبر) و**Average** (المتوسط). وانتبه إلى التسمية – يستخدم المنتج *Average* ولا يستخدم «avg» أبدًا، ولا تتوافر دوال للوسيط الحسابي أو لعدّ القيم المتمايزة أو للأول أو الأخير أو للانحراف المعياري.

ويتوقّف ما يجوز لك اختياره منها على الحقل وحده:

إذا كان الحقل…	جاز لك اختيار
حقل **Number** (رقم)	الدوال الخمس جميعًا

إذا كان الحقل…	جاز لك اختيار
أيّ نوع حقل آخر	**Count** فقط
حقلًا **مشفَّرًا**	لا شيء إطلاقًا – الحقل مرفوض، بما في ذلك Count

لماذا تبقى Count متاحة على حقل غير رقمي: خلافًا للدوال الأربع الأخرى، لا تحتاج Count إلى قراءة رقم من الحقل لتجمعه أو تقارنه أو تقسمه – بل تكتفي بالتحقّق من أن للحقل **قيمةً أصلًا**. لذا يكون استخدامها مع حقل نصّي أو حقل تاريخ أو مرجع ذا معنى، حتى حين لا يكون للمجموع معنى. وهذا أيضًا سبب كون نتيجة Count ليست مجرّد عدد الصفوف: فالمُدخل الذي لا يحمل قيمة في الحقل المعدود لا يُحتسَب فيها، تمامًا كما لا يُحتسَب في Sum أو Average على الحقل نفسه. `Since 1.10` – قبل هذا الإصدار، كانت Count تتجاهل بصمت الحقل الذي ضُبطت عليه، وتَعدّ كل مُدخل مطابق بصرف النظر عن قيمة الحقل؛ لذا كان يمكن أن تستند Count وAverage المُعرَّفتان على الحقل نفسه إلى مجموعتَين متعارضتين من المُدخلات؛ أمّا الآن فكلتاهما تتّفقان على معنى «بلا قيمة».

ولماذا يُرفض الحقل المشفَّر رفضًا صريحًا بدل إخفائه: لأن القيمة المخزَّنة نصّ مُعمّى، فلا يوجد رقم حقيقي خلف القناع – وليس ثمة ما يُحجب، بل حسابٌ بلا معنى. ويقع الرفض عند حفظ التعريف، لا بصمت عند القراءة. ويُرفض الحقل الذي تجري إزالة التشفير عنه للسبب نفسه، لأن المُدخلات التي لم تُرحَّل بعد ما تزال تحمل نصًّا مُعمّى.

ويُعامَل الحقل **المرتبط بمصدر (mapped)** تمامًا كأيّ حقل عادي من نوعه – فالربط لا يغيّر شيئًا في الدوال المسموح بها. ويقبل الحقل **داخل كتلة** حساب المحصّلات عليه (تُعرَض حقول الكتلة الطرفية مسطَّحةً خارجها)؛ أمّا حاوية الكتلة نفسها فلا.

ولا يوجد **حدّ أقصى لعدد المحصّلات على الكيان**، ولا قاعدة تقصر الحقل على محصّلة واحدة: يمكن أن توجد *مجموع المبلغ* و*متوسط المبلغ* معًا، وتعرضهما القائمة جنبًا إلى جنب في الخلية نفسها. لذا امنح كل واحدة اسم عرض – إذ تستخدم المحصّلة التي لا اسم لها اسم الحقل نفسه، مما يجعل التسميتين غير واضحتين حين تستهدف محصّلتان الحقل نفسه.

تُعرَّف مرّة، وتُفعَّل لكل معرض

يتبع التعريف الكيان، أمّا قرار *إظهاره* فيُتّخذ لكل معرض على حدة، عبر قائمة مربّعات اختيار على بطاقة قائمة المُدخلات.

والمعرض الذي لا يفعّل أيّ محصّلة لا يحصل على أيّ منها. فهو لا يرث مجموعة محصّلات الكيان، ولا يكتسب محصّلة لاحقًا لمجرّد أن أحدهم أضافها إلى الكيان. وهذا التقاطع **حدّ متعمَّد، لا تسهيل**: إذ تحدّد أسس المحصّلاتِ التي يجوز لطلبٍ ما حسابها بأخذ تقاطع ما طلبه المُستدعي مع ما فعّله المعرض، وتفعل ذلك على الخادم. فلا يستطيع عميلٌ توسيع المجموعة بمجرّد أن يطلب المزيد، ولا يمكن لمعرض منشور أو مفتوح للزوار المجهولين أن يبدأ بكشف إجمالي جديد دون علمك.

ولهذا الضمان تبعة عملية تستحقّ التخطيط: تبقى المحصّلة المُنشأة حديثًا غير مرئية حتى تفعّلها على كل معرض ينبغي أن يعرضها، ولا يوجد مفتاح «لكل المعارض».

وتترتّب على القاعدة نفسها نتيجتان:

- **في قسم الإدارة** – عند استعراض مُدخلات الكيان نفسه دون ربط بمعرض – تُحسب كل محصّلة معرَّفة على الكيان، إذ لا يوجد معرض يضيّق النطاق.
- **المحصّلة المحذوفة تُسقَط بتسامح.** فالمعارض التي فعّلتها لا تُعاد كتابتها ولا تُصدر أخطاء؛ وإنّما يتعذّر حلّ المعرّف فحسب.

الإجمالي الكلي والمجاميع الفرعية لكل مجموعة

يُحسب كلا الرقمين **على الخادم**، في مسار واحد، على كل المنصّات، ولا ينتج أيّ منهما عن تجميع يجريه العميل لما يظهر على الشاشة.

- يُحسب **الإجمالي الكلي** على **كل ما تطابقه القائمة** – أي كل مُدخل يبقى بعد تطبيق المرشحات والبحث والمرشحات السريعة والترشيح المسبق وحالة المُدخل وإذن المخزن – *لا* على صفوف الصفحة الحالية، ولا على صفحة المجموعات المعروضة.
- وتحمل كل **مجموعة** المحصّلات نفسها مرّة أخرى بوصفها **مجموعها الفرعي**، محسوبةً على مُدخلات تلك المجموعة.

ويرث الملخّص نطاق قائمة المُدخلات نفسه بدل إعادة بنائه، فلا يمكن لإجمالي أن يَعدّ صفًّا ترفض القائمة نفسها إظهاره.

ويترتّب على ذلك أمران قد يثيران الاستغراب:

- **التنقّل بين الصفحات لا يغيّر شيئًا.** فالانتقال إلى الصفحة الرابعة لا يغيّر الإجمالي الكلي، لأنه لم يكن في أيّ وقت إجمالي الصفوف الظاهرة.
- **المجاميع الفرعية على الشاشة لا تُجمَع لتساوي الإجمالي الكلي** في الحالة العامة. فجمع مجاميع المجموعات يصحّ، أمّا حساب متوسط المتوسطات فلا، وكذلك أيّ حساب يقتصر على صفحة واحدة من المجموعات.

وحين ينتج عن التجميع **مجموعة واحدة** فقط، تُسقِط القائمة صفَّ الإجمالي الكلي – إذ سيكرّر المجموع الفرعي لتلك المجموعة تمامًا.

الفراغ والشرطة و `****`

هذه ثلاث إجابات مختلفة، وعلى باني التطبيق ألّا يخلط بينها:

ما يظهر في الخلية	ما يعنيه
`-` (خلية فارغة تُعرض كشرطة)	**لا شيء يُجمَع.** لا توجد مُدخلات مطابقة، أو لا يحمل أيّ منها قيمة لذلك الحقل. فمتوسط صفرٍ من الصفوف ليس صفرًا، ولا توجد قيمة صغرى لمجموعة خالية، ولذا تمتنع المنصّة عن اختلاق قيمة.
`0`	صفر حقيقي – نتيجة محسوبة فعلًا.
`****`	**حُسِب ثم حُجِب.** الحقل محميّ ولا يجوز لهذا القارئ رؤية القيمة. انظر الحقول المحمية.

فلا تفسّر `-` بمعنى «مخفيّة»، ولا تفسّر `****` بمعنى «فارغة». تفصل المنصّة بينهما عمدًا، ولا تضع نصًّا مخصّصًا للعرض داخل حقل بيانات، فيرى المُكامِل الذي يقرأ عبر الواجهة البرمجية التمييزَ نفسه: للقيمة الغائبة والقيمة المحجوبة علامتان منفصلتان.

وضع المعاينة

في وضع المعاينة [بناء التطبيقات]، تتضمّن استجابة الملخّص دائمًا `matchedCount: 0`، وتظهر كل محصّلة بالقيمة `-`، ولا تتضمّن الاستجابة أيّ مجموعات – لكنّ الحقل الذي كان سيُحجب عن المستدعي في وضع المعاينة يظلّ موسومًا بأنه محجوب. فالرقم الزائف الذي يبدو معقولًا أسوأ من فراغ واضح، ولذا لا تختلق المعاينة رقمًا؛ ولأن قرار الحجب يظلّ محسوبًا على نحو صحيح، لا تُظهر المعاينة أبدًا محصّلةً كانت الاستجابة الحيّة ستحجبها أو تُسقطها كليًا.

قواعد إعداد التجميع

- **مستويات التجميع** (**إضافة تجميع**) – قائمة حقول مرتّبة. **المستوى الأول هو المجموعة الخارجية**، ويقسّم المستوى التالي المُدخلات داخلها، وهكذا. ولا يمكن اختيار الحقل نفسه أكثر من مرّة، ولا حدّ أقصى لعدد المستويات التي تضيفها.
- **طي المجموعات افتراضيًا** – تظهر كل مجموعة مطويّةً أول مرّة يرى فيها القارئ القائمة. معطّل افتراضيًا.
- **تغيير التجميع** – يمكن للمستخدمين النهائيين اختيار مستوياتهم الخاصة من لوحة **إعدادات العرض** في القائمة نفسها. معطّل افتراضيًا. وما دام معطّلًا، فلا يستطيع القارئ إضافة مستوى أو حذفه؛ ويقتصر التجميع على إعداد المعرض.

وإيقاف بطاقة قائمة المُدخلات يُخفي هذه الإعدادات في المصمّم فحسب، ولا يمسحها شيء؛ فتعود كما هي عند تشغيل البطاقة مجدّدًا.

وثمّة قاعدة واحدة تقيّد ما يمكنك حفظه: **يجب أن يكون مستوى التجميع من حقول المعرض المسرودة.** فالتجميع حسب حقل لا يسرده المعرض سيُظهر عمودًا أخفيته عمدًا، ولذلك يُرفض هذا الإعداد. وينطبق ذلك أيضًا على الحقل المحميّ: فطيّ المستوى، كما هو موضّح أدناه، لا يجيز الوصول إلى حقل لم يسرده المعرض أصلًا.

الحقول التي يمكن التجميع حسبها، ولماذا

يمكنك التجميع حسب النص، والنص المترجم، والرقم، والتاريخ، والتصنيف، والمُحوّل، وحقل **مرجع أحادي القيمة**، وحقل **مرتبط بمصدر (mapped)**، وأيّ حقل داخل كتلة **غير قابلة للتكرار**.

ولا يمكن التجميع حسب أربعة أصناف من الحقول. وهذا **مستحيل**، وليس مجرّد أمر «غير مدعوم»؛ إذ يُخلّ كل صنف منها بأحد الضمانات التي يجب أن توفّرها الميزة:

الحقل الذي…	لماذا لا يصحّ التجميع حسبه
يحمل **عدّة قيم** في آن واحد	ينتمي المُدخل إلى أكثر من مجموعة في الوقت نفسه.
يتكرّر	للسبب نفسه: مُدخل واحد، وقيم عدّة، ومجموعات عدّة.
يقع **داخل كتلة قابلة للتكرار**	للسبب نفسه أيضًا؛ إذ يحمل المُدخل قيمة لكل تكرار من تكرارات الكتلة.
هو حقل **بحث/ADT**	القيمة المخزَّنة فيه معرّف خيار يُحَلّ بالرجوع إلى قائمة البحث الخاصة بالحقل، وليست مؤشّرًا إلى مُدخل؛ لذا لا تتضمّن القيمة ما يستند إليه التجميع.

وتتعذّر معالجة الأصناف الثلاثة الأولى لسبب مشترك: لا تستطيع القائمة إظهار المُدخل إلا في **مكان واحد**، ولا يمكن إدراجه ضمن حدود مجموعة واحدة وهو ينتمي إلى ثلاث مجموعات في الوقت نفسه. وفوق ذلك، سيُحتسب هذا المُدخل في عدد المُدخلات وفي كل مجموع فرعي مرّة عن كل مجموعة ينتمي إليها، فلا يعود مجموع أعداد المُدخلات في المجموعات مساويًا لعدد المُدخلات في القائمة نفسها. أمّا الكتلة **غير القابلة للتكرار** فلا إشكال فيها؛ إذ تحمل قيمة واحدة بالضبط لكل مُدخل، فلا يحدث شيء من ذلك.

أمّا الصنف الرابع فله سبب مختلف. فحقل البحث/ADT يبدو كحقل مرجع، وهو من النوع نفسه في الأساس، لكن تجميعه بالطريقة المتّبعة مع المرجع سيؤدّي إلى قراءة جزء من القيمة لا وجود له في أيّ مُدخل. ونتيجة لذلك، ستُدرَج كل المُدخلات بصمت في مجموعة واحدة بمفتاح فارغ **ودون أيّ خطأ** ينبّهك. لذا فإن استبعاده صراحةً هو الخيار الأمين.

كما تُستبعَد كتلة **أوصاف المُدخل** والحقول التابعة لها؛ فهي افتراضية، وتُركَّب مع كل طلب بدلًا من تخزينها. ويستبعد منتقي الحقول في المصمّم كذلك الحقول **ذات المدى**، لأنها تحمل زوجًا من القيم «من - إلى»، لا قيمة واحدة.

حقول التصنيف متعدّدة القيم `Since 1.10`

يمكن الآن ضبط حقل التصنيف ليحمل **عدّة قيم**، أي مصفوفة من الخيارات بدلًا من خيار واحد. ويندرج هذا الحقل ضمن الصف الأول من الجدول أعلاه؛ فالمُدخل الذي يحمل *أ* و*ب* سينتمي إلى مجموعتين، ولذلك لا يصلح مستوىً للتجميع. وتُفحص هذه القاعدة مرتين: عند حفظ المعرض، ثم **عند قراءة الملخّص** أيضًا، لأن حقل التصنيف قد يتحوّل إلى حقل متعدّد القيم بعد حفظ معرض يجمّع حسبه؛ وعندئذ يتوقّف تطبيق تجميع ذلك المعرض بدلًا من احتساب المُدخلات مرتين.

ويُرفض الحقل نفسه في المواضع ذات الصلة التي تتطلّب قيمة واحدة بالضبط لكل مُدخل: **الفرز الافتراضي** للمعرض، و**التجميع** في إحصائية لوحة المعلومات، وحقل أعمدة **لوحة المهام** أو تصنيفها الفرعي، وتصنيف **الشريط الزمني**، و**التصنيف الرئيسي / الفرعي** للكيان (الخطأ `MULTI_VALUE_CATEGORY_FIELD_NOT_ALLOWED`، ويحدّد `details.usage` الموضع). وينطبق القيد في الاتجاه المعاكس أيضًا: يفشل تحويل حقل تصنيف قائم إلى متعدّد القيم إذا كان مستخدمًا في أيّ من تلك المواضع، ويظهر الخطأ `CATEGORY_FIELD_MULTI_VALUE_IN_USE`؛ لذا أزِل تلك الاستخدامات أولًا. وانظر معارض البيانات والواجهات [بناء التطبيقات › معارض البيانات وأوجهها] للاطّلاع على إعدادات المعرض.

التجميع حسب حقل مرجع

التجميع حسب مرجع أحادي القيمة مدعوم بالكامل، ويختلف سلوكه قليلًا عن التجميع حسب حقل عادي:

- **مفتاح** المجموعة هو المُدخل المرتبط نفسه، لذا يمكنك إعادة استخدام المفتاح بوصفه قيمة ترشيح دون معرفة كيفية تخزين المرجع.
- **عنوان** المجموعة هو العنوان الرئيسي لذلك المُدخل المرتبط، ويُحَلّ على الخادم ضمن الاستعلام نفسه بعملية بحث واحدة لكل مجموعة، لا لكل صف.
- وإذا كان المُدخل المرتبط قد حُذف أو أُرشِف، أو كان في مخزن لا يستطيع هذا القارئ فتحه، **تحتفظ المجموعة بمُدخلاتها وتظهر بلا عنوان فحسب.** وهذا سلوك مقصود، وليس خطأً؛ فمُدخلات تلك المجموعة متاحة للقارئ، ولهذا وُجدت المجموعة أصلًا، وحذف المجموعة كلّها لإخفاء عنوانها سيُخفي بيانات يحقّ للقارئ الاطّلاع عليها.
- **ترتيب المجموعات** في مستوى مرجع يتبع العنوان الذي جرى حلّه. وخلافًا لمستوى تصنيف أو مُحوّل، ليس للمرجع ترتيب خيارات أعدّه المصمّم ليُرجَع إليه.
- أمّا ترتيب *الصفوف* حسب عنوان المرجع فيتطلّب، مع كل طلب، المرور على المُدخلات المرتبطة المتمايزة، بحدّ أقصى يبلغ **500** مُدخل مرتبط متمايز. وبعد هذا الحدّ، يظلّ تجميع القائمة وحساب إجمالياتها صحيحين، ويُفقَد فقط ترتيب الصفوف حسب العنوان في ذلك المستوى.

التجميع حسب حقل نص مترجم

يدعم حقل النص المترجم التجميع، ويجري التجميع حسب **اللغة التي يقرأ بها القارئ**.

ولهذا نتيجة عملية ينبغي توضيحها لمن يطلب هذا المعرض: **شخصان يقرآن القائمة نفسها بلغتين مختلفتين يريان مفاتيح مجموعات مختلفة**، بل قد يريان عددًا مختلفًا من المجموعات. فإذا تشارك مُدخلان القيمة الإنجليزية واختلفا في العربية، اجتمعا لدى قارئ وافترقا لدى آخر. وهذا هو النهج نفسه الذي تتّبعه المنصّة أصلًا عند ترتيب حقل مترجم، مما يُبقي مجموعات القائمة متّسقة مع ترتيب إظهار صفوفها.

ولأن المفتاح *هو* النص نفسه، فلا يوجد عنوان منفصل يحتاج إلى حلّ. ويُدرَج المُدخل الذي لا قيمة له بلغة القارئ الحالية في مجموعة «بلا قيمة» لدى ذلك القارئ.

أيّ تجميع يفوز، وكم يدوم

القاعدة واضحة، ولها شقّان مختلفان في طبيعتهما:

- **إعداد المصمّم دائم، ويفوز دائمًا عند التحميل.** فهو مخزَّن في المعرض كأيّ إعداد آخر، ويبدأ به كل قارئ يفتح المعرض.
- **وتجميع القارئ الخاص له وحده، ولا يدوم إلا للجلسة.** لا يُخزَّن منه شيء على الخادم، وإعادة فتح المعرض في سياق جديد تعيد التجميع المُعدّ للمعرض؛ أي إن اختيار القارئ يُلغى عند إعادة الفتح.

وفيما يلي معنى «الجلسة» عمليًا لقارئ اختار تجميعه الخاص:

الحدث	هل يبقى اختياره؟
إعادة تحميل الصفحة	نعم
الانتقال إلى موضع آخر داخل البوابة ثم العودة	نعم
تبويب أو نافذة متصفّح جديدة	لا
تسجيل الخروج	لا
جهاز آخر، أو زيارة لاحقة	لا
تعديلك لتجميع المعرض	**لا – يُسقَط اختياره**

وللصفّ الأخير أهمّية عند تعديل المعرض؛ فهو يضمن ألّا يستمرّ أيّ قارئ في استخدام تجميع لم يعد يطابق المعرض الذي يراه. وكانت نسخة سابقة تحفظ تجميع القارئ على الخادم، فتمنحه الأولوية دائمًا على أيّ تعديل لاحق تجريه على المعرض، مما يؤدّي إلى تعارض بين قائمة القارئ ومعاينتك. وقد أُزيل ذلك السلوك، وأصبح إعداد المعرض المصدر الدائم الوحيد.

كيف تُرتَّب المجموعات

عند ترتيب قائمة مُجمَّعة، تُرتَّب المجموعات والصفوف داخلها:

- **دون ترتيب مطبَّق**، تتبع المجموعات الترتيب الخاص بمفاتيحها: ترتيب الخيارات الذي أعددته لمستوى تصنيف أو مُحوّل، والعنوان الذي جرى حلّه لمستوى مرجع.
- **وبمجرّد أن يرتّب القارئ**، تُرتَّب كل مجموعة بقيمة واحدة مأخوذة من الحقل المستخدم في الترتيب: **أكبر** قيمة في المجموعة عند الترتيب التنازلي، و**أصغرها** عند الترتيب التصاعدي. وهي القيمة التي ستظهر *أولًا داخل المجموعة* وفق اتجاه الترتيب الذي اختاره القارئ؛ فأيّ اختيار آخر سيضع مجموعةً قبل أخرى تحتوي صفًّا ظاهرًا أعلى منها في الترتيب، فيبدو كأن التجميع ألغى الترتيب.
- **ولا يرتّب المجموعاتِ إلا حقل الترتيب الأساسي.** أمّا حقل الترتيب الثانوي فيرتّب الصفوف داخل المجموعة؛ إذ إن ترتيب المجموعات بقيمة تتساوى فيها معظمها لن يُحدِث فرقًا في ترتيبها.
- **وتُحسب قيمة الترتيب فوق الصفوف التي يستطيع القارئ رؤيتها فعلًا لا غير**: أي بعد تطبيق المرشحات والبحث والمرشحات السريعة والأذونات والترشيح المسبق. فلا يستطيع صف استُبعِد بالترشيح تقديم مجموعته إلى رأس القائمة، ولا يمكن أن يخالف ترتيب رؤوس المجموعات ترتيب مجموعات الصفوف تحتها.

الحقول المحمية

تلتزم المحصّلات والتجميع بحماية الحقول، كلٌّ بالطريقة التي تكشف أقلّ قدر من المعلومات. انظر الحقول الحساسة [بناء التطبيقات › حماية حقل] لمعرفة كيف يصبح الحقل حسّاسًا أو مشفَّرًا، وكيف تعمل المنح وجلسات الكشف.

- **تبقى المحصّلة فوق حقل حسّاس ظاهرةً، باسم عرضها، مع حجب قيمتها على هيئة `****`.** فإزالتها تكشف معلومات بحدّ ذاتها؛ إذ يُخبر غيابها القارئ بأن الحقل موجود وأنه محميّ. أمّا إبقاؤها بهذه الصورة فيُعلِمه بوجود رقم قد لا يحقّ له الاطّلاع عليه، وهذا هو القدر المقصود من المعلومات.
- **والمحصّلة فوق حقل مشفَّر لا تُتاح أصلًا**؛ فلا يمكنك تعريفها، ومن ثمّ لا توجد محصّلة لإخفائها، ولا رقم خلف القناع لحجبه.
- **والتجميع حسب حقل محميّ – حسّاس أو مشفَّر – ينطوي إلى مجموعة واحدة مُقنَّعة** بالضبط، عنوانها `****`، وتضمّ كل مُدخل مطابق؛ ويشمل عدد مُدخلاتها ومجاميعها الفرعية جميع المُدخلات المطابقة.
  - وهو **موحَّد للجميع**: فالقارئ الذي يملك منحة الحقل *ومعها* جلسة كشف نشطة يرى المجموعة الواحدة نفسها التي يراها زائر مجهول. فالتجميع إعداد مشترك محفوظ في المعرض، وليس إعدادًا خاصًّا بكل قارئ؛ لذا لا يمكن الجمع بين «قابلية التجميع لكل مُشاهِد» و«الإعداد المشترك».
  - **والمُدخلات التي لا قيمة لها في الحقل تقع في المجموعة نفسها** عمدًا، ولا تُدرَج في مجموعة «بلا قيمة» منفصلة إلى جانبها. فالمجموعة المنفصلة ستكشف لكل قارئ المُدخلات التي يحتوي فيها الحقل على قيمة؛ وغالبًا ما تكون هذه المعلومة نفسها حسّاسة في حقل راتب أو حقل طبّي أو تأديبي.
  - كذلك لا تكشف المجموعة الواحدة شيئًا عن خصائص قيم الحقل: لا عدد القيم المتمايزة فيه، ولا كيفية توزيع المُدخلات عليها، ولا ترتيبها النسبي؛ وهي معلومات كان سيكشفها وجود عدّة رؤوس مجموعات محجوبة.
  - وهذا المستوى **إعداد صالح وقابل للحفظ**؛ فلم تعد أسس ترفضه. ويظهر هذا المستوى، لكن دون فائدة، ويمكنك ملاحظة ذلك في المصمّم والتراجع عنه.
- **وعلى معرض يستطيع زائر مجهول فتحه**، تُستبعَد المحصّلة فوق حقل حسّاس بالكامل بدلًا من حجب قيمتها؛ إذ تحجب المنصّة حتى تعريفها، لأن معرّفها واسم عرضها واسم حقلها كلّها تصف حقلًا حسّاسًا لشخص لا تستطيع المنصّة إسناد أيّ شيء إليه.

وفي المصمّم، يعرض منتقي التجميع الحقل الحسّاس، فيُطوى مستواه كما سبق، لكنه يُخفي الحقل المشفَّر، أو الحقل الذي يجري تشفيره؛ أمّا المستوى المحفوظ سابقًا على مثل هذا الحقل فيظلّ مطويًّا، ويُشار في المصمّم إلى أنه لم يعد متاحًا.

سطح واجهة البرمجة

تَرِد كل استجابة أدناه ضمن الغلاف المعتاد `{success, result}`؛ انظر استدعاء واجهة HTTP.

تعريفات المحصّلات

الطريقة	المسار	الغرض
`POST`	`/v1/entities/{entity}/aggregates`	إنشاء محصّلة على الكيان
`GET`	`/v1/entities/{entity}/aggregates`	سرد محصّلات الكيان (بلا ترقيم صفحات)
`PATCH`	`/v1/entities/{entity}/aggregates/{id}`	تعديل محصّلة واحدة؛ وتُقبَل الحمولة الجزئية
`DELETE`	`/v1/entities/{entity}/aggregates/{id}`	حذف محصّلة واحدة؛ وتستبعدها المعارض التي فعّلتها دون خطأ
ويقبل `{entity}` معرّف الكيان أو اسمه النموذجي.		

لا يوجد مسار نسخ على الخادم `Removed 1.10`. كانت إصدارات 1.10 السابقة للإطلاق تتضمّن `POST /v1/entities/{entity}/aggregates/{id}/copy`، ثم أُزيل قبل الإطلاق. وأصبح أمر *نسخ* في المصمّم يفتح نافذة الإنشاء معبّأةً بقيم المحصّلة الأصلية (مع إلحاق « - Copy» باسمها)، ويحفظ عبر المسار المعتاد `POST`

`/v1/entities/{entity}/aggregates`: لذا ينسخ المُكامِل المحصّلة بالطريقة نفسها: يقرؤها، ثم ينشئ محصّلة جديدة بالقيم نفسها لـ `field` و`aggregationFunction` و`name`.

ملخّص المُدخلات

تَرِد الإجماليات ورؤوس المجموعات في استجابة واحدة، عبر أحد ثلاثة مسارات `GET` بحسب السياق:

المسار	السياق
`/v1/entities/{entity}/entries/summary`	قسم الإدارة، دون ربط بمعرض
`/v1/entities/{entity}/views/{id}/entries/summary`	معرض منشور أو معرض في البوابة
`/v1/entities/{entity}/entries/{entryId}/relations/{relationId}/summary`	قائمة مُدخلات مرتبطة

وتشترك هذه المسارات في المعاملات التالية: `aggregates` و`groupBy` و`groupSkip` و`groupLimit` و`filterCriteria` و`search` و`include_virtual` و`appliedQuickFilters`. ويقبل المسار المُقيَّد بالمعرض أيضًا `filterFirst` و`parameters`، فيما يقبل المسار على مستوى الكيان أيضًا `collectionId`.

- **واختيارك للمسار حدّ أمني**، وليس مجرّد تسهيل: فالمسار المُقيَّد بالمعرض يحصر المحصّلات في المجموعة التي فعّلها المعرض، أمّا المسار على مستوى الكيان فلا يفعل ذلك.
- ويُؤخَذ تقاطع `aggregates` مع مجموعة المحصّلات المفعَّلة في المعرض، ولا يُؤخَذ اتحادهما أبدًا. فإغفاله يعني «مجموعة المحصّلات المفعَّلة في المعرض»، وإرساله فارغًا يعني «لا شيء»؛ ولكلّ منهما نتيجة مختلفة.
- ولا يُعتدّ بـ `groupBy` إلا حين يسمح المعرض بتجميع المستخدم النهائي؛ وإلا استُخدمت مستويات المعرض نفسه. **وإرساله فارغًا لا يعني «بلا تجميع»**؛ إذ تؤدّي القائمة الفارغة إلى استخدام المستويات المُعدَّة. ولهذا تعطّل تطبيقات العميل الخاصة بالمنتج العرض المُجمَّع من جانبها.
- ويقسّم `groupSkip` / `groupLimit` رؤوس المجموعات إلى صفحات حين يرسلهما المُكامِل. **ولا يرسلهما عميل الويب ولا عميل الهاتف أبدًا**؛ لذا لا تبنِ تصميم المعرض على ترقيم صفحات المجموعات، وخطّط بدلًا من ذلك لمستوى تجميع يضمّ عددًا من القيم المتمايزة يمكن التعامل معه.

ويُتاح معامل `groupBy` الاختياري نفسه على مسارات `v3` لسرد المُدخلات. وهو لا يغيّر هناك محتوى النتائج، بل يغيّر **الترتيب** وحده، بحيث تَرِد كل مُدخلات المجموعة متجاورة؛ وهذا ما يحتاجه جدول مُجمَّع برؤوس فرعية لإظهار رأس واحد لكل مجموعة.

تحميل معرض المستخدم النهائي

يتضمّن `GET /v2/views/end-user/{id}` الخاصية **`availableAggregates`**؛ أي *تعريفات* المحصّلات (المعرّف والاسم والحقل والدالّة) التي يحتاجها المعرض في البوابة لبناء قائمة الاختيار الخاصة به، لأن المحصّلة غير المفعَّلة لا قيمة محسوبة لها يمكن الاستناد إليها لإدراجها في القائمة. أمّا `GET /v1/views/end-user/{id}` الأقدم فلا يتضمّنها.

ويُضيّق الخادم نطاق هذه القائمة مسبقًا؛ فهي مجموعة المحصّلات المفعَّلة في المعرض بعد استبعاد كل ما لا يحقّ لهذا المُستدعي الاطّلاع عليه. **فلا تُعِد تقاطعها مع مصفوفة `aggregates` في المعرض**: فقد يكون المعرّف موجودًا في إعداد المعرض على نحو صحيح، لكنه يغيب عن هذه القائمة، وإعادة الترشيح لن تفعل سوى إعادة تعارض حسمه الخادم. وإذا تعذّر حلّ القائمة، أُعيدت فارغة بدلًا من إفشال تحميل المعرض.

وما لا تغطّيه أيّ نقطة نهاية: لا يتضمّن **تصدير** المُدخلات تجميعًا أو إجماليات؛ فتصدير XLSX لا يتيح معامل `groupBy` أو `aggregates`، والتصدير عبر قالب يعرض المُدخلات دون الملخّصات. وينطبق ذلك على كل مسارات التصدير، بما فيها التصدير السريع من المعرض في 1.10 (XLSX أو CSV؛ انظر معارض البيانات والواجهات [بناء التطبيقات ‹ معارض البيانات وأوجهها]):

فمعامل `groupBy` المُرسَل إليه **يرتّب** الصفوف فقط بحيث تتجاور مُدخلات كل مجموعة، ولا يكتب ترويسات مجموعات أو إجماليات. فإذا كان لا بدّ من إخراج إجمالي من المنصّة، فاحسبه من استدعاء ملخّص بدلًا من توقّع وجوده في الملف.

ملاحظات الإصدار

كل ما تتناوله هذه الصفحة – تعريفات المحصّلات، وتفعيلها لكل معرض، وتجميع المُدخلات، ومسارات الملخّص، و`availableAggregates` – متاح **Since 1.10**. ولا يتوافر أيّ منه في الإصدار 1.8 أو أيّ إصدار أسبق. أمّا الإصلاح الوحيد الذي يستحقّ ذكره على حدة فهو الآتي: `Since 1.10`، صُحِّحت `Count` لتَعدّ المُدخلات التي لها قيمة في الحقل المضبوط، بدلًا من عدّ كل مُدخل مطابق بصرف النظر عن قيمة الحقل – انظر الدوال وأنواع الحقول التي تقبلها أعلاه.

مرجع واجهة برمجة قوالب التصدير

هذه الصفحة موجّهة إلى **المُكامِلين والمطوّرين** الذين يصدّرون المُدخلات باستخدام قالب عبر الواجهة البرمجية، أو يقرؤون تعريفات القوالب أو يكتبونها، أو يحتاجون إلى معرفة كيفية تصيير القالب بدقّة.

هل تصمّم القوالب من الواجهة؟

إن كنت من بُناة التطبيقات، فابدأ بصفحة قوالب التصدير [بناء التطبيقات]؛ إذ تشرح كتابة ملف القالب، وتبويب **القوالب**، وإعدادات المعرض بالتسميات التي يعرضها المصمّم.

مفهومان للقالب

تُعرّف أسس **مفهومين مختلفين للقالب**، يُخزَّن كلاهما داخل مستند `DataEntity` (لا في مجموعات مستقلة):

1. **قوالب التصدير** – ملفات Word أو HTML أو Excel مرفوعة تملؤها أسس ببيانات المُدخل لإنتاج مستندات DOCX أو PDF أو XLSX، وهي التي تشغّل نقطة النهاية `entries/export/{templateId}`.
2. **قوالب العرض ومعاملات القوالب** – `templateId` مع مجموعة من الارتباطات المضبوطة بمعاملات (`TemplateParameter`) تُلحَق بكيان أو بمعرض، وتحدّد عبر هذه المعاملات طريقة تصيير المُدخل أو القائمة.

المُخرجات المدعومة هي **DOCX وHTML وXLSX**، مع التحويل **من DOCX/HTML إلى PDF**. ولا تُنتج تصديرات القوالب ملفات CSV أبدًا؛ إذ لا تتاح صيغة CSV إلا في التصديرين السريعين لمعرض أو لمُدخلات كيان (`format=xlsx|csv`، `Since 1.10`؛ انظر التصدير السريع لمُدخلات كيان [بناء التطبيقات › إنشاء كيان وتصميمه]). وتُملأ قيم القالب باستخدام رموز وتعبيرات تُحَلّ بالرجوع إلى المُدخل وقت التصدير، ويمكن استخدام JavaScript والدوال المخصّصة داخل القالب.

لا يصيّر هذا المحرّك التقارير. فقد أدرجت نسخة سابقة من صفحة بُناة التطبيقات التقاريرَ ضمن مواضع استخدام القوالب، وذكرت أنها «تُصيَّر بالمحرّك نفسه». أما في الرماز، فيُصيَّر التقرير عبر `ReportService` ← `ExportService` (`data-service/src/services/report.service.ts`)، لا عبر `DataEntryTemplateExportService`؛ انظر التقارير [بناء التطبيقات].

الحقول الحسّاسة في تصدير القالب `Since 1.10`

يحمل القالب سياسته الخاصة `sensitiveDataAccessPolicy` (`Prohibited` أو `Privileged` أو `User-Based` أو `Auto`؛ وغيابها يعني `Privileged`). وتُعطّل قراءة القالب الإخفاء المرتبط بنطاق المستدعي (`disabledStages.hideSensitiveFields: true`)؛ فإذا كانت النتيجة النهائية لسياسة القالب لا تتيح الكشف الكامل، مرّرت بدلًا منه `elementSensitiveAccess` مع نطاقات المنح المحلولة. وتسري سياسة القالب نفسه حتى عندما يصيّره إجراء **إنشاء ملف** في تفاعل (فسياسة التفاعل لا تتجاوزها). وإذا تعذّر تحديد المنفّذ، أخفى قالبُ `User-Based` أو `Auto` القيم (إغلاق عند الفشل). أما الحقول المشفّرة فيُخفيها `maskFieldsExcludedFromBuilders` دون شرط قبل تقييم أي تعبير أو تصيير أي شيء. انظر سياسة الوصول إلى البيانات الحسّاسة [بناء التطبيقات › حماية حقل].

صياغة ملف القالب

ترد صياغة قوالب HTML (EJS) وWord وExcel، مع الأمثلة، في قوالب التصدير – كتابة ملف القالب [بناء التطبيقات › قوالب التصدير].

المفاهيم ونموذج البيانات

قالب التصدير (DataEntityExportTemplate)

تُخزَّن قوالب التصدير في `DataEntity.templates` (`IDataEntityExportTemplate[]`). ويشير كل قالب إلى ملف مصدر مرفوع، ويحدّد طريقة تصييره وتسليمه.

الحقل	النوع	إلزامي	المعنى
`id`	UUID	نعم	معرّف القالب (يُستخدم بوصفه `templateId` في نقطة نهاية التصدير)
`name`	`LocalizedValue`	نعم	اسم القالب المترجَم، وهو أيضًا الاسم الافتراضي لملف المُخرَج
`level`	`TemplateLevelEnum`	نعم	`SingleEntry` (0) أو `MultipleEntries` (1)
`type`	`TemplateTypeEnum`	نعم	صيغة المُخرَج: `Word` (0)، `PDF` (1)، `Excel` (2)
`file`	`FileModel`	نعم	ملف القالب المصدر المرفوع (يحتوي على `id` و `mimeType` و `name`)
`exportType`	`TemplateExportTypeEnum`	نعم	`UI` (0) = تنزيل متزامن، `Background` (1) = غير متزامن عبر ناقل الرسائل
`additionalProperties`	`Record<string,string>`	لا	متغيّرات مخصّصة (تعبيرات) وأعلام ميزات مثل `extended`
`query`	مرجع أو `null`	لا	`Since 1.10` . استعلام محفوظ يجب أن يستوفيه المُدخل (لمستوى المُدخل الواحد فقط) – انظر التصدير الشرطي
`criteria`	`FilterExpression`	لا	`Since 1.10` . مرشِّح يجب أن يستوفيه المُدخل (لمستوى المُدخل الواحد فقط)
`nonMatchEntriesMode`	تعداد	لا	`Since 1.10` . `Disable` (0، الافتراضي) أو `Hidden` (1) للمُدخلات التي لا تستوفي الشرط
`sensitiveDataAccessPolicy`	`SensitiveDataAccessPolicyEnum`	لا	`Since 1.10` . ما يجوز للقالب كشفه من الحقول الحسّاسة؛ وغيابه يعني `Privileged` . انظر الحقول الحسّاسة في تصدير القالب

ملاحظات جرى التحقّق منها بالرجوع إلى الرماز:

- `type` (صيغة **المُخرَج** المطلوبة) مستقلّ عن نوع MIME للملف **المصدر**. ففي حالة `type === PDF` تتفرّع الخدمة بحسب `file.mimeType` : فمصدر DOCX يُصيَّر ثم يُحوَّل من DOCX إلى PDF، ومصدر HTML يُصيَّر ثم يُحوَّل من HTML إلى PDF. ولا يُنتج قالب PDF أي ملف إذا كان للمصدر نوع MIME آخر.
- يتحكّم `level` في المساعِدات الخاصة بكل مُدخل؛ إذ تعيد `getEntryComments` و `getEntryEmails` القيمة `[]` ما لم يكن `level === SingleEntry` (انظر الدوال المخصّصة).

محرّكات القوالب بحسب النوع

تحدّد صيغة المُخرَج محرّك القوالب المستخدَم:

`type`	الملف المصدر	المحرّك	المحدِّدات
`Word`	DOCX	`docx-templates` `createReport`	`{{` … `}}` (`cmdDelimiter`)، `fixSmartQuotes: true`
`PDF` (من DOCX)	DOCX	`docx-templates` ، ثم تحويل DOCX→PDF	`{{` … `}}`
`PDF` (من HTML)	HTML	`ejs.render` (غير متزامن)، ثم تحويل HTML→PDF	`openDelimiter: '{{'` ، `closeDelimiter: '}}'`
`Excel`	XLSX	`xlsx-template` (`substitute(1, data)`)؛ أما القالب القابل للبثّ فيكتبه كاتب البثّ في ExcelJS بدلًا منه (`Since 1.4` ، انظر التصديرات الكبيرة)	الافتراضي في المكتبة (عناصر نائبة `${...}`)

كائن البيانات المُمرَّر إلى كل محرّك هو `{ entries, entity }` (إضافةً إلى مجموعات أخرى في التصدير الموسّع – انظر أدناه). أما في Excel فتُسطَّح المُدخلات أولًا بواسطة `mapEntriesForExcelTemplate`؛ إذ تُختزَل مصفوفات المراجع المحلولة إلى سلاسل `mainTitle` مفصولة بفواصل لكل لغة متاحة، وتُستبدَل محارف الجدولة العمودية بمسافات.

معاملات القالب (`TemplateParameter`)

Since 1.4.

تستخدم قوالب العرض والقوالب المضبوطة بمعاملات `TemplateParameter`، المعرَّف في `models/template-parameter.model.ts` وفي `@osos/shared`.

الحقل	النوع	إلزامي	المعنى
`name`	string	نعم	اسم المعامل، ويجب أن يطابق اسم المعامل في القالب
`valueType`	`TemplateParameterValueTypeEnum`	نعم	`Fixed` (0)، `Expression` (1)، `Field` (2)
`value`	any	لا	قيمة ثابتة، أو تعبير مُدخل مثل `value.my_block[scope.index].my_field` حين يكون النوع `Expression`
`component`	string	لا	مرجع اختياري إلى مكوّن
`source`	string	لا	مرجع اختياري إلى مصدر

دلالات `valueType` كما يوردها توثيق JSDoc للنموذج: **Fixed** – تُستخدم `value` كما هي؛ **Expression** – `value` تعبير مُدخل يُقيَّم لإنتاج القيمة؛ **Field** – نوع القيمة `2` (عضو التعداد موجود، ويتولّى المستهلكون حلّه).

تُجمَّع المعاملات بحسب سياق التصيير في `TemplateParameterTypes`:

```
interface TemplateParameterTypes {
  listEntry?: TemplateParameter[];   // list rendering
  viewEntry?: TemplateParameter[];   // read/view rendering
  editEntry?: TemplateParameter[];   // edit rendering
}
```

تُلحَق هذه الحزمة في موضعين، كلاهما بالشكل `{ templateId, parameters }`:

- `EntityDisplayTemplate` → `DataEntity.displayTemplate`
- معرض بيانات → `ViewDisplayTemplate`

نقاط النهاية في الواجهة البرمجية

تصدير المُدخلات بقالب

```
GET /v1/entities/{entity}/entries/export/{templateId}
```

المتحكّم: `DataEntryTemplateExportController.exportEntriesWithTemplateV2`. يتطلّب المصادقة (استراتيجية `Auth2`)، ويعيد ملفًا للتنزيل (`application/vnd.openxmlformats-officedocument.spreadsheetml.sheet` أو `...wordprocessingml.document` أو `application/pdf`) أو إقرارًا بصيغة JSON.

معاملات المسار:

الاسم	الموضع	المعنى
`entity`	path	معرّف كيان البيانات
`templateId`	path	معرّف قالب التصدير (الحقل `id` في `DataEntity.templates[*]`)

معاملات الاستعلام:

الاسم	النوع	المعنى
`mode`	number (`ExportWithTemplateSourceMode`)	`SelectedItems` (0) أو `source` (1)
`itemsToExport`	[]string	معرّفات المُدخلات المراد تصديرها؛ وهو **إلزامي** حين يكون `mode === SelectedItems`
`filterCriteria`	object	معايير الترشيح المطبَّقة حين يكون `mode === source`
`search`	string	بحث نصّي حرّ (يُتجاهَل، أو تُفرض له القيمة `undefined`، في وضع `SelectedItems`)
`sourceCollection`	string	معرّف المجموعة؛ وتُضاف عناصرها `items` بوصفها مرشِّحًا من نوع `_id IN (...)` في وضع `source`
`appliedQuickFilters`	[]string	معرّفات المرشِّحات السريعة

السلوك (جرى التحقّق منه):

1. يحمّل الكيان، ثم يستدعي `getEntityTemplate(dataEntity, templateId)`، فيرمي الاستثناء `TemplateNotExists` (HTTP 404، الرمز `TEMPLATE_NOT_EXIST`) إن لم يوجد على الكيان قالب بهذا المعرّف.
2. **القوالب الموسّعة** – إذا كان `additionalProperties.extended === 'true'`، يُحال الطلب إلى `DataEntryExtendedTemplateExportService.exportFromTemplate` (انظر التصدير الموسّع)، ويكون التصدير متزامنًا دائمًا.
3. **التصدير في الخلفية** – وإلا، فإذا كان `exportType === Background`، يرسل `ExportTemplateInBackgroundCommand` عبر ناقل الرسائل ويعيد فورًا `200 { "success": true }`. ويولّد `ExportTemplateInBackgroundHandler` الملف على نحو غير متزامن، ثم يُسلَّم الملف عبر `FileGeneratedEvent`.
4. **التصدير من الواجهة** – وإلا فإنه يصيّر الملف على نحو متزامن عبر `exportEntriesWithTemplate`، ويرفع النتيجة إلى التخزين، وينشر `FileGeneratedEvent`، ويضبط ترويسَتَي `Content-Type` والمرفق، ثم يبثّ محتوى الملف في الاستجابة.

وفي الحالتين 2 و4 يُرفع الملف المولَّد أيضًا إلى خدمة التخزين ويُنشر `FileGeneratedEvent` (يحمل الملف، ومعرّف القالب، و`exportType`، ومعرّف الكيان، وقائمة معرّفات المُدخلات المصدَّرة).

حالات الخطأ:

الحالة	الاستثناء	HTTP
لا يوجد على الكيان قالب بالمعرّف `templateId`	`TemplateNotExists`	404
`mode === SelectedItems` مع `itemsToExport` فارغ أو غائب	`MustProvideITemsToExportProperty`	400
المُدخل لا يستوفي شرط القالب، أو استُدعي قالب ذو شرط في وضع `source` (`Since 1.10`)	`TEMPLATE_CONDITIONS_NOT_MET`	422

التحقق مما إذا كان القالب مستخدَمًا

```
GET /v1/templates/{id}/used
```

المتحكّم: `TemplatesController.isTemplateUsed`. يعيد قيمة منطقية. *Long-standing*.

تتعامل نقطة النهاية هذه مع القوالب المخزَّنة بوصفها **مُدخلاتِ** بيانات في الكيان الداخلي `templates` (`StringConstants.TEMPLATES_ENTITY_NAME = 'templates'`)، أي قوالب الرسائل والبريد الإلكتروني التي تشير إليها مشغّلات الكيانات، ولا تتعامل مع قوالب التصدير `DataEntityExportTemplate` المذكورة أعلاه.

المنطق (جرى التحقّق منه):

1. البحث عن مُدخل القالب بالمعرّف `id` ، وإعادة `false` إن لم يوجد.
2. إذا كان `value.template_type.id === 'EMAIL_PART_TEMPLATE'` للقالب، فالبحث عن القوالب الرئيسة التي تشير إلى هذا الجزء بوصفه ترويسةً أو تذييلًا لها (`value.template_details.template_header.dataEntry` / `template_footer.dataEntry` يساوي `id`)، وجمع معرّفاتها. وإلا فمجموعة المرشَّحين هي معرّف هذا القالب وحده.
3. إعادة `true` إن كان لأي `DataEntity` إجراء مشغِّل مرتبط بأحد معرّفات تلك القوالب (`triggers.actions.options.template IN (...)`)، وإلا إعادة `false` .

مراحل التصيير

`exportEntriesWithTemplate` هو المُصيِّر الأساسي الذي يستخدمه مسار التصدير من الواجهة، ومعالِج الخلفية، والمعالِج الداخلي `GenerateFileFromTemplateHandler` . وينفّذ الخطوات الآتية:

1. **جلب الملف المصدر** من التخزين بصيغة `arraybuffer` .
2. **تحديد اسم ملف المُخرَج** – باستخدام الاسم المُمرَّر `selectedFileName` ، وإلا فاسم القالب المترجَم `name` .
3. **جمع بيانات التصدير** (`getTemplateExportData`):
   - `mode === source` : يبني المرشِّحات من عناصر المجموعة المصدر `items` (`_id IN`) ومن أي معايير في `filterCriteria.criteria` .
   - `mode === SelectedItems` : يشترط وجود `itemsToExport` ، ويفرض `search = undefined` ، ويرشّح باستخدام `_id IN itemsToExport` .
   - **التصدير الذي يُطلقه معرض يقرأ عبر أعمدة المعرض نفسه.** فالحقول `fields` المهيّأة في المعرض هي الأعمدة التي يعرضها، ويستخدم `buildScope` القائمة نفسها لتقييد حقول العلاقات وADT التي يحلّها، ولبناء `$project` الخاص بالقراءة؛ لذا إذا أشار القالب إلى حقل لا يعرضه المعرض المُطلِق، صُيِّر ذلك الحقل فارغًا (قيد معروف لم يُعالَج بعد). وقد ذكرت نسخة سابقة من هذه الصفحة أن نطاق التصدير وُسِّع ليشمل كل حقل غير حسّاس `Since 1.8` ؛ ثم أُلغي هذا التوسيع، لأن `$project` الذي يسمّي كتلة وحقلها الفرعي معًا (`value.address` و `value.address.street`) يُحدث في MongoDB تصادمًا بين المسارات، مما أدّى إلى إنتاج ملف خالٍ من أي مُدخل (`data-entry-template-export.service.ts` ، توثيق JSDoc للدالة `getTemplateExportData`). أما القيود التي يفرضها المعرض على الصفوف (قواعد المتقادم وغير الصالح والمؤرشف والبحث، ومصدر بيانات الاستعلام، والفرز الافتراضي) فما زالت سارية.
   - يشغّل `dataEntryService.findWithResolvesFactory` مع `disabledStages.hideSensitiveFields:` `true` – وهذا *يعطّل* الإخفاء المقيَّد بالمستدعي، فيعود القرار إلى سياسة الوصول إلى البيانات الحسّاسة الخاصة بالقالب (`elementSensitiveAccess`)؛ ثم تُخفى الحقول المشفّرة دون شرط.
   - إذا صرّح القالب بـ `additionalProperties` ، يقيّمها لكل مُدخل (انظر المتغيّرات المخصّصة).
4. **التصيير** بالمحرّك المطابق لـ `type` (انظر الجدول أعلاه)، مع حقن الدوال المخصّصة المشتركة.
5. **الإعادة** `{ fileName, fileContent, mimeType, exportEntriesIds }` .

المتغيّرات المخصّصة

Long-standing.

`additionalProperties` من النوع `Record<string, string>`، وكل قيمة فيه تعبير مغلَّف بـ `{{%= … %}}`. وفي أثناء جمع البيانات، يزيل `parseAdditionalProperties` علامتَي التغليف `{{%=` / `%}}`، ويقيّم التعبير الداخلي لكل مُدخل عبر `executeExpression`، ثم يدمج النتائج في `entry.value` لكل مُدخل تحت مفتاح الخاصية، فتُتاح للقالب كأي حقل عادي.

المفتاح المحجوز `extended` (`additionalProperties.extended === 'true'`) **علَم ميزة** وليس متغيّرًا، إذ يوجّه التصدير إلى الخدمة الموسّعة.

الدوال المخصّصة في القوالب

تتلقّى كل عملية تصيير (`docx` و `ejs` / `html` وتقييم التعبيرات) مجموعة دوال مشتركة من `getSharedCustomFunctions(entity, templateConfig)`. وفيما يلي الدوال التي جرى التحقق منها:

الدالة	الغرض
`today(manipulateAmount?)`	التاريخ الحالي معدَّلًا بحسب المنطقة الزمنية، مع إمكانية إجراء حسابات على التاريخ
`date(value, formatString?, tzOffset?)`	تنسيق تاريخ (الافتراضي `Constants.DATE_FORMAT`) معدَّلًا بحسب المنطقة الزمنية
`loc(localizedValue)`	حلّ `LocalizedValue` إلى لغة المستخدم
`formula(text)`	لـ Excel فقط: الطريقة الوحيدة لإخراج خلية صيغة (السلسلة العادية التي تبدأ بـ `=` تبقى نصًّا)
`mdToText(value)`	تحويل Markdown إلى نص عادي (مع توحيد الشطب بموجة مفردة وفواصل الأسطر أيضًا)
`fetchEntry(ref)`	حلّ مُدخل واحد مُشار إليه من `{ dataEntity, dataEntry }`
`fetchEntriesById(entityId, entriesIds)`	حلّ عدة مُدخلات بمعرّفاتها
`fetchEntries(entityId, filter)`	حلّ مُدخلات باستخدام أي مرشِّح
`getStepName(entry, stepId)`	اسم خطوة سير العمل (مع الرجوع إلى الخطوات الإضافية (ad-hoc) إن لم يجدها)
`getCurrentStepOwner(entry)`	مالك الخطوة الحالية النشطة (إن كان له `mainTitle` محلول)
`getStepOwner(entry, stepId)`	مالك خطوة من سجل سير العمل
`getStepReceivedAt(entry, stepId?)`	الطابع الزمني لاستلام خطوة (يتعامل مع `recievedAt` / `receivedAt`)
`getStepActions(entry, stepId)`	الإجراءات المعرَّفة على خطوة
`getStepActionName(entry, stepId, actionId)`	اسم إجراء معيّن في خطوة
`getStepHistory(entry, stepId)`	جميع السجلات التاريخية لخطوة في سير العمل النشط
`getStepLastHistory(entry, stepId)`	أحدث سجل تاريخي لخطوة
`getEntryComments(entry, lang?)`	تعليقات المُدخل (مع حلّ الإشارات)؛ وتعيد `[]` ما لم يكن `level === SingleEntry`
`getEntryEmails(entry)`	رسائل البريد الإلكتروني المرتبطة بالمُدخل؛ وتعيد `[]` ما لم يكن `level === SingleEntry`

وتُتاح لعمليات تصيير DOCX وHTML، إضافةً إلى ذلك، الدوال الآتية:

- **DOCX** – تعيد `getImage(image, width, height, entryId?, alt?)` بايتات الصورة لتضمينها (الامتدادات المدعومة: `png` و `gif` و `jpg` و `jpeg` و `svg`؛ وتُسجَّل الامتدادات غير المدعومة وتُتخطّى)، وتصيّر `mdToHtml(value)` نص Markdown إلى HTML مغلَّف بـ `<body>…</body>` (مع دعم الشطب بموجة مفردة ومزدوجة).
- **HTML/EJS** – تعيد `getImageSrc(image, entryId?)` عنوان `data:` للصورة بترميز base64، وتُتاح أيضًا `mdToHtml(value)` (مهيّأة بـ `breaks: true`).

التصدير في الخلفية

حين يكون `exportType === Background`، تُدرج نقطة النهاية الأمرَ `ExportTemplateInBackgroundCommand` في الطابور (ويحمل معرّف الكيان، و`entityTemplate` كاملًا، و`mode`، و`itemsToExport`، و`sourceCollection`، و`filterCriteria`، و`search`، و`view`، و`appliedQuickFilters`) مع سمات لاصقة للمستخدم والمساحة والتفويض، ثم تعيد `{ success: true }`.

ويعيد `ExportTemplateInBackgroundHandler` تحميل الكيان، ويستدعي `exportEntriesWithTemplate` نفسها، ويرفع النتيجة، وينشر `FileGeneratedEvent` (مع `exportType = Background`) إضافةً إلى `CommandExecutedMessage` عبر المقبس. وتُسجَّل الأخطاء دون تمريرها (فلا يُنشر في هذا المعالِج أي حدث فشل عند وقوع خطأ).

داخلي: توليد ملف من قالب

يعالج `GenerateFileFromTemplateHandler` الأمر `GenerateFileFromTemplateCommand` (الذي تستخدمه إجراءات سير العمل والأتمتة، وليس مسار HTTP). إذ يحمّل معرضًا بواسطة `viewId` إن وُجد، ويحلّ الكيان والقالب، ويستدعي `exportEntriesWithTemplate` مع `command.fileName`، ويرفع الملف إلى التخزين إن ضُبط `uploadToStorageService`، ثم يعيد الملف (أو سجل الملف المرفوع) في `CommandExecutedMessage`. وعند وقوع خطأ، يُصدر `CommandExecutedMessage` مع `success: false` والخطأ.

التصدير الموسّع

Since 1.6.

يُستخدم `DataEntryExtendedTemplateExportService.exportFromTemplate` حين يكون `additionalProperties.extended === 'true'`. وهو يشغّل مراحل الجمع والتصيير نفسها، لكن في وضع `source` دائمًا، ويُثري كائن `data` الخاص بالقالب بسياق من نموذج الكيان يتجاوز `{ entries, entity }`:

```
{ entries, entity, module, reactions, notifications, workflows,
  lists, queries, views, relations }
```

تُحلّ هذه العناصر من كيان البيانات (`getModule`، `getReactions`، `getNotifications`، `getWorkflows`، `getLists`، `getQueries`، `getViews`، `getRelations`)، مما يجعل هذا التصدير مناسبًا لتوليد ملفات توثيق أو مواصفات لتهيئة الكيان نفسها. وتشترك الخدمة الموسّعة مع الخدمة الأساسية في اختيار المحرّك ومجموعة الدوال المخصّصة. ويعمل هذا المسار تزامنيًا دائمًا (فلا فرع له للتنفيذ في الخلفية).

تصديرات Excel المبثوثة `Since 1.4`

يُكتب تصدير قالب Excel لعدد كبير من المُدخلات **صفحةً صفحة** (500 مُدخل في كل مرة، وتُكتب كل صفحة وتُحرَّر قبل قراءة التالية) متى كانت بنية القالب بسيطة بما يكفي؛ وأي بنية أخرى تُصيَّر في الذاكرة كما في السابق. ويتّضح الفرق مع ازدياد الحجم: ففي القياس المرافق للتغيير، استغرق تصدير 25,000 مُدخل 12.8 ثانية بدلًا من 323 ثانية، باستخدام نحو ربع الذاكرة، وأصبح حجم الملف 3 ميغابايت بدلًا من 24 ميغابايت. وقد سُلّم هذا أيضًا إلى الخطوط 1.4 و1.6 و1.8 بوصفه تحديثًا.

لا يُبثّ القالب إلا إذا تحقّقت **كل** الشروط الآتية:

- يحتوي على ورقة واحدة بالضبط.

- يملأ كل عنصر نائب خلية كاملة، ويكون بصيغة `${table:entries.<key>}` البسيطة (بلا نوع فرعي)، وتوجد العناصر النائبة كلها في الصف نفسه؛ ولا يحتوي ذلك الصف على أي شيء آخر.
- لا توجد أي قيمة تحت ذلك الصف، ولا يمتدّ أي نطاق مدموج إلى صفّ الجدول أو ما تحته.
- لا تحتوي الورقة على صيغ ولا صور ولا جداول Excel ولا تنسيق شرطي ولا تحقق من البيانات ولا مرشِّح تلقائي ولا أسماء معرَّفة.

وحين لا يستوفي القالب الشروط، تسجّل data-service السبب بمستوى INFO (`template <id> of entity <id> is … rendered in memory: it`)، مما يوضّح ما يجب إزالته. ويستطيع المشغّلون إعادة كل القوالب إلى التصيير في الذاكرة باستخدام الإعداد `TEMPLATE_EXPORT_XLSX_STREAMING_DISABLED=true`. وفي المسارين، تبقى قيمة المستخدم التي تبدأ بـ `=` نصًّا؛ ولا تُكتب أي قيمة بوصفها صيغة إلا إذا أنتجتها دالة التعبير `formula()`.

القوالب على الكيانات الوهمية `Since 1.6`

يتاح تبويب القوالب أيضًا على **الكيانات الوهمية** مثل المجموعات؛ إذ يمكن إدارة القوالب على كيان وهمي إن لم يكن ضمنيًا، أو كان ضمنيًا لكنه يسمح بالتحرير (`implicitOptions.allowEdit`)؛ وإلا فتسري قاعدة التحرير العادية للكيان. ويُحفظ القالب على كيان وهمي من نوع الوكيل أو الموصِّل أو المجمِّع عبر نقطة نهاية الكيان الوهمي (`PUT /v1/entities/virtual/{id}`). ويتضمّن تصدير القالب على كيان وهمي الصفوفَ التي حدّدها المستخدم. وقد سُلِّم الأمران أيضًا إلى الخطّين 1.6 و1.8 بوصفهما تحديثًا (وسُلِّم إصلاح الصفوف المحدّدة إلى الخط 1.4 كذلك). ولبقية العمليات الإدارية على الكيانات الوهمية، انظر الكيانات الوهمية: العمليات الإدارية [بناء التطبيقات › الكيانات الوهمية].

التصدير الشرطي `Since 1.10`

يمكن أن يتضمّن قالب التصدير من مستوى **مدخل واحد** شرطًا، فلا تُصدَّر باستخدامه إلا المُدخلات التي تستوفي ذلك الشرط. وفي نموذج القالب، يُضبط الشرط في قسم **الترشيح**، وتحدّد قائمة **من أجل بقية المدخلات** ما يراه المستخدمون عند التعامل مع مُدخل لا يستوفيه: **إلغاء تفعيل خيار التصدير** (الافتراضي) أو **إخفاء خيار التصدير**. ولا يمكن أن يتضمّن قالب **عدة مدخلات** شرطًا (فعناصر الترشيح فيه معطّلة). أما القالب الذي لا شرط له، فيعمل كما كان تمامًا.

يُخزَّن الشرط على القالب:

الخاصية	المعنى
`query`	استعلام محفوظ يجب أن يستوفيه المُدخل (وتؤدّي `null` إلى محوه).
`criteria`	تعبير ترشيح يجب أن يستوفيه المُدخل.
`nonMatchEntriesMode`	`Disable` (0، الافتراضي عند الغياب) أو `Hidden` (1) – وهي القيم نفسها التي تستخدمها أزرار النظام.

يُعدّ القالب «ذا شرط» إذا كان من مستوى المُدخل الواحد، وكان له استعلام محفوظ **أو** قاعدة معايير واحدة على الأقل؛ ولا تُعدّ مجموعة القواعد الفارغة شرطًا. والاستعلام المحفوظ الذي يتعذّر تحميله، أو المعايير التي يتعذّر تحليلها، لا تطابق أي مُدخل (ويُسجَّل ذلك).

- **مسار الحالة.** يعيد `POST /v1/entities/{entity}/templates/status/entries` مع `{dataEntriesIds: string[]}` (من 1 إلى 100 معرّف) القائمة `[{templateId, isDisabled, isHidden}]` لكل قالب ذي *شرط*. وعند استخدامه عبر معرض، لا يُبلغ إلا عن القوالب التي يعرضها ذلك المعرض، ولا يقيّم الشروط إلا إذا كان المعرض قادرًا على قراءة كل المُدخلات المطلوبة؛ أما خارج المعرض، فيُتاح لعارضي النظام ومن فوقهم، شأنه شأن مسار التصدير. وتستخدمه قوائم الصف وتفاصيل المُدخل لإظهار كل قالب مفعّلًا أو معطّلًا (مع السبب عند التمرير) أو مخفيًا، ولا

تستدعيه إلا إذا كان لأحد القوالب المعروضة شرط. وتُضمّن المعارض التي تعرض قالبًا هذا المسار في خطة تنفيذها (وقد أضافه الترحيل `1.10.0-a22` إلى المعارض القائمة).

- **مسار التصدير.** يرفض `GET /v1/entities/{entity}/entries/export/{templateId}` المُدخلَ غير المطابق، وأيَّ `mode` غير العناصر المحدّدة لقالب ذي شرط، بالاستجابة `422 TEMPLATE_CONDITIONS_NOT_MET`. ويجري الفحص قبل أي تصيير أو إدراج في الطابور.
- **خارج نطاق الشرط.** إجراء التفاعل **توليد ملف من قالب** لا يطبّق شروط القالب.

ملاحظات الإصدارات

الميزة	منذ	الدليل
معاملات القالب ونماذج قالب العرض (`TemplateParameter`، `EntityDisplayTemplate`، `ViewDisplayTemplate`)	1.4	أُدخلت في `1.4.0-alpha.01`
التصدير الموسّع للقالب (`exportFromTemplate` / العلَم `extended`)	1.6	أول وسم `1.6.0`
التصدير في الخلفية (`TemplateExportTypeEnum`، `ExportTemplateInBackgroundCommand`، `FileGeneratedEvent`)	Long-standing	قبل 1.2
المتغيّرات المخصّصة (`additionalProperties` / `parseAdditionalProperties`)	Long-standing	قبل 1.2
`GET /v1/templates/{id}/used`	Long-standing	قبل 1.2
تصديرات قوالب Excel المبثوثة (500 مُدخل لكل صفحة)	1.4	نُشرت أولًا على الخط 1.4، وسُلِّمت أيضًا إلى الخطّين 1.6 و1.8
تضمين الصفوف الوهمية المحدّدة في تصدير القالب	1.4	سُلِّم إلى الخطوط 1.4 و1.6 و1.8
تبويب القوالب على الكيانات الوهمية	1.6	الواجهة الأمامية، الخطّان 1.6 و1.8
التصدير الشرطي (`query` / `criteria` / `nonMatchEntriesMode`، ومسار الحالة، و`TEMPLATE_CONDITIONS_NOT_MET`)	1.10	على develop فقط (الترحيل `1.10.0-a22`)
سياسة الوصول إلى البيانات الحسّاسة لكل قالب (`sensitiveDataAccessPolicy`)	1.10	لا يظهر عنصر التحكم في النموذج إلا إذا كان في الكيان حقل حسّاس

المجموعات والوحدات التنظيمية والانتماء

هذه الصفحة موجّهة إلى المُكامِلين والمهندسين الذين يقرؤون بيانات الانتماء إلى المجموعات والهيكل التنظيمي أو يكتبونها برمجيًا، أو يحتاجون إلى معرفة الكيفية الدقيقة التي تحلّ بها أسس مرجع مجموعة أو مدير إلى شخص بعينه. وللاطّلاع على النموذج نفسه من منظور مسؤول النظام، انظر إدارة المجموعات [الإدارة] والوحدات التنظيمية [الإدارة]. أما أذونات المجموعات، فتتناولها واجهة برمجة الأذونات.

المجموعات

ما المجموعة

ليس `GROUP` مجموعة بيانات مادية مستقلة، بل هو **طبقة عرض وهمية توحّد عدة كيانات مصدرية فعلية للهوية**. فحين تختار «مجموعة» في الواجهة، فأنت تختار صفًّا من أحد هذه الكيانات. لذا يُحدَّد أعضاء المجموعة إما **مباشرةً** (مجموعة مستخدمين تُملأ يدويًا أو عبر مزامنة الدليل) وإما **اشتقاقًا** (موظفو وحدة تنظيمية، أو حاملو دور، أو نتيجة استعلام مستخدمين محفوظ). ويُدار كل نوع منها في شاشة كيانه المصدري.

ولأن `GROUP` وهمي، فهو **كيان هوية أساسي** موجود في كل بيئة، والمجموعات **مستبعَدة من تصدير الحزم واستيرادها** بوصفها موارد قائمة بذاتها (انظر المجموعات والحزم).

أنواع المجموعات

النوع (`group_type.en`)	الوصف	الكيان المصدري	معرّف الكيان
مجموعة مستخدمين	مجموعة عادية تُنشأ وتُدار يدويًا بإضافة الأعضاء.	مجموعات المستخدمين	`USERS_GROUP`
مجموعة الأدوار	تُنشأ تلقائيًا لكل دور، ويُشتقّ الانتماء إليها من إسنادات الدور.	الدور	`ROLE`
مجموعة الوحدات التنظيمية	تُنشأ تلقائيًا لكل وحدة تنظيمية، ويُشتقّ الانتماء إليها من الهيكل التنظيمي.	الوحدة التنظيمية	`ORGUNIT`
مجموعة الاستعلام	مجموعة **ديناميكية**: ينضمّ إليها كل مستخدم يستوفي الاستعلام، ويغادرها كل مستخدم لم يعد يستوفيه، دون أي إسناد يدوي.	مجموعة استعلام المستخدمين	`USER_QUERY_GROUP`
مجموعة وهمية	مجموعة محسوبة يديرها النظام لأغراض داخلية، ولا تُنشأ ولا تُعدَّل يدويًا أبدًا.	–	–

تسميات الحقول المزروعة هي: في `USERS_GROUP`، الحقول **اسم المجموعة** (`name`) و**وصف المجموعة** (`description`) و**أيقونة المجموعة** (`icon`)؛ وفي `USER_QUERY_GROUP`، الحقول **اسم المجموعة** و**التوصيف** و**أيقونة** و**الاستعلام** (منتقٍ لاستعلام بيانات على المستخدمين).

كيف يُخزَّن الانتماء

يُسجَّل الانتماء في صورة صفوف إسناد، لا في مصفوفة مضمّنة في المجموعة:

اسم الكيان	معرّف الكيان	الغرض
`group_assignments`	`GROUP_ASSIGNMENTS`	وهمي: يربط **مجموعة** (من أي نوع) بـ**مستخدم**.
`user_group_assignments`	`USER_GROUP_ASSIGNMENTS`	يربط **مجموعة مستخدمين** تحديدًا بـ**مستخدم** (`value.users_group`، `value.user`).

اسم الكيان	معرّف الكيان	الغرض
`user_query_group_assignment`	`USER_QUERY_GROUP_ASSIGNMENT`	الانتماء المجسَّد لمجموعات الاستعلام، وتكتبه معالجات تعمل في الخلفية. `Since 1.8`

يحتوي صف `group_assignments` على الحقول `value.group.dataEntry` (معرّف المجموعة) و `value.user.dataEntry` (العضو) و `value._metaOrder.value` (مفتاح ترتيب يُستخدم لفرز الأعضاء فرزًا حتميًا). ولكل صف أيضًا حقل `state` على المستوى الأعلى وعلَم `isInvalid` . ولا تُحتسب ضمن الانتماء النشط إلا الصفوف التي يكون فيها `state = Active` و `isInvalid != true` .

التفرّد. لكلٍّ من `user_group_assignments` و `user_query_group_assignment` و `space_users` فهرس مركّب يشترط أن يكون الزوج النشط `(user, group)` (أو `(user, space)`) **فريدًا**، فيُرفض أي إسناد نشط ثانٍ للزوج نفسه. `Since 1.8` – أزال الترحيل `1.8.0-a16` التكرارات القائمة، وأعاد الترحيل `1.8.0-a17` إنشاء الفهارس، إذ كانت قد أُسقطت دون إشعار في البيئات التي عُدِّل فيها الكيان.

وتعرض شاشات الإدارة ذلك في تبويب **المستخدمون** ضمن مُدخل مجموعة المستخدمين، وتبويب **المجموعات** ضمن مُدخل المستخدم (وهما العلاقتان المزروعتان إلى `USER_GROUP_ASSIGNMENTS`).

مزامنة الدليل

في المجموعات المرتبطة بالدليل، تتولّى أدوات استيراد الدليل تسجيل الانتماء؛ إذ تُطابَق كل مجموعة مستخدمين في أسس موسومة بأنها خارجية مع مجموعة في الدليل عبر `value.externalGroupId` ، ويُطابَق كل عضو في الدليل مع مستخدم في أسس عبر `account_information.externalUserId` ، ويُنشأ صف `user_group_assignments` لكل تطابق. أما المجموعات غير الموجودة في أسس والأعضاء الذين لا يقابلهم مستخدم، فتُتخطّى ويُسجَّل ذلك. وترد التفاصيل الكاملة، بما فيها الحقول التي تحتاجها مجموعة المستخدمين لذلك، في عمليات مزامنة الدليل.

سرد أعضاء مجموعة

تتيح خدمة البيانات نقطة نهاية واحدة لقراءة الانتماء إلى المجموعات، تحلّ المجموعة أيًّا كان نوعها إلى **مستخدميها الأعضاء النشطين دون تكرار**:

```
GET /v1/group/{groupId}/users?includeResolved={true|false}
```

- **المصادقة:** مطلوبة (استراتيجية مصادقة أسس).
- **التفويض:** يتولّاه مصوِّت الكيانات القياسي.
- **معامل المسار** `groupId` : معرّف المجموعة.
- **معامل الاستعلام** `includeResolved` (منطقي): حين يكون `true` ، يُثرى كل مرجع مستخدم مُعاد بكتلة `_resolved` تتضمّن بيانات العرض الوصفية المفهرسة للمستخدم؛ وحين يكون `false` أو غير موجود، لا يُعاد إلا المرجع المجرّد.

يقصر المتحكّم دائمًا المستخدمين الأساسيين على من تكون حالتهم `state = 1` (Active)، فلا تُعاد أبدًا حسابات المستخدمين غير النشطة (المؤرشفة أو المسودّة). أما علَم المستخدم `is_deactivated` فهو حقل قيمة، وليس حقل حالة؛ ولذلك يُعاد المستخدمون **المعطَّلون** ضمن الأعضاء.

شكل الاستجابة

كل عنصر هو مرجع مستخدم:

الحقل	النوع	يوجد حين	المعنى
`dataEntity`	string	دائمًا	معرّف كيان المرجع (مستخدم).

الحقل	النوع	يوجد حين	المعنى
`dataEntry`	string	دائمًا	معرّف المستخدم العضو.
`_resolved`	object	`includeResolved=true`	بيانات عرض وصفية محلولة من فهرس المُدخلات.

قواعد حلّ الانتماء

تبني `getUsersInsideGroup` تجميعة MongoDB على `group_assignments` تُطبّق ما يلي بالترتيب:

1. **المطابقة** مع الصفوف التي يكون فيها `value.group.dataEntry` أحد معرّفات المجموعات المطلوبة، و`state = Active`، و`isInvalid != true` . ويمكن تمرير عدة معرّفات مجموعات دفعةً واحدة، فتُعامَل بوصفها اتحادًا.
2. **ترشيح مسبق اختياري للمستخدمين**: إذا مُرِّر مرشِّح للمستخدمين (وتمرّر نقطة نهاية HTTP القيمة `state = 1`)، تحلّ الخدمة أولًا معرّفات المستخدمين المطابقين من الكيان `USER` ، وتقصر الإسنادات عليهم. وبهذا يُستبعد المستخدمون غير النشطين.
3. **إزالة التكرار** بحسب `value.user.dataEntry` مع الاحتفاظ بالصف الأول: فالمستخدم المنتمي إلى مجموعة عبر أكثر من إسناد يظهر مرة واحدة.
4. **الفرز** بحسب `value._metaOrder.value` تصاعديًا، ليبقى ترتيب الأعضاء ثابتًا.
5. **الحلّ الاختياري**: عند طلب `includeResolved` ، تُجلب البيانات الوصفية للأعضاء من فهرس كيان البيانات في دفعة واحدة وتُرفق بوصفها `_resolved` .

وثمة أيضًا وضع داخلي `activeOnly` (غير متاح عبر نقطة نهاية HTTP): فإذا اقترن بـ `includeResolved` ، استُبعد تمامًا كل عضو لا يكون مُدخله في الفهرس بحالة `Active` ، بدلًا من إعادته دون بيانات وصفية محلولة. وإن لم يطابق أي إسناد الشروط، أعادت الخدمة قائمة فارغة.

ويُستخدم الحلّ نفسه حيثما كانت المجموعة جمهورًا أو مستلِمًا (الإشعارات، ومشغّلات البريد الإلكتروني والرسائل النصية، والتعليقات)، ولذلك يؤثّر ضبط الانتماء مرة واحدة في إسناد الملكية وتسليم الإشعارات وتوجيه التعليقات على نحو متّسق.

المجموعات مالكةً لخطوات سير العمل

تُصرّح الحقول المرجعية التي تقبل مالكًا بأن أنواعها المقبولة هي `USER,GROUP` ، ومنها مثلًا حقل **المالك الأصلي** في خطوة سير العمل. وهذا ما يتيح لباني التطبيق إسناد ملكية خطوة إلى مجموعة.

وحين تُنفَّذ خطوة مالكها **مجموعة**، يحلّ المحرّك المجموعة إلى مستخدم فعلي في `OwnerTypeProcessor` ، وذلك بحسب خيارات المالك في الخطوة:

خيار المالك	الأثر حين يكون المالك مجموعة
الافتراضي (مالك واحد)	يحلّ المجموعة ويُسند **أول عضو فيها** (`groupUsers[0]`) مالكًا.
`selectionCriteria = Manager`	يحلّ **مدير الوحدة التنظيمية** للمجموعة عبر `getOrgUnitManager` ؛ وإن لم يُعثر على مدير، يُمسح المالك.
`selectionCriteria = Manual`	يُمسح المالك المحلول تلقائيًا ليختار شخصٌ المالكَ يدويًا.
`isParallel = true`	يُمسح المالك الواحد، وتُعالَج الخطوة بوصفها إسنادًا متوازيًا متعدد المالكين.
مصفوفة مالكين من المجموعات (متعدد ديناميكي)	يوسّع كل مجموعة إلى أعضائها عبر معالج المالكين المتعددين الديناميكي؛ وإن كانت كل المُدخلات المحلولة مستخدمين، اختير أولها مالكًا نشطًا (ما لم يكن الخيار Manual أو متوازيًا).

تبحث `getOrgUnitManager` أولًا عن إسناد نشط على الوحدة التنظيمية نفسها تكون فيه `value.isManager = true` ، ويكون المستخدم المُشار إليه فعّالًا. فإن لم يوجد، تصعد **في سلسلة آباء الوحدة التنظيمية** (الأقرب أولًا) وتعيد أول مدير تجده.

ولأن إسناد المالك الافتراضي يعتمد على العضو *الأول*، فإن ترتيب `_metaOrder` لأعضاء المجموعة ليس شكليًا: فهو الذي يحدّد من يملك الخطوة المسندة إلى مجموعة.

المجموعات والحزم

يوجد `GROUP` وكيانات الهوية الكامنة خلفه (`USER` و `ORGUNIT` و `WORKING_HOURS`) وجودًا مستقلًا في كل بيئة، ولذلك فهي **كيانات أساسية لا تُصدَّر أبدًا ضمن حزمة**. وحين تشير حزمة مستوردة إلى مجموعة لا توجد بعد في البيئة الهدف، فإن الإنشاء التلقائي **لا** ينشئ صف `GROUP` مباشرةً، بل ينشئ مُدخلًا في الكيان المصدري الفعلي الذي يقتضيه `group_type` للمجموعة (مجموعة مستخدمين أو دور أو وحدة تنظيمية أو مجموعة استعلام)، ويعتمد `USERS_GROUP` إن كان النوع مجهولًا.

وحين تنقل بين البيئات سير عمل أو معرضًا يُسند مجموعة، فإن تعريف المجموعة والانتماء إليها لا ينتقلان مع الحزمة. لذا يجب أن توجد المجموعة المكافئة في البيئة الهدف، أو أن يُعاد توجيه المرجع بعد الاستيراد. انظر الحزم.

ملاحظات تشغيلية

- **النشط وحده في كل موضع.** يجب أن يكون المستخدمون الأعضاء وصفوف إسنادهم جميعًا في حالة Active ليُحتسبوا. وتُستبعد حالتا المسودّة (`0`) والمؤرشف (`2`).
- **تُدمج التكرارات** وقت القراءة، ومنذ 1.8 تمنعها الفهارس الفريدة المذكورة أعلاه وقت الكتابة.

الوحدات التنظيمية

نظرة عامة على المفهوم

الوحدة التنظيمية مُدخل بيانات عادي في الكيان المدمج `ORGUNIT` (الاسم الموحّد `org_unit`). وتشير الوحدات التنظيمية بعضها إلى بعض عبر حقل `parent` ذي المرجع الذاتي، فتشكّل مجموعة الوحدات شجرة. ويرتبط المستخدمون بالوحدات التنظيمية عبر مُدخلات **الإسناد** (الكيان `ASSIGNMENT`)، إذ يربط كلٌّ منها مستخدمًا بدور ووحدة تنظيمية، ويحمل مستوى الإسناد (رئيسي / إضافي)، ونوع الإسناد (أصلي / تفويض)، وعلَم *هل هو مدير*.

> **مفهومان لـ«المنظمة».** يشكّل `ORGUNIT` الهيكل الهرمي. أما كيان **المنظمة** المفرد (`ORGANIZATION` ، `OrganizationDataEntry` / `OrganizationDataEntryValue`) فلا يحمل إلا العلامة التجارية للقاطن وهويته: `name` (مترجم، إلزامي)، و `description` ، و `logo` ، و `contact_information` (كتلة تضمّ الهاتف والبريد الإلكتروني)، و `organization_header` ، و `organization_footer` ، و `address` ، و `motto` . وينشئ الإعداد الأولي مُدخلًا واحدًا هو «المنظمة الافتراضية». ويحمل مستند **OrgConfig** منفصل في users-service العلامة التجارية لصفحة تسجيل الدخول (الاسم، والشعار، والأيقونة المفضّلة، وCSS مخصّص، ورابط تسجيل اختياري). ولا يتضمّن أيٌّ من هذه علاقة أب أو تسلسلًا هرميًا.

يدعم التسلسل الهرمي ثلاث قدرات:

القدرة	يحرّكها	نقطة الدخول
تحديد النطاق («مستخدمو وحدتي التنظيمية»)	الوحدة التنظيمية للإسناد الرئيسي للمستخدم، وأبوها، وكل الوحدات المتفرّعة منها	`GET /v1/users/my-orgunit`
تحديد المدير المباشر	الصعود في سلسلة `parent` بحثًا عن أقرب إسناد مدير	`GET /v1/users/direct-manager`
مالك خطوة سير العمل (النسبي)	الصعود نفسه، وفق *الخيارات النسبية* للخطوة	محرّك سير العمل (تحديد المالك)

كيان الوحدة التنظيمية

يُزرع `ORGUNIT` كيانًا ضمنيًا (نظاميًا) في أثناء الإعداد الأولي. وهو **شجري البنية**: إذ يتضمّن تعريفه `treeStructure.supported = true` مع ضبط حقل الشجرة على `parent`.

الحقول (`orgunit.json`، ويعكسها النموذج `OrgUnitDataEntryValue`):

الحقل (normalizedName)	التسمية	النوع	ملاحظات
`name`	الاسم	نص مترجم	إلزامي، **فريد**، ويُعرض عنوانًا للمُدخل
`description`	التوصيف	نص مترجم	اختياري
`icon`	الأيقونة	أيقونة	اختياري، ويُستخدم صورةً للمُدخل
`parent`	الأب	علاقة ← `ORGUNIT`	رابط التسلسل الهرمي. `sourceOptions.denyCircularRelation: true`، فلا يمكن أن تصبح الوحدة سلفًا لنفسها
`org_unit_level`	مستوى الوحدة التنظيمية	علاقة ← مجموعة `ORGANIZATION_UNIT_LEVELS`	مستوى الوحدة
`isMainAssignment`	مستوى الإسناد رئيسي	منطقي	الافتراضي `false`. وحين يكون `true` لا تقبل الوحدة إلا الإسنادات الرئيسية (`OrgUnitAcceptJustMainLevelException`)
`manager`	المدير	علاقة ← `USER`	**اختياري**. حقل غير مطبَّع، يُزامَن تلقائيًا من الإسنادات

الكيان **مقيَّد** (`restrictedEntityIds` يتضمّن `ORG_UNIT_ENTITY_ID`): يقرؤه عارضو النظام والمستخدمون المقيَّدون الحاصلون على المنح، ولا يعدّله إلا المديرون. وبما أن `implicitOptions.allowEdit = true`، يستطيع بُناة التطبيقات إضافة حقول إليه.

مستويات الوحدات التنظيمية

يشير `org_unit_level` إلى مجموعة ADT النصية `ORGANIZATION_UNIT_LEVELS` (تسميتها **مستويات الوحدات التنظيمية**). وتضمّ عند التثبيت مستوى واحدًا:

معرّف المستوى	التسمية (en)	`level`	`index`
`5cf7d2cc-…-461e636f`	First level	0	0

المستويات تسميات وصفية؛ أما التسلسل الهرمي الفعلي فيحدّده `parent`، لا قيمة المستوى.

الجذر المزروع

ينشئ الإعداد الأولي وحدة تنظيمية جذرية واحدة هي **«الإدارة التنفيذية»** (`org-root.json`)، مع `parent: ""`، والمستوى الأول المزروع، وأيقونة `home`.

الإسنادات

يربط كل **إسناد** (الكيان `ASSIGNMENT`، `assignment.json`) مستخدمًا واحدًا بوحدة تنظيمية واحدة، مع تحديد دور:

الحقل	التسمية	النوع	المعنى
`role`	الدور	علاقة ← `ROLE`	الدور الذي يشغله
`assign_level`	مستوى الإسناد	اختيار مفرد	`MAIN` أو `ADDITIONAL` (`AssignmentLevelTypes`)
`assign_type`	نوع الإسناد	اختيار مفرد	`ORIGINAL` أو `DELEGATION` (`AssignTypes`)
`user`	المستخدم	علاقة ← `USER`	المستخدم المُسنَد
`org_unit`	الوحدة التنظيمية	علاقة ← `ORGUNIT`	الوحدة التنظيمية

الحقل	التسمية	النوع	المعنى
`isManager`	مدير	منطقي	هل يدير هذا المستخدم الوحدة التنظيمية

ويفرض مدقّق الإسناد عند الحفظ الضوابط التالية:

- الالتزام بقيمة `isMainAssignment` للوحدة (لا يُسمح فيها إلا بالإسناد الرئيسي)؛
- الدور ذو نوع الإسناد **فريد** لا يُسند إلا مرة واحدة في كل وحدة تنظيمية، والدور **المرتبط بوحدة تنظيمية واحدة** لا يوجد إلا في وحدة تنظيمية واحدة؛
- يتطلّب الإسناد **الرئيسي** دورًا من نوع *منصب وظيفي* ("role must be of type job position if the the assignment level is main")، ولا يكون للمستخدم إلا إسناد رئيسي واحد غير مؤرشف ("each person can have only main assignment")؛
- يجب أن يكون المستخدم حساب منظمة ("User must be an organization account")؛
- يُمسح حقل `org_unit` في الإسناد إذا كان دوره من نوع *غير المرتبط بوحدات تنظيمية*.

وترتبط الأدوار بالهيكل عبر `RoleLinkWIthOrgUnitTypes` : *غير مرتبط بوحدة تنظيمية*، و*مرتبط بوحدة تنظيمية واحدة*، و*مرتبط بعدة وحدات تنظيمية*.

الوحدة التنظيمية للمستخدم هي وحدة إسناده الرئيسي. تعيد `getUserOrgUnitId()` قيمة `org_unit` للإسناد النشط الوحيد الذي يحقّق `assign_level.id === MAIN` . وحين يُنشأ إسناد، ينسخ معالجُ دورَ الإسناد الرئيسي ووحدته التنظيمية إلى الحقلين `main_assignment_role` و`org_unit` في مُدخل المستخدم، وينسخ دور الإسناد الإضافي إلى `additional_assignment_role` . أما `direct_manager` في مُدخل المستخدم فحقل محسوب يُشتق بالصعود في سلسلة الآباء.

الحقل `manager` المُزامَن تلقائيًا

حقل `manager` في الوحدة التنظيمية نسخة غير مطبَّعة تُبقيها معالجات الأحداث محدَّثة. فحين **يُنشأ إسناد أو يُحدَّث أو يُؤرشف**، يتحقّق معالج (`update-orgunit-manager-on-assignment-entry-*`) مما إذا كان الإسناد **رئيسيًا** وكانت قيمة `isManager` فيه صحيحة؛ فإن تحقّق الشرطان، حدّث حقل `manager` في الوحدة التنظيمية ليشير إلى ذلك المستخدم. (Long-standing.)

أما *تحديد* المدير لسير العمل والإشعارات، فهو **لا** يقرأ هذا الحقل المساعد، بل يستعلم عن الإسنادات في أثناء الصعود في سلسلة الآباء (القسم التالي).

زرع الحقل `manager` `Since 1.8`

وجود `manager` في `ORGUNIT` مضمون في كل بيئة منذ 1.8؛ فالبيئة الجديدة تزرعه مع تعريف الكيان، ويضيفه ترحيلٌ إلى البيئات التي جُهِّزت قبل ذلك. أما في البيئات التي جُهِّزت قبل تنفيذ ذلك الترحيل، فكان الحقل غائبًا، رغم أن معالجات الإسناد كانت تكتب فيه على الدوام.

والسبب أن ترحيل الإعداد الأولي يقرأ **كل** ملف JSON في المجلد `entity/` ويتخطّى أي معرّف كيان أُدرج من قبل. وكان ملفان يعرّفان `ORGUNIT` : الملف `org-unit.json` (بلا `manager`) والملف `orgunit.json` (مع `manager` و`treeStructure` كاملًا). ولأن سرد محتويات المجلد كان يُظهر الملف ذا الشَّرطة أولًا، أُدرج التعريف الناقص بذلك المعرّف وتُخطّي `orgunit.json` . وقد عالج ذلك تغييران:

- حُذف الملف المكرّر القديم `org-unit.json` من مصادر الترحيل، فصارت البيئة الجديدة تزرع `ORGUNIT` من `orgunit.json` ، وهو الآن ملف الزرع الوحيد للكيان.
- يستكمل الترحيل `1.8.0-a31-add-manager-field-to-org-unit-entity` تعريف `ORGUNIT` بإضافة الحقل `manager` بأثر رجعي في البيئات القائمة. وله الأثر نفسه عند تكرار تنفيذه (إذ لا يُضيف الحقل إلا حيث يغيب `manager`)، ودالة

`down()` فيه لا تنفّذ أي إجراء عمدًا، لأن إزالة الحقل قد تُتلف بيانات سبق تسجيلها في مُدخلات الوحدات التنظيمية.

تحديد المدير (الصعود في الشجرة)

تحدّد `getUserDirectManager(userId, relationGrade = 1)` مدير المستخدم بالصعود في التسلسل الهرمي:

1. إيجاد الإسناد **الرئيسي** للمستخدم وقيمة `org_unit` الخاصة به.
2. تنفيذ `$graphLookup` صعودًا في سلسلة `parent` من تلك الوحدة، مع ترتيب النتائج بحسب العمق (الأقرب أولًا، بما في ذلك وحدة المستخدم نفسه عند العمق 0).
3. المرور على تلك القائمة، والبحث في كل وحدة عن إسناد فيه `isManager: true` لمستخدم **غير** المستخدم المعنيّ.
4. إعادة المدير **رقم N** بين المديرين الذين يُعثر عليهم، حيث **`relationGrade = N`** («عدد مستويات المديرين صعودًا»، بدءًا من 1 للمدير المباشر).
5. إن لم يُعثر على مدير عند الدرجة المطلوبة، تُعاد الوحدة التنظيمية وحدها (بلا `directManger`).

ويُتاح ذلك للبوابة عبر `GET /v1/users/direct-manager` . ويعتمد **نموذج أذونات المدير** كذلك على المدير المباشر للمستخدم؛ ويشمل هذا النموذج المنح المقيَّدة بنطاق المدير التي تتيح له التصرّف في مُدخلات المستخدمين الذين يديرهم (انظر واجهة برمجة الأذونات: أذونات المدير).

مالكو الخطوات النسبيون في سير العمل

يمكن لخطوات سير العمل أن تعرّف **خيارات مالك نسبية** (`StepOwnerRelativeOptions`): `enabled` و `relationType` و `relationGrade` («عدد المستويات المطلوب صعودها في التسلسل الهرمي، بدءًا من 1»). وفي أثناء تحديد المالك، حين تكون قيمة `source.relativeOptions.enabled` للخطوة صحيحة (ولم تكن الخطوة الأخيرة)، يستبدل المحرّك المالك الذي جرى تحديده بنتيجة `getUserDirectManager(owner, relationGrade)` . وإن لم يُعثر على مدير عند تلك الدرجة، تُترك الخطوة **بلا مالك** (ويُسجَّل تحذير) بدلًا من اللجوء إلى بديل من دون تنبيه.

ويرتبط بالتسلسل الهرمي رمزان من رموز **مصدر المستلمين** في سير العمل، تستخدمهما الإشعارات ومشغّلات البريد الإلكتروني:

الرمز	الثابت	المعنى
`$OWNER.DIRECT_MANAGER`	`CURRENT_STEP_OWNER_DIRECT_MANAGER_RECIPIENT_SOURCE_VALUE`	المدير المباشر لمالك الخطوة
`$OWNER.ORG_UNIT_MANAGER`	`CURRENT_STEP_OWNER_ORGUNIT_MANAGER_RECIPIENT_SOURCE_VALUE`	مدير الوحدة التنظيمية لمالك الخطوة

تحديد النطاق («مستخدمو الوحدة التنظيمية ذوو الصلة»)

تعيد `getUserRelatedOrgUnitsUsers(userId)` المستخدمين الذين يُعدّون «ذوي صلة» بمستخدم ما عبر الهيكل. فبدءًا من الوحدة التنظيمية لإسناده الرئيسي، يجمع استعلامٌ تجميعي المستخدمين النشطين المُسنَدين إلى وحدته **نفسها**، وإلى أبيها **المباشر**، وإلى **كل** الوحدات المتفرّعة منها (عبر `$graphLookup` نزولًا في سلسلة `parent`). ولا تُدرَج إلا الإسنادات والمستخدمون في حالة Active، وتُزال التكرارات من النتائج. وتتيح `GET /v1/users/my-orgunit` هذه النتائج، ويستخدمها مثلًا جالب بيانات منتقي المستخدمين لتحديد نطاق الأشخاص القابلين للاختيار.

إدارة التسلسل الهرمي عبر الواجهة البرمجية

لا يوجد متحكّم REST مخصّص للوحدات التنظيمية؛ إذ تُدار الوحدات التنظيمية والإسنادات عبر عمليات CRUD العامة لمُدخلات البيانات في الكيانين `org_unit` و `assignment` (انظر كتابة البيانات)، إضافةً إلى مساعِدات القراءة المذكورة أعلاه. ابنِ الشجرة

باستخدام `parent` (واترك قيمة `parent` للجذر فارغة)، وعيّن المدير بإسناد رئيسي فيه `isManager: true`، وامنح كل مستخدم إسنادًا رئيسيًا واحدًا بالضبط، ووسّع `ORGANIZATION_UNIT_LEVELS` لوصف العمق.

الأسلاف المجسَّدة

لأن الوحدات التنظيمية تشكّل شجرة، تحفظ أسس قائمة أسلاف كل عقدة محسوبةً مسبقًا، تجنّبًا لتكرار عمليات البحث التعاودية. ويسري ذلك على كل كيان يتضمّن تعريفه `treeStructure.supported = true`.

- **`DataEntryTreeService`** يحسب الأسلاف بعملية `$graphLookup` في MongoDB على `value.<treeField>.dataEntry → _id`، مع ترتيب النتائج بحسب العمق، ويخزّن معرّفات الأسلاف مرتَّبةً في `_meta.ancestors.value.<treeField>` (وفي الوحدات التنظيمية: `_meta.ancestors.value.parent`). ويتيح `calculateAncestors` و`findDescendants` و`updateNodeAncestors` و`updateNodeAndDescendantsAncestors` و`initializeAllAncestors` و`rebuildAncestors`. `Since 1.2` (مع ترحيل الملء الرجعي `1.2.0-a02-calculate-ancestors`).
- **معالجات المزامنة** (`sync-entry-ancestors-data-entry-created` / `-updated`) تعمل عند إنشاء المُدخل أو تحديثه، فتتحقّق من `treeStructure.supported`، وتعيد حساب الأسلاف لكل حقل شجري. وعند نقل شجرة فرعية، يُعاد حساب الأسلاف للعقدة وكل العقد المتفرّعة منها.
- تستخدم عمليات الكتابة `$set` الخام على `_meta.ancestors...` (ولا تستخدم `updateById` أبدًا)، اتساقًا مع قاعدة أمان الترحيل في التحديثات الجزئية.

ملخّص نموذج البيانات

- `ORGUNIT`: الاسم الموحّد `org_unit`، ونوع الكيان `11`، ضمني، مقيَّد، قابل للتعديل (`implicitOptions.allowEdit = true`). نموذج القيمة `OrgUnitDataEntryValue`.
- `ASSIGNMENT`: الاسم الموحّد `assignment`. يربط المستخدم بالوحدة التنظيمية مع تحديد الدور والمستوى والنوع و`isManager`.
- `ORGANIZATION_UNIT_LEVELS`: مجموعة ADT نصية لتسميات المستويات.
- الثوابت ذات الصلة في `libs/constants.ts`: `ORG_UNIT_ENTITY_ID='ORGUNIT'`، و`ORG_UNIT_ENTITY_NAME='org_unit'`، و`ASSIGNMENT_ENTITY_ID/NAME`، و`AssignmentLevelTypes`، و`AssignTypes`، و`RoleLinkWIthOrgUnitTypes`، ورمزا المستلمين `$OWNER.*`.

إطار البوابة: الصفحة الرئيسية والشريط الجانبي وروابط المشاركة (الواجهة البرمجية)

هذه الصفحة موجّهة إلى المُكامِلين والمطوّرين الذين يبنون آلية تنقّل خاصة بهم فوق المنصة، أو يحتاجون إلى فهم أسباب ظهور عناصر معيّنة في الشريط الجانبي لمستخدم ما. وتتناول تحديد الصفحة الرئيسية، وقواعد الظهور، ونقطة نهاية الشريط الجانبي الوحيدة وحمولتها، وقائمة المستخدم المقيَّد، وروابط المشاركة (روابط النظام).

للاطّلاع على ما يراه المستخدم النهائي، راجع تخطيط البوابة والتنقل [استخدام أسس]. ويتوفّر مرجع نقاط النهاية المولَّد في side-bar و system-link.

الصفحة الرئيسية

يؤدّي فتح `/` (أو اختيار شعار المنظمة) إلى عرض الصفحة الرئيسية للمستخدم. وتُحدَّد هذه الصفحة **لكل مستخدم** وفق الإعدادات (`DefaultPageResolverService`)، لذا قد يصل مستخدمون مختلفون إلى صفحات مختلفة. وتعرض الصفحة الرئيسية صفحةً عامة (generic view) قابلة للتهيئة (أي صفحة مبنية من قطع)، ويمكنها تشغيل **قواعد إعادة توجيه** تنقل المستخدم إلى وجهة أخرى فور وصوله.

أقسام الشريط الجانبي

يتكوّن الشريط الجانبي، من الأعلى إلى الأسفل، من أقسام مستقلة لا يظهر كلٌّ منها إلا إذا تضمّن محتوى يحقّ للمستخدم رؤيته. ويفصل بينها خط رفيع، ويمكن تمرير كتلة الوحدات والمعارض على نحو مستقل حين يتجاوز ارتفاعها ارتفاع منطقة العرض.

الترتيب	القسم	يظهر حين
1	قائمة الإدارة	يملك المستخدم وصولًا إداريًا بمستوى العرض على الأقل، أو منحة مقيَّدة (انظر أدناه)
2	قائمة المساحات	لا يكون لدى المستخدم تفويض نشط ولا يكون مقيَّدًا (ولديه أكثر من مساحة)
3	اختصارات الخدمات	يكون تنقّل الخدمات مفعّلًا ولا يكون المستخدم مقيَّدًا
4	الوحدات والمعارض	تتوفّر للمستخدم روابط (وحدات، صفحات، معارض، تقارير)

وتُرشَّح مُدخلات الإدارة بحسب مستوى الصلاحية الإدارية للمستخدم.

اختصارات الخدمات

الاختصار (التسمية الإنجليزية)	المسار	العدّاد الحيّ
Services Catalog	`/s`	–
Outbox	`/my`	الطلبات الجارية
Inbox	`/inbox`	عناصر الوارد الجارية
Watching	`/rm`	الطلبات المتابَعة الجارية

وما زالت مفاتيح المسارات والصفحات القديمة تسمّي Outbox باسم «My Requests»، وWatching باسم «Request Monitoring». ويُحذف اختصار **المتابعة** من القائمة كليًا، بدلًا من عرضه بالرقم صفر، حين لا يتابع المستخدم شيئًا. وللاطّلاع على قواعد الوصول إلى الطلبات، راجع الطلبات والخدمات.

الوحدات والمعارض

الوحدة حاوية فقط: فاختيارها يوسّعها أو يطويها. ولا تُعرض الوحدة إذا لم يكن فيها أي عنصر فرعي ظاهر. وتضمّ **المعارض** و**الصفحات** و**التقارير**. ولكل مُدخل أيقونة وعنوان مترجَم، وتُرتَّب المُدخلات وفق الترتيب الذي حدّده باني التطبيق، ثم أبجديًا بحسب العنوان. وتحت كل معرض، يسرد الشريط الجانبي أيضًا **الاستعلامات المحفوظة** للمستخدم، و**مجموعاته الخاصة**، و**المجموعات المشارَكة معه** (عبر مجموعاته) لذلك المعرض.

وتتوفّر التقارير على الويب فقط؛ أما في الجوال، فيُحوَّل رابط التقرير إلى عنصر نائب لرابط غير مدعوم ويُستبعد من الشريط الجانبي.

وعلى الشاشات الصغيرة، يتضمّن الشريط الجانبي في تخطيط الجوال حقل بحث يبحث تعاوديًا في عناوين الروابط والاستعلامات المحفوظة والمجموعات.

قواعد الظهور

لا يظهر الرابط إلا إذا استوفى جميع الشروط التالية:

- **منشور ونشط**: لم يُلغَ نشره، وغير مؤرشف.
- **نوع المصادقة**: يرى المستخدمون المسجِّلون دخولهم الروابط الموسومة *Authenticated* أو *Everyone* أو *Restricted* التي تستهدفهم (مباشرةً أو عبر مجموعة). ولا يرى الزوار المجهولون إلا روابط *Anonymous* أو *Everyone*.
- **النطاق**: ينطبق الرابط على كل النطاقات أو على النطاق الحالي.
- **المشاركة**: الرابط Public، أو Private/Shared مع المساحة الحالية.
- **التفويض**: عند وجود تفويض، تُزال الروابط إلى المعارض أو التقارير أو الصفحات التي لم تُفوَّض.

الحدّ المقيَّد `Since 1.8`

تُضبط بعض الحسابات على حدّ **مقيَّد** (Restricted) على مستوى النظام، فتختصر البوابة إطارها لهؤلاء المستخدمين:

- تُخفى قائمة المساحات واختصارات الخدمات؛
- تحلّ محلّ قائمة الإدارة قائمة مصغّرة: **الوحدات** (إن مُنح المستخدم وصولًا إلى الوحدات) و/أو **الكيانات** (إن مُنح وصولًا إلى الوحدات أو الكيانات). ويستلزم الوصول إلى الوحدات الوصول إلى الكيانات، لذا تُظهر منحة الوحدة الاثنين معًا.

أما المستخدم المقيَّد الذي ليست لديه منحة على وحدة أو كيان، فلا يرى أيًّا من الكتلتين، بل يرى الوحدات والمعارض التي تتيحها منحه فقط. وللاطّلاع على نموذج الأذونات، راجع: الأذونات [الإدارة › الأذونات والوصول].

`GET /v2/side-bar`

تجلب البوابة الشريط الجانبي كاملًا من نقطة نهاية واحدة.

الاسم	الموضع	المعنى
`x-domain-id`	header	النطاق الحالي
`x-space-id`	header	المساحة الحالية
`include`	query	الأجزاء المطلوب حسابها

include	الاسم	يعيد
0	All (الافتراضي)	الروابط **و**حالة معارض الخدمات
1	Links	الروابط فقط (وتكون قيمة serviceViewsStatus هي null)
2	OnlyService	حالة معارض الخدمات فقط

الاستجابة (SidebarDtoV2):

- links : شجرة الوحدات والصفحات والمعارض والتقارير.
- serviceViewsStatus : hasServiceCatalog ، و hasRequests ، و hasInbox ، و hasObserverRequests ، والأعداد onGoingRequests و onGoingInbox و onGoingObserverRequests .
- isRestricted : هل يخضع المستخدم للحدّ المقيَّد.
- restrictedAccess : { hasModuleAccess, hasEntityAccess } ، ويُستخدم لبناء قائمة الإدارة المصغّرة للمستخدم المقيَّد. وتكون قيمة كلٍّ منهما صحيحة متى كان للمستخدم وصول بمستوى العرض على الأقل إلى أي وحدة أو كيان.

ويحمل كل رابط GenericLink الحقول id و order و title و description و icon و type و url و target و children . ويُشتق url من النوع:

نوع الرابط	القيمة	URL
Module	0	لا شيء (حاوية فقط)
View	1	/v/<targetId>
Page	2	/<targetId>
Report	3	/reports/<targetId> (على الويب فقط)
Untyped	4	عنوان target.url الخارجي الخاص بالرابط

وتتضمّن روابط المعارض أيضًا userQueries و userCollections و sharedCollections ، وكلٌّ منها رابط خفيف له url خاص به (وفي حالة المجموعة: /v/<viewId>/c/<collectionId>/entries). ويقرأ تطبيق الجوال الشجرة نفسها؛ وللاطّلاع على كيفية تعامله مع كل نوع، راجع بنية تطبيق الجوال.

روابط المشاركة (روابط النظام)

يفتح **رابط النظام** معرضًا عبر عنوان URL قصير قابل للمشاركة يحمل رمزًا مولَّدًا يُستخدم بدلًا من تسجيل الدخول. ومساره في البوابة /vt/:token . ويولّده باني التطبيق، وتبلغ صلاحيته الافتراضية سنة واحدة.

نقطة النهاية	المصادقة	الغرض
GET /v1/system-link/generate	مصادَق عليه	توليد رمز فريد جديد عند إنشاء رابط
GET /v1/system-link/resolve/{token}	مسموح للمجهول	حلّ الرمز إلى معرض البيانات أو المعرض العام الذي يشير إليه

الشريط العلوي

- تُظهر شارة العناصر غير المقروءة في الجرس العدد الدقيق حتى 99، ثم 99+ .
- يُفتح **البحث** بالضغط على **Ctrl + F** أو **F3**؛ ويُغلق بالنقر خارج اللوحة.
- لا يظهر زر **أسس AI** إلا حين يكون الذكاء الاصطناعي مفعّلًا للقاطن. ويفتح لوحة جانبية تُضمّن تطبيقًا خارجيًا في iframe، يُفتح بلغة الواجهة الحالية، لذا يعتمد على إمكانية الوصول إلى خدمة الذكاء الاصطناعي؛ وهو ليس جزءًا من إطار

البوابة نفسه. انظر مساعد أسس AI.

- يمكن أن تظهر **قطع العدّادات** (عدّادات حيّة قابلة للتهيئة) بجوار الجرس.
- في صفحات المُدخلات، يتضمّن مسار التنقّل (breadcrumb) أداة ترقيم سريعة للانتقال بين المُدخلات.

مسارات الملف الشخصي

- `/user/profile` : صفحة **تفضيلات المستخدم** (المعلومات الأساسية، ومعلومات المنظمة والحساب، والأمان، وعمليات تسجيل الدخول). وإلى جانبها `/user/settings` .
- `/edit-profile` : محرّر خفيف لمُدخل الملف الشخصي يحذّر قبل مغادرة الصفحة إن وُجدت تغييرات غير محفوظة.
- `/welcome` : صفحة الترحيب والتعريف الأولي، ويمكن الوصول إليها من **المساعدة** في قائمة الملف الشخصي.

وتحفظ لوحة الإعدادات (زر الترس) الإعدادات لكل حساب (تُكتب في تفضيلات المستخدم، فتنتقل معه بين الأجهزة) أو لكل جلسة (تُخزَّن في جلسة المتصفح، ويُعاد تحميل الصفحة). ويؤدّي اختيار **إعادة تعيين** إلى مسح إعدادات الجلسة واستعادة الإعدادات الافتراضية للحساب. وتعرض اللوحة أرقام الإصدارين الحاليين للويب والخلفية.

الكيانات الوهمية: النموذج والواجهة البرمجية والبنية الداخلية

هذه الصفحة موجّهة إلى **المُكامِلين والمطوّرين** الذين ينشئون الكيانات الوهمية أو يقرؤونها عبر الواجهة البرمجية، أو يكتبون عبر كيان وكيل، أو يحتاجون إلى معرفة الكيفية الدقيقة لبناء المُدخلات الوهمية والحفاظ على تزامنها.

تبني كيانًا وهميًا من الواجهة؟

ينبغي لبُناة التطبيقات البدء بدليل الكيانات الوهمية [بناء التطبيقات]؛ إذ يشرح متى يُستخدم كل نوع وكيف يُنشأ، مستخدمًا التسميات التي يعرضها المصمّم.

الأنواع والتخزين

للكيانات الوهمية ثلاثة أنواع، يُميَّز بينها باستخدام `DataEntity.type` (`DataEntityTypeEnum`):

النوع	`DataEntity.type`	قيمة Enum	ما يقوم به
وكيل	`VirtualProxy`	`50`	وكيل بعلاقة 1:1 مع كيان مصدر واحد. يُجسَّد كل مُدخل مصدر مؤهّل **مادياً** بوصفه مُدخل وكيل في مجموعة تخزين داعمة. `Since 1.10` ، يدعم الكتابة عند ارتباطه بمعرض وكيل حاكم يسمح بذلك.
مُوصِّل	`JoinerProxy`	`51`	وكيل يعرض أيضًا كيانات مرتبطة عبر العلاقات (`scope`). إسقاط للقراءة فقط.
مُجمِّع	`VirtualAggregator`	`52`	كيان يُنشأ بتجميع/ربط عدة مصادر مع تعيينات حقول صريحة. إسقاط للقراءة فقط.

تتشارك الأنواع الثلاثة البنية نفسها لمستند `DataEntity` ؛ إذ تُحدَّد قواعد الاشتقاق في `DataEntity.typeOptions` ، ويتحدّد نوعه الفعلي بحسب `type` الكيان (`VirtualProxyEntityOptions` للوكيل والمُوصِّل، و `AggregatorVirtualEntityOptions` للمجمِّع).

ملاحظة

توثّق هذه الصفحة آلية *الكيان الوكيل/الوهمي* (`type` بقيمة 50/51/52 مع `typeOptions`). وتختلف هذه الآلية عن خيار `DataEntity.virtual` على مستوى الحقل (`DataEntityVirtualOptions` : `enabled` و `source` و `pipeline` و `dependsOn`)، وهو ميزة مستقلة لخط أنابيب الحقول المحسوبة لا تتناولها هذه الصفحة.

كيف تبقى الكيانات الوهمية متزامنة

يسجّل كل كيان وهمي مُدخلات المصدر التي بُني منها كل مُدخل من مُدخلاته. وعندما يُنشأ مُدخل مصدر أو يُحدَّث أو يُؤرشف أو يُحذف، أو عندما يتغيّر استعلام يغذّي كيانًا وهميًا، تحدّث أسس المُدخلات الوهمية المتأثّرة. وتُعيد الكيانات المُجمِّعة، إضافةً إلى ذلك، إجراء الحساب عند تغيّر مصادرها. ويجري هذا التزامن في الخلفية ليعكس الكيان الوهمي دائمًا الحالة الراهنة لمصادره. أما تفاصيل هذه الآلية للكيانات الوكيلة فموضّحة في دورة الحياة.

> **الاتساق التدريجي**
>
> التحديث **مدفوع بالأحداث**: تُعاد بناء المُدخلات الوهمية المتأثّرة استجابةً لأحداث الناقل المنشورة بعد تثبيت تغيير المصدر. لذا توقّع تأخّرًا قصيرًا في الانتشار بدلًا من تحديث فوري ومتزامن؛ إذ يتقارب الكيان الوهمي نحو الحالة الراهنة لمصادره، ولا يُحدَّث ضمن المعاملة نفسها التي تغيّر المصدر. وتختلف عن ذلك الكتابة التي تمرّ **عبر** كيان وكيل قادر على الكتابة: فالكتابة إلى المضيف متزامنة، ويحصل المستدعي فورًا على معاينة للصفّ بصيغة الوكيل، لكن الصفّ الذي يقرؤه لاحقًا يظلّ الصفّ نفسه الذي يكتبه هذا التحديث الخلفي. انظر الكتابة عبر التوجيه.

نموذج بيانات خيارات النوع

الوكيل والمُوصِّل: `VirtualProxyEntityOptions`

تكون `typeOptions` لكيانات `VirtualProxy` و`JoinerProxy` **قائمة مسطّحة من المصادر**. المصدر الأول/الأساسي هو الجذر، ويشير كل مصدر آخر إلى المصدر الأصل عبر `scope`.

```
interface VirtualProxyEntityOptions {
  sources: VirtualProxyEntityOptionsSource[];
}

interface VirtualProxyEntityOptionsSource {
  alias: string | null;                 // unique; used as field-name prefix "alias$fieldNormalizedName"
  type: VirtualProxyEntitySourceTypeEnum; // Primary = 0 | Additional = 1
  sourceEntityNormalizedName: string;
  sourceQuery?: string | null;          // legacy; VirtualProxy pre-migration fallback only — see below
  sourceViewId?: string | null;         // VirtualProxy only, Since 1.10 — the bound Proxy view; see below
  scope: { alias: string | null; relation: string } | null; // join to a parent source via a relation
  level: number;                        // Primary = 1; child = parent.level + 1
  isMandatory?: boolean;                // default false
}
```

دلالات الحقول التي جرى التحقّق منها (استنادًا إلى مُزخرِفات النموذج وتعليقات JSDoc في `source.model.ts` وواجهة `@osos/shared`):

- **`alias`** – يساوي الاسم المعياري لكيان المصدر في مصدر `Primary`؛ أما في مصادر `Additional`، فهو الاسم المعياري للعلاقة مضافًا إليه لاحقة لإزالة الالتباس. ويُستخدم بادئةً للأسماء المعيارية للحقول المربوطة، بالصيغة `alias + "$" + fieldNormalizedName`.
- **`type`** – `VirtualProxyEntitySourceTypeEnum.Primary` (`0`) أو `Additional` (`1`). يُتوقَّع وجود مصدر `Primary` واحد بالضبط، وتحدّده المعالِجات باستخدام `sources.find(x => x.type === Primary)`.
- **`sourceViewId`** (لمصدر `VirtualProxy` الأساسي فقط، `Since 1.10`) – معرّف معرض `kind: Proxy` المربوط الذي يحكم سقف قدرات هذا الكيان، ويحكم أيضًا نطاق تجسيده عندما يكون المعرض مبنيًّا على استعلام. انظر حوكمة معرض الوكيل. وهو مطلوب لكل مصدر أساسي لـ `VirtualProxy` يُنشأ أو يُعاد ربطه من الآن فصاعدًا.
- **`sourceQuery`** (لمصدر `VirtualProxy` الأساسي، **قديم**) – معرّف `DataQuery`، `Since 1.10`. **يُرفَض** أيّ مصدر أساسي جديد أو مُعاد ربطه لـ `VirtualProxy` لا يزال يحمل `sourceQuery`؛ انظر الربط. ولا يبقى إلّا **حلًّا احتياطيًا لما قبل الترحيل**: فالكيان السابق للإصدار 1.10 الذي لم يُربَط بعد بـ `sourceViewId` لا يزال يحدّد نطاق تجسيده من هذا

الحقل. وتستمر مصادر `JoinerProxy` الإضافية/المحدَّدة النطاق في استخدام `sourceQuery` كما كانت؛ فالإلغاء يخصّ مصدر `VirtualProxy` **الأساسي**.

- **`scope`** – يربط `{ alias, relation }` مصدر `Additional` بمصدر أصل عبر علاقة مسمّاة (`relation` = الاسم المعياري للعلاقة). وتكون قيمته `null` للمصدر الأساسي. ويُمثَّل بالنموذج `VirtualProxyEntityOptionsSourceScope`.
- **`level`** – العمق في شجرة المصادر: تكون قيمة `Primary` هي `1`، وقيمة كل مصدر ابن هي `parent.level + 1`.

المجمِّع: `AggregatorVirtualEntityOptions`

```
interface AggregatorVirtualEntityOptions {
  sources: AggregatorVirtualEntityOptionsSource[]; // [0] is the primary source
}

interface AggregatorVirtualEntityOptionsSource {
  alias: string | null;
  type: VirtualProxyEntitySourceTypeEnum;   // Primary | Additional
  sourceEntityNormalizedName: string;
  sourceQuery: string | null;
  includedFields: string[];                 // default []
  fieldMappings: AggregatorEntityFieldMapping[]; // required
}

interface AggregatorEntityFieldMapping {
  normalizedName: string;                   // field in THIS (aggregator) entity
  type: AggregatorEntityFieldMappingType;   // Fixed=0 | Dynamic=1 | NotSet=2 | Expression=3
  value: string | null | unknown;           // source field name, fixed value, expression, or null
  optionsMappings?: OptionsMapping[];        // category option-id remapping
}

interface OptionsMapping {
  sourceOptionId: string;                   // option id in the SOURCE category field
  targetOptionId: string | null;            // corresponding option id in THIS entity
}
```

تحدّد قيم `type` في تعيين الحقل (`AggregatorEntityFieldMappingType`) كيفية تفسير `value`:

`type`	القيمة	معنى `value`
`Fixed`	`0`	قيمة حرفية ثابتة تُكتب في كل مُدخل مُجمَّع.
`Dynamic`	`1`	الاسم المعياري لحقل في كيان المصدر تُنسخ منه القيمة.
`NotSet`	`2`	يُترك الحقل دون ضبط.
`Expression`	`3`	تعبير يُقيَّم لإنتاج القيمة.

`optionsMappings` *(Since 1.6)* يتيح للمجمِّع تعيين معرّفات خيارات حقل الفئة عندما يستخدم كيانا المصدر والهدف معرّفات خيارات مختلفة للقيمة الدلالية نفسها. ويربط كل `OptionsMapping` معرّف `sourceOptionId` بمعرّف `targetOptionId` (الذي يقبل القيمة null). ولا يكون مطلوبًا إلا عند تجميع حقول الفئة التي تختلف معرّفات خياراتها بين المصدر والهدف.

مصادر المجمِّع مصفوفة *موضعية*؛ إذ يمثّل `sources[0]` الكيان الأساسي الذي يبدأ منه التجميع، أما العناصر المتبقية فهي مصادر إضافية مربوطة به.

حوكمة معرض الوكيل: أسقُف القدرات `Since 1.10`

`Since 1.10`، يتحكّم **معرض وكيل** مربوط بكيان `VirtualProxy` تحكّمًا كاملًا في قدرته على الكتابة: هل يُسمح أصلًا بالإنشاء أو التعديل أو الأرشفة أو الاستعادة أو الحفظ كمسوَّدة، وعبر أيّ حقول. ولا تُحكَم هذه القدرة بأيّ بيانات مخزَّنة على الكيان نفسه. والقاعدة الحاكمة هي: **يُشتقّ السقف عند كل قراءة، ولا يُخزَّن أبدًا**: فلا يحمل الكيان الوكيل أيّ سجلّ مخزَّن بما يُسمَح له بفعله حاليًّا، ولذلك يسري إبطال أيّ قدرة من المعرض الحاكم فورًا أينما استُخدم الوكيل.

معرض `kind: Proxy` وسطح محرِّره المقيَّد

يُضاف إلى معرض البيانات نوع (`kind`) ثالث، لتصبح الأنواع: `Implicit` (ينشئه النظام)، و `UserDefined` (معرض اعتيادي)، و`Proxy` ** – معرض مُنشأ خصيصًا لحوكمة كيان `VirtualProxy` . ويثبت `kind` المعرض بمجرد ضبطه، فلا يمكن تغييره لاحقًا، حتى باستخدام حمولة تُغفله فحسب.

تُقيَّد خيارات محرِّر معرض `kind: Proxy` الإداري عمدًا مقارنةً بخيارات المعرض الاعتيادي. وتقتصر كل عملية حفظ على قائمة سماح تضمّ الهوية (الاسم، الأيقونة، الوصف)، وكيان/استعلام المصدر، وقائمة الحقول، ومفاتيح تبديل عمليات الإنشاء/التحرير/الأرشفة (بقائمة حقول خاصة بكل عملية)، وخيار الإنشاء كمسوَّدة؛ و**تعود كل خاصية أخرى إلى قيمتها الافتراضية بصمت** عند الحفظ في الواجهة الخلفية، فلا يقتصر الأمر على إخفائها في واجهة الإدارة. وعلى وجه الخصوص، لا يملك معرض Proxy **حالة نشر، ولا خيارات وصول مجهول/موثَّق، ولا أذونات خاصة به**: إذ تتطابق مشاركته دائمًا مع مشاركة الكيان المضيف، لأنّ أذونات الكيان نفسه تتحكّم في الوصول إلى الكيان الذي يحكمه المعرض، لا أذونات المعرض الحاكم.

عندما يُنشأ كيان `VirtualProxy` دون تحديد معرض حاكم موجود، تُنشئ أسس **معرضاً افتراضياً تلقائياً** للقراءة فقط (لا عمليات مفعَّلة)، يشمل نطاقه كل حقل يملكه كيان المصدر حاليًا، ويُعرض كجدول. ويُفعِّل المسؤول لاحقًا عمليات الإنشاء/التحرير/الأرشفة فيه لجعل الوكيل قادرًا على الكتابة.

ربط كيان وكيل بمعرض مصدره `Since 1.10`

يُتحقَّق من ربط كيان `VirtualProxy` بمعرض حاكم (أو إعادة ربطه) عند كل إنشاء وكل تحديث:

- يجب أن يكون المعرض المحدَّد من النوع `kind: Proxy` فعلًا، وأن يحكم **نفس** كيان المصدر الذي يستند إليه الكيان الوكيل؛ فلا يمكن لوكيل الارتباط بمعرض مُنشأ لكيان آخر.
- يجب ألّا يحمل مصدر الكيان بعد الآن قيمة قديمة في `sourceQuery` ؛ انظر نموذج بيانات خيارات النوع أعلاه.
- يجب ألّا يشكّل الربط **حلقة (cycle)**، وألّا يتجاوز عمق سلسلته **10 معارض**، وأن ينتهي حلّ السلسلة إلى مضيف حقيقي غير وكيل. ويُسمَح لمعرض Proxy بأن يحكم كيان `VirtualProxy` آخر (وكيل مربوط بوكيل)، ما دامت السلسلة تنتهي إلى بيانات مصدرية حقيقية.

يُفرَض سقف الحقول في **الاتجاهين كليهما**: يُرفَض تضييق قائمة حقول معرض حاكم إذا كان سيحرم كيانًا وكيلًا مربوطًا من حقل لا يزال يحتاج إليه، ويُرفَض تضييق قائمة حقول الكيان الوكيل نفسه إذا كان سيتجاوز ما لا يزال معرضه الحاكم يغطّيه.

بمجرد الربط، لا يمكن إلغاء ربط كيان `VirtualProxy` ؛ إذ يُرفَض بالكامل أيّ تحديث يُغفل المعرض الحاكم لكيان مربوط بالفعل، والطريقة الوحيدة لتغييره هي إعادة الربط بمعرض `kind: Proxy` آخر.

اشتقاق القدرات

عند القراءة، تشتقّ أسس **سقف قدرات** الوكيل بتتبّع سلسلة ربطه (عبر كل حلقة وسيطة يرتبط فيها وكيل بوكيل) وإجراء **تقاطع (intersect)** للقدرات التي يمنحها كل معرض مربوط؛ فزيادة طول السلسلة لا يمكن إلّا أن تُضيِّق نطاق المسموح به،

ولا يمكن أن توسِّعه أبدًا. وأيّ فشل أثناء تتبّعها – كيان يتعذّر حلّه، أو سلسلة لا تنتهي إلى مضيف حقيقي، أو أيّ خطأ آخر – يُنتِج سقفًا **مغلَقاً كلياً**: لا تُمنَح أيّ قدرة، ولا تُكشَف أيّ حقول. ولا يُفترَض احتياطيًا أنّ السقف «غير مقيَّد».

يغطّي السقف المُشتَقّ:

القدرة	المعنى
`bound`	تكون قيمته `false` فقط عندما يتعذّر حلّ سلسلة الربط كليًا.
`fields`	مجموعة مسارات الحقول الناتجة عن التقاطع التي يجوز للوكيل كشفها.
الشاشات (قائمة المُدخلات / تفاصيل المُدخل / اللوحة)	الشاشات التي تسمح بها السلسلة أصلًا.
`allowCreateEntryAsDraft`	ما إذا كان يمكن بدء مسوَّدة على الوكيل.
الإنشاء / التحرير / الأرشفة	ما إذا كانت كل عملية ممنوحة؛ فالعملية الغائبة عن كل معرض في السلسلة لا تمنح شيئًا، ولا يوجد تجاوز منفصل.
الحفظ رغم عدم الصلاحية عند الإنشاء / التحرير	ما إذا كانت تُقبَل، لكل عملية، كتابة رغم إخفاقها في التحقّق.
عرض المُدخلات غير الصالحة، ومرشِّحاه الاثنان	تُحكَم معًا؛ فلا يمكن تفعيل المرشِّحَين إلّا عندما تكون شاشة المُدخلات غير الصالحة نفسها مفعَّلة.

يُطبَّق هذا السقف نفسه، قبل الاستخدام، على أيّ معرض مربوط فعليًا بالكيان الوكيل، أي المعرض الذي يفتحه المستخدم النهائي للعمل مع مُدخلاته: فتُحصر قائمة حقوله ضمن السقف، وتُحصر عملياته في ما يمنحه السقف، وتُقاطَع قائمة حقول كل عملية متبقية مع السقف أيضًا. وبذلك لا يمكن لمعرض يحدّد «كل الحقول» لعملية ما أن يتجاوز ما يُسمَح به للوكيل فعليًا. وهذا ما يجعل السقف **مفروضًا، لا استرشاديًا فحسب**: إذ يطبّقه كل مسار قراءة يحلّ معرضًا يستند إلى كيان وكيل، ولا يقتصر تطبيقه على محرِّر الإدارة.

`Since 1.10`، تتضمّن تفاصيل كيان `VirtualProxy` أيضًا هذا السقف المُشتَقّ في الحقل `proxyCapabilities` (عند قراءة كيان واحد فقط، لا عند قراءة القوائم)، ليعرف المستدعي مسبقًا ما يُسمَح له بفعله دون الحاجة إلى الاستقصاء. ويكون الحقل **غائباً** في الكيان غير الوكيل، وحاضرًا بقيمة `bound: false` في الوكيل الذي تعذّر حلّ ربطه؛ والحالتان متمايزتان عمدًا. انظر استدعاء واجهة HTTP للاطّلاع على نقطة النهاية.

> **ملاحظة**
>
> لا يزال `proxyCapabilities` غير مُدرَج في مواصفة OpenAPI المنشورة لنقطة نهاية الكيانات. اعتمد هذا الدليل مرجعًا موثوقًا لبنيته إلى أن تُحدَّث المواصفة لتشمله.

تتالي منتقيات الحقول المُوجَّه بالسقف

يُضيِّق السقف نفسه أيضًا نطاق ما يراه المسؤول أثناء **بناء** قائمة حقول الكيان الوكيل نفسه؛ إذ يُستبعَد أيّ حقل يحجبه سقف المعرض الحاكم من المنتقيات المستخدَمة لضبط الكيان الوكيل (ومن أيّ شيء يُبنى على قائمة الحقول تلك، مثل أعمدة الجدول أو قائمة حقول عملية ما). ويُترك الحقل الجذري المخصّص للعرض فقط دون فحص، لأنه لا يقرأ بيانات فعلية ولا يكتبها مطلقًا، لكن يُفحَص كل حقل متداخل داخل كتلة على حدة، لأنّ حارس الحفظ يفحص الحقول بالطريقة نفسها؛ فإعفاء حقل بصمت في المنتقي وحده سيؤدّي إلى فشل حفظه لسبب لا يظهر في المنتقي. انظر معارض البيانات والواجهات [بناء التطبيقات › معارض البيانات وأوجهها] لمعرفة كيف يُظهر مُصمِّم المعرض نفسه هذا التضييق أثناء بناء المعرض الحاكم.

واجهة API

تقع جميع نقاط النهاية ضمن `DataEntitiesController` في data-service (`/v1/entities/...`). وتطابق متون الطلبات أدناه بُنى DTO التي تخضع للتحقّق.

إنشاء كيان وهمي

`POST /v1/entities/virtual`

متن الطلب (`CreateVirtualDataEntityDtoModel`):

الحقل	النوع	مطلوب	ملاحظات
`name`	`LocalizedValue`	نعم	
`description`	`NullableLocalizedValue`	لا	
`normalizedName`	`string`	نعم	فريد؛ يُتحقَّق من مطابقته لـ `NormalizedNameValidation` (الطول + النمط). تُرفض الأسماء المقيّدة.
`permissions`	`Permission[]`	لا	يُخزَّن في حقوق الوصول، لا في وثيقة الكيان.
`type`	`DataEntityTypeEnum`	نعم	يجب أن تكون قيمته إحدى القيم التالية: `VirtualProxy` (50) أو `JoinerProxy` (51) أو `VirtualAggregator` (52).
`typeOptions`	انظر أدناه	نعم	
`icon`	`Icon`	نعم	
`module`	`UUID`	نعم	يجب أن **لا** يكون معرّف الوحدة الأساسية (وإلا يُرمى `ImplicitModuleException`).
`entryNames`	`DataEntityEntryNames`	نعم	
`displayOption`	`DataEntityDisplayOptionEnum`	لا	القيمة الافتراضية `KeyValue` .

يقبل `typeOptions` شكلين (يُعالَجان في `createVirtual`):

1. **مسطّح / تفاعلي** – `{ sourceEntityNormalizedName, sourceQuery?, sourceViewId? }` (أُضيف `sourceViewId` `Since 1.10` ، لـ `VirtualProxy` فقط – انظر ربط الوكيل، وقت الإنشاء أدناه). يغلّف المتحكّم هذه البنية في مصدر واحد من نوع `VirtualProxyEntityOptionsSource` ، أساسي (`Primary` ، `alias: null` ، `type: Primary` ، `scope:` `null`).
2. **مُخزَّن / تمريري** – `{ sources: [...] }` . إذا كان `typeOptions.sources` مصفوفةً، فإنه يُمرَّر **دون تعديل**. يستخدم استيراد التغليف هذا المسار لإعادة تشغيل كيان مُصدَّر سابقًا (مصادر وكيل/موصِّل مع `scope` ، أو مصادر مجمِّع مع `fieldMappings`): إذ إنّ إعادة بنائه بصيغة `VirtualProxyEntityOptions` ستجرّده من الحقول الخاصة بالمجمِّع.

تُنشئ أسس الكيان مباشرة وتعيده.

مثال مبسّط (وكيل تفاعلي لكيان `tickets` ، دون تحديد معرض حاكم – تُنشئ أسس واحدًا تلقائيًا):

```
POST /v1/entities/virtual
{
  "name": { "en": "Open Tickets" },
  "normalizedName": "open_tickets",
  "type": 50,
  "typeOptions": {
    "sourceEntityNormalizedName": "tickets"
  },
  "icon": { "...": "..." },
  "module": "…module-uuid…",
  "entryNames": { "...": "..." }
}
```

ربط الوكيل، وقت الإنشاء `Since 1.10`

عند إنشاء `VirtualProxy` تحديدًا، تحدّد أسس المعرض الحاكم **قبل** إنشاء الكيان نفسه، استنادًا إلى `sourceQuery` / `sourceViewId` للمصدر الأساسي:

1. **تسمّي `sourceViewId` .** يجب أن يكون المعرض المحدَّد موجودًا وأن يجتاز فحوص الربط أعلاه. لا يُنشأ معرض افتراضي.
2. **لا تسمّي `sourceViewId` ولا `sourceQuery` .** يُنشأ **معرض وكيل افتراضي للقراءة فقط تلقائياً**، يشمل نطاقه كل حقل يملكه كيان المصدر حاليًا، ويُربَط بالكيان الجديد. وإذا فشل إنشاء الكيان بعد ذلك، يُحذَف المعرض المُنشَأ تلقائيًا — إلّا إذا كان الكيان قد حُفظ فعليًا رغم الخطأ؛ فعندئذٍ يُحتفَظ بالمعرض عمدًا لتجنّب ترك كيان محفوظ ذي ربط معلَّق.
3. **تسمّي `sourceQuery` قديمًا مضمَّنًا دون `sourceViewId` .** تُرفَض هذه الحالة تمامًا عند إنشاء كيان **جديد**. الاستثناء الوحيد هو **استيراد تغليف** يعيد تشغيل حزمة بُنيت قبل 1.9، وقد لا تزال تحمل `sourceQuery` مضمَّنة؛ إذ يُقبَل هذا الاستيراد، ويُؤجَّل نقل الاستعلام إلى معرض مربوط إلى عملية ترحيل لاحقة، بدلًا من إفشال الاستيراد.

يُرفَض `sourceViewId` تمامًا عند إنشاء `VirtualAggregator` / `JoinerProxy` — فالربط خاص بـ `VirtualProxy` فقط.

تحديث كيان وهمي

`PUT /v1/entities/virtual/{id}`

متن الطلب (`UpdateVirtualDataEntityDtoModel`): `name` و`description` و`permissions?` و`typeOptions?` و(`VirtualProxyEntityOptions | AggregatorVirtualEntityOptions`) و`icon` و`entryNames` و`displayOption?` و`fields[]` و`internalEntities[]` و`internalCollections[]` و`templates?` و`updateIdentifier`** و** مطلوب.

فحوص التزامن والسلامة التي تُجرى قبل إرسال أمر التحديث:

- لا يمكن تحرير الكيانات الضمنية (`ImplicitEntityUpdateException`).
- يجب أن يساوي `updateIdentifier` القيمة المخزّنة، وإلا يُرمى `UpdateIdentifierConflictException` (تزامن تفاؤلي).
- يجب ألّا يكون مستخدم آخر قد أقفل الكيان (`LockedObjectException`).

يُصدَر الأمر `UpdateVirtualDataEntityCommand`، ويُعاد الكيان.

`Since 1.10`، إذا حدّدت `typeOptions` الواردة في تحديث لـ `VirtualProxy` قيمة جديدة لـ `sourceViewId`، تُجرى فحوص الربط نفسها المذكورة أعلاه مجدّدًا على المعرض الجديد — بما فيها التحقّق من أنّ مجموعة حقول الكيان نفسها (التي ربما حُدِّثت أيضًا) لا تزال ضمن سقف المعرض الجديد. ويُرفَض إغفال `sourceViewId` عند تحديث كيان **مربوط بالفعل**: فلا يمكن إلغاء ربط وكيل مربوط بالفعل عبر تحديث.

قراءة تفاصيل الكيان

يعيد طلب `GET` إلى نقطة نهاية تفاصيل الكيان كائنًا من نوع `DataEntityDto` تُستخلَص فيه قيمة `typeDetails` من `typeOptions`:

- **`VirtualAggregator`** ← `DataEntityAggregatorDetailsDto` مع `primarySource` و`additionalSources[]` و`mappedSources` (تُحسَب عبر `DataEntityService.getVirtualEntityMappedSources`).
- **`VirtualProxy` / `JoinerProxy`** ← `DataEntityJoinerDetailsDto` مع `primarySource`، وعنصر في `sources[]` لكل مصدر يتضمّن `scope.relation`، بعد حلّه ليحمل DTO العلاقة.

إذا لم يتضمّن الكيان `typeOptions.sources`، يُسجَّل تحذير وتُعاد التفاصيل مع `typeDetails` فارغة.

`Since 1.10`، تتضمّن تفاصيل كيان `VirtualProxy` أيضًا `proxyCapabilities` – انظر اشتقاق القدرات أعلاه.

الحذف

يؤرشف `DELETE /v1/entities/{id}` الكيان (نقطة نهاية حذف مشتركة، وليست خاصة بالكيانات الوهمية). وابتداءً من `Since 1.10`، يستطيع مسؤول النظام أيضًا إزالة كيان وهمي نهائيًا باستخدام `DELETE /v1/entities/{id}/permanent` – انظر العمليات الإدارية.

تحديد النطاق: أيّ مدخلات المصدر تُوكَّل

`Since 1.10`، يُحدَّد النطاق الفعّال لكيان `VirtualProxy` عبر **معرضه الوكيل المربوط**، لا عبر استعلام مرفق بالكيان مباشرة:

- **مربوط (`sourceViewId` مضبوط – الحالة الاعتيادية منذ 1.10)** – يُحمَّل المعرض المربوط. فإن كان المعرض **مبنياً على استعلام**، كان استعلامه هو النطاق الفعّال. وإن كان المعرض **مبنياً على مخزن**، فلا يُطبَّق مرشِّح، ويُوكَّل كل مُدخل من المصدر. أمّا المعرض المبني على استعلام دون تحديد أيّ استعلام، أو الربط الذي يتعذّر حلّه تمامًا، فيُعامَل كلٌّ منهما على أنّه **"لا تُحدِّث/تكتب شيئاً"**، وليس «لا مرشِّح» – أي إنّ الربط المعطوب يمنع العملية عند الفشل، ولا يتّسع نطاقه ضمنيًا أبدًا ليحاكي المصدر كاملًا.
- **غير مربوط بعد (كيان يسبق 1.10 فقط)** – يرجع مباشرة إلى الحقل القديم `sourceQuery` (`null` تعني «كل المُدخلات»)، تمامًا كما كان قبل هذا الإصدار.

بمجرد تحديد معرّف الاستعلام الفعّال (أو `null`، أو القيمة الحارسة «لا شيء»)، تبني أسس معايير الجلب بالطريقة السابقة نفسها:

- إذا حُدِّد معرّف مُدخل مصدر بعينه، فإنها تحصر النتائج في ذلك المُدخل.
- إذا كان هناك استعلام نطاق سارٍ يُحمَّل معيار الاستعلام ويُدمَج باستخدام العامل AND.
- إذا لم يُحدَّد أيٌّ منهما، تُعاد **كل** مُدخلات المصدر.

تُجلَب مُدخلات المصدر دون حلّ العلاقات أو الفئات، مع الاقتصار على الحقول التي يحتاجها الوكيل فعليًا: المعرّف، والكيان، والقيمة، وحالة دورة الحياة، والحالة، والسجل التاريخي للحالة، والمخزن، وعَلَم عدم الصلاحية وسببه، ومعلومات الإنشاء/التحديث.

يحكم النطاق نفسه أيضًا كل **كتابة** تمرّ عبر الوكيل – انظر الكتابة عبر التوجيه.

كيف يُبنى مدخل وكيل

في كيانات `VirtualProxy`، يحوّل `ProxyDataEntryService.refreshEntry` كل مُدخل مصدر مؤهّل إلى مُدخل وكيل من نوع `DataEntry`:

- **معرّف حتمي** – `hashId(normalizedName, sourceId)` = `Utils.hashString("${dataEntity.normalizedName}_${sourceEntry.id}")`. وبذلك يبقى معرّف الوكيل ثابتًا ومتطابقًا عند إعادة التوليد لكل زوج (كيان وكيل، مُدخل مصدر).
- **`entryType`** = `EntryTypeEnum.Virtual` (`1`).
- **`basedOn`** = `ReferenceProperty { dataEntity, dataEntry }` واحد يشير إلى مُدخل المصدر (يُستخدم لاحقًا للعثور على مُدخلات الوكيل المقابلة لمُدخل مصدر معيّن عبر فهرس الكيان `basedOn.dataEntry`).
- **`value`** – يُنسخ حقلًا بحقل من مُدخل المصدر باستخدام `copyValuesByPaths` على `dataEntity.getAllFieldsPaths()`. يتتبّع الناسخ المسارات المنقّطة ويتوغّل داخل المصفوفات، فيُعيد بناء الحقول المتداخلة والحقول ذات القيم المصفوفية في الهدف.
- **مُنعكِس من المصدر** – `isInvalid` و `invalidReason` و `state` و `stateHistory` و `store`، ومعلومات الفاعل في `creationInfo` / `updateInfo` (`doneBy` / `delegatee`، مع الرجوع إلى `creationInfo` عند غياب `updateInfo`). تُضبط `entityVersion` على قيمة `version` لكيان الوكيل.
- عند تمرير `originalEntry`، تُحدَّث تلك النسخة في موضعها بدلًا من إنشاء نسخة جديدة (فيُحتفَظ بالمعرّف و `basedOn` و `creationInfo` عند التحديث).

الكتابة عبر التوجيه: كتابات المدخل على كيان وكيل `Since 1.10`

`Since 1.10`، عندما يسمح سقف القدرات الخاص بكيان `VirtualProxy` بذلك، يُعاد توجيه الكتابات التي تُجرى مباشرة على الوكيل **توجيهها إلى الكيان المضيف (المصدري)** بدلًا من رفضها. ولا يتاح أيّ مما يلي على وكيل غير مربوط أو يتعذّر حلّ ربطه.

ما يمرّ عبر التوجيه إلى المضيف

العملية	تمرّ عبر التوجيه؟
إنشاء مُدخل واحد (إنشاء اعتيادي، أو عبر معرض بيانات؛ ويمكن الوصول إليه أيضًا من إجراء «إنشاء مدخل بيانات» لتفاعل)	نعم
الإنشاء بالدُفعة	نعم للصفّ الرئيسي؛ وتُرفَض الصفوف المرتبطة بدُفعة الإنشاء تمامًا، لأنّها ستحتاج إلى الإشارة إلى صفّ الوكيل قبل أن يُنتجه حدث تجسيد كتابة المضيف نفسه.
تحديث كامل (`PUT`) وتصحيح (`PATCH`، عمليات JSON-Patch)	نعم – تُقارَن القيمة الواردة وتُترجَم، ثم تُطبَّق عملية الكتابة على المضيف.
التصحيح بالدُفعة	نعم، صفًّا بصفّ، ضمن المعاملة نفسها التي تشمل بقية الدُفعة.
الأرشفة والاستعادة – فرديًا وبالدُفعة	نعم – يُحلّ صفّ الوكيل إلى مُدخله المضيف، ثم يُؤرشَف ذلك المُدخل أو يُستعاد.
إنشاء/تحديث مسوَّدة	نعم، ويخضع ذلك بصورة مستقلة لعَلَم المسوَّدة الخاص بالسقف. وتأخذ المسوَّدة الجديدة مخزنها وسير عملها من الكيان المضيف، لأنّ الوكيل لا يملك أيًّا منهما.
كتابات `JoinerProxy` / `VirtualAggregator`	**لا** – الكتابة عبر التوجيه خاصة بـ `VirtualProxy` فقط؛ ويظل النوعان الوهميان الآخران إسقاطات للقراءة فقط. والتصحيح (Patch) هو الاستثناء القديم الوحيد: فقد كان مُدخل المُجمِّع يدعم التصحيح دائمًا، مع ترجمته عبر تعيينات حقوله، لا عبر سقف.

كيف تُبنى الكتابة المُعاد توجيهها وتُحدَّد نطاقها

تتبع جميع مسارات الكتابة عبر التوجيه النمط نفسه:

1. يجب أن يمنح السقف القدرة المعنيّة (الإنشاء/التحرير/الأرشفة)، وإلّا تُرفَض الكتابة تمامًا.
2. تُحصَر الكتابة في مجموعة الحقول المسموح بها – أي حقول السقف، مع تضييقها أكثر وفق أيّ معرض استخدمه المستدعي فعليًا لإجراء الكتابة.
3. يُرفَض أيّ شيء **خارج** تلك المجموعة، ولا يُتجاهَل ضمنيًا أبدًا. (ثمة استثناء واحد محدود ومتعمَّد: تُعفى بضعة حقول من البيانات الوصفية لسير العمل والترتيب، التي يعيد نموذج تحرير اعتيادي إرسالها دائمًا، من فحص السقف – لكن فقط عندما تُستعاد حرفيًا من المُدخل المخزَّن، وعلى مسار التحرير العادي وحده؛ أمّا المستدعي الذي يقدّم قيمًا بديلة من عنده لتلك البيانات الوصفية فلا يحصل على إعفاء.)
4. تُفحَص الكتابة مقابل النطاق الفعّال للوكيل، استنادًا إلى القيمة التي ستُنتجها فعليًا، وقبل حفظ أيّ شيء. وتُرفَض أيّ كتابة ستقع خارج نطاق المعرض الحاكم أو ستنقل مُدخلًا إلى خارجه، بدلًا من تنفيذها خارج الحدود أو تجاهلها ضمنيًا.
5. تصل الكتابة إلى الكيان المضيف ضمن مساحة المعرّفات الخاصة به؛ فمعرض الوكيل الحاكم لا صلة له بقواعد الكتابة الخاصة بالمضيف على مستوى المعرض.

لا تضيف الكتابة عبر التوجيه أيّ منطق تزامن جديد: فأحداث الإنشاء/التحديث/الأرشفة/الاستعادة المعتادة الناتجة عن كتابة المضيف نفسها هي التي تحدّث صفّ الوكيل لاحقًا، كما يحدث مع أيّ تغيير آخر في المصدر – انظر دورة الحياة. والاستثناء الوحيد هو استجابة HTTP للإنشاء، إذ تتضمّن فورًا معاينة للصفّ المكتوب للتوّ بصيغة وكيل، دون انتظار وصول ذلك التحديث؛ أمّا صفّ الوكيل الفعلي المحفوظ، فيظلّ هو الصفّ الذي يكتبه معالِج التحديث بمجرد وصول الحدث.

تزامن الحقول: إبقاء تعريفات حقول الوكيل المنسوخة متسقة `Since 1.10`

تعريفات حقول كيان `VirtualProxy` هي **نسخة** مأخوذة من مصدره وقت إنشاء الكيان الوكيل نفسه أو تحديث مجموعة حقوله، وليست مرجعًا حيًّا. `Since 1.10` ، تحافظ أسس أيضًا على اتساق تلك النسخة عندما يتغيّر تعريف الحقل **المصدري** لاحقًا: فقبل ذلك، كان الحقل الذي يصبح إلزاميًا على المضيف يظلّ اختياريًا على كل وكيل نُسخ منه، مما قد يؤدي إلى رفض كتابة عبر الوكيل بسبب متطلَّب لم يُظهره نموذجه إطلاقًا.

تقتصر إعادة المزامنة من المصدر على مجموعة محدودة ومقصودة من الخصائص – نوع البيانات، والتحقّقات، وخيارات العرض، و(بالنسبة لعَلَم `unique` تحديدًا) الجزء الوحيد من خيارات عرض الحقل الذي يمثّل قاعدة صحّة لا خيار عرض. ويظلّ كل ما عدا ذلك – كل إعداد عرض، وأيّ متطلَّب لسير العمل – خاصًا بالوكيل: أمّا **المتطلَّب** تحديدًا فلا يُورَث أبدًا، لأنّ الكيان الوكيل لا يملك سير عمل خاصًا به يُنفَّذ المتطلَّب في سياقه.

تُطبَّق عملية إصلاح تُنفَّذ مرة واحدة هذه القاعدة نفسها على جميع كيانات `VirtualProxy` التي كانت تعريفات حقولها قد انحرفت بالفعل قبل هذا الإصلاح، دون توليد إشعارات تغيير خاصة بها؛ إذ تقتصر على استدراك تعريفات الحقول المخزَّنة التي تلتقطها الفهرسة الاعتيادية أصلًا عند التحديث التالي للكيان.

حالة القفل المشتقّة من سير عمل المصدر على مدخل وكيل `Since 1.10`

لا يملك كيان `VirtualProxy` سير عمل خاصًا به، شأنه شأن أيّ كيان وهمي؛ لذا لا يُظهر صفّ الوكيل أيّ حالة لسير العمل ما لم تشتقّها أسس. `Since 1.10` ، تشتقّها المنصة بالفعل: إذ يتبع قفل تحرير صفّ الوكيل الآن سير العمل النشط لمُدخله **المصدري**، بالطريقة نفسها التي تتبع بها رؤية إجراء التعديل على مدخل التجميع سير عمل مصدره بالفعل.

- إذا بلغ سير عمل المصدر حالة نهائية، يُقفَل كل حقل غير كتلة على الوكيل.

- إذا وُجدت خطوة نشطة في سير العمل ولم يكن المستخدم الحالي مالكها، يُقفَل كل حقل أيضًا.
- وإلّا، تُطبَّق متطلَّبات خطوة المصدر وقواعد الحقول المطلوبة الخاصة بالمعرض الحاكم، ويُقفَل كل حقل في الوكيل يكون نظيره في المصدر مخفيًا أو مقفولًا، لمطابقة حالته – ولا **يُضيف** ذلك إلّا أقفالًا إلى أقفال المعرض الحاكم نفسه، ولا يزيل أيًّا منها أبدًا.
- إذا تعذّر حلّ مصدر الصفّ، أو لم يكن للمصدر سير عمل أصلًا، تبقى حالة قفل الصفّ كما يحدّدها المعرض الحاكم نفسه؛ ولا تُشتَق أيّ حالة إضافية.

دورة الحياة: إبقاء مدخلات الوكيل متزامنة

مدخلات `VirtualProxy` **مُجسَّدة**، وتُحافِظ معالِجات الأحداث على ناقل الخدمة على تزامنها. يحدّد كل معالِج الكيانات الوكيلة المتأثّرة باستعلام بالصيغة التالية:

```
where: {
  type: DataEntityTypeEnum.VirtualProxy,
  'typeOptions.sources.sourceEntityNormalizedName': <sourceEntity.normalizedName>,
}
```

> **ملاحظة**
>
> تُرشِّح معالِجات التحديث هذه صراحةً حسب `type: VirtualProxy` – فهي خاصة بمدخلات الوكيل. أما مدخلات `JoinerProxy` و `VirtualAggregator` فلا تُجسِّدها هذه الخدمة **ماديًا**، لكنها تُحتسَب أيضًا: فلكلٍّ منها خدمة تجسيد مستقلة تُشغِّل خط أنابيب التجميع الخاص بها. انظر كيف تُجسَّد مدخلات المُوصِّل أدناه للاطّلاع على آلية المُوصِّل، وكيف تُجسَّد مدخلات المُجمِّع للاطّلاع على آلية المُجمِّع.

الحدث المُطلِق	المعالِج	السلوك
إنشاء **كيان** مصدر (من النوع `VirtualProxy`)	`RefreshProxyEntriesOnDataEntityCreatedHandler`	يحدّد كيان المصدر الأساسي، ويجلب جميع مدخلات المصدر المطابقة، ويطبّق `refreshEntry` على كلٍّ منها، ثم ينفّذ `createAll` عليها في مجموعة الوكيل؛ وينشر `DataEntryCreatedEvent` لكل مدخل. وتفشل الرسالة إذا كان كيان المصدر مفقودًا.
إنشاء **مدخل** مصدر	`RefreshProxyEntryOnDataEntryCreated`	لكل كيان وكيل يستند إلى ذلك المصدر: إذا لم يطابق المدخل الجديد النطاق الفعّال للوكيل، يتخطّاه المعالِج؛ وإلا يُنشئ مدخل الوكيل وينشر `DataEntryCreatedEvent` .
تحديث **مدخل** مصدر	`RefreshProxyEntryOnDataEntryUpdated`	يستخدم فهرس الكيان (`basedOn.dataEntry`) للعثور على مدخل الوكيل الموجود، ثم يوفّق حالته: يُنشئه إن أصبح مطابقًا، ويحذفه إن لم يعد يطابق النطاق الفعّال، وإلا يحدّثه في مكانه. وينشر Created/Updated/Deleted تبعًا لذلك.
حذف **مدخل** مصدر نهائيًا	`DeleteProxyEntryOnDataEntryPermanentlyDeleted`	يعثر على مدخلات الوكيل عبر `basedOn.dataEntry` ويحذفها بالتتابع مع صفوف فهرسها.
تحديث **استعلام** مصدر (تغيّرت المعايير)	`RefreshProxyEntriesOnQueryUpdatedHandler`	يعمل فقط عند `criteriaChanged` . ولكل كيان وكيل مرتبط بذلك الاستعلام، يُعيد **بناء المجموعة كاملةً**: يحذف جميع مدخلات الوكيل، ويعيد إنشاءها من نتيجة الاستعلام الحالية، وينشر أحداث Created/Updated/Deleted بمقارنة المعرّفات.

توجد معالِجات إضافية للمدخلات عند الانتقال بين حالات الأرشفة / الحذف / الإبطال (`refresh-proxy-entry-on-data-entry-archived` و `-data-entries-invalidated` و `-deleted` و `archived`)، وتنقل تغيّرات الحالة هذه إلى نسخ الوكيل.

تتّبع المعالِجات نهجًا لمعالجة الأخطاء: يُعامَل `ENTITY_NOT_FOUND` على أنه خطأ غير قاتل (يُسجَّل ويُتخطّى / تفشل الرسالة)، أما الأخطاء الأخرى فتُسجَّل ويُعاد رميها ليتمكّن الناقل من إعادة المحاولة.

كيف تُجسَّد مدخلات المُوصَّل

بينما تُرشِّح معالِجات تحديث الوكيل أعلاه حسب `type: VirtualProxy` فقط، تُجسَّد مدخلات `JoinerProxy` عبر خدمة منفصلة (`JoinerDataEntryService`). يبني `buildJoinerStages` فيها خط أنابيب تجميع MongoDB يُنفَّذ بعد إلحاق مرحلة `$out` به (يُلحق `DataEntryRepository.refreshVirtualEntity` المرحلة `{$out: <normalizedName>}`)، فتُستبدَل مجموعة التخزين التي يستند إليها المُوصِّل بالصفوف المرتبطة حديثًا.

يُبنى خط الأنابيب من قائمة `sources` المسطّحة للمُوصِّل:

- `$lookup` **لكل مصدر إضافي** – يُربط المصدر الأساسي بكل مصدر إضافي عبر علاقة `scope` الخاصة به. ففي الرابط من **المستوى 0** (غير المصفوفي)، يستخدم البحث `localField` / `foreignField` ؛ أما الرابط **ذو القيمة المصفوفية** فيستخدم خط أنابيب `let` + `$expr` بدلًا من ذلك.
- `$unwind` **لكل مستوى مصفوفة** – تُولَّد عملية تفكيك واحدة لكل مستوى مصفوفة على امتداد المسار المرتبط (مع مراعاة الكتلة والحقل وتعدّد القيم). ويُطبَّق `destructTargetField` عندما يكون الحقل الهدف قابلًا للتكرار، ولكن **ليس** عندما يكون متعدّد القيم.
- **ربط داخلي مقابل ربط أيسر** – يضبط المصدر الإضافي **الإلزامي** (`isMandatory`) (أو مرشّح معرّف إضافي) القيمة `preserveNullAndEmptyArrays: false` ، أي لا تُحفَظ إلا المدخلات التي لها مطابقة في ذلك المصدر (**ربط داخلي**). وإلا فتكون القيمة `true` ، فتُحفَظ المدخلات حتى دون مطابقة (**ربط أيسر**). وهكذا يُنفَّذ، عند القراءة، خيار الإلزام أو الاختيار المحدّد عند إنشاء مُوصِّل [بناء التطبيقات ‹ الكيانات الوهمية].
- **التتبّع (Provenance)** – يسجّل كل صف مرتبط مدخلات المصدر التي بُني منها في `basedOn[]` ، مستخدمًا `includeArrayIndex` لتكون المعرّفات المُولَّدة حتمية.

كما في مدخلات الوكيل، تُحدَّث مدخلات المُوصِّل عند تغيّر كيانات/مدخلات المصدر وعند **تحديثات الاستعلامات المحفوظة** (إذ يمكن أن يستند المُوصِّل إلى استعلام)، وتُزال عند الحذف النهائي لمدخل مصدر – وذلك عبر معالِجات الناقل ضمن `joiner-entity/*` و `joiner-entries/*` و `refresh-joiner-entries-on-query-updated` .

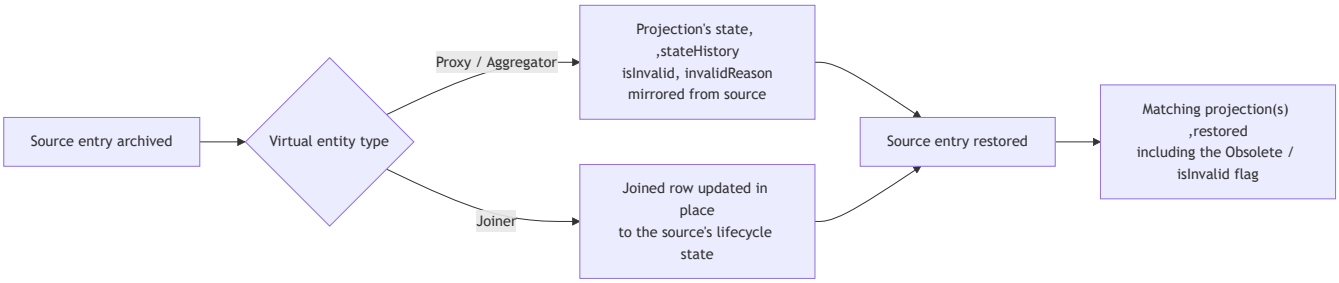
كيف تُجسَّد مدخلات المُجمِّع

تُجسَّد مدخلات المُجمِّع بالطريقة نفسها التي تُجسَّد بها مدخلات المُوصِّل: إذ تُشغِّل خدمة مخصّصة (`AggregatorDataEntryService`) خط أنابيب تجميع MongoDB خاصًا بها – يُبنى من مصدر المُجمِّع الأساسي ومصادره الإضافية وتعيينات حقولها – وتحدّث مجموعة التخزين التي يستند إليها المُجمِّع بنتائجه، بالتوازي مع `ProxyDataEntryService` لـ `VirtualProxy` و `JoinerDataEntryService` لـ `JoinerProxy` .

وكما في المُوصِّل، تطابق عملية إعادة بناء المُجمِّع الآن مدخلات المصدر ذات الحالة **Active** أو **Archived** (مع استثناء `Draft` كما في السابق)، بدلًا من `Active` وحدها – انظر انتشار حالة الأرشفة/الاستعادة لمعرفة سبب أهمية ذلك.

انتشار حالة الأرشفة/الاستعادة، لكل أنواع الكيانات الوهمية `Since 1.8`

Since 1.8، أصبحت أرشفة مدخل مصدر واستعادته تنقلان حالة دورة حياته على نحو صحيح إلى إسقاطات **الأنواع الثلاثة** من الكيانات الوهمية – الوكيل والمُوصِّل والمُجمِّع – بدلًا من الوكيل وحده. ويختلف ذلك عن التجسيد/الإنتاج (فلكل نوع مسار تجسيد خاص به، كما سبق)؛ إذ يتعلّق تحديدًا بإبقاء حالة أرشفة/استعادة صف مُجسَّد بالفعل (أو مرتبط أو مُجمَّع) مطابقةً لحالة مصدره.



- **معالجة أرشفة الوكيل والمجمِّع تعكس الآن دورة الحياة كاملةً، لا `state` فقط.** قبل 1.8، كانت أرشفة مدخل مصدر لوكيل أو مُجمِّع تغيّر `state: Archived` فقط في الإسقاط. وأصبحت الآن تنسخ `state` و`stateHistory` و`isInvalid` و`invalidReason` مباشرةً من مدخل المصدر – بما يطابق تمامًا ما يكتبه مسار الاستعادة – بحيث يحتفظ المدخل الذي أصبح باليًا قبل أرشفته، على سبيل المثال، بتلك الحالة على نحو صحيح.
- **صفوف المُوصِّل تُحدَّث في مكانها بدلاً من حذفها.** كانت أرشفة مصدر تحذف سابقًا صف المُوصِّل المطابق (ومدخل فهرسه) حذفًا نهائيًا، مما يعني أن الاستعادة لا تجد شيئًا لتُعيده. وأصبح صف المُوصِّل الآن يُحدَّث في مكانه ليطابق حالة دورة حياة المصدر بدلًا من ذلك.
- **معالجة استعادة موجودة الآن لكل نوع.** تُستعاد إسقاطات الوكيل والمُوصِّل والمُجمِّع كلٌّ على حدة – بما في ذلك عَلَم البالي/غير الصالح – عند استعادة مدخل مصدرها.
- **لم تعد عمليات إعادة البناء تُسقِط الصفوف المؤرشَفة.** أصبحت عملية إعادة بناء المُوصِّل أو المُجمِّع (التي يُطلِقها تعديل أي حقل فيه، وليس إعادة تعيين المصدر وحدها) تطابق مدخلات المصدر ذات الحالة Active أو Archived، بدلًا من Active وحدها. وقبل ذلك، كانت إعادة بناء كيان مُوصِّل أو مُجمِّع تُسقِط نهائيًا كل صف كان مصدره مؤرشفًا.
- **صف المُوصِّل "حيّ" فقط عندما تكون كل مصادره المشارِكة كذلك.** يمكن لصف المُوصِّل أن يجمع عدة مصادر. فإذا كان أي مصدر مشارك مؤرشفًا، تُخفَّض الآن حالة الصف نفسه إلى Archived (مع اعتماد السجل التاريخي لحالة ذلك المصدر)، حتى لو كان المصدر الأساسي لا يزال فعّالًا – وتكون الأولوية حتميًا لأول مشارك مؤرشف عند انطباق الشرط على أكثر من مصدر واحد.

رؤية إجراء التعديل على مدخل التجميع تتبع سير عمل المصدر `Since 1.8`

ليس لمدخل التجميع سير عمل خاص به يمكن فحصه: فكيان المُجمِّع لا يتضمّن `workflows`، والمدخل المُسقَط/المُجمَّع لا يتضمّن بيانات وصفية للخطوة الحالية. أما سير العمل الذي يحكم فعليًا إمكانية التعديل فيُنفَّذ على مدخل **المصدر** – وهو ما يتّسق مع اقتصار تعديل صف التجميع على إعادة التوجيه إلى صفحة تعديل مدخل المصدر.

Since 1.8، تحدّد المنصّة الحالة النهائية لسير عمل ذلك المصدر وتستخدمها لإخفاء إجراء التعديل بمجرد اكتمال سير عمل المصدر – سواء لمدخل واحد مفتوح أو لكل صف في معارض القوائم (قائمة إدارة الكيانات لدى المسؤول وبوّابة العرض العامة الخاصة بمعرض البيانات) – بما يطابق السلوك المطبّق بالفعل في معرض/قائمة مدخل المصدر نفسه.

العمليات الإدارية على الكيانات الوهمية `Since 1.10`

تُوثَّق الميزات الإدارية على مستوى الكيان في 1.10 مرة واحدة في نمذجة البيانات: الكيانات [بناء التطبيقات › إنشاء كيان وتصميمه].

ويقتصر هذا الجدول على توضيح كيفية تعاملها مع الأنواع الوهمية الثلاثة – الوكيل والمُوصِّل والمُجمِّع:

الميزة	على كيان وهمي
النقل إلى وحدة أخرى (مفردًا وجماعيًا)	**مدعوم.** لا يغيّر النقل سوى الوحدة، ويُبقي فحوص الحزمة المثبَّتة والأرشفة، وينشر حدث تغيّر الوحدة المعتاد. وكان نقل أي كيان وهمي قبل 1.10 يفشل بـ `500`.

الميزة	على كيان وهمي
الحذف النهائي	**مدعوم** للأنواع الثلاثة، مع فحص التبعيات نفسه «افصل الروابط أولًا» المطبّق على أي كيان آخر.
الفهارس المخصّصة (تبويب الفهارس)	**غير مدعومة** – فليس للكيان الوهمي مجموعة خاصة به يمكن فهرستها. لا يظهر التبويب، ويُرجع `index-paths` القيمة `[]`، ويُرفض إرسال فهارس مخصّصة بـ `ENTITY_INDEXES_NOT_SUPPORTED`. وللسبب نفسه، لا تُضاف فهارس قراءة القوائم المُرتَّبة (collated) في 1.10 إلى الكيانات الوهمية.
قوالب التصدير [بناء التطبيقات]	`Since 1.6`. يمكن إدارة القوالب على كيان وهمي غير ضمني (أو تكون قيمة `implicitOptions.allowEdit` فيه true)، وتُحفظ عبر تحديث الكيان الوهمي. ويتضمّن تصدير القالب الصفوف التي حدّدها المستخدم.

ويعالج إصلاحان في الإصدار نفسه كيانات وهمية كانت قد توقّفت عن التحديث دون ظهور أخطاء؛ ويُنفَّذ كلاهما مرة واحدة ضمن ترحيل الترقية، ولا يتطلّبان أي إجراء:

- **`group_assignments`.** كان خط المعالجة الخاص بهذا الكيان الوهمي المضمَّن قد يُنتج معرّفات مكرّرة لمستخدم له إسنادان رئيسيان تحت سلف واحد، مما يُجمِّد المجموعة المُجسَّدة (وكل منحة مشتقّة منها) عند آخر حالة سليمة لها. استُبدل خط المعالجة وأُعيد بناء كل كيان وهمي.
- **مصادر المُجمِّع بلا تعيينات حقول.** كانت المُجمِّعات المنشأة تفاعليًا قد تخزّن مصدرًا بلا `fieldMappings`، فيتخطّى كل تحديث هذا التجميع. وأصبحت تلك المصادر تُزوَّد بقائمة تعيينات فارغة، وأُعيد بناء التجميعات.

قابلية تحرير المدخلات الوهمية

بعد إنشائه، يتصرّف كل كيان وهمي كأي كيان آخر عند **القراءة**: يمكنك بناء المعارض عليه، وإضافته مصدرًا لحقل مرجع، وترشيحه والبحث فيه، واستخدامه في الاستعلامات والإحصائيات والتقارير.

أما عند **الكتابة**، فتختلف الأنواع الثلاثة:

- تبقى مدخلات **المُوصِّل** و**المُجمِّع** نماذج قراءة محسوبة – ولتغيير ما تعرضه، غيِّر بيانات مصدرها أو إعداداته، لا المدخل نفسه.
- يمكن إنشاء مدخل **الوكيل** وتعديله وأرشفته واستعادته مباشرةً، تمامًا كمدخل اعتيادي – **عندما يسمح سقف قدرات معرضه الحاكم بذلك** (انظر حوكمة معرض الوكيل). ويكون كيان الوكيل المُنشَأ حديثًا للقراءة فقط افتراضيًا؛ ويجب على المسؤول تفعيل الإنشاء/التحرير/الأرشفة صراحةً على معرضه الحاكم قبل أن يتمكّن المستخدمون النهائيون من الكتابة عبره.

> **ملاحظة**
>
> المدخلات الوهمية إسقاطات محسوبة في جوهرها، لكن لا يُفترَض أن قابليتها للتحرير موحّدة: ففي كيانات **المُجمِّع**، يتحقّق مُصيِّر المدخلات من قابلية تحرير كل حقل عبر الواجهة الخلفية باستخدام `GET .../editable/{fieldId}` قبل تحديد ما إذا كان قابلًا للتحرير، بدلًا من اعتبار المدخل بأكمله للقراءة فقط على نحو مطلق؛ أما في كيان **وكيل**، فتقتصر قابلية التحرير على ما يتيحه حاليًا سقف معرضه الحاكم.

ملاحظات الإصدار

- نموذج مصدر الوكيل/المجمِّع (`VirtualProxyEntitySourceTypeEnum` و`scope` و`isMandatory` و`level`) و`ProxyDataEntryService` مكوّنات **أساسية** و**Long-standing**؛ فهي تسبق مخطط المراحل 1.2، لذا لا تُسنَد إليها شارة مرحلة.

- كذلك يعود نوعا الكيان `JoinerProxy` (51) و`VirtualAggregator` (52)، وكذلك `AggregatorEntityFieldMappingType`، إلى مرحلة **Long-standing** (تسبق 1.2).
- **Since 1.6** – أُدخِل `OptionsMapping` (إعادة تعيين معرّفات خيارات الفئة ضمن تعيينات حقول المجمِّع).
- **Since 1.8** – الحذف النهائي المتسلسل لمُدخلات الوكيل (`DeleteProxyEntryOnDataEntryPermanentlyDeleted`).
- **Since 1.8** – توسيع انتشار حالة دورة حياة الأرشفة/الاستعادة (`state` و`stateHistory` و`isInvalid` و`invalidReason`) ليشمل المُوصِّل والمُجمِّع أيضًا، بعدما اقتصر على الوكيل؛ انظر انتشار حالة الأرشفة/الاستعادة.
- **Since 1.8** – إخفاء إجراء تعديل مُدخل التجميع بمجرد اكتمال سير عمل المصدر، سواء لمُدخل واحد أو لكل صف في معارض القوائم؛ انظر ظهور إجراء تعديل مُدخل التجميع.
- **Since 1.10** – **حوكمة معرض الوكيل والكتابة عبر التوجيه على المدخلات**: معرض `kind: Proxy`، وواجهة محرِّره المقيَّدة، واشتقاق سقف القدرات وإنفاذه، وضوابط الربط/الحلقات، وحقل `proxyCapabilities` الجديد في استجابة الكيان، والكتابة عبر التوجيه لعمليات الإنشاء/التحديث/التصحيح/الأرشفة/الاستعادة/الإنشاء بالدُفعة على مُدخلات الوكيل. انظر حوكمة معرض الوكيل والكتابة عبر التوجيه.
- **Since 1.10** – يعمل النقل (المفرد والجماعي) والحذف النهائي على الأنواع الوهمية الثلاثة؛ والفهارس المخصّصة غير مدعومة؛ انظر العمليات الإدارية.
- **Since 1.10** – مزامنة حقول الوكيل المنسوخة مع مصدرها؛ انظر مزامنة الحقول.
- **Since 1.10** – حالة قفل لمُدخل الوكيل مشتقّة من سير عمل المصدر؛ انظر حالة القفل المشتقّة من سير عمل المصدر.
- **Since 1.10** – **أُلغي** `sourceQuery` في مصدر `VirtualProxy` الأساسي لصالح تحديد النطاق المستند إلى المعرض، ولم يعد يُستخدم إلّا حلًّا احتياطيًا لما قبل الترحيل؛ انظر نموذج بيانات خيارات النوع وتحديد النطاق.

تاريخ تغيّر السلوك

- **إنشاء الكيان الوهمي متزامن.** ذكرت النسخ السابقة من هذا التوثيق أن إنشاء الكيان يُصدر `CreateVirtualDataEntityCommand` على ناقل الخدمة. وقد استُعيض عن هذا الوسيط باستدعاء مباشر ومتزامن: وبذلك لا يتضمّن مسار الإنشاء رحلة ذهاب وإياب عبر الناقل.

واجهة التفويض البرمجية وآلية فرضه

تتوجّه هذه الصفحة إلى مطوّري التكامل والمشغّلين والمهندسين الذين يحتاجون إلى العمل برمجيًا بموجب تفويض، أو إلى فهم آلية فرض التفويض بدقة. أما ما يخص المدير (إنشاء التفويض وتحديد نطاقه وتحويله وإنهاؤه) فيرد في صفحة التفويض [الإدارة]. وتُدرج إضافة البوابة كذلك مع سائر الإضافات في عمليات البوابة، بينما ترد كيفية حلّ المحرّك لأذونات المفوَّض له في المحرّك: الأذونات.

يسبق كيانُ التفويض الأساسي ونقطةُ نهاية التفويضات النشطة مرحلةَ 1.2؛ أما ترشيح الأذونات بحسب النطاق ووراثة أذونات الإدارة فأُضيفا لاحقًا، لكنهما متاحان كذلك منذ زمن طويل (قبل 1.2).

سجل التفويض

التفويض مُدخل بيانات عادي في الكيان **`delegation`** (المعرّف النظامي `DELEGATION`)، ويتضمن الحقل `value` فيه ما يلي:

الحقل	التسمية	النوع	مطلوب	المعنى
`delegator`	المفوض	مرجع (مستخدم)	نعم	المستخدم الذي تُفوَّض مهامه وأذوناته.
`delegated_to`	المفوض له	مرجع (مستخدم)	نعم	المفوَّض له.
`from_date`	من تاريخ	تاريخ ووقت	نعم	موعد بدء سريان التفويض.
`to_date`	إلى تاريخ	تاريخ ووقت	لا (يقبل null)	موعد انتهائه؛ وتعني القيمة `null` أو غيابها أن التفويض مفتوح المدة.
`is_active`	نشط	تصنيف (`DelegationTypeEnum`)	نعم	`ACTIVE` أو `INACTIVE` : مفتاح التشغيل والإيقاف اليدوي.
`delegation_scope`	نطاق التفويض	كائن	نعم	ما يشمله التفويض (انظر أدناه).

نطاق التفويض

الحقل	التسمية	النوع	المعنى
`base`	أساس التفويض	تصنيف (`DelegationScopeEnum`)	الوضع الافتراضي: `ALLOW_ALL` أو `DENY_ALL` .
`exception`	ماعدا	مصفوفة مراجع	كائنات النظام التي *لا* ينطبق عليها الوضع الأساسي. القيمة الافتراضية `[]` .
`allow_administration`	الإدارة	مصفوفة تصنيفات (`DelegationAdministrationEnum`)، اختياري	إذا احتوت على `ALLOWED` ، ورث المفوَّض له أيضًا أذونات الإدارة (النظام) التي يملكها المفوِّض.
`forward_notifications`	حول الإشعارات	منطقي، معطَّل افتراضيًا	يُسلَّم نسخة من إشعارات المفوِّض داخل التطبيق إلى المفوَّض له. `Since 1.10`
`forward_emails`	حول البريد الالكتروني	منطقي، معطَّل افتراضيًا	يُرسل إلى المفوَّض له نسخة من رسائل البريد الإلكتروني الموجّهة إلى المفوِّض. `Since 1.10`

قيم التعدادات: `DelegationTypeEnum` = `ACTIVE` ، `INACTIVE` ؛ و `DelegationScopeEnum` = `ALLOW_ALL` ، `DENY_ALL` ؛ و `DelegationAdministrationEnum` = `ALLOWED` .

يحمل الكيان `USER` علاقة ضمنية باسم **التفويض** («التفويضات التي منحها المستخدم»، `USER_GIVEN_DELEGATIONS_RELATION`) تربطه بالكيان `DELEGATION` عبر الحقل `delegator` . `Since 1.6`

ما يعرضه منتقي الاستثناءات `Since 1.8`

تقتصر الخيارات المعروضة في **ماعدا** على ما يستطيع المدير الذي ينشئ المُدخل الوصول إليه:

- تُعرض للمستدعي الذي يملك منحة على **مستوى النظام** (مستعرض النظام أو محرّره أو مديره) **شجرة التطبيقات المنشورة كاملةً**: الوحدات والمعارض والصفحات والتقارير وإجراءات الخدمات.
- أما أي مستدعٍ آخر فلا يرى إلا المعارض والصفحات والتقارير التي يملك حق الوصول إليها، إضافةً إلى إجراءات الخدمات التي يجوز له تقديمها أو مراقبتها، والعُقد الأصلية اللازمة لإبقاء هذه العناصر متصلة بالشجرة.
- وتُعرض للمستدعي المجهول المعارض والصفحات التي فُعّلت للمجهولين، ولا يُعرض شيء لمن لا يملك أي إذن.

لا يحدّد هذا إلا ما *يعرضه* المنتقي؛ إذ يُطبَّق `delegation_scope.exception` وقت الطلب بوصفه مرشّحًا لأذونات **المفوِّض نفسه** (انظر فرض النطاق)، ولذلك لا يمكن لقائمة خيارات أوسع أن توسّع الوصول الفعلي للمفوَّض له.

متى يكون التفويض نشطًا؟

لا يُعتدّ بالتفويض إلا إذا تحققت الشروط التالية **كلها**:

1. قيمة `is_active` هي `ACTIVE`.
2. `from_date <= now`.
3. قيمة `to_date` هي `null` **أو** `to_date >= now`.
4. حالة المُدخل `state` هي `Active` (غير مؤرشف ولا محذوف).

يُرجع استعلام التفويضات النشطة التفويضاتِ التي يكون فيها المستخدم المعني هو **المفوَّض له** (أي يطابقه `delegated_to`). وإذا تكرر المفوِّض نفسه، أُزيل التكرار من النتائج، فيرى المفوَّض له تفويضًا واحدًا لكل مفوِّض. ويُحلّ مرجع `delegator` في كل تفويض يُرجعه الاستعلام مع بيانات وصفية للعرض.

كيف يفرض أسس التفويض

يُطبَّق التفويض **لكل طلب على حدة** عبر ترويسة الطلب `X-DELEGATION-ID`. فالعميل الذي يريد العمل بموجب تفويض يرسل معرّف مُدخل التفويض في هذه الترويسة؛ ولا يحتفظ الخادم بحالة «وضع تفويض»، بل إن وجود الترويسة في كل طلب هو ما يفعّل التفويض لذلك الطلب. وتتوزع آلية الفرض بين بوابة الـ API (ربط المفوَّض له، والنافذة الزمنية، وتبديل الهوية) وخدمة البيانات (`is_active`، وحالة المُدخل، والنطاق).

في تطبيق الهاتف المحمول، يُحفظ التفويض الذي يختاره المستخدم ويُعاد تحميل التطبيق، ثم يُرفق معترض من جهة العميل (`DelegationInterceptor`) ترويسة التفويض بكل طلب. أما على الويب، فيُنهي تسجيلُ الخروج التفويضَ النشط. `Since 1.6`

الفرض في البوابة

تفرض بوابة الـ API (Kong) التفويض عبر الإضافة المخصصة `user-delegation` (المحزَّمة باسم `kong-plugin-user-delegation-0.1.0-1.rockspec`، والمعتمدة على `lua-resty-redis-connector 0.11.0`). وتعمل بـ**الأولوية 1000**: بعد استبطان الرمز ومصادقة رمز التطبيق (اللذين يحدّدان `x-user-id`)، وقبل `user-space-checker` (999) و`osos-access` (997). وتسير المعالجة على النحو التالي:

1. **شرط التشغيل.** إذا خلا الطلب من ترويسة `X-DELEGATION-ID`، مرّرته الإضافة دون تعديل.

2. **اشتراط مستخدم مُصادَق.** إذا غابت `x-user-id`، أعادت `400 MISSING_USER_ID`.

3. **تحميل تفاصيل التفويض.** تبحث عن التفويض في Redis تحت مفتاح **مقيَّد بالقاطن** هو `osos:delegation:v2:<delegation_id>`، يُركَّب مع `envPrefix` ومعرّف القاطن (ويُرفض الطلب الذي لم يُحدَّد القاطن الخاص به بالخطأ `500 APP_ERROR` قبل أي بحث). فإن لم تجده، تستدعي `GET <data_service_url>/api/v1/delegations/<id>` بترويسات الخدمة الداخلية (هوية خدمة البوابة والمفتاح المشترك، إضافةً إلى القاطن)، ثم تُخبّئ النتيجة مدة `cache_ttl` (الافتراضي **600 s**). حالات الإخفاق: الاستجابة بغير 200 تُمرَّر بوصفها `DELEGATION_API_ERROR`؛ والطلب الذي يُخفق كليًا يُعيد `500 DELEGATION_API_REQUEST_FAILED`؛ و`success: false` يُعيد `500 DELEGATION_API_LOGIC_ERROR`؛ والجسم الذي يتعذّر فك ترميزه يُعيد `500 DELEGATION_API_DECODE_ERROR`؛ وإخفاق الاتصال بـ Redis يُعيد `500 REDIS_CONNECTION_ERROR`. وتُسجَّل تفاصيل البنية التحتية (المضيفات الداخلية وأخطاء Redis) في البوابة، ولا تُعاد إلى المستدعي.

4. **ربط التفويض بالمستدعي.** `Since 1.6` تقارن الإضافة `delegated_to` بـ `x-user-id`. فإن كان التفويض ممنوحًا لشخص آخر، رُفض بالخطأ `403 DELEGATION_FORBIDDEN` ("Delegation does not belong to the current user")؛ ويُعامَل التفويض الخالي من `delegated_to` على أنه ناقص (`500 DELEGATION_DATA_ERROR`). انظر ربط المفوَّض له.

5. **التحقق من النافذة الزمنية.** تُحلَّل `from_date` و`to_date` من سلاسل تاريخ ووقت بصيغة ISO 8601 (مع أجزاء الثانية أو دونها، ومنتهيةً بـ `Z` أو بإزاحة `±HH:MM` تُطبَّق) إلى طوابع زمنية بتوقيت UTC، ويُرفض التاريخ التقويمي غير الممكن بدل ترحيله إلى تاريخ لاحق. فإذا كان الوقت الحالي سابقًا لـ `from_date`، أو لاحقًا لـ `to_date` عند وجودها، أعادت الإضافة **403** `DELEGATION_EXPIRED`. وغياب `to_date` أو كونها null يعني عدم وجود حد أعلى. أما التواريخ التي يتعذّر تحليلها فتُعيد `500 DELEGATION_DATE_PARSE_ERROR`.

6. **تبديل الهوية.** عند النجاح، تعيد الإضافة كتابة الترويسات المتجهة إلى الخدمات اللاحقة، فتراها هذه الخدمات صادرة عن المفوِّض:

الترويسة	القيمة
`x-user-id`	معرّف **المفوِّض**
`x-delegatee-id`	المستدعي الأصلي (المفوَّض له)
`x-is-delegation`	`"true"`

ربط المفوَّض له `Since 1.6`

يرسل العميل ترويسة `X-DELEGATION-ID`، وتجلب البوابة التفويض باستدعاء خدمة داخلي. أما الآن فتشترط الإضافة أن يكون المستدعي هو `delegated_to` في التفويض، وإلا استجابت بـ `403 DELEGATION_FORBIDDEN`.

يتطلب الفحص شقّين معًا: أن تُعيد خدمة البيانات `delegated_to` في `GET /v1/delegations/{id}` (وقد أُتيح ذلك أيضًا بتحديث فروع الإصدارات 1.4 و1.6 و1.8)، وأن تخبّئ البوابة التفاصيل تحت مفتاح `v2` الجديد، فلا تُقرأ أبدًا التفاصيل المخبوءة قبل التغيير، وهي خالية من `delegated_to`، ولا تُرفض الطلبات بسببها. حدِّث خدمة البيانات قبل البوابة.

لا يكشف `DELEGATION_EXPIRED` التفويض. لا يحمل جسم الاستجابة `403` إلا `details.error = "Delegation period is not current."`، دون المفوِّض أو الفترة أو ساعة البوابة. ويُسجَّل الرفض بمستوى info مع معرّف التفويض والمستدعي ومعرّف التتبع، ومن هذا السجل يتعرّف المشغّل على المفوِّض والفترة. ولا يقرأ عميل الويب إلا الرمز، ثم يُلغي التفويض النشط.

فحوص الوصول في ظل التفويض

تقرأ الإضافة اللاحقة `osos-access` (الأولوية 997) الترويسة `X-DELEGATION-ID`، فإن وجدتها جلبت ملف مستخدم مقيّدًا بالتفويض من خدمة البيانات، وخبّأته في Redis تحت `shared:user-profile-<userId>_<delegationId>`. ومع وجود معرّف تفويض، يُحدَّد الوصول إلى المعرض مباشرةً بحسب ما إذا كان المفوِّض يملك إذن الوصول إلى ذلك المعرض، بدلًا من الفحص المعتاد القائم على الملكية. وتتجاوز مجموعة ثابتة من معرّفات المعارض مسار التفويض وتخضع دائمًا للفحص المعتاد:

```
SERVICE_CATALOG, ONGOING_INBOX, INBOX_REQUESTS,
ONGOING_MY_REQUESTS, COMPLETED_MY_REQUESTS,
ONGOING_OBSERVER, COMPLETED_OBSERVER
```

فرض النطاق في خدمة البيانات

تحلّ خدمة البيانات الأذونات الفعلية للمفوَّض له في `getAllDelegateePermissions(userId, delegationId)`، وهنا يُطبَّق `delegation_scope`:

1. تُحمِّل التفويض النشط (وفق قواعد النشاط نفسها المذكورة أعلاه). فإن لم تجده، حصل المفوَّض له على أذونات فارغة.
2. تقرأ `delegation_scope.base` وتجمع `delegation_scope.exception` في مجموعة للاستثناءات.
3. تنطلق من أذونات المفوِّض الكاملة، وتُرفق بالأذونات المحلولة كتلة `delegation` (`{ delegationId, scope, exceptions }`).
4. **الإدارة**: تُورَّث أذونات النظام كلها إذا احتوى `allow_administration` على `ALLOWED`. وإلا أُفرغت `system`، باستثناء حالة واحدة `Since 1.10`: تفويض `ALLOW_ALL` بلا استثناءات، إذ يحتفظ بمنح النظام الأدنى من Admin ولا يُسقط إلا System Admin. أما أذونات الوحدات والكيانات فتبقى دائمًا.
5. **المعارض والصفحات والتقارير** تُرشَّح وفق الاستثناءات:
   - `DENY_ALL` بلا استثناءات: لا معارض ولا صفحات ولا تقارير.
   - `ALLOW_ALL` بلا استثناءات: جميع معارض المفوِّض وصفحاته وتقاريره.
   - مع وجود استثناءات، يُطابَق كل عنصر مع أنماط الاستثناء (بمعرّف العنصر أو بوحدته الأم، إضافةً إلى الأنماط العامة `so:all` و`so:app` و`so:app:module`). ففي وضع `ALLOW_ALL` **تستبعد** المطابقةُ العنصر، وفي وضع `DENY_ALL` **تُدرجه**.

ترشيح القوائم وقت الطلب

في نقاط نهاية قوائم الطلبات (صندوق الوارد، والمراقبة، ودليل الخدمات)، يضيّق مرشّح طلبات التفويض نطاق الاستعلام حين يكون التفويض نشطًا؛ إذ يطابق عناوين معارض محددة ويُدرج مرشّح `where` على حقل خاص بكل معرض:

المعرض	نوع الإجراء	حقل الترشيح
`SERVICE_CATALOG`	Submit	`value.relatedEntityId`
Ongoing/Completed My Requests	MyRequest	`value.reference.dataEntity`
Ongoing/Completed Inbox	Inbox	`value.reference.dataEntity`
Ongoing/Completed Observer	Observe	`value.reference.dataEntity`

يتوقف المعامل المستخدم على النطاق: يستخدم `ALLOW_ALL` المعامل `nin` (استبعاد خدمات الاستثناء)، ويستخدم `DENY_ALL` المعامل `inq` (الاقتصار على خدمات الاستثناء). وتختصر الاستثناءات العامة `so:all` و`so:service` هذه

المعالجة كلها: فهي تعني في ظل `ALLOW_ALL` «بلا مرشّح»، وفي ظل `DENY_ALL` «استبعاد الكل». ويسري المرشّح على `/entries` و`/count`.

التحويل إلى المفوَّض له `Since 1.10`

عند تفعيل `forward_notifications` أو `forward_emails` أو كليهما، يتلقى المفوَّض له نسخة مما يُوجَّه إلى المفوِّض ما دام التفويض نشطًا، وفي حدود العمل الذي يشمله التفويض فقط. ومصادر التحويل هي إجراءا التفاعل **إرسال إشعار** و**إرسال البريد الإلكتروني**، ويُضاف إليهما في البريد الإلكتروني إرسالُ التقارير المجدولة. وتُنسَب كل رسالة إلى الخدمة أو الوحدة أو التقرير أو المعرض الذي تتعلق به، ويُطبَّق النطاق وفق هذه النسبة (Allow All: تُرسل ما لم تُستثنَ؛ Deny All: لا تُرسل إلا إذا منح استثناءٌ تلك الخدمة أو الوحدة أو التقرير أو أحد معارض الكيان؛ والرسالة المرتبطة بمعرض بعينه تحتاج إلى منح ذلك المعرض). ويُتخطّى التحويل في الحالات التالية: غياب المفوَّض له أو تعطيله، وكونه المفوِّض نفسه، واحتساب زوج المفوِّض والمفوَّض له نفسه مرتين، وكون المفوَّض له مستلمًا مباشرًا للإشعار أصلًا. ولا تُحوَّل النسخة مرة أخرى أبدًا. وإذا تعذّر تحديد المفوَّض لهم، يُتخطّى التحويل ويُسجَّل ذلك، أما التسليم الأصلي فيستمر في جميع الأحوال. تفاصيل جانب الإشعارات: تهيئة الإشعارات [الإدارة › الإشعارات وقوالب البريد الإلكتروني].

نقاط النهاية

تتطلب نقطتا النهاية العامتان كلتاهما مستخدمًا مُصادَقًا (استبطان OAuth2)، وتقعان في متحكّم التفويضات في خدمة البيانات.

`GET /v1/delegations/{id}`

تُعيد تفويضًا واحدًا مُسقَطًا على `MyDelegationsDto`، وتستخدمها الإضافة `user-delegation` في البوابة:

الحقل	النوع	ملاحظات
`id`	string	معرّف مُدخل التفويض.
`delegator`	مرجع	المستخدم المفوِّض.
`from_date`	تاريخ ووقت	بداية النافذة.
`to_date`	تاريخ ووقت | null	نهاية النافذة (تقبل null).
`delegated_to`	مرجع	المفوَّض له. `Since 1.4`، وهو لازم لربط المفوَّض له.

لا يتضمن هذا الإسقاط `is_active` ولا `delegation_scope`.

`GET /v1/delegations/my`

تُعيد تفويضات المستخدم الحالي **النشطة** (التي يكون فيها هو المفوَّض له) في مصفوفة من `MyDelegationsDto`. ولا تُعاد إلا التفويضات التي تجتاز فحص النشاط الكامل، بعد إزالة التكرار لكل مفوِّض.

تهيئة إضافة البوابة

تُهيَّأ الإضافة `user-delegation` على مستوى البوابة كلها في `config/kong.yml`:

الإعداد	النوع	مطلوب	الافتراضي	الغرض
`data_service_url`	string	نعم	–	عنوان URL الأساسي لخدمة البيانات التي تُجلب منها تفاصيل التفويض.

الإعداد	النوع	مطلوب	الافتراضي	الغرض
`shared_key`	string	نعم	–	المفتاح المشترك الداخلي المُرسَل في `X-SHARED-KEY`.
`cache_ttl`	number	لا	`600`	مدة بقاء تفاصيل التفويض المخبوءة في Redis (بالثواني).
`redis.url`	string	لا	`redis://localhost:6379`	عنوان URL للاتصال بـ Redis.
`redis.ssl`	boolean	لا	`false`	تفعيل TLS في الاتصال بـ Redis.
`redis.ssl_verify`	boolean	لا	`false`	التحقق من شهادة خادم Redis.
`redis.useSentinel`	boolean	لا	`false`	استخدام Redis Sentinel بدل خادم مستقل.
`redis.sentinel.sentinels[]`	مصفوفة من `{host, port}`	–	–	نقاط نهاية Sentinel.
`redis.sentinel.username` / `masterName` / `password`	string	–	–	مصادقة Sentinel واسم الخادم الرئيسي.
`envPrefix`	string	لا	–	بادئة تُضاف إلى مفاتيح الخبء في Redis (لفصل أسماء كل بيئة).

سجل الإجراءات والتدقيق

يُنفَّذ كل إجراء بموجب تفويض **باسم المفوِّض**، في حين يسجّل أسس أن الفاعل الحقيقي هو **المفوَّض له**، مما يتيح لمسار التدقيق التمييز بينهما دائمًا. وتُرفق معرّفات التفويض بكل طلب يخص سير العمل أو التعليقات، ويُمرَّر معرّف المفوَّض له (`x-delegatee-id`، مع `x-is-delegation: true`) إلى الخدمات لأغراض التدقيق. انظر المراقبة: السجلات والتدقيق [الإدارة]، وعلى مستوى المُدخل، الشريط الزمني [بناء التطبيقات › الشريط الزمني والسجل التاريخي].

يُنسب السجل التاريخي للنقر على الأزرار إلى المفوَّض له `Since 1.8`. قبل هذا الإصلاح، كان النقر على زر نظام أثناء العمل بموجب تفويض يُنتج سجل `ButtonClickedHistory` بفاعل فارغ؛ إذ كان `delegatee` يُسقط لأن المتحكّم الذي يعالج النقرة لم يمرّره قط إلى استدعاء إنشاء السجل، مع أن البوابة كانت قد حلّته ومرّرته. ولم تتأثر قط تغييرات القيم ولا إعادة إسناد المالك التي جرت بموجب تفويض.

ملاحظات تشغيلية وحالات حدّية

- **الخبء.** تخبّئ البوابة تفاصيل التفويض مدة `cache_ttl` (600 s افتراضيًا)، ولا تعيد جلبها إلا إذا لم تجدها في الخبء. وحين **يُحرَّر مُدخل التفويض أو يُؤرشف**، تحذف خدمة البيانات النسخة المخبوءة لدى البوابة، فيسري التغيير مع الطلب التالي. و`Since 1.8` تحذف كلا جيلَي المفتاح، أي المفتاح الحالي `osos:delegation:v2:` والمفتاح الأقدم `osos:delegation:`، فتشمل بذلك الخلية التي تشغّل بوابات من الجيلين أثناء الترقية المتدرجة؛ وقبل ذلك كان التفويض المحرَّر أو المؤرشف قد يبقى صالحًا للاستخدام لدى بوابة أحدث مدةً تصل إلى `cache_ttl`.
- **الأذونات المفوَّضة تتبع تغييرات الأذونات** `Since 1.4`. تُمسح الأذونات المفوَّضة المخبوءة للمفوَّض له (التي تُحفظ حتى 300 s) عند تغيّر حقوق الوصول والمعارض والتقارير التي تتضمنها، فلا يبقى الإذن المسحوب من المفوِّض ساريًا لدى المفوَّض له.
- **الفرض الموزَّع.** تتحقق البوابة من ربط المفوَّض له ومن النافذة الزمنية وتنفّذ تبديل الهوية؛ أما `is_active` و`state` و`delegation_scope` فتفرضها خدمة البيانات.
- **التفويضات المفتوحة المدة.** التفويض الذي تكون قيمة `to_date` فيه null لا ينتهي من تلقاء نفسه؛ فأنهِ التفويض بتعيين `is_active` على `INACTIVE`، أو بأرشفة المُدخل، أو بتعيين `to_date`.

- **الهوية لدى الخدمات اللاحقة.** في ظل تفويض نشط، ترى الخدمات اللاحقة `x-user-id` = المفوِّض، بينما تحمل `x-delegatee-id` و `x-is-delegation: true` المستدعيَ الحقيقي والعَلَم.
- **المستخدمون الضيوف (المجهولون)** ليست لهم تفويضات.
- **رموز الـ API** آلية منفصلة مخصصة للأنظمة الخارجية لا للأشخاص. انظر رموز الـ API.

مرجع واجهة برمجة السجل التاريخي والشريط الزمني

هذه الصفحة موجّهة إلى **المُكامِلين والمطوّرين** الذين يقرؤون السجل التاريخي لمُدخلٍ أو خلاصة شريط زمني عبر الواجهة البرمجية، أو يثبّتون المنشورات أو ينشرونها من نظام آخر، أو يحتاجون إلى فهم آلية تسجيل السجل التاريخي وترشيحه بدقّة.

هل تُعِدّ ذلك من المصمّم؟

إن كنت من بُناة التطبيقات وتختار ما يعرضه المعرض من السجل التاريخي للمُدخل، أو تضيف شريطًا زمنيًا إلى كيان، فابدأ بصفحة الشريط الزمني والسجل التاريخي [بناء التطبيقات]؛ إذ تشرح الميزات نفسها بالتسميات التي تظهر في المصمّمات.

تتيح أسس مفهومين متمايزين ومترابطين، يجيب كلٌّ منهما عن سؤال «ماذا حدث لهذا المُدخل، ومتى؟»:

1. **السجل التاريخي للمُدخل** – خلاصة التدقيق والنشاط الخاصة بمُدخل بيانات واحد (تغييرات الحقول، وإجراءات سير العمل، وإعادة إسناد المالكين، وتجاوزات اتفاقية مستوى الخدمة، وتشغيلات التفاعلات، ونقرات الأزرار، والإنشاء والأرشفة). تقدّمه **data-service**.
2. **الشريط الزمني** – *خلاصة منشورات* منتقاة، عامة أو داخلية، تُولَّد من مُدخلات البيانات عبر ربطٍ خاص بكل كيان يُسمّى «مساهمة الشريط الزمني». ويمكن تثبيت المنشورات ونشرها وحجبها (يدويًا، أو وفق جدول زمني، أو عبر سير العمل)، وتتضمّن طبقة اجتماعية (تفاعلات وتعليقات). تقدّمه **timeline-service**.

توثّق هذه الصفحة كليهما من منظور باني التطبيقات والمطوّر: نقاط نهاية HTTP، وبنية البيانات التي تتلقّاها، وكيفية تدوين القيود، والضوابط التي تحكم الظهور.

> **ملاحظة حول النطاق:** تركّز هذه الصفحة على واجهات القراءة، ونموذج القيد، ومسار التسجيل. أما محرّك سير العمل الذي *يُنتج* `_metaWorkflowHistory`، ومحرّك التفاعلات الذي *يُنتج* مستندات `TriggerHistory`، فيقعان خارج هاتين الخدمتين، ويقتصر الوصف هنا على إسهامهما في خلاصة السجل التاريخي.

الجزء الأول – السجل التاريخي للمُدخل (data-service)

المفهوم

تتراكم لكل مُدخل بيانات قيودٌ تكوّن خلاصة سجلّه. ولا تُستمدّ الخلاصة التي تعيدها نقطة نهاية السجل التاريخي من مجموعة بيانات واحدة، بل **تُجمَّع وقت القراءة** من عدة مصادر، يُحوَّل كلٌّ منها إلى صيغة `DataEntryHistory` موحّدة:

المصدر	الأصل	أنواع السجل التاريخي الناتجة
قيود تغيير الحقول	المجموعة `data_entry_history` (تُحفظ عند الإنشاء والتحديث)	`Update`
قيود إعادة إسناد المالك	المجموعة `data_entry_history`	`ReassignOwner`
قيود نقر الأزرار	المجموعة `data_entry_history`	`ButtonClickedHistory`
إجراءات خطوات سير العمل	`value._metaWorkflowHistory` في المُدخل (محسوب)	`WorkflowAction`
الإنشاء / الأرشفة	`stateHistory` في المُدخل (محسوب)	`Create`، `Archive`
أحداث تجاوز اتفاقية مستوى الخدمة	خدمة SLA	`SlaOverdue`

المصدر	الأصل	أنواع السجل التاريخي الناتجة
تشغيلات التفاعلات	المجموعة `trigger_history` (محسوب)	`TriggerHistory`

تُحفظ قيود السجل التاريخي في مجموعة MongoDB مخصّصة عبر `DataEntryHistoryRepository`؛ أما شرائح سير العمل والحالة واتفاقية مستوى الخدمة فتُشتق آنيًا من مستند المُدخل ومن خدمات أخرى، ثم يدمجها المتحكّم.

قيد `DataEntryHistory`

المصدر: `models/data-entry/data-entry-history.model.ts`.

الحقل	النوع	ملاحظات
`id`	string	مولَّد
`specificationVersion`	number	حاليًا `1`
`relatedTo`	`ReferenceProperty`	`{dataEntity, dataEntry}` الذي ينتمي إليه القيد
`historyType`	`HistoryTypeEnum` (number)	انظر التعداد أدناه
`actionInfo`	`ActionInfo`	من نفّذ الإجراء ومتى – `{doneAt, doneBy, delegatee?}`
`additionalData`	متعدّد الأشكال	يتوقّف شكله على `historyType` (انظر أدناه)
`isValueResolved`	boolean	يصبح true عند حلّ المراجع وتخزين تسمياتها في القيد نفسه
`isInvalid`	boolean | null	كان المُدخل في حالة غير صالحة حينها
`invalidReason`	null | `DataEntryInvalidReasonEnum`	سبب عدم الصلاحية

`ActionInfo` (`action-info.model.ts`): `doneAt` (سلسلة ISO، إلزامي)، و`doneBy` (`ReferenceProperty`، إلزامي)، و`delegatee` (مرجع يقبل null – يوجد حين يُنفَّذ الإجراء نيابةً عن مستخدم آخر).

`HistoryTypeEnum`

المصدر: `packages/osos-shared/src/models/enums.ts`. **هذه قيم تعداد عددية** – فالواجهة البرمجية تعيد الرقم لا الاسم:

القيمة	الاسم	المعنى
0	`Update`	تغيّرت قيمة حقل أو أكثر
1	`WorkflowAction`	نُفِّذت خطوة سير عمل
2	`StateChange`	(معرَّف؛ لا يُصدره مُجمِّع السجل التاريخي)
3	`ReassignOwner`	تغيّر مالك خطوة سير عمل
4	`ButtonClickedHistory`	نُقر زر نظام
5	`TriggerHistory`	شُغِّل تفاعل أو أتمتة
6	`Create`	أصبح المُدخل Active
7	`Archive`	أُرشف المُدخل
8	`SlaOverdue`	جرى تجاوز موعد استحقاق في اتفاقية مستوى الخدمة
9	`PermanentDelete`	(معرَّف)
10	`Restore`	(معرَّف)

> لا تُصدر شريحة `state` حاليًا إلا `Create` و`Archive` (انظر `getDataEntryStateHistory` – فهي تحوّل `newState === Active` إلى `Create`، وما عداه إلى `Archive`). أما `PermanentDelete` / `Restore` فمعرَّفان في التعداد، لكن المُجمِّع الحالي لا يُنتجهما.

أشكال `additionalData` بحسب النوع

- **`Update`** ← `UpdateAdditionalData[]` . يمثّل كل عنصر عملية فرق JSON:

```
{
  "op": "replace",          // just-diff Operation: add | replace | remove
  "path": ["field", "sub"], // path segments into the entry value
  "value": { "old": <any>, "new": <any> }
}
```

وعند الكتابة الأولى لا يحمل الحقل عادةً إلا `new` (بلا `old`).

- **`ReassignOwner`** ← `ReassignOwnerChanges` : `{workflowId, stepId, oldOwner, newOwner, originalOwner?, note?, permanentChange?}`.
- **`ButtonClickedHistory`** ← `{buttonId, buttonName}` .
- **`TriggerHistory`** ← `TriggerHistoryDto` (انظر السجل التاريخي للتفاعلات).
- **`SlaOverdue`** ← `{stepId, dueDate, receivedDate, status, stepName}` .
- **`WorkflowAction`** ← محتوى `StepHistory` المخزَّن، إضافةً إلى `stepName` وكائن `workflowContext` بالصيغة `{workflowId, instanceId, parentInstanceId, isSubWorkflow}` .

تغييرات الحقول متعددة اللغات في خلاصة السجل التاريخي `Since 1.8`

حين يكون الحقل متعدد اللغات، يظهر التعديل الذي يمسّ إحدى لغاته فقط الآن في السجل التاريخي للمُدخل بسطر مستقل – `label: old → new` – لكل لغة تغيّرت فعلًا.

كان طرفا الفرق يُحلّان سابقًا إلى لغة الواجهة الحالية قبل عرضهما. فإذا عدّلت القيمة العربية وحدها ثم قرأت السجل التاريخي والواجهة بالإنجليزية، ظهر في الطرفين النص الإنجليزي نفسه دون تغيير، فبدا التغيير فارغًا رغم تسجيل التعديل. أما القيد المخزَّن فكان يحمل دائمًا القيم الكاملة قبل التعديل وبعده لكل لغة (انظر شكل `Update` أعلاه)، وكان الخلل مقتصرًا على العرض.

ما تعرضه بطاقة السجل التاريخي الآن:

- **اللغات التي تغيّرت فعلًا وحدها** هي التي تُدرج؛ إذ تُتخطّى اللغة التي تتطابق قيمتها على الطرفين، وتُعامل القيمة الفارغة معاملة القيمة الغائبة.
- **تتبع اللغات ترتيب اللغات المهيّأ في المساحة**، وتُلحق في النهاية أي لغة خارج تلك القائمة.
- **تُعرض تسمية اللغة بلغة الواجهة الحالية، أما القيم فلا** – إذ تُعرض كل قيمة بلغتها دون ترجمة؛ لذا يُقرأ التعديل نفسه بالصورة ذاتها أيًّا كانت اللغة التي تعرض بها السجل التاريخي.
- **اللغة المضافة في أثناء التعديل تُعرض إضافةً صريحة** – القيمة الجديدة وحدها، بلا سهم ولا طرف «قديم» – واللغة التي أُفرغت تُعرض حذفًا. ويتحدّد ذلك بحسب حالة كل لغة قبل التعديل وبعده، لا بحسب نوع تغيير الحقل ككل، فتُعرض إضافة لغة في أثناء تعديل أخرى عرضًا صحيحًا في السطرين كليهما.

كيف يُدوَّن السجل التاريخي

يُكتب السجل التاريخي **تفاعليًا، استجابةً لأحداث ناقل الرسائل** – فالمتحكّمات لا تكتبه ضمن معالجة الطلب نفسه. وتشترك المعالجات الموجودة ضمن `handlers/events/data-entries/history/` في أحداث النطاق:

الحدث	المعالج	الإجراء
`DataEntryCreatedEvent`	`AddHistoryOnDataEntryCreatedHandler`	`createHistoryForValueChanges(...)` باستخدام `creationInfo.doneAt`
`DataEntryUpdatedEvent`	`AddHistoryOnDataEntryUpdatedHandler`	`createHistoryForValueChanges(...)` باستخدام `updateInfo.doneAt`
`WorkflowStepOwnerChangedEvent`	`AddHistoryOnWorkflowStepOwnerChangedEventHandler`	`createHistoryForOwnerChanges(...)`
`DataEntryDeletedEvent`	`DeleteHistoryOnDataEntryDeletedHandler`	`deleteHistoryForDataEntry(entryId)`
(المُدخلات المحذوفة نهائيًا)	`delete-history-on-data-entries-permanently-deleted.handler.ts`	`deleteHistoryForDataEntry` بالجملة

أما نقرات الأزرار فتُكتب من `system-button.controller.ts` عبر `createHistoryForButtonClicked(...)` (باستخدام `Utils.dateTime()` قيمةً لـ `doneAt`).

> **تصحيح – Since 1.8.** أصبح نقر الزر في أثناء سريان تفويض [الإدارة › التفويض] يُنسب على نحو صحيح إلى المفوَّض إليه الذي نقره فعلًا، لا إلى المفوِّض. فقد كان المتحكّم يحلّ الفاعل الحقيقي في حدث الناقل الذي ينشره، لكنه لم يكن يمرّره عند كتابة السجل التاريخي، فكان كل قيد `ButtonClickedHistory` يُسجَّل بقيمة فارغة في `actionInfo.delegatee` – فلا يُظهر السجل التاريخي إلا المستخدم المفوِّض، ولا يتيح معرفة من نقر الزر فعلًا. أما الآن فيُسجَّل `delegatee` في السجل التاريخي لنقر الزر بالطريقة نفسها المتّبعة في سجلَّي تغيير القيم وإعادة إسناد المالك.

أبرز سلوكيات التدوين (`data-entry-history.service.ts`):

- **تُتخطّى قيود Update / Create حين يكون الفرق فارغًا** – إذ يعود كلٌّ من المعالج و `createHistoryForValueChanges` مبكرًا إن كان `valueDiff` خاليًا، فلا يُنتج الحفظ الذي لا يغيّر شيئًا أي صف في السجل التاريخي.
- **يُمرَّر `delegatee`** من `stickyAttributes.delegateeId` في الرسالة، فيوثّق السجل التاريخي حالة «س يتصرّف نيابةً عن ص».
- **يُنسخ `isInvalid` / `invalidReason`** من المُدخل إلى قيد Update، فتعكس الخلاصة ما إذا كان المُدخل صالحًا عند تلك المراجعة.
- **تُحلّ المراجع وقت الكتابة.** إذ تستدعي `createHistoryForValueChanges` و `...ForOwnerChanges` و `...ForButtonClicked` جميعًا `resolveHistoryValues(...)` وتحفظ القيود مع `isValueResolved: true`، فتُخزَّن تسميات المراجع (أسماء المستخدمين، وعناوين عناصر المجموعات) داخل القيد المخزَّن نفسه، ولا تحتاج إلى حلّ جديد عند القراءة.
- **الحماية من التكرار.** تُسجَّل أخطاء تكرار المفتاح في MongoDB (`code === 11000`) التي تواجهها عمليات الإدراج، ويُكتفى بمعالجتها دون إعادة رميها، فلا تُعطِّل رسالةٌ أعاد الناقل تسليمها المعالجَ.

تمرّ `resolveHistoryValues` على `actionInfo.doneBy` / `delegatee` وعلى قيم `additionalData`، فتجمع أهداف `ReferenceProperty` ومعرّفات عناصر المجموعات من نوع UUID، ثم تحمّلها دفعةً واحدة (`DataEntityIndexRepository` للمراجع، و `DataCollectionsRepository` مع `internalCollections` الخاصة بالكيان لعناصر المجموعات)، وتعيد كتابة كلٍّ منها في صورة `ReferencePropertyWithResolved` يحمل تسمية `_resolved`. وتُترك القيم المحلولة سلفًا كما هي.

> **التدوين غير متزامن.** تتولّى معالجاتُ الناقل التي تستجيب لأحداث `DataEntry*Event` إنتاج السجل التاريخي للمُدخل، ولا يُنتَج السجل التاريخي ضمن الطلب نفسه على نحو متزامن – لذا توقّع اتساقًا نهائيًا بين استجابة الكتابة وقراءة السجل التاريخي.

قراءة السجل التاريخي للمُدخل (v1)

```
GET /v1/entities/{entity}/entries/{id}/history
```

تعيد نقطة النهاية هذه الخلاصةَ المجمَّعة **كاملة** لمُدخل واحد (بلا تقسيم إلى صفحات). ويدمج المتحكّم (`findEntryHistoryById`) بالترتيب:

1. السجل التاريخي المحفوظ (`getEntryHistory(id)`).
2. السجل التاريخي لسير العمل (`getEntryWorkflowHistory`).
3. السجل التاريخي للتفاعلات – **فقط** حين يكون المستدعي عارض نظام بلا معرض، أو حين يكون في المعرض النشط `entriesConfiguration.showTriggerHistory.enabled` .
4. السجل التاريخي للحالة (الإنشاء والأرشفة) – فقط حين يُسمح برؤية السجل التاريخي للحالة.
5. السجل التاريخي لتجاوز اتفاقية مستوى الخدمة.

ثم **يفرز النتائج تصاعديًّا بحسب** `actionInfo.doneAt` ، ويحلّ أي قيود غير محلولة، ويُخفي الحقول الحسّاسة قبل إعادة النتائج.

قراءة السجل التاريخي للمُدخل (v2، مقسَّمًا إلى صفحات)

```
GET /v2/entities/{entity}/entries/{id}/history?lastTimeStamp=<ISO>
```

Long-standing (`findEntryHistoryByIdV2`). تضيف نقطة النهاية v2 تقسيمًا إلى صفحات بالمفتاح (keyset)، وهي التي ينبغي الاعتماد عليها للخلاصات الكبيرة:

- **حجم الصفحة ثابت عند** `limit = 12` (ثابت في الخادم، ولا يستطيع العميل تغييره).
- `lastTimeStamp` هو المؤشّر: مرِّر `actionInfo.doneAt` لآخر عنصر تلقّيته لتحصل على الصفحة التالية. وتطبّق كل شريحة شرط الترشيح `doneAt < lastTimeStamp` .
- تُفرز النتائج **تنازليًّا** بحسب `doneAt` (الأحدث أولًا).
- **معالجة تساوي الطوابع الزمنية:** بعد الاقتطاع إلى `limit` ، *تمدّد* الحلقة الحدّ ما دام القيد التالي له مباشرةً يحمل قيمة `doneAt` نفسها التي يحملها آخر قيد مُحتفَظ به – فلا تتوزّع القيود ذات الطوابع الزمنية المتطابقة على صفحتين أبدًا.
- **وضع المعاينة:** حين يكون الطلب في وضع المعاينة، تعيد نقطة النهاية سجلًا مصطنعًا (معرّفات كيان وسير عمل حقيقية، ومحتوى مصطنع) عبر مولّد البيانات المصطنعة.

يطبّق الإصداران v1 وv2 الترشيح نفسه بحسب المعرض، وحلّ المراجع نفسه، وإخفاء الحقول الحسّاسة نفسه، كما يرد أدناه. ويُفضَّل استخدام v2 (مؤشّر `lastTimeStamp` ، وحجم صفحة ثابت 12، و `doneAt DESC`) لسجلات المُدخلات الكبيرة.

الظهور والترشيح بحسب المعرض

تقابل مفاتيح تبديل السجل التاريخي في مصمّم المعرض المفاتيح البرمجية الآتية: **عرض تاريخ المدخل** = `entriesConfiguration.history.enabled` ، و**عرض التعديلات** = `showChanges` ، و**عرض من قام بالتعديل** = `showChanger` ، و**استعراض التفاعلات** = `showTriggerHistory.enabled` ، و**استعراض تفاصيل التفاعلات** = `showTriggerHistory.showTriggerDetails` ، و**عرض تاريخ سير العمل** = `showWorkflowHistory.enabled` ، و**إظهار اسم مالك الخطوة** = `showStepOwners` (**الكل** = `AllOwners` ، و**المملوك للمستخدم فقط** = مقصور على المالك).

حين يتضمّن الطلب **معرض** بيانات (سياق المستخدم النهائي)، يُرشَّح السجل التاريخي عبر `clearEntryHistoryAgainstView(...)` والمساعِدات المرتبطة به:

- **السجل التاريخي معطّل كليًا.** إذا غاب `view.entriesConfiguration.history` أو كان `history.enabled === false`، تعيد نقطة النهاية `[]`.
- **الترشيح على مستوى الحقل.** تُقلَّص قيود `Update` لتقتصر على الحقول التي يعرضها المعرض (`entriesConfiguration.fields`) *وليست* مخفية. ويُقيَّم إخفاء الحقول لكل قيد على حدة، مع مراعاة `hidden.expression` الشرطي (يُقيَّم عبر `ExpressionService`). ويُسمح دائمًا بـ `_metaOrder.value`. ويُسقط قيد Update إذا استُبعدت كل فروقه بالترشيح.
- **`showChanges` / `showChanger`.** إذا كان `history.enabled` صحيحًا و`showChanges === false`، يُحتفَظ بالحدث في قيود Update وتُحذف `additionalData` (فترى *أن* شيئًا تغيّر، لا *ما* الذي تغيّر). وإذا كان `showChanger === false`، يُقلَّص `actionInfo` إلى `doneAt` وحده (فيُخفى الفاعل).
- **التحكّم في السجل التاريخي لسير العمل.** يتحكّم `showWorkflowHistory.enabled` في ظهور قيود `ReassignOwner` / `WorkflowAction` أصلًا. ويفرض `showStepOwners` قيدًا إضافيًا على ظهور المالك:
  - `AllOwners` ← خطوات الجميع ظاهرة.
  - وإلا فلا يُعرض القيد أو يُنسب إلا إذا كان المستخدم الحالي (أو إحدى مجموعاته) ضمن `executedBy` / `owner` / `originalOwner` للخطوة (في `WorkflowAction`) أو `newOwner` / `oldOwner` (في `ReassignOwner`). وحين لا يُسمح بذلك، تُزال `executedBy` و`owner` و`originalOwner` و`actionInfo.doneBy` من القيد بدلًا من حذف القيد.
- **التحكّم في السجل التاريخي للحالة.** لا يُدرج السجل التاريخي للإنشاء والأرشفة إلا إذا لم يوجد معرض، أو كان المعرض يسمح بالمُدخلات المؤرشفة، أو كان «العرض بصفة مدير» مفعّلًا.
- **التحكّم في السجل التاريخي للتفاعلات.** كما سبق – يتطلّب عارض نظام (بلا معرض) أو `showTriggerHistory.enabled`؛ ويتحكّم `showTriggerDetails` في إدراج حمولة مُدخلات التفاعل ومُخرجاته وخطئه.

إخفاء الحقول الحسّاسة

Since 1.8 – `maskSensitiveHistoryFields(...)` (`AUDIT_SENSITIVE_MASK`، `buildSensitiveFieldScope`).

تُنفَّذ هذه المعالجة في **كل** قراءة للسجل التاريخي (v1 وv2)، بعد حلّ المراجع، بوصفها الخطوة الأخيرة قبل إعادة النتائج. ويكون سلوكها كالآتي:

- تبني مجموعة قيم `fullNormalizedName` للحقول التي فيها `options.sensitive === true` في الكيان. فإن لم يوجد أيٌّ منها، تنتهي المعالجة دون تنفيذ أي إجراء.
- في كل فرق من قيد `Update` يطابق `path` فيه حقلًا حسّاسًا، تُستبدل قيمتا `old` و`new` بـ `AUDIT_SENSITIVE_MASK` – **ما لم** يكن لدى المستخدم صاحب الطلب منحة حقل حسّاس مطابقة (`SensitiveFieldsService.canAccessSensitiveField`) على نطاق الوحدة والكيان والمسار الخاص بالحقل).
- يُطبَّق الإخفاء لكل خاصية على حدة: ففرق الكتابة الأولى الذي لا يحمل إلا `new` لا تُضاف إليه قيمة `old` مُخفاة لم تكن موجودة أصلًا.
- تُعاد القيود من غير نوع `Update` دون تغيير.

ويعامل الرماز ذلك على أنه **غير مشروط ودائم في سجل التدقيق**: فمتى كُتبت قيمة في السجل التاريخي عُرضت مُخفاة لكل من لا يملك المنحة، فلا تتسرّب القيم الحسّاسة التاريخية عبر واجهة السجل التاريخي البرمجية.

السجل التاريخي للتفاعلات

المصدر: `trigger-history.service.ts`، والنموذج `trigger-history.model.ts`. السجل التاريخي للتفاعلات **Long-standing** (`TriggerHistoryService`).

تُحفظ تشغيلات التفاعلات في صورة مستندات `TriggerHistory` (المجموعة `trigger_history`) بالصيغة `{triggerId, spaceId, eventName, relatedTo, triggerName, timestamp, doneBy, delegatee?, result: ActionResult[]}`.

تُجري `getTriggersHistory(entityId, entryId, showDetails?, limit?, lastTimeStamp?)` الاستعلام بشرط OR، فيظهر التشغيل في السجل التاريخي للمُدخل إذا تحقّق **أيٌّ** من الآتي:

- `relatedTo.dataEntry === entryId` (المُدخل الذي أطلقه)، **أو**
- `result.effectedEntities.entry.dataEntry === entryId` (مُدخل *تأثّر* بالتفاعل).

وقد يحتوي كل تشغيل مخزَّن على عدة نتائج إجراءات في `result`؛ فتسطّحها الخدمة (`flattenDeep`) إلى قيد `DataEntryHistory` واحد لكل نتيجة إجراء، مع `historyType = TriggerHistory` و`additionalData` من النوع `TriggerHistoryDto`:

حقل `TriggerHistoryDto`	ملاحظات
`triggerId`، `triggerName`	التفاعل المعني
`eventType`	الحدث الذي أطلقه (`eventName`)
`actionType`	نوع الإجراء الذي نُفِّذ
`success`	علَم نجاح لكل إجراء
`details.actionName`	موجود دائمًا
`details.{eventName, entity, entry, startTimeStamp, input, output, error, doneBy}`	**فقط حين يكون `showDetails` صحيحًا**

تتحدّد قيمة `showDetails` بقيمة `showTriggerHistory.showTriggerDetails` في المعرض (ويحصل عارضو النظام بلا معرض على details = true). ويحاكي `limit` + `lastTimeStamp` عقد التقسيم إلى صفحات في v2 (`timestamp DESC`، `timestamp < lastTimeStamp`).

الشرائح المشتقّة: الحالة وسير العمل واتفاقية مستوى الخدمة

هذه الشرائح **تُحسب وقت القراءة من مستند المُدخل ومن خدمات أخرى**، ولا تُخزَّن في مجموعة السجل التاريخي:

- **الحالة** (`getDataEntryStateHistory`) – تحوّل عناصر `entry.stateHistory` التي فيها `newState > 0` إلى قيود `Create` (Active) أو `Archive`، باستخدام `changedAt` / `changedBy` / `delegatee`.
- **سير العمل** (`getEntryWorkflowHistory`) – تمرّ على `entry.value._metaWorkflowHistory` **كله** (سير العمل الرئيس **وكل** تدفق فرعي)، فتحلّ كل `stepHistory` إلى قيد `WorkflowAction`. ويحمل كلٌّ منها `workflowContext` `{workflowId, instanceId, parentInstanceId, isSubWorkflow}`. و**Since 1.4** يعالج المُجمِّع كل سجلات سير العمل، لا سير العمل النشط وحده، فتظهر خطوات التدفق الفرعي في الخلاصة. ويكون الفرز تنازليًا بحسب الطابع الزمني، مع قاعدة لفضّ التساوي تضع الانتقالات التلقائية للتدفق الفرعي (`StepType.SubWorkflowStep`) قبل إجراءات المستخدم الواقعة في اللحظة نفسها.
- **اتفاقية مستوى الخدمة** (`getEntryOverdueSlaHistory`) – تجلب قيود التجاوز من `SlaService` للمُدخل، وترفق اسم الخطوة من سير العمل النشط، وتُصدر قيود `SlaOverdue` منسوبة إلى مستخدم النظام. **Long-standing**.

وتقبل الشرائح الثلاث جميعًا `limit` / `lastTimeStamp` اختياريين لمسار v2. ويؤدي حذف المُدخل إلى حذف سجله المحفوظ (`deleteHistoryForDataEntry`) عند أحداث الحذف والحذف النهائي)؛ أما هذه الشرائح المشتقّة فتزول بزوال المُدخل أو القيود الأساسية.

الجزء الثاني – الشريط الزمني (timeline-service)

المفهوم

الشريط الزمني خلاصة نشاط تتألّف من **منشورات**. ويُخزَّن المنشور نفسه بوصفه مُدخلَ بيانات ضمن كيان مخصّص للمنشورات (`Constants.POSTS_ENTITY_ID`)، ويُنتَج تلقائيًا من مُدخل مصدر متى هُيّئت **مساهمة شريط زمني** لكيان ذلك المُدخل. لذا لا تُنشئ المنشورات يدويًا، بل تُعرّف إعدادات الشريط الزمني على الكيان، ويتولّى النظام إبقاء المنشورات متزامنة بوصفها **نموذج قراءة قائمًا على الأحداث**. وتُحدِّد «خيارات مساهمة الشريط الزمني» الخاصة بالكيان كيفية الربط: أيّ حقل يُستخدم للعنوان والوصف والسمات والتصنيف والأيقونة، وكيف تُضبط إعدادات النشر والتثبيت.

ويستند التسجيل هنا إلى الأحداث، على غرار السجل التاريخي للمُدخل:

حدث المصدر	معالج الشريط الزمني	الأثر
`DataEntryCreatedEvent`	`PostDataEntryCreatedHandler` ← `TimelineService.submitPost`	بناء منشور وإنشاؤه
`DataEntryUpdatedEvent`	معالج التحديث ← `TimelineService.updatePost`	إعادة بناء المنشور من المُدخل
أرشفة المُدخل أو حذفه	معالجات الأرشفة والحذف ← `archivePost` / `deletePost`	إزالة المنشور أو سحبه
بلوغ خطوة سير عمل	`trigger-workflow-step-reached-event.handler.ts`	تحديد حالة النشر المرتبطة بسير العمل

يبني `submitPost` / `updatePost` المنشور عبر `buildPost(...)`، ويُرسلان `CreateDataEntryCommand` / `UpdateDataEntryCommand` عبر الناقل (مع `skipValidation = true`)، فتمرّ المنشورات بمسار الكتابة نفسه في data-service كأي مُدخل؛ ولهذا أيضًا **يحصل المنشور على سجل تاريخي للمُدخل خاص به** (الجزء الأول) *ويشارك* في خلاصة الشريط الزمني. لذا ميِّز بين واجهة السجل التاريخي للمُدخل (data-service) وواجهة منشورات الشريط الزمني (timeline-service).

تفعيل شريط زمني (المصمّم)

يُضاف الشريط الزمني إلى كيان موجود. افتح الكيان، واختر **تعديل الكيان** (السهم المجاور لـ **أنشئ مدخل**)، ثم اختر **الشريط الزمني** من الشريط العلوي (التفاصيل - التفاعلات - البنية - سير العمل - الشريط الزمني). وأضف الشريط الزمني من وسط الشاشة؛ فيُعيَّن له اسم وأيقونة افتراضيان مأخوذان من الكيان، ويمكنك تغيير كليهما.

بمجرد حفظ إعدادات الشريط الزمني وتفعيلها على الكيان، يبدأ النظام مزامنة المنشورات مع مُدخلات الكيان تلقائيًا: فتُنشأ منشورات للمُدخلات الجديدة، وتُحدَّث حين يتغيّر المُدخل، وتُؤرشف أو تُحذف حين يُؤرشف المُدخل أو يُحذف. ويؤدّي إيقاف الشريط الزمني على كيان إلى حذف إعداداته المرتبطة به.

قوالب المنشورات

يُربط كل حقل من حقول المنشور بمصدر قيمته بأحد أنواع الربط الآتية في `MappingTypeEnum`: **ثابت** (قيمة حرفية واحدة لكل المنشورات)، أو **متغيّر/ مربوط بالمُدخل** (`EntryMapping` – يأخذ القيمة من مسار في قيمة مُدخل المصدر)، أو **مخفي** (`Hidden` – لا يظهر في المنشور). أما الحقول الإلزامية فلا تقبل إلا الثابت أو المتغيّر؛ إذ لا يتاح لها خيار «مخفي»، لأن المستخدم ملزَم بتقديم قيمة.

وفي باني الشريط الزمني، تختار القالب ضمن **إعدادات المنشور**: افتح **قالب المنشور** واختر **منشورات عامة** أو **منشورات التقويم** (والاسم الإنجليزي «Calender Post» هو التهجئة المعتمدة في المنتج لعنوان التصنيف المزروع). وتتغيّر مجموعة الحقول المتاحة في المنشور بحسب القالب المختار. وخيارات **النوع** للربط في المصمّم هي **حقل** (= `EntryMapping`) و**ثابت** و**مخفي**.

لكل قاطن تصنيفان للمنشورات: **العامة** و**التقويم (الأحداث)** (انظر تصنيفات المنشورات المزروعة لمعرفة كيفية إنشائهما):

- **المنشورات العامة** تتيح **العنوان** و**الوصف** و**المرفقات**.
- **منشورات التقويم** تتيح، إضافةً إلى ذلك، **تاريخ ووقت البدء** و**تاريخ ووقت الانتهاء** وصورة **الغلاف** و**الموقع** (مكان فعلي أو رابط لحدث افتراضي).

> **ملاحظة حول القوالب:** لا يُعرض حاليًا في واجهة الويب بصورة كاملة إلا قالب **منشورات عامة**؛ أما تصنيف منشورات التقويم فمعرَّف في النموذج، لكن لم يُخصَّص له بعد قالب عرض.

تصنيفات المنشورات المزروعة `Since 1.10`

أصبح إنشاء الكيانين الضمنيين للشريط الزمني، **المنشورات** و**تصنيفات المنشورات**، والتصنيفين الافتراضيين، يجري لكل قاطن عبر **سلسلة الترحيلات** في (1.10.0-a21-seed-timeline-post-categories) data-service، وتُشغَّل هذه السلسلة أيضًا في أثناء تهيئة القاطن. وقبل 1.10، كان إنشاؤها يجري بعملية زرع عند إقلاع timeline-service، تستدعي data-service عبر HTTP؛ وكان القاطن قيد التهيئة يرفض تلك الاستدعاءات، مما قد يترك القاطن المهيَّأ على خلية متعددة القاطنين **بلا أي تصنيف منشورات**.

ما يعنيه ذلك للمُكامِلين:

- لا ينشئ الزرع **إلا ما هو غائب** (بالمطابقة على الأسماء الموحّدة للكيانات وعلى المعرّفات الثابتة للتصنيفات)، لذا لا تؤدّي إعادة تشغيله على قاطن مزروع إلى كتابة أي شيء.
- معرّفات التصنيفات **ثابتة ومتطابقة في كل قاطن**؛ لذا يمكنك الإشارة إليها كما تشاء، لكن لا تُنشئ بنفسك تصنيفات بهذه المعرّفات.
- لم تعد timeline-service تزرع شيئًا عند الإقلاع. وأي تغيير مستقبلي على هذه التعريفات التي يديرها الرماز يُوزَّع في صورة ترحيل في data-service، ويُطبَّق بالترحيل لا بإعادة التشغيل.

قراءة الخلاصة

تتطلّب جميع نقاط نهاية الشريط الزمني المصادقة (`Auth2AuthenticationStrategyName`). المصدر: `timeline.controller.ts`.

الطريقة والمسار	الغرض	مصوِّتو التفويض
`GET /v1/posts`	سرد المنشورات في الخلاصة	`canShowPostVoter`، `systemViewerVoter`
`GET /v1/posts/{id}`	منشور واحد	`canShowPostVoter`، `systemViewerVoter`
`GET /v1/posts/count`	عدّ المنشورات	`canShowPostVoter`، `systemViewerVoter`
`GET /v1/posts/entities/{entityId}`	منشورات كيان مصدر معيّن	`denyEndUserVoter`، `canViewPostEntityVoter`
`GET /v1/posts/entities/{entityId}/count`	العدد لكيان مصدر	`denyEndUserVoter`، `canViewPostEntityVoter`
`PATCH /v1/posts/{id}/pin`	تثبيت / إلغاء تثبيت	`canPinPostVoter`، `canEditEntityVoter`
`PATCH /v1/posts/{id}/publish`	نشر / حجب / جدولة	`canPublishPostVoter`، `canEditEntityVoter`

يقبل `GET /v1/posts` مرشِّح LoopBack القياسي `Filter<Post>` (معامل الاستعلام `filter`). ويدمج المتحكّم تلقائيًا **مرشّحات الجمهور** عبر `TimelineService.additionalFilters()`:

- يُطبَّق دائمًا `state === Active`، إضافةً إلى شرط OR على:
  - `value.anonymous.enabled === true` (منشورات عامة)، أو
  - مصادَق عليه + غير مقيَّد، أو
  - مصادَق عليه + مقيَّد **و** المستخدم الحالي (أو إحدى مجموعاته) مُدرج في `value.authenticated.entitledEntries`.

فالخلاصة مقيَّدة بالجمهور بطبيعتها: لا يرى المستدعي المجهول إلا المنشورات المتاحة للمجهولين، ويرى المستدعي المصادَق عليه، إضافةً إليها، المنشورات غير المقيَّدة والمنشورات المقيَّدة التي يحقّ له الاطّلاع عليها. وحين يوجد **معرض**، يُدمج `EndUserService.postFilter` أيضًا، وإذا كان عرض الشريط الزمني في المعرض لا يتيح `allowPublishPost`، تُزال الحقول `publishState.workflow.currentStep(Name)` من كل منشور مُعاد.

أما `GET /v1/posts/entities/{entityId}` فيتيح عرض المنشورات للمديرين وبُناة التطبيقات (ويُمنع منه المستخدمون النهائيون)، ويرشّحها بحسب `value.reference.dataEntity === entityId`.

وضع المعاينة: تتجاوز جميع نقاط نهاية الخلاصة والعدّ والسرد بحسب الكيان المعالجة المعتادة، وتعيد منشورات مصطنعة (`FakePostGeneratorService`) حين يُضبط `previewMode`. وتتطابق نتائج نقاط نهاية العدّ دائمًا مع نتائج نقطة نهاية السرد المقابلة، لأنها تعدّ الخلاصة المصطنعة نفسها.

نموذج `Post`

المصدر: `post.model.ts`. يحاكي المنشور مُدخل بيانات (`id`، `specificationVersion`، `dataEntity`، `value`، `creationInfo`، `updateInfo`، `state`). ويقع المحتوى المهم في `value` (`PostDataEntryValue`)، ولا سيما:

- `title` و`description` و`attributes` – تُربط بمُدخل المصدر وفق `MappingTypeEnum` في المساهمة (`Fixed` = قيمة حرفية، `EntryMapping` = مسار في قيمة مُدخل المصدر، `Hidden` = null).
- `reference` `{dataEntity, dataEntry}` – مؤشّر يرجع إلى مُدخل المصدر.
- `module`، `timelineConfig` `{category, title, icon, publishConfig, pinnedConfig}`.
- `anonymous`، `authenticated` – إعدادات الجمهور (تتحكّم في `additionalFilters`).
- `pinned` (bool) + `pinnedState`، `published` (bool) + `publishState`.
- `publishInfo` / `unpublishInfo`، `pinnedInfo` / `unpinnedInfo` – أختام `ActionInfo` لآخر إجراء نشر أو تثبيت.

التثبيت

```
PATCH /v1/posts/{id}/pin
Body: { "pinned": true | false }
```

القيود التي يفرضها المتحكّم:

- `pinned` إلزامي (وإلا `missingRequired('pinned')`).
- إذا حدّدت المساهمة `pinnedConfig.pinnedDate` للتثبيت التلقائي، فلا يمكنك التثبيت يدويًا (`UnablePinnedPostManuallyException`)؛ وبالمثل، يمنع إعداد `unpinnedDate` إلغاء التثبيت يدويًا (`UnableUnPinnedManuallyPostException`).

- يؤدّي ضبط `pinned` على قيمته الحالية إلى رمي `DuplicatePinnedStatusPostException`.

وعند النجاح، يستدعي المتحكّم `TimelineService.handlePinnedPost(spaceId, entry, manual, {userId, isPinned})`، الذي يعيد حساب `pinnedState`، ويضبط `pinned`، ويسجّل الختم في `pinnedInfo` / `unpinnedInfo`، ويحفظ عبر `UpdateDataEntryCommand`.

النشر والجدولة

في قسم **نشر** في المصمّم، يحدّد **النشر عند الخطوات** خطوات سير العمل التي يُنشر عندها المنشور (`publishStepIds`)؛ وتركه فارغًا يُبقي `publishConfig.manual` مفعّلًا. ويُستخدم **تاريخ النشر** / **تاريخ الحجب** لربط حقلي تاريخ من مُدخل المصدر. وتوضّح تلميحات المصمّم القاعدة المركّبة: في النشر اليدوي: «سيتم نشر المنشور عندما تقوم بالضغط على زر النشر و يكون تاريخ النشر قد مضى»، وفي النشر عبر سير العمل: «سيتم نشر المنشور عندما يصل المنشور إلى خطوة النشر و يكون تاريخ النشر قد مضى» (`calculatePublishedState`). أما تواريخ التثبيت وإلغاء التثبيت التلقائية (`pinnedConfig`) فلا يتاح لها عنصر تحكّم في المصمّم.

```
PATCH /v1/posts/{id}/publish
Body (one of):
  { "publish": true | false }                // manual publish/unpublish
  { "publishDate": "<ISO>" }                  // schedule publish
  { "unpublishDate": "<ISO>" }                // schedule unpublish
```

يمرّ المنشور بعدة حالات: **مسودّة ‹ منشور ‹ محجوب**. ويدعم النظام ثلاث طرائق للتحكّم في النشر، تُتتبَّع في `publishState` ويحدّد حالتها `getPostPublishState`: **يدوي**، و**مجدول** `{publishDate, unpublishDate, published}`، و**عبر سير العمل** `{currentStep, currentStepName, published}`، مثل نشر المنشور تلقائيًا حين يبلغ سير العمل خطوة موافقة المدير.

القواعد (من `publishPost` + `manualPublishValidations`):

- `publishDate` و`unpublishDate` **متنافيان** في الطلب الواحد (`TimelinePublishDateConflictException`).
- إذا لم يكن الطلب طلبَ جدولة، فإن `publish` إلزامي.
- يتطلّب النشر والحجب اليدويان `timelineConfig.publishConfig.manual` في المساهمة، وإلا `PublishNotAllowedException` / `UnpublishNotAllowedException`.
- لا يمكنك جدولة نشر أو حجب لحقل يتحكّم فيه أصلًا ربط مُهيّأ لـ `publishDate` / `unpublishDate` (`UnableScheduleManualPublishUnpublishPostException`).
- يجب أن تكون التواريخ المجدولة **في المستقبل**؛ وإن أُعطي كلاهما، وجب أن يكون `publishDate < unpublishDate`.
- يُرفض نشر منشور منشور أصلًا (أو حجب منشور محجوب أصلًا).

تُرسل الجدولة `ScheduleJobCommand` (`JobName.ScheduleJob`) بالمعرّف `publish-<postId>` / `unpublish-<postId>` للموعد المستقبلي؛ وتُشتق القيمة المنطقية الفعلية `published` عبر `calculatePublishedState(...)`. وتعمل جدولة التثبيت على نحو مماثل (`pinned-<postId>` / `unpinned-<postId>`)، وإن كان فرع التثبيت اليدوي في `getPostPinnedState` معلَّقًا في الرماز حاليًا؛ لذا تتحكّم التواريخ في جدولة التثبيت. أما النشر عبر سير العمل فيضبطه معالج بلوغ خطوة سير العمل، الذي يمرّر الخطوة التي بُلِغت إلى `handlePublishPost(..., workflow, {currentStep, currentStepName, isPublish})`.

التفاعلات والتعليقات

تُضيف المنشورات طبقة اجتماعية إلى الخلاصة:

- **التفاعلات** – تفاعلات بالرموز التعبيرية على المنشور، مع عدّاد للتفاعلات وإظهار تفاعل المستخدم الحالي نفسه؛ ويُحفظ تفاعل واحد لكل مستخدم. وتتيح واجهة الويب مجموعة الرموز الكاملة، بينما يقتصر تطبيق الجوال حاليًا على تفاعل واحد (`like`).
- **التعليقات** – يمكن التعليق على المنشورات، ويُعرض عدد التعليقات مع المنشور. وتدعم التعليقات **الإشارات (@)** (يُختار الشخص عبر مربع بحث، ويُشعَر المستخدمون المُشار إليهم) وتحديد الجمهور لكل تعليق. وأنواع مرفقات التعليقات **يضبطها المدير لكل معرض** (بحدّ افتراضي للحجم يقارب 1MB)، بدلًا من الاعتماد على قائمة ثابتة من أنواع الملفات.

> **تمييز مهم:** تفاعلات الشريط الزمني هنا **اجتماعية** (رموز تعبيرية)، وتختلف تمامًا عن «التفاعلات» بمعنى الأتمتة (المشغّلات والإجراءات المستندة إلى الأحداث).

وضع المعاينة

عند تصميم كيان أو معرض، يمكنك استعراض الشريط الزمني في وضع المعاينة؛ إذ يعرض النظام خلاصة مصطنعة حتمية (منشورات نموذجية بمحتوى مؤقت) بدلًا من البيانات الحقيقية، لتعاين الخلاصة قبل النشر. ويماثل ذلك سلوك المعاينة في نقاط نهاية السجل التاريخي للمُدخل، التي تعيد سجلًا مصطنعًا مطابقًا في الشكل عند ضبط العلَم نفسه.

ملاحظات عامة للمطوّرين

- **التعدادات تُرسَل أرقامًا.** يُسلسَل كلّ من `historyType` وحالة المُدخل `state` في صورة أرقام. فحوِّل هذه القيم في جهة العميل (`HistoryTypeEnum` ، `DataEntityStateEnum`).
- **«سجلّان» مختلفان للمنشور.** المنشور مُدخل بيانات، لذا له خلاصة `.../entries/{id}/history` خاصة به، *ويشارك* كذلك في خلاصة الشريط الزمني. فأبقِ الواجهتين منفصلتين.
- **التسجيل غير متزامن.** تتولّى معالجات الناقل التي تستجيب لأحداث `DataEntry*Event` إنتاج السجل التاريخي للمُدخل ومنشورات الشريط الزمني، ولا يُنتجهما الطلب نفسه على نحو متزامن؛ لذا توقّع اتساقًا نهائيًا بين استجابة الكتابة وقراءة السجل التاريخي أو الشريط الزمني.
- **الحذف.** يؤدّي حذف المُدخل إلى حذف سجله المحفوظ؛ أما الشرائح المشتقّة (سير العمل والحالة واتفاقية مستوى الخدمة والتفاعلات) فتزول بزوال المُدخل أو القيود الأساسية.
- **عقد التقسيم إلى صفحات.** فضِّل v2 (مؤشّر `lastTimeStamp` ، وحجم صفحة ثابت 12، و `doneAt DESC`) للسجل التاريخي للمُدخل؛ أما نقاط نهاية خلاصة الشريط الزمني فتستخدم `Where` / `Filter` القياسيين في LoopBack.

ملاحظات وقيود

- لا يُعرض حاليًا بصورة كاملة إلا قالب **منشورات عامة**؛ ولم يُخصَّص بعد قالب عرض لتصنيف منشورات التقويم.
- الشريط الزمني **نموذج قراءة قائم على الأحداث**؛ فإن فات حدثٌ من الناقل، فقد تبدو الخلاصة غير محدَّثة لبعض الوقت إلى أن يصل التحديث التالي.
- تعتمد خدمة الشريط الزمني على خدمة البيانات الأساسية؛ فإن تعذّر الوصول إليها، فقد يفشل تحميل الخلاصة.
- في الجوال، تقتصر التفاعلات حاليًا على `like` واحد، بينما تتاح المجموعة الكاملة على الويب.

الإشعارات وتسليمها (الواجهة البرمجية والتفاصيل الداخلية)

هذه الصفحة موجّهة إلى المُكامِلين والمطوّرين الذين يقرؤون إشعارات مستخدم عبر عميلهم الخاص، أو يسجّلون الأجهزة لتلقّي الإشعارات الفورية، أو يحتاجون إلى فهم سبب وصول إشعار إلى شخص ما أو عدم وصوله إليه. وتتناول الإشعار المخزَّن، وواجهة الإشعارات البرمجية الموجّهة إلى المستخدم، وتسجيل الإشعارات الفورية وتسليمها، والتسليم اللحظي عبر اتصال الوقت الحقيقي، ومسارَي البريد الإلكتروني والرسائل النصية.

للاطّلاع على ما يراه المستخدم النهائي، انظر استقبال الإشعارات [استخدام أسس]. ولمعرفة كيفية تهيئة المؤسسة للقوالب والمشغّلات والقنوات، انظر إعداد الإشعارات [الإدارة › الإشعارات وقوالب البريد الإلكتروني]. ويتوفّر مرجع نقاط النهاية المولَّد ضمن notification-api.

القنوات

القناة	أين يراها المستخدم	مصدرها
داخل التطبيق	لوحة الجرس في البوابة وصفحة **جميع الإشعارات**؛ وتبويب الإشعارات في تطبيق الجوال	إشعار يُنشأ للمستخدم
الإشعارات الفورية	أجهزة المستخدم **المحمولة** المسجَّلة، حتى حين يكون التطبيق مغلقًا	الإشعار نفسه داخل التطبيق، وتُرسَل نسخة منه إلى رموز أجهزة المستخدم عبر Firebase Cloud Messaging
البريد الإلكتروني	صندوق بريد المستخدم	أحداث البريد: رسائل الأتمتة، وإعادة تعيين كلمة المرور، وتأكيد رمز API
الرسائل النصية	هاتف المستخدم	أحداث الرسائل النصية من الأتمتة (إرسال فقط؛ فلا صندوق وارد للرسائل النصية)

يحمل الإشعار المخزَّن قائمة `channels`، لكن قيمة القناة الوحيدة المعرَّفة هي **داخل التطبيق** (`NotificationChannelTypeEnum.InApp = 0`). أما البريد الإلكتروني والرسائل النصية فلا يُخزَّنان في سجل الإشعار، بل يُعالَجان بوصفهما أحداثًا منفصلة.

بوابة الويب لا تتلقّى الإشعارات الفورية. فالواجهة الأمامية تتضمّن عميلًا مولَّدًا لنقاط نهاية رموز الإشعارات الفورية، لكنها لا تسجّل أي رمز، ولا تدعم الإشعارات الفورية في المتصفح ولا رسائل عامل الخدمة (service worker). ويتلقّى مستخدمو الويب الإشعارات في لوحة الجرس لحظيًا ما دامت البوابة مفتوحة (انظر التسليم اللحظي). أما الإشعارات الفورية فقناة خاصة بتطبيق الجوال وحده.

كيف تُنتَج الإشعارات

تُنتَج الإشعارات استجابةً للأحداث: فحين يقع حدث (إنشاء مُدخل، أو بلوغ خطوة في سير العمل، أو طلب إعادة تعيين كلمة مرور، أو اكتمال توليد ملف)، تحدّد خدمة الإشعارات القناة أو القنوات، والمستلمين، والمحتوى، ثم ترسل الإشعار. ويستند المحتوى إلى القوالب، إذ تحتوي القوالب على رموز تُستبدل بقيمها وقت الإرسال، فيُنتج القالب الواحد رسالة مخصّصة لكل مستلم. انظر القوالب [الإدارة › الإشعارات وقوالب البريد الإلكتروني].

ويُخزَّن كل إشعار داخل التطبيق بوصفه **مُدخلَ بيانات**، فيُدار كأي مُدخل آخر. ويتضمّن:

- **عنوانًا** و**محتوى**، كلاهما مترجَم (بالإنجليزية والعربية مثلًا)؛

- **أيقونة** (مثل `file` لعملية تصدير مكتملة)؛
- **مرفقًا** اختياريًا (مثل الملف المولَّد)؛
- **إشارة** إلى العنصر الذي يتعلّق به، إضافةً إلى **الوحدة**، ليتمكّن العميل من الربط به مباشرةً؛
- **رابطًا** اختياريًا داخل التطبيق `Since 1.10` (انظر الروابط)؛
- `readAt`، ويبقى فارغًا حتى يفتح المستخدم الإشعار.

تقييد النطاق

ينتمي كل إشعار إلى مستخدم واحد ومساحة واحدة. ولا تعيد الواجهة البرمجية إلا إشعارات المستدعي نفسه في المساحة الحالية وحدها، ولا تعيد الإشعارات المؤرشفة أبدًا. ويُفرض ذلك على الخادم (`NotificationController.additionalFilters` يرشّح بمعرّف المستخدم والمساحة الحالية ويستبعد المُدخلات المؤرشفة)، وليس في الواجهة وحدها.

الواجهة البرمجية الموجّهة إلى المستخدم

تُجرى جميع الاستدعاءات بهوية المستخدم الذي سجّل دخوله، وفي المساحة التي يحدّدها مضيف الطلب.

السرد: `GET /v1/my`

يعيد سجلات إشعارات المستدعي في المساحة الحالية. ويسري عليها الترشيح القياسي (التقسيم إلى صفحات، أو الاقتصار على غير المقروءة). وتطبّقه البوابة نيابةً عن المستخدم.

العدّ: `GET /v1/my/count`

```
{
  "count": 42,
  "unreadCount": 5
}
```

- `count`: عدد الإشعارات المطابقة للعرض الحالي.
- `unreadCount`: عدد الإشعارات التي لا يتضمّن سجلّها `readAt` أو تكون قيمته `null`.

وهذا ما يحدّد العدد في شارة الجرس (وتظهر `99+` عندما يتجاوز العدد 99).

`Since 1.10` أصبح الاستدعاءان يستندان إلى فهرس مخصّص (للمستخدم والمساحة)، بعدما كان كل استدعاء يمسح مجموعة الإشعارات كاملة. وأُصلح أيضًا عدم تطابق في حساسية حالة الأحرف في استعلام العدّ، فأصبح `unreadCount` متّسقًا مع ما تعيده القائمة.

التمييز كمقروء: `PATCH /v1/my/{id}/mark-as-read`

يضبط `readAt` على الوقت الحالي ويَسِم الإشعار بأنه مقروء، فلا يعود يُحتسب ضمن `unreadCount`. ولا يمكن تمييز إلا إشعار المستدعي نفسه في المساحة الحالية؛ ويُرفض الطلب إذا لم يطابق أي إشعار.

لا توجد نقطة نهاية لـ«تمييز الكل كمقروء»، **ولا** نقطة نهاية للحذف. فالتمييز يجري لإشعار واحد في كل مرة، ولا تغادر الإشعارات القائمة إلا حين تُؤرشف في موضع آخر. (تحتوي ملفات ترجمة البوابة على نص «تحديد الكل كمقروء»، لكن لا تستخدمه أي شاشة.)

وفي تطبيق الجوال، يؤدّي النقر على إشعار إلى تمييزه بأنه مقروء وإنقاص عدّاد غير المقروءة **تفاؤليًّا**، قبل ورود تأكيد الخادم. فإن رُفض `mark-as-read`، تراجع التطبيق عن التغيير ليبقى العدّاد دقيقًا.

الروابط `Since 1.10`

قد يتضمّن الإشعار `link` إلى صفحة داخل التطبيق. ويؤدّي النقر عليه في لوحة الجرس أو في صفحة الإشعارات إلى تمييزه بأنه مقروء ونقل المستخدم إلى تلك الصفحة. ولا تُتبع إلا العناوين الواقعة داخل التطبيق؛ أما أي عنوان آخر فيُتجاهل، ويُفتح الإشعار وفق السلوك السابق (المُدخل أو المعرض الذي يشير إليه). ويستخدم مسار النشر في المتجر هذه الآلية لتوجيه مقدّمي الحزم ومراجعيها إلى صفحة الحزمة الصحيحة. ولإعداد الرابط، انظر: رابط داخل التطبيق في الإشعار [الإدارة › الإشعارات وقوالب البريد الإلكتروني].

وتُفتح صفحة **جميع الإشعارات** دائمًا على صفحتها الأولى `Since 1.6`. (كانت العودة إليها سابقًا تعرض مُدخلات الصفحة التي غادرها المستخدم، بينما يشير محدِّد الصفحات إلى الصفحة 1.)

الإشعارات المحوَّلة `Since 1.10`

حين يُفعَّل التحويل في تفويض ما، يتلقّى المفوَّض إليه نسخة من إشعارات المفوِّض (ورسائل بريده) ضمن نطاق التفويض ما دام نشطًا، ولا يتلقّى أبدًا نسخة مكرّرة من شيء موجّه إليه أصلًا. وتتضمّن النسخة التفويضَ، والمفوِّض، والهدفَ الأصلي. ويصنّفها العملاء كما يلي:

الحالة	المعنى	سلوك العميل
حالية	الجلسة تعمل أصلًا نيابةً عن ذلك المفوِّض	يفتح الهدف الأصلي مباشرةً
قابلة للتبديل	يتطلّب الفتح الانتقال إلى التفويض	يطلب تأكيد الانتقال، ثم يفتح الهدف
غير متاحة	انتهى التفويض، أو ثمة تفويض آخر نشط	يُعرض الإشعار معطَّلًا

ولا يميّز تطبيق الجوال الإشعار المحوَّل بأنه مقروء إلا إذا جرى فتحه فعلًا، ولا ينزّل مرفق المفوِّض أبدًا. أما رسائل البريد المحوَّلة فتُرسَل منها نسخة يُضاف فيها المفوَّض إليه إلى CC (أو BCC إن كان المفوِّض في BCC). للإعداد، انظر: نسخ المفوَّض إليه [الإدارة › الإشعارات وقوالب البريد الإلكتروني]. ولمعالجة الحالات في تطبيق الجوال، انظر: الإشعارات المحوَّلة بالتفويض.

التسليم اللحظي

ما دام العميل متصلًا بخدمة الوقت الحقيقي، يُرسَل الإشعار الجديد أيضًا **لحظيًّا** عبر ذلك الاتصال، فتُحدَّث القائمة والشارة دون إعادة تحميل. وتعرض بوابة الويب تنبيهًا منبثقًا يتضمّن عنوان الإشعار ومحتواه (أو رسالة عامة هي «إشعار جديد»)، وتزيد عدّاد غير المقروءة.

`Since 1.10` تتجاهل بوابة الويب الإشعار اللحظي الذي ينتمي إلى **مساحة غير** المساحة الحالية: فلا يظهر إشعار منبثق، ولا تتغيّر الشارة. ويظل الاتصال اللحظي ينقل جميع إشعارات المستخدم؛ والبوابة هي التي تقارن، داخل المتصفح، مساحة كل إشعار بالمساحة الحالية. أما الإشعار الذي لا يتضمّن مساحته فيُعرض ولا يُستبعَد. وقبل ذلك، كان يمكن للأتمتة التي يشمل تنفيذها عدة مساحات أن تسلّم الإشعار نفسه إلى كل جلسة مفتوحة لمساحة: حدث واحد ينتج عنه تنبيهان منبثقان متطابقان، واحتساب الإشعار مرتين في الشارة. أما إعادة تحميل القائمة (`GET /v1/my`) فلم تتأثّر قط، لأن الخادم يقصرها على المساحة الحالية.

للاطّلاع على تفاصيل المقبس، انظر: الوقت الحقيقي.

الإشعارات الفورية

تسجيل جهاز: `POST /v1/push-tokens`

```
POST /v1/push-tokens
Body: { "token": "<device push token>" }
```

السلوك (`PushNotificationsTokensController.create`):

- يُخزَّن الرمز **الجديد** ويُربط بالمستدعي، مع `lastUsedDate`.
- الرمز **الموجود مسبقًا** لكنه تابع لمستخدم آخر (جهاز مشترك) يُعاد إسناده إلى المستدعي ويُحدَّث `lastUsedDate` الخاص به.
- لا يتكرّر الرمز نفسه أبدًا.

يحصل تطبيق الجوال على رمز الجهاز من مزوّد الإشعارات الفورية، ويسجّله حين يسجّل المستخدم دخوله ويمنح إذن الإشعارات. ويؤدّي إبطال الإذن على الجهاز إلى إيقاف الإشعارات؛ أما الرمز المخزَّن فيتوقّف عن تلقّي أي شيء. ويتوفّر مرجع نقطة النهاية، بما فيه عمليتا الحذف والإبطال `Since 1.10`، ضمن PushNotificationsTokens.

التسليم

حين يُنشأ إشعار داخل التطبيق، تنسخه خدمة الإشعارات (`CreatedNotificationHandler`) إلى أجهزة المستخدم إذا:

1. كانت للإشعار القناة **داخل التطبيق**؛
2. وكان للمستخدم رمز إشعارات فورية مسجّل واحد على الأقل؛
3. وكان للإشعار عنوان صالح للاستخدام.

وترسل الخدمة العنوان والمتن إلى كل رمز مسجّل عبر **Firebase Cloud Messaging** (الخدمة `fcm-sender`)، التي توزّعهما على جميع الأجهزة دفعةً واحدة (`sendEachForMulticast`) وتشغّل صوت التنبيه الافتراضي على أجهزة Apple. ويتضمّن الإشعار الفوري أيضًا بيانات للربط العميق: العنصر المُشار إليه، والوحدة، والأيقونة، وأي مرفق، وعنوان URL الأساسي للمساحة.

ويجري التسليم **بأفضل جهد**: إذ يُسجَّل تقادم الرمز أو رفض FCM ويُحتسب، لكن ذلك لا يمنع أبدًا ظهور الإشعار في القائمة داخل التطبيق.

`Since 1.10` يتلقّى كل جهاز الإشعار الفوري **مرة واحدة**، حتى لو سُجِّل رمزه أكثر من مرة، وتُنقَّح التسجيلات: فالرمز الذي يُبلغ FCM عن عدم صلاحيته يُزال بعد الإرسال، والرمز الذي لم يُحدَّث منذ **90 يومًا** (`PUSH_TOKEN_STALE_AFTER_DAYS`) تزيله مهمة الصيانة. ويؤدّي تسجيل الدخول على الجهاز مجددًا إلى تسجيل الرمز من جديد. للاطّلاع على منظور المشغّل، انظر: تسليم الإشعارات الفورية وصيانة رموز الأجهزة [الإدارة › الإشعارات وقوالب البريد الإلكتروني].

لغة الإشعار الفوري

تُخزَّن العناوين والمتون بكل لغة، ويختار مُرسِل الإشعارات الفورية لغة المستلم، ويلجأ إلى أول لغة متاحة بدلًا من عدم إرسال أي شيء.

قيد معروف. تحديد لغة المستخدم في خدمة الإشعارات ليس إلا تنفيذًا مؤقتًا: إذ تعيد `getUserLanguage` دائمًا `en` (`services/notification-service/src/services/user.service.ts`). وعمليًا، تُرسَل الإشعارات الفورية بالإنجليزية متى وُجدت نسخة إنجليزية، وإلا فبأول لغة متاحة. أما القائمة داخل التطبيق فتعيد جميع اللغات المخزَّنة، ويعرض العملاء المحتوى بلغة المستخدم.

البريد الإلكتروني

ترسل خدمة `email-sender` أحداث البريد عبر SMTP. ومن الرسائل المعتادة:

- **رسائل الأتمتة (المشغّلات)**: موضوع، ومتن HTML، ومرفقات، ورابط عميق إلى العنصر.
- **إعادة تعيين كلمة المرور**: رابط إعادة تعيين آمن (`UserPasswordResetHandler`).
- **إنشاء رمز API**: تأكيد يُرسل إلى العنوان المرتبط، يتضمّن الرمز وتذكيرًا بحفظه بأمان (`AppTokenGeneratedHandler`).

ويُبنى الرابط إلى المنصة باستخدام مضيف المساحة نفسه، و**يُختصر** قبل تضمينه. وإن فشل الاختصار، استُخدم الرابط الكامل. ولا تظهر رموز إعادة التعيين ورموز API في سجل البريد؛ انظر إعداد الإشعارات [الإدارة › الإشعارات وقوالب البريد الإلكتروني].

الرسائل النصية

ترسل خدمة `sms-sender` أحداث الرسائل النصية (التي تأتي عادةً من الأتمتة)، وتدعم عدة شركات اتصالات وبوابات إقليمية، مثل EAND Bulkg وSmart Dubaig SDGg. وتُختصر الروابط أولًا. وتنتج عن كل عملية إرسال حالة تسليم مسجَّلة (نجاح أو فشل لكل مستلم) تُتتبَّع داخليًا. لإعداد القناة، انظر: قناة الرسائل النصية [الإدارة › الإشعارات وقوالب البريد الإلكتروني].

إشعارات جاهزية الملف

عند إنتاج ملف في الخلفية (كتصدير XLSX أو PDF)، ينشئ `FileGeneratedEventHandler` إشعارًا داخل التطبيق بعنوان **«الملف الخاص بك جاهز للتحميل»** مرفقًا به الملف، وتُرسَل نسخة منه كذلك عبر الإشعارات الفورية.

`Since 1.10` إذا فشل تصدير قائمة في الخلفية قبل اكتماله، فلا يُنتَج أي ملف، **ولا** يُرسَل إشعار جاهزية الملف، وبذلك لا يتسلّم المستخدم أبدًا مصنّفًا تالفًا أو مكتوبًا جزئيًا.

التفضيلات

لا توجد واجهة برمجية لتفضيلات الإشعارات أو الاشتراكات لكل مستخدم. فواجهة الإشعارات البرمجية لا تتيح إلا السرد والعدّ والتمييز كمقروء وتسجيل رموز الأجهزة؛ ولا شيء فيها يتيح تشغيل القنوات أو إيقافها، أو كتم الفئات، أو ضبط وتيرة الملخّصات.

- يتوقّف تلقّي الإشعارات داخل التطبيق والفورية على ما إذا أُنشئ إشعار للمستخدم، وما إذا كان له جهاز مسجّل مع إذن ممنوح من نظام التشغيل.
- يتوقّف تلقّي البريد الإلكتروني والرسائل النصية على إعداد الأتمتة التي تُطلقهما، وبيانات التواصل في الملف الشخصي للمستخدم.

مرجع سريع

إجراء المستخدم	ما يفعله العميل
رؤية الإشعارات	`GET /v1/my` (إشعاراته، في المساحة الحالية، غير المؤرشفة)
رؤية عدد غير المقروءة	`GET /v1/my/count` ← `count` / `unreadCount`
فتح إشعار أو إزالته	`PATCH /v1/my/{id}/mark-as-read` ← يضبط `readAt` على وقت القراءة
تلقّي الإشعارات المنبثقة على الهاتف	يسجّل تطبيق الجوال الجهاز عبر `POST /v1/push-tokens` بعد منح الإذن
إيقاف الإشعارات المنبثقة	إبطال إذن الإشعارات على الجهاز

إجراء المستخدم	ما يفعله العميل
تسلّم تصدير	إشعار جاهزية الملف مرفقًا به الملف
فتح إشعار محوَّل	يُفتح مباشرةً، أو يُطلب تأكيد الانتقال إلى التفويض أولًا

نموذج التقرير ومرجع واجهته البرمجية

هذه الصفحة موجّهة إلى **مطوّري التكامل والمطوّرين** الذين ينشئون التقارير أو يشغّلونها أو يصدّرونها عبر الواجهة البرمجية، أو يحتاجون إلى فهم دقيق لكيفية إنتاج النسخة المُصدَّرة. وتوثّق الصفحة نموذج التقرير، وقواعد التحقّق ورموز الأخطاء، ونقاط النهاية، ومراحل معالجة التصدير، والتسليم المجدوَل.

> **تبني تقريرًا من الواجهة؟**
>
> ينبغي لبُناة التطبيقات البدء بصفحة التقارير [بناء التطبيقات]: فهي تغطّي الميزات نفسها بالتسميات التي يعرضها المصمّم.

التقرير إسقاط جدولي محفوظ على كيان بيانات، يربط **استعلام بيانات** (مصدر الصفوف) بقائمة مرتّبة من **الحقول** (الأعمدة)، إضافةً إلى خيارات العرض والجدولة والإشعار والوصول. وبعد تعريفه، يمكن **تنفيذه** لإرجاع الصفوف الحالية، و**عدّ** صفوفه، و**تصديره** إلى نُسخ بصيغة PDF أو XLSX.

- **الويب فقط.** تتاح التقارير في تطبيق الويب فقط، ولا يُفتَح رابط التقرير إلا على الويب؛ أمّا على الجوّال، فيُحَلّ إلى `NotSupportedLink` ولا يمكن الانتقال إليه.

المفاهيم ونموذج البيانات

يُخزَّن التقرير بوصفه `DataEntry` تابعًا لكيان **Report** الأساسي. أمّا النموذج الفعلي فهو `ReportEntry`، الذي تحتوي خاصيته `value` على كائن `ReportEntryValue`:

```
ReportEntry (extends DataEntry)
  value: ReportEntryValue
```

ويفرض مُنشِئ `ReportEntry` القيمة `dataEntity = REPORT_ENTITY_ID` دائمًا، لذا يُنسب كل صف تقرير إلى كيان Report الأساسي، أيًّا كانت الحمولة.

حقول ReportEntryValue

من `report-entry-value.model.ts`:

الحقل	النوع	مطلوب	ملاحظات
`name`	`LocalizedValue`	نعم	اسم التقرير المعروض (لكل لغة). يجب أن يكون فريدًا ضمن كل كيان بيانات مرتبط؛ راجع قواعد التحقّق.
`normalizedName`	`string`	لا	معرّف برمجي يُولَّد تلقائيًا من `name`؛ فريد ضمن كل `DataEntity` مرتبط.
`description`	`NullableLocalizedValue`	لا	وصف مُترجَم؛ يُستخدَم في نص ترويسة PDF.
`icon`	`Icon`	(`require: true`)	أيقونة التقرير.
`dataSource`	`DataEntrySource`	(`require: true`)	مصدر الصفوف. يُدعَم النوع **Query** فقط (انظر أدناه).
`fields`	`string[]`	(`require: true` ، `minItems: 1`)	قائمة مرتّبة بمسارات حقول الكيان لإسقاطها في صورة أعمدة، مع الحفاظ على ترتيبها.

الحقل	النوع	مطلوب	ملاحظات
`allowChangeFields`	`boolean`	لا	إذا كانت قيمته true، يمكن لمستدعي التصدير تقليص مجموعة الأعمدة المُصدَّرة عبر `selectedColumns`.
`relatedTo`	`DataEntityReferenceProperty`	(`require: true`)	الكيان الذي يُشغِّل التقرير عليه (`{dataEntity, dataEntry}`).
`authenticated`	`AuthenticatedViewOption`	لا	إعداد الوصول والمصادقة. القيمة الافتراضية `{isRestricted: false, enabled: true}`.
`isPublished`	`boolean`	لا	القيمة الافتراضية `false`. ويتوقّف اشتراط الأذونات والتحقّق من الجدولة على حالة النشر.
`schedule`	`ReportScheduleModel`	لا	مؤقّت اختياري للتسليم المجدوّل.
`inAppNotification`	`SendReportNotificationActionOptions`	لا	إعداد تسليم الإشعار داخل التطبيق.
`emailNotification`	`SendReportEmailActionOptions`	لا	إعداد التسليم عبر البريد الإلكتروني.
`displayOrder`	`number`	لا	القيمة الافتراضية `0`. مؤشِّر للترتيب.
`design`	`ReportDesignOptions`	لا	خيارات تخطيط PDF والترويسة والتذييل.
`domain`	`ReferenceProperty`	لا	مرجع نطاق اختياري.
`sharing`	`ApplicationSharing`	نعم	بيانات وصفية لمشاركة التطبيق.
`help`	`ReportHelp`	لا	نص مساعدة سياقي. **Since 1.2**

> **ملاحظة حول `require` مقابل `required`.** تستخدم عدّة خصائص في النموذج مفتاح المُزخرِف المكتوب خطأً `require:` (مثل `true` و `icon` و `dataSource` و `fields` و `relatedTo`). تقرأ آلية التحقّق من الخصائص في LoopBack المفتاح `required` (وليس `require`)، لذا لا يفرض مخطّط إطار العمل هذه الخصائص بوصفها إلزامية، فهي **غير** إلزامية فيه؛ وقواعد العمل في `ReportService` (أدناه) هي التي تفرض فعليًا نوع `dataSource`، ومحتوى `fields`، و `relatedTo`.

مصدر البيانات (مصدر الصفوف)

`dataSource` هو من النوع `DataEntrySource`:

الحقل	النوع	ملاحظات
`type`	`DataCollectionTypeEnum`	يجب أن يكون `Query` (قيمة التعداد `2`). يُرفَض `Store` و `Collection`.
`store`	`string`	اسم المخزن (مُقيَّد بنمط).
`collection`	`UUID`	معرّف التجميعة.
`query`	`UUID`	معرّف استعلام البيانات الذي يُنتج الصفوف.

`DataCollectionTypeEnum = { Store = 0, Collection = 1, Query = 2 }`. تتطلّب قواعد التحقّق من التقرير أن يكون `type === Query`، لذا يجلب كل تقرير صفوفه عبر استعلام بيانات [بناء التطبيقات › الاستعلامات والمرشّحات].

ويُقيّد `report-reference-property.model.ts` مرجع التقرير بحيث يشير دائمًا إلى كيان `DataEntity` الأساسي (`dataEntity` هو ثابت `const` في مخطّط JSON قيمته `CoreEntities.DataEntity`)، مع `dataEntry` من نوع UUID. أمّا نوع `ReportEntryValue.relatedTo` فهو `DataEntityReferenceProperty` الأعمّ، لكن البنية نفسها تنطبق عليه: `{dataEntity, dataEntry}`.

الوصول والمصادقة

`authenticated` هو من النوع `AuthenticatedViewOption`:

الحقل	النوع	ملاحظات
`enabled`	`boolean`	ما إذا كان الوصول المُصادَق عليه مُفعَّلًا (القيمة الافتراضية في المخطّط `false`).
`defaultView`	`boolean`	ما إذا كان هذا هو المعرض الافتراضي (القيمة الافتراضية في المخطّط `false`).
`isRestricted`	`boolean`	ما إذا كانت للتقرير أذونات صريحة مرفقة. **محسوب فقط**: لا يُقبَل مباشرةً من حمولات الإنشاء أو التحديث لهذا الغرض، ويستند إليه التحقّق من أذونات النشر.

خيارات التصميم

تقابل خيارات **تصميم التقرير** في المصمّم خصائص `ReportDesignOptions`، وقيمها الافتراضية هي `pageHeader = true`، و`organizationHeader = true`، و`pageFooter = true`، و`organizationFooter = false`.

الأذونات

لا تُخزَّن الأذونات في مستند التقرير، بل في حقوق الوصول (انظر الأمور الداخلية والثوابت). ويُرفَض نشر تقرير **مُقيَّد** دون اختيار أي أذونات.

قواعد التحقّق

تمرّ جميع مسارات الإنشاء والتحديث عبر `ReportService.validateReportValue` و`validateNameUniqueness`.

`validateReportValue(value)`

1. **يجب** أن يكون `value.dataSource.type` مساويًا لـ `DataCollectionTypeEnum.Query`، وإلا أُطلِق الاستثناء `ReportDataSourceSupportedException` (الخطأ `REPORT_DATA_SOURCE_NOT_SUPPORTED`).
2. عندما يكون `relatedTo.dataEntity === CoreEntities.DataEntity`:
   - يجب أن يكون الاستعلام (`dataSource.query`) موجودًا، وإلا أُطلِق الاستثناء `DataQueryNotFound`.
   - يجب أن يكون الكيان المرتبط (`relatedTo.dataEntry`) موجودًا، وإلا أُطلِق الاستثناء `EntryNotExistException`.
   - يجب ألّا يكون الكيان مؤرشفًا، وإلا أُطلِق الاستثناء `EntityArchivedException`.
   - يجب أن يساوي `dataEntity` الخاص بالاستعلام قيمة `relatedTo.dataEntry`، وإلا أُطلِق الاستثناء `DataQueryNotBelongToEntity`.
   - **التحقّق من الحقول** (`validateReportFields`): يجب ألّا تكون `fields` فارغة (`ReportFieldsLengthException`، `REPORT_FIELDS_LENGTH`)، ويجب أن يكون كل عنصر فيها موجودًا في `getAllFieldsPaths()` الخاص بالكيان؛ وإذا وُجدت حقول غير معروفة، أُطلِق الاستثناء `ReportFieldsNotExistException` (`REPORT_FIELDS_NOT_EXIST`) مع سرد المسارات المُخالِفة.
3. إذا كان التقرير `isPublished` وكانت جدولة إرساله لمرة واحدة مفعَّلة عبر `schedule.enabled`، وكان الطابع الزمني `oneTime` في الماضي، أُطلِق الاستثناء `ReportSendingTimeCanNotBeInThePastException` (`REPORT_SENDING_TIME`).

`validateNameUniqueness(report, selfId?)` – لكل لغة مُهيّأة، إذا كانت قيمة `name[lang]` معيَّنة، تبحث الخدمة عن تقرير موجود تتطابق معه قيمة `value.relatedTo.dataEntry` وقيمة `value.name.<lang>`. عند PATCH، يُمرَّر `selfId` لاستبعاد

التقرير الحالي (`id neq selfId`). وإذا وُجد تعارض، أُطلِق الاستثناء `ReportExistException` (`REPORT_EXISTS`). وبذلك يكون التفرّد **ضمن كل كيان بيانات مرتبط، ولكل لغة على حدة**.

نقاط النهاية

توجد جميع المسارات في `ReportEntryController` ، وتخضع للمصادقة باستراتيجية OAuth2 في أسس، ويُفحص التخويل لكل مسار على حدة. ويمثّل `id` معرّف المُدخل الخاص بالتقرير.

الأسلوب	المسار	الغرض
`POST`	`/v1/reports`	إنشاء تقرير.
`GET`	`/v1/reports`	سرد التقارير (`filter` في LoopBack).
`GET`	`/v1/reports/count`	عدّ التقارير (`where` في LoopBack).
`GET`	`/v1/reports/{id}`	جلب تقرير واحد (يضمّ الأذونات).
`PATCH`	`/v1/reports/{id}`	تحديث تقرير.
`PATCH`	`/v1/reports/{id}/permissions`	تحديث الوصول إلى التقرير وأذوناته فقط.
`DELETE`	`/v1/reports/{id}`	حذف تقرير.
`GET`	`/v1/reports/{id}/entries`	تنفيذ التقرير وإرجاع الصفوف (ترشيح v1).
`GET`	`/v2/reports/{id}/entries`	تنفيذ التقرير وإرجاع الصفوف (معايير + بحث).
`GET`	`/v1/reports/{id}/entries/count`	عدّ الصفوف (where في v1).
`GET`	`/v2/reports/{id}/entries/count`	عدّ الصفوف (معايير + بحث).
`GET`	`/v2/reports/{id}/export/{format}`	تصدير الصفوف إلى ملف (`pdf` أو `xlsx`).

الإنشاء (`POST /v1/reports`) – جسم الطلب هو `ReportEntryDto` دون `id` (أي `NewReportEntry`): كائن `ReportEntryValue` مع الخاصية الاختيارية `permissions: Permission[]` . يستدعي مسار المعالجة `validateNameUniqueness` ثم `validateReportValue` ، ويزيل `permissions` ، ويرسل أمر `CreateDataEntryCommand` **على ناقل الخدمة** (غير متزامن، بنمط CQRS) لكيان Report، وينتظر `invokerService.registerToken(messageId)` للحصول على النتيجة المُنشأة، ثم يخزّن الأذونات منفصلةً في حقوق الوصول. ويتولّى مُعالج الأمر إنشاء صفّ التقرير، ولا ينشئه الطلب مباشرةً.

القراءة (`GET /v1/reports/{id}`) – يُرجِع التقرير المُخزَّن مع مصفوفة `permissions` **تُضَمّ وقت القراءة** من حقوق الوصول (`convertAccessRightToPermissions(Report, id)`)؛ فالأذونات ليست مُخزَّنة في مستند التقرير.

التحديث (`PATCH /v1/reports/{id}`) – جسم الطلب هو `ReportEntryDto` جزئي. بعد التحقّق، يُطبَّق **فحص أذونات النشر**: إذا كان `value.isPublished` **و** `authenticated.enabled` **و** `authenticated.isRestricted` **و** كانت `permissions` المُقدَّمة فارغة **و** لا توجد أذونات سابقة، أُطلِق الاستثناء `ReportPermissionsNotSelectedException` (HTTP 422، `REPORT_PERMISSION_NOT_SELECTED`). ثم يُحفَظ التقرير عبر `updateById` ؛ وتُوفَّق حقوق الوصول مع التحديث **قبل النشر** (والترتيب مهمّ، إذ تُعيد عملية مزامنة الاعتماديات جلب التقرير عبر `GET /v1/reports/{id}`): إذا كان `isRestricted === false`، تُمحى الأذونات المُخزَّنة؛ وإلا فتُخزَّن مجموعة الأذونات المُقدَّمة إذا كانت غير فارغة. وأخيرًا، يُنشَر `DataEntryUpdatedEvent` حاملًا علامة تغيّر الحقل `isPublished` إذا تغيّر.

تحديث الأذونات (`PATCH /v1/reports/{id}/permissions`) – يُمنَح التخويل عبر `systemAdminVoterFactory()` (لمسؤول النظام فقط). جسم الطلب هو `UpdateResourcePermissionsDto` (`permissions` + `authenticated`). ويطبّق المسار

فحص النشر نفسه، ويكتب كتلة `authenticated` الجديدة، وينشر `DataEntryUpdatedEvent`، ثم يوفّق حقوق الوصول مع التحديث.

الحذف (`DELETE /v1/reports/{id}`) – يُرسِل أمر `DeleteDataEntryCommand` على الناقل، وينتظر رمزه المميّز، ثم ينشر `ReportDeletedEvent`. فالحذف، كالإنشاء، يجري عبر الأوامر.

تنفيذ التقرير

تقبل **v1** (`/v1/reports/{id}/entries` ، `/v1/reports/{id}/entries/count`) مرشّحات LoopBack عبر `filter` / `where`. ويحدّد المتحكّم الكيان المرتبط من `report.value.relatedTo.dataEntry`، ثم يفرض الشروط `state = Active`، و`virtual != true` (ما لم يكن `include_virtual=true`)، و`dataEntity = <entity id>`، ويستدعي `dataSourceService.find(entity, report.value.fields, filter, count?, report.value.dataSource)`.

تقبل **v2** (`/v2/reports/{id}/entries` ، `/v2/reports/{id}/entries/count`) معايير مُهيكلة عبر `filterCriteria` / `whereCriteria` (شجرة `FilterExpression`)، إضافةً إلى معامل البحث النصّي الحرّ `search`. ويبني المتحكّم تعبير `AND` يفرض دائمًا `state = Active` و`virtual != true` (ما لم يكن `include_virtual`)، ويدمجه اختياريًا مع معايير المستدعي باستخدام العامل AND، ثم يفوّض التنفيذ إلى `dataSourceService.findV2` / `countV2`. وتمرّر نقطة نهاية المُدخلات `revealSensitiveFields = true`.

معاملات الاستعلام الشائعة:

المعامل	النوع	المعنى
`include_virtual`	boolean	تضمين الصفوف المُعلَّمة `virtual` (تُستبعَد افتراضيًا).
`search`	string (v2)	بحث نصّي حرّ في الحقول المُسقَطة.
`filterCriteria` / `whereCriteria`	object (v2)	تعبير ترشيح مُهيكل.

نُسخ التصدير

إطلاق التصدير من الواجهة

يبدأ المستخدمون النهائيون التصدير من قائمة إجراءات التقرير في تطبيق الويب، باختيار **التصدير إلى Excel** أو **التصدير إلى PDF**. ويستدعي كلا الخيارين نقطة نهاية التصدير أدناه.

نقطة نهاية التصدير

`GET /v2/reports/{id}/export/{format}` – يأخذ `format` قيمة من `ExportFormatEnum`: إمّا `pdf` أو `xlsx` (لا يتوفر تصدير CSV). ومعاملات الاستعلام هي: `filterCriteria` و`search` و`include_virtual` و`selectedColumns` (مصفوفة سلاسل). وتأتي الاستجابة على هيئة ملف للتنزيل، ويُؤخذ نوع المحتوى و`Content-Disposition: attachment;filename=...` من نتيجة المُصدِّر. ونوعا المحتوى المُعلَنان هما `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet` (XLSX) و`application/pdf`. و**Since 1.2** يحمل رابط التصدير `{format}` جزءًا من المسار.

نطاق الصيغ: لا تُصدِّر التقارير CSV أبدًا

ينتمي `format` في مسار التقرير إلى النوع `ExportFormatEnum` (`pdf` | `xlsx`)، وتقتصر أنواع المحتوى المُعلَنة على XLSX وPDF، كما أنّ المُصدِّرين المسجّلين الوحيدين هما مُصدِّرا XLSX وPDF. أمّا صيغة CSV التي أُضيفت في 1.10 فتخصّ **التصدير السريع** من المعرض ونظيره في الإدارة في صفحة الكيان (`GET /v3/entities/{entity}/views/{id}/entries/export` و`GET /v3/entities/{entity}/entries/export`، وكلاهما `format=xlsx|csv`)، وهما موثّقان في معارض البيانات والواجهات [بناء التطبيقات ‹ معارض البيانات وأوجهها]. ولا يؤثّر ذلك التغيير في التقارير؛ إذ تظلّ مقتصرة على XLSX / PDF ومتاحة على الويب فقط (لا تتوفر تقارير في تطبيق الجوال).

يحمّل `ReportService.exportReportV2` التقرير، ويشتقّ اسم الملف من قيمة `name` للتقرير، مُترجَمةً بلغة المستدعي، ويبني مرشّح `AND` نفسه المستخدم في تنفيذ v2، بالشروط `state = Active` و `virtual != true` الاختياري، ويدمجه باستخدام AND مع أيّ `filterCriteria` . ثم يختار الأعمدة، ويجلب الصفوف عبر `dataSourceService.findV2(..., revealSensitiveFields = true)`، ويستدعي `ExportService.export` .

اختيار الأعمدة وتنسيق القيم

تُحدَّد الأعمدة المُصدَّرة افتراضيًا من `report.value.fields` . إذا زُوِّد `selectedColumns` **وكان غير فارغ و** كانت قيمة `report.value.allowChangeFields` هي true، تُحدَّد الأعمدة المُصدَّرة من تقاطع `selectedColumns` مع `fields` الخاصة بالتقرير نفسه (فلا يمكن للتقرير أبدًا تصدير عمود لا يعرّفه). وإذا كانت قيمة `allowChangeFields` هي false، يُتجاهَل `selectedColumns` . **Since 1.2** (`selectedColumns` + `allowChangeFields`).

يحدّد `ExportService.export` المُصدِّر الفعلي عبر `ExporterFactoryService` (وتؤدي الصيغة غير المعروفة إلى `InvalidExportFormatException`). وقبل تسليم الصفوف، يُسطّح `convertDataEntryToExportableData` كل `DataEntry.value` إلى سلاسل جاهزة للعرض بحسب نوع الحقل:

نوع الحقل	عرض التصدير
ADT string / date / datetime / time	قيمة التجميعة بعد حلّها؛ التواريخ `dd/MM/yyyy` ، والتاريخ والوقت `dd/MM/yyyy hh:mm` ، والوقت `hh:mm` (وفق المنطقة الزمنية للمستخدم أو UTC).
Reference / User	`_resolved.mainTitle` (مُترجَم).
Localized text	القيمة المُترجَمة بلغة المستخدم.
DateTime	`dd/MM/yyyy hh:mm` .
Date	`dd/MM/yyyy` .
Date / DateTime range	طرفا النطاق مفصولان بـ `-` .
Time	`hh:mm` .
Month-Year	`MM/yyyy` .
Category	`_resolved.label` (مُترجَم).
Blob / file	قيمة `name` للملف.
Map	`JSON.stringify(geometry.coordinates)` .
Switch	`_resolved.mainTitle` (مُترجَم).
Password	مُقنَّع بـ `****` .
أخرى (icon، phone، …)	القيمة الخام.

وفي الحقول القابلة للتكرار / متعدّدة القيم، يُعرَض كل عنصر وفق القاعدة نفسها، وتُوصَل العناصر بـ `, ` (ويُوصَل تاريخا كل نطاق بـ `-`). ويشترك تنفيذ التقرير والتصدير في هذا المُحوِّل، لذا تطابق القيم المُصدَّرة ما تُرجِعه نقطة نهاية المُدخلات.

الحقول الحساسة. يطلب التنفيذ والتصدير قيم الحقول الحساسة (`revealSensitiveFields = true`)، لكن يظلّ كل حقل حسّاس مُقنَّعًا (`****`) لكل من لا يملك إذن الوصول إليه ضمن الحقول الحساسة [بناء التطبيقات › حماية حقل]، وتُقنَّع الحقول من نوع password دائمًا. ويُطبَّق التقنيع بالطريقة نفسها في الملف المُصدَّر، والتقرير المعروض على الشاشة، والمرفق المُرسَل بالبريد.

تصدير XLSX

يبني `ExportExcel` (الصيغة `xlsx` ، ExcelJS) ورقة عمل واحدة تحمل اسم التقرير/الكيان:

- تُستمدّ الأعمدة من حقول الكيان بعد ترشيحها للاقتصار على مجموعة حقول التقرير (`getAllFields(true)`)، ويبلغ عرض كلٍّ منها 20 وحدة، وتحمل الترويسة اسم الحقل المُترجَم.
- **حقول الكتلة** تُنتِج صف ترويسة فرعية مدموجًا يمتدّ عبر أعمدتها الفرعية. أمّا **الكتل القابلة للتكرار** فتُوسَّع أفقيًا؛ إذ تكرّر كل نسخة من الكتلة الأعمدة الفرعية (`<field>.<index>`)، ويُحدَّد حجم التوسّع وفق أقصى عدد من النُسخ في الصفوف المُصدَّرة.
- إذا وُجدت صورة **ترويسة منظمة** افتراضية، تُوضَع في مساحة مدموجة `A1:D1` ، وتبدأ البيانات من الصف 2.
- تُنسَّق خلايا الترويسة بخط عريض ومحاذاة وسطية وحدود رفيعة؛ وتُنسَّق خلايا الترويسة الفرعية بخط عريض وحدّ سفلي.
- يُرجِع `{ fileName: "<name>.xlsx", file: <ArrayBuffer>, mimeType: spreadsheetml.sheet }` .

لا يُفرَض حدّ لعدد الأعمدة في XLSX.

تصدير PDF

يستخدم `ExportPdf` (الصيغة `pdf`) محرّك `@osos/fluentreports` :

- **حدّ الأعمدة.** يُعرَض عدد من الأعمدة لا يتجاوز `MAX_COLUMNS_COUNT` (الافتراضي **7**)؛ وتُستبعَد عناصر `fields` بعد العمود السابع (`rptFields.slice(0, 7)`). وهذا حدّ صارم خاصّ بـ PDF فقط.
- تستخدم تسميات ترويسة الأعمدة اسم الحقل المُترجَم (تُفهرَس الأعمدة داخليًا بـ `#<normalizedName>` `<localizedName>`؛ وتُعرَض حقول الكتلة بصيغة `block.field`).
- **التخطيط** يتبع `design.layout` (`ReportLayoutEnum` : `Portrait = 0` ، `Landscape = 1`)، ويكون أفقيًا عندما يكون `layout === Landscape` .
- **أقسام الترويسة/التذييل** يحدّدها `ReportDesignOptions` : يجلب `organizationHeader` / `organizationFooter` صورتي `organization_header` / `organization_footer` للمنظمة الافتراضية؛ ويطبع `pageHeader` نص اسم التقرير/وصفه (الافتراضي `true`)؛ ويطبع `pageFooter` أرقام الصفحات `Page {0} of {1}` / `صفحة {0} من {1}` (الافتراضي `true`)، وتُطبَع حتى عند غياب `design`)؛ ويحمل `reportHeader` / `reportFooter` قيمة من نوع `ReportHeaderFooterValue` ، تكون قيمة `type` فيها إمّا `Text` (`0`) أو `Image` (`1`).
- يستخدم دعم **العربية/RTL** صورًا مُحاذاة إلى اليمين وخطّ Tajawal (المُسجَّل من `FONTS_PUBLIC_DIR` ، الافتراضي `public/fonts`).
- يُقتطَع نصّ القسم عند `SECTION_MAX_CHAR_COUNT` (الافتراضي 171)، ويُوزَّع على `SECTION_MAX_LINE_COUNT` (الافتراضي 3) أسطر.
- عند حدوث أيّ خطأ في التصيير، يسجّل المُصدِّر الخطأ ويُطلق `DataExportException` .
- يُرجِع `{ file: <Buffer>, fileName: "<name>.pdf", mimeType: "application/pdf" }` .

تضبط خيارات `PDFExporterOptions` المُمرَّرة إلى المُصدِّر القيمتين `pdfDesign = report.design` و`pdfHeaderText = "<report name> \n <description>"`، مع تمرير `pdfDesignConfigs` فارغًا؛ فيرجع المُصدِّر عندئذٍ إلى قيمه الافتراضية المُدمَجة.

الجدولة والإشعارات

قد يتضمّن التقرير إعدادات تسليم تُخزَّن فيه، لكن تتولّى تنفيذها أنظمة فرعية أخرى (خدمات المؤقّت/الإشعار)، وليست نقاط النهاية أعلاه.

- **`schedule`** (`ReportScheduleModel` يرث من `TimerOptionsEventConfig`): `enabled: boolean`، إضافةً إلى `oneTime?: string` و`repeat?` (كائن `DataEntityTriggerRepeat` يتضمّن `expression` بصيغة cron أو بصيغة مقروءة بشريًا، و`timezone` / `startDate` / `endDate` اختيارية). يُرفَض عند الحفظ أيّ تقرير منشور ذي جدولة لمرة واحدة تكون فيها `enabled` مفعّلة ويقع موعدها في الماضي (انظر التحقّق).
- **`emailNotification`** (`SendReportEmailActionOptions`): قوائم المستلمين `to` / `cc` / `bcc`، و`template` (UUID أو `EmailTemplateValue` مضمَّن)، و`tokensMap`، و`link` اختياري، و`attachment` / `attachments`، وإعداد `from` اختياري.
- **`inAppNotification`** (`SendReportNotificationActionOptions`): `channels`، و`recipients`، و`module` / `notificationType` اختياريان، و`template` (UUID أو `NotificationTemplateValue` مضمَّن)، و`tokensMap`.

التسليم المجدوَل والمرفق – تصحيح `Since 1.8`

عند إطلاق مهمّة مؤقّت التقرير `schedule`، يُحمِّل المُعالِج `ScheduledReportJobTriggeredHandler` التابع لـ `data-service` التقريرَ، ولا يرسل البريد الإلكتروني المُهيَّأ إلا عندما يكون كلٌّ من `isPublished` و`schedule.enabled` مفعّلًا. ويُعدّ `emailNotification.attachment.enabled` و`emailNotification.link.enabled` مفتاحين مستقلّين؛ والتهيئة الشائعة هي إرسال رابط فقط مع تعطيل المرفق.

قبل هذا الإصلاح، كان المُعالِج يقرأ نتيجة رفع المرفق دون شرط، حتى عندما تكون قيمة `attachment.enabled` هي `false`، فتؤدي القراءة إلى خطأ ويُتخطّى إرسال البريد بصمت. وكان try/catch المحيط يعيد رمي الاستثناء إلى مستهلك ناقل الرسائل دون إظهار أيّ واجهة للمستخدم، ثم يعيد الناقل المحاولة قبل أن يتوقّف عنها. وأصبح تصدير المرفق/رفعه الآن **بأفضل جهد**: يُنفَّذ ضمن try/catch خاص به، ويؤدي انقطاع التخزين أو فشل التصيير أو تعطيل المرفق إلى إرسال البريد **دون** مرفق بدلًا من إسقاط الإشعار. ويظلّ الفشل مُسجَّلًا كتحذير مُهيكَل (`{reportId, err}`)، مما يتيح تشخيص فشل المرفق لكل تقرير.

مرجع الأخطاء

الاستثناء	الرمز	HTTP	السبب
`ReportDataSourceSupportedException`	`REPORT_DATA_SOURCE_NOT_SUPPORTED`	4xx (تحقّق)	`dataSource.type` ليس `Query`.
`ReportFieldsLengthException`	`REPORT_FIELDS_LENGTH`	4xx (تحقّق)	`fields` فارغة.
`ReportFieldsNotExistException`	`REPORT_FIELDS_NOT_EXIST`	4xx (تحقّق)	حقل أو أكثر من `fields` غير موجود في الكيان.
`ReportSendingTimeCanNotBeInThePastException`	`REPORT_SENDING_TIME`	4xx (تحقّق)	تقرير منشور مُجدوَل للإرسال مرة واحدة في موعد يقع في الماضي.

الاستثناء	الرمز	HTTP	السبب
`ReportExistException`	`REPORT_EXISTS`	4xx (تحقّق)	الاسم يتعارض مع تقرير آخر ضمن الكيان/اللغة نفسهما.
`ReportPermissionsNotSelectedException`	`REPORT_PERMISSION_NOT_SELECTED`	422	نشر تقرير مُقيَّد دون اختيار أذونات.
`DataQueryNotFound`	–	4xx	`dataSource.query` غير موجود.
`DataQueryNotBelongToEntity`	–	4xx	كيان الاستعلام ≠ `relatedTo.dataEntry` .
`EntryNotExistException`	–	4xx	كيان `relatedTo.dataEntry` مفقود.
`EntityArchivedException`	–	4xx	الكيان المرتبط مؤرشف.
`InvalidExportFormatException`	–	4xx	لا يوجد مُصدِّر مربوط بصيغة التصدير `format` .
`DataExportException`	–	5xx	فشل تصيير XLSX/PDF.

الأمور الداخلية والثوابت

- **مصادر Query فقط.** يجلب تنفيذ التقرير الصفوف دائمًا عبر استعلام بيانات، وتُرفَض مصادر store/collection؛ و `ReportDataSourceSupportedException` هو الفحص الوحيد لذلك.
- **الحقول هي عقد الأعمدة.** تحدّد `fields` ، مع مراعاة ترتيبها، كلًا من إسقاط الحقول عند التنفيذ وأعمدة التصدير. ويقتصر PDF بصمت على 7 أعمدة؛ ولا يُفرَض حدّ مماثل في XLSX.
- **الأذونات في حقوق الوصول، لا على المستند.** تُضمَّن الأذونات عند القراءة، وتُسوَّى عند الكتابة بصورة منفصلة ومقصودة *قبل* النشر، حتى تحصل إعادة الجلب أثناء مزامنة الاعتماديات على المجموعة الجديدة.
- **CQRS للإنشاء/الحذف.** يمرّ الإنشاء والحذف عبر أوامر ناقل الخدمة (`CreateDataEntryCommand` / `DeleteDataEntryCommand`) وينتظران رمز الاستدعاء المميّز؛ أمّا التحديث فيكتب إلى المستودع مباشرةً ويُصدر `DataEntryUpdatedEvent` .
- **الحقول الحساسة مكشوفة عند التنفيذ/التصدير ثم مُقنَّعة حسب الإذن.** يمرّر تنفيذ v2 والتصدير `revealSensitiveFields = true` ؛ ويظلّ كل حقل حسّاس مُقنَّعًا (`****`) لكل من لا يملك إذن الوصول إليه، وتُقنَّع الحقول من نوع password دائمًا.
- **معالجة `virtual` القديمة.** تستخدم المرشّحات `virtual != true` (وليس `virtual == false`) لإرجاع الصفوف الموجودة مسبقًا التي تفتقر إلى الحقل.

التسلسلات ذاتية الزيادة وقطع العدّادات: البنية الداخلية والواجهة البرمجية

تخاطب هذه الصفحة **المُكامِلين ومهندسي المنصة** الذين يحتاجون إلى فهم دقيق لكيفية تخصيص أرقام التسلسل وتخزينها، وما يفرضه الخادم عند حفظ كيان يحتوي على حقل تسلسل، وما تُعيده نقطة نهاية العدّادات.

> **تضبط الترقيم أو عدّادًا من الواجهة؟**
>
> ينبغي لبُناة التطبيقات البدء من العدّادات وأرقام المُدخلات [بناء التطبيقات].

قطع العدّادات مقابل التسلسلات

يسهل الخلط بين الميزتين لاشتراكهما في الاسم، غير أنّهما مستقلّتان تمامًا في الرماز وتخدمان أغراضًا مختلفة:

	قطعة العدّاد	تسلسل الزيادة التلقائية
الغرض	عرض رقم *ملخّص* حيّ (عدد المُدخلات المطابقة)	تعيين رقم *متسلسل* فريد لكل مُدخل جديد
مبنيّ على	معرض، مع ترشيح اختياري باستعلام محفوظ	نموذج البيانات `CounterSequence` / `Counters`
مكانه	الواجهة (الشريط العلوي، لوحات المعلومات، الصفحات)	`autoIncrementOptions` لحقل في كيان البيانات
تغيّره بمرور الوقت	يُعاد حسابه مع تغيّر البيانات	يُخصَّص مرّة واحدة عند إنشاء المُدخل

تعدّ قطعة العدّاد مُدخلات **معرض بيانات** مرتبط بها، مع ترشيحها اختياريًا **باستعلام محفوظ** (`DataQuery`) – ويستند معيار العدّ فيها إلى استعلام، لا إلى نموذج `StatisticQuery` المنفصل الذي يدعم التجميع وتستخدمه لوحة المعلومات في بطاقات الرقم/الخلاصة.

بدءًا من `Since 1.4`، أصبحت قائمة قطع العدّادات لدى المستخدم النهائي تقرأ القطع المرشّحة أولًا، ثم تفحص معارضها وحدها، بدلًا من تحميل جميع المعارض مسبقًا؛ وبذلك لم تعد تتباطأ مع تراكم المعارض لدى القاطن (وأُتيح هذا التغيير أيضًا ضمن تحديث لخطوط 1.4 و1.6 و1.8). انظر لوحات المعلومات والخلاصات وقطع العدّادات [بناء التطبيقات › الخلاصات ولوحات المعلومات والعدّادات] للاطّلاع على قواعد إدراج القطع في القائمة.

كيف تعمل تسلسلات الزيادة التلقائية

تقوم ميزة الزيادة التلقائية على عنصرين:

- **خيار الزيادة التلقائية** على حقل في كيان البيانات (`options.autoIncrementOptions` ، النموذج `FieldOptionsAutoIncrement`)، وهو يحدّد أنّ الحقل يستمدّ قيمته من عدّاد، ويحدّد اسم العدّاد المراد استخدامه.
- **عدّاد** (`Counters`) إضافةً إلى **تسلسل عدّاد** واحد أو أكثر (`CounterSequence`)، تُخزَّن جميعها في قاعدة بيانات data-service، وتحمل الإعدادات والقيمة الحالية أو القيم الحالية للتسلسل.

الرقم الناتج عدد صحيح بسيط، ولا يُخزَّن على العدّاد نفسه أيّ قناع للبادئة أو اللاحقة أو الحشو – فالتنسيق (الحشو بالأصفار، البادئات) ليس جزءًا من نموذج العدّاد.

وتجري العملية كاملةً على النحو الآتي:

1. يحتوي حقل في كيان البيانات على `options.autoIncrementOptions`. وعندما تكون قيمة `enabled` هي `true`، يكون الحقل حقل زيادة تلقائية.
2. عند **إنشاء الكيان أو تحديثه**، يوفّق `DataEntityService` مستند `Counters` الذي يستند إليه كل حقل مُفعَّل مع إعدادات الحقل (إنشاءً أو ترحيلًا أو تحديثًا – انظر دورة حياة العدّاد وقت حفظ الكيان). أمّا الإعدادات الخاصة بكل عدّاد (`startAt`، `incrementBy`، `sharingLevel`) فـ**تُنسَخ إلى العدّاد وتُزال من خيارات الحقل**.
3. عند **إنشاء مدخل بيانات**، تعثر `DataEntryWriterService.handleAutoIncrementFields` على كل حقل زيادة تلقائية مُفعَّل، وتخصّص القيمة التالية لكلّ منها (بالتوازي)، وتكتبها في `entry.value.<fullNormalizedName>`.
4. يمرّ التخصيص عبر `SequenceService.getNextValue`، الذي يعثر على العدّاد، ويُقيّم أيّ مفاتيح شرطية، ثم يزيد قيمة `CounterSequence` المطابق ذريًّا عبر `CounterSequenceRepository.getNextValue`.

تُخصَّص قيم الزيادة التلقائية **عند الإنشاء فقط** – فلا يُعاد تخصيصها عند التحديث أو التعديل الجزئي. (تُستدعى `handleAutoIncrementFields` من مسار الإنشاء ومن مسار الإنشاء الدفعي؛ ولا تُستدعى من `updateDataEntry` أو `patchDataEntry`.)

سلوك الفشل

تُحيط `handleAutoIncrementFields` عملياتها بكتلة `try/catch` **يسجّل أيّ خطأ ويبتلعه**. فإذا فشل تخصيص قيمة العدّاد، يُنشأ المُدخل رغم ذلك، ويُترك الحقل دون رقم مُولَّد. لذا لا تعوّل على فشل إنشاء المُدخل عند تهيئة العدّاد تهيئةً غير صحيحة.

وداخل المستودع، إذا رمى التحديث الذرّي استثناءً، يُعاد رميه على هيئة `GeneratingCounterSequenceException` (`GENERATING_COUNTER_SEQUENCE_ERROR`)، غير أنّه يصل في النهاية إلى كتلة catch المذكورة أعلاه، التي تبتلعه.

نموذج بيانات التسلسل

`Counters` (المجموعة `_counters`)

يوجد مستند واحد لكل عدّاد، ومعرّف المستند `id` **هو مفتاح العدّاد** – أي النصّ نفسه الذي يشير إليه `autoIncrementOptions.key` الخاص بالحقل. يُخزَّن المستند في مجموعة MongoDB المسمّاة `_counters`.

الحقل	النوع	الافتراضي	ملاحظات
`id`	string	–	المفتاح الأساسي؛ لا يُولَّد. وهو *مفتاح* العدّاد.
`lastValue`	number	–	**مهملة (Deprecated)**. أصبحت القيمة الحالية محفوظة في `CounterSequence`، لا هنا.
`startAt`	number	`1`	أوّل قيمة تُعاد عندما لا يكون التسلسل قد استُخدم بعد.
`incrementBy`	number	`1`	المقدار المُضاف عند كل تخصيص لاحق.
`sharingLevel`	number (`AutoIncrementSharingLevelEnum`)	`0` (None)	انظر مستويات التشاركية.
`referencedBy`	`CountersReferencedBy[]`	–	حقول الكيانات التي تشير إلى هذا العدّاد (رابط تهيئة).
`usedBy`	`CountersUsedBy[]`	–	حقول الكيانات التي استهلكت قيمة فعلًا، وهي التي تتحكّم في تغييرات التهيئة.
`conditionalIncrement`	boolean	`false`	عندما تكون قيمته `true`، يُشغّل العدّاد عدّة مسارات مستقلّة تميّزها `keys`.
`keys`	`CountersKey[]`	`[]`	تعريفات المفاتيح المستخدمة لاختيار مسار التسلسل أو تقسيمه.

CountersReferencedBy (نموذج فرعي): `module` ، `entity` ، `field` – وكلّها معرّفات UUID تحدّد الحقل الذي يشير إلى هذا العدّاد.

CountersUsedBy (نموذج فرعي): سلسلة نصية واحدة هي `entityField` ، تُضبط عند أوّل استخدام فعلي للعدّاد لإنتاج قيمة لحقل. وصيغتها `<entityId>_<field.normalizedName>` (مثلًا `xyz-uuid_requestNumber`). ولا تتغيّر بعد ضبطها. ويمنع وجود مُدخلات في `usedBy` بعض تغييرات التهيئة (انظر قواعد دورة الحياة).

CounterSequence (المجموعة _counters_sequence)

يحمل القيمة الجارية الفعلية للعدّاد (أو قيمه الجارية). ويمكن أن توجد مستندات تسلسل **كثيرة** لكل عدّاد عندما يكون `conditionalIncrement` مُفعَّلًا – مستند واحد لكل تركيبة متميّزة من قيم المفاتيح. ويُخزَّن كل مستند في مجموعة MongoDB المسمّاة `_counters_sequence` .

الحقل	النوع	ملاحظات
`id`	string	معرّف UUID عشوائي (v4)، يُولَّد عند أوّل عملية upsert.
`counterId`	string	معرّف `id` للعدّاد المالك.
`lastValue`	number	آخر قيمة صادرة لهذا المسار.
`[key: string]: any`	–	أعمدة مفاتيح ديناميكية. عندما يكون العدّاد شرطيًا، تُضاف خاصيّة واحدة لكل `CountersKey.keyName` ، تحمل القيمة المُقيَّمة لذلك المفتاح، وتُستخدم لمطابقة المسار الصحيح.

CountersKey

يعرّف مفتاحًا واحدًا يقسّم عدّادًا شرطيًا إلى مسارات مستقلّة.

الحقل	النوع	ملاحظات
`keyName`	string	فريد؛ يُتحقّق من مطابقته لـ `NormalizedNameValidation` (الحدّ الأدنى والأقصى للطول + نمط الاسم المعياري). يصبح اسم خاصيّة في `CounterSequence` .
`calculationMethod`	number (`TokenSourceType`)	كيفية تفسير `value` . القيمة الافتراضية في المخطّط هي `Expression` . **يُقبل `Expression` فقط.**
`value`	string	التعبير المراد تقييمه (لطريقة `Expression`).

قيم `TokenSourceType` : `Fixed = 0` ، `Field = 1` ، `Expression = 2` ، `Pointer = 3` . **يجب أن تكون جميع مفاتيح العدّادات من نوع `Expression` .** وأيّ مفتاح تكون قيمة `calculationMethod` فيه غير `Expression` يتسبّب في رمي `UnsupportedConditionalIncrementCalculationMethod` (`UNSUPPORTED_CONDITIONAL_INCREMENT_CALCULATION_METHOD`) عند حفظ الكيان.

FieldOptionsAutoIncrement (خيار الحقل)

يوجد ضمن حقل في كيان البيانات، في `options.autoIncrementOptions` .

الحقل	النوع	الافتراضي	ملاحظات
`enabled`	boolean	– (مطلوب)	يحدّد أنّ الحقل حقل زيادة تلقائية. تُرجع `FieldService.isAutoIncrementField` القيمة `!!options.autoIncrementOptions?.enabled` .
`key`	string	– (مطلوب)	مفتاح العدّاد (`Counters.id ==`) المراد تخصيص القيم منه.
`sharingLevel`	number	`0`	يُنسخ إلى العدّاد، ثم يُزال من خيارات الحقل عند الحفظ.
`startAt`	number	`1`	يُنسخ إلى العدّاد، ثم يُزال من خيارات الحقل.

الحقل	النوع	الافتراضي	ملاحظات
`incrementBy`	number	`1`	يُنسخ إلى العدّاد، ثم يُزال من خيارات الحقل.
`conditionalIncrement`	boolean	`false`	يُنسخ إلى العدّاد ويُحتفظ به على الحقل.
`keys`	`CountersKey[]`	`[]`	تُنسخ إلى العدّاد ويُحتفظ بها على الحقل.

بعد حفظ الكيان، يُعاد بناء `autoIncrementOptions` للحقل للاحتفاظ **فقط** بـ `enabled` و `key` و `conditionalIncrement` و `keys` – أمّا الإعدادات العددية (`sharingLevel` ، `startAt` ، `incrementBy`) فمرجعها العدّاد، لا الحقل.

مستويات التشاركية

`AutoIncrementSharingLevelEnum` :

القيمة	الاسم	المعنى
`0`	`None`	غير مشترَك.
`1`	`Module`	مشترَك على مستوى الوحدة.
`2`	`System`	مشترَك على مستوى النظام.

يُخزَّن `sharingLevel` على العدّاد، وقيمته الافتراضية `None` . ملاحظة: مشاركة عدّاد واحد بين عدّة حقول ليست مُطبَّقة بالكامل في الرماز – إذ يعتمد إنشاء العدّاد حصرًا على `key` ، وهناك `TODO` في الرماز يشير إلى أنّ إنشاء عدّاد مفتاحه موجود مسبقًا يتسبّب حاليًا في رمي استثناء، وأنّ المشاركة «ينبغي التعامل معها لاحقًا». لذا تعامَل مع المشاركة بين الحقول بوصفها جزئية/غير مكتملة.

تخصيص القيمة التالية

`SequenceService.getNextValue(field, dataEntry)` :

1. يقرأ `key = field.options.autoIncrementOptions?.key` . وإن غاب، يسجّل خطأً ويُرجع `undefined` .
2. يحمّل العدّاد بواسطة `countersRepository.findById(key)` . وإن لم يُعثر عليه، يسجّل ذلك ويُرجع `undefined` .
3. يستدعي `evaluateConditionalKeys(counter.keys, dataEntry)` – ولكل مفتاح يتحقّق فيه الشرط `calculationMethod === Expression` ، يستبدل `key.value` بنتيجة `ExpressionService.evaluateExpression(key.value, null, dataEntry)` . وبذلك يحوّل التعبير المخزَّن إلى القيمة الفعلية الخاصة بـ *هذا* المُدخل.
4. يستدعي `counterSequenceRepository.getNextValue(counter)` للحصول على الرقم التالي بعملية ذرّية.
5. يسجّل الاستخدام: تُضيف عملية `$addToSet` على العدّاد `{ entityField: '<dataEntry.dataEntity>_<field.normalizedName>' }` إلى `usedBy` ، فيُسجَّل على العدّاد أنّ هذا الحقل استهلك قيمة منه.
6. يُرجع القيمة العددية.

وتنفّذ `CounterSequenceRepository.getNextValue(counter)` الزيادة الذرّية:

- إذا كانت `counter.conditionalIncrement` و `counter.keys` موجودتين، يبني كائن `keys` الذي يربط كل مفتاح بقيمته وفق `keyName -> key.value` (القيمة المُقيَّمة مسبقًا). وإلا فإنّ `keys` يكون فارغًا.

- يُنفّذ `findOneAndUpdate` في MongoDB على `_counters_sequence` على النحو الآتي:
  - **المُرشّح (filter)**: `{ counterId: counter.id, ...keys }` – يطابق المسار الخاص بهذا العدّاد وبهذه التركيبة الدقيقة من المفاتيح.
  - **التحديث (update)** (بأسلوب مسار التجميع `$set`):
    - `_id`: القيمة `_id` الموجودة أو معرّف UUID v4 جديد (في حالة upsert).
    - `counterId`: القيمة الموجودة أو `counter.id`.
    - حقول `keys`.
    - `lastValue`: إذا كانت قيمة `lastValue` الحالية هي null ← `counter.startAt`؛ وإلا `lastValue + counter.incrementBy`.
  - **الخيارات (options)**: `{ upsert: true, returnDocument: 'after' }`.
- يُرجع `result.value.lastValue`.

ولأنّ القراءة والتعديل والكتابة تُنفَّذ بعملية `findOneAndUpdate` ذرّية واحدة محصورة في `(counterId + key values)` (معرّف العدّاد مع قيم المفاتيح)، فلا يمكن لعمليات إنشاء المُدخلات المتزامنة أن تُصدر الرقم نفسه مرتين للمسار نفسه. فأوّل تخصيص لمسار يُنتج `startAt` بالضبط، وكلّ تخصيص لاحق يضيف `incrementBy`.

وعند حدوث أيّ خطأ، يسجّله المستودع ويرمي `GeneratingCounterSequenceException`.

مثال محلول

العدّاد `requestNumber` غير شرطي، وقيمة `startAt = 100`، وقيمة `incrementBy = 5`:

رقم الإنشاء	`_counters_sequence.lastValue`	القيمة المكتوبة في المدخل
الأوّل	(لا شيء) ← `100`	`100`
الثاني	`100 + 5` ← `105`	`105`
الثالث	`105 + 5` ← `110`	`110`

ومع `conditionalIncrement` ومفتاح `year` (تعبير تُقيَّم نتيجته إلى سنة المُدخل)، يكون لكل سنة متميّزة مستند تسلسل خاص بها وتسلسل ترقيم مستقل يبدأ عند `startAt`.

التسلسل الشرطي (متعدّد المسارات)

يتيح تفعيل `conditionalIncrement` مع مفتاح واحد أو أكثر في `keys` لعدّاد واحد أن يحتفظ بـ **مسارات ترقيم مستقلّة**. ويُعرَّف كل مسار بتركيبة قيم المفاتيح المُقيَّمة، ولكلّ مسار مستند `CounterSequence` خاص به. ومن الاستخدامات الشائعة: إعادة بدء الترقيم أو تقسيمه لكل سنة، أو لكل قسم، أو لكل فرع، إلخ، إذ يستنتج تعبير المفتاح التقسيم من المُدخل الجاري إنشاؤه.

القواعد المفروضة عند حفظ الكيان (`validateConditionalIncrement`):

- يجب أن يكون كل `CountersKey.calculationMethod` **من نوع** `Expression`؛ وإلا يُرمى `UnsupportedConditionalIncrementCalculationMethod`.
- إذا كان العدّاد قد استُخدم سابقًا (`usedBy` غير فارغ)، فإنّك **لا تستطيع** تغيير `conditionalIncrement` أو مجموعة `keys` – وإلا يُرمى `CanNotChangeConditionalIncrementConfigurations`

(`CAN_NOT_CHANGE_CONDITIONAL_INCREMENT_CONFIGURATIONS`). وبذلك تصبح التهيئة الشرطية ثابتة فعليًا بمجرّد إصدار الأرقام.

العلاقة مع التفرّد: يُشترط في حقل الزيادة التلقائية العادي أن تكون قيمة `options.unique = true` . وعندما يكون `conditionalIncrement` مُفعَّلًا (في الخيارات الواردة أو على العدّاد الموجود)، يُضبط الحقل على `options.unique = false` — لأنّ الرقم نفسه يمكن أن يتكرّر على نحو مشروع في مسارات مختلفة.

دورة حياة العدّاد وقت حفظ الكيان

تُستدعى `DataEntityService.handleAutoIncrementField(field, dataEntity)` لكل حقل تكون فيه `autoIncrementOptions.enabled` مفعّلة عند إنشاء الكيان أو تحديثه. وتبحث عن العدّاد الموجود باستخدام `referencedBy.field === field.id` ، ثم تتصرف على النحو الآتي:

- **لا يوجد عدّاد ←** أنشئ `Counters` جديدًا بقيمة `id = key` ، وقيم `sharingLevel` / `startAt` / `incrementBy` (بالقيم الافتراضية `1` / `1` / `0`)، و `conditionalIncrement` ، و `keys` ، ومُدخل `referencedBy` واحد لهذا الحقل. (تجري عملية الإنشاء داخل `try/catch` يتجاهل الأخطاء؛ ووفقًا لـ `TODO` في الرماز، يتسبّب المفتاح المكرَّر حاليًا في رمي استثناء، ولم تُعالَج المشاركة بعد.)
- **عدّاد موجود، المفتاح تغيّر ←**
  - إن **لم** يكن حقل الكيان هذا قد استهلك العدّاد القديم، فأنشئ عدّادًا جديدًا للمفتاح الجديد، وإن كان هذا الحقل وحده يشير إلى العدّاد القديم (`referencedBy.length === 1` ، «نفاية»)، فاحذفه.
  - إن كان حقل الكيان هذا **قد** استهلك العدّاد القديم، فارم `FailToAddFieldException` ("can not update auto increment key after it is used"). **فمفتاح العدّاد غير قابل للتغيير بمجرّد استخدامه.**
- **عدّاد موجود، المفتاح نفسه ←** إن **لم** يكن العدّاد قد استُخدم بعد، فحدّث `incrementBy` / `startAt` / `sharingLevel` (عند توفيرها فقط)، و `conditionalIncrement` ، و `keys` ، وامسح `usedBy` . وإن كان قد استُخدم بالفعل، فيُترك الضبط كما هو (تُتخطّى الكتلة).

باختصار: تبقى `startAt` و `incrementBy` و `sharingLevel` قابلة للتعديل بحرّية حتى إصدار أوّل قيمة؛ وبعد ذلك يصبح ترقيم العدّاد ثابتًا فعليًا، ولا يمكن تغيير المفتاح مطلقًا.

واجهة API للتسلسلات

للعدّادات نقطة نهاية HTTP واحدة فقط، وهي **للقراءة فقط**؛ إذ تُنشأ العدّادات وتُعدَّل بوصف ذلك أثرًا جانبيًا لحفظ كيانات البيانات، وليس عبر واجهة كتابة مخصّصة.

`GET /v1/auto-increment-counters`

تسرد عدّادات الزيادة التلقائية، وهي معرَّفة في `CountersController.find` .

- **المصادقة (Auth):** تتطلّب المصادقة عبر استراتيجية `Auth2` (`@authenticate`)، والتفويض عبر `entityVoterFactory()` (`@authorize`).
- **الاستعلام (Query):** تقبل مُرشِّح `filter` قياسيًا من نوع LoopBack لـ `Counters` (`@param.filter(Counters)`)، مما يتيح الترشيح وتحديد حدّ لعدد النتائج واختيار الحقول والترتيب بحسب حقول العدّاد الموثّقة أعلاه.
- **يُرجع:** `Counters[]` (مستندات العدّاد الخام).

لا توجد نقطة نهاية لإنشاء عدّاد أو تحديثه أو حذفه في هذا المتحكّم، و**لا توجد** نقطة نهاية تعرض قيم `_counters_sequence` مباشرةً.

```
GET /v1/auto-increment-counters?filter={"where":{"conditionalIncrement":true}}
Authorization: Bearer <token>
```

ملاحظات للمهندسين

- **مجموعتان، مسؤولية مقسَّمة.** تخزّن `_counters` التهيئة، وتخزّن `_counters_sequence` القيمة الحالية أو القيم الحالية. أمّا `Counters.lastValue` فمهملة؛ فلا تقرأها ولا تكتبها بوصفها المرجع المعتمد.
- **الذرّية على مستوى التسلسل.** تضمن عملية upsert باستخدام `findOneAndUpdate` على `_counters_sequence`، والمقيّدة بـ `counterId` + قيم المفاتيح المُقيَّمة، سلامة التزامن. ويجب أن يحافظ أيّ تغيير هنا على تنفيذ تلك الزيادة بصورة ذرّية في اتصال واحد بقاعدة البيانات.
- **المفاتيح تُقيَّم لكل مدخل.** تستبدل `evaluateConditionalKeys` القيمة المخزّنة في الذاكرة `counter.keys[].value` بالنتيجة المُقيَّمة قبل بناء مُرشِّح المسار؛ فالعدّاد المخزَّن يحتفظ بـ *التعبير*، بينما يخزّن مستند التسلسل القيمة *المُقيَّمة*.
- **تتبّع الاستخدام يحكم التحرير.** يحدّد `usedBy` (على العدّاد) وفحص «النفاية» `referencedBy.length === 1` ما إذا كان تعديل التهيئة مسموحًا، وما إذا كان العدّاد اليتيم سيُحذف. وتأخذ سلسلة `usedBy` الصيغة `<entityId>_<field.normalizedName>`.
- **القيمة تُكتب بالاسم المعياري الكامل.** تُنفّذ `set(dataEntry, 'value.` `incrementField` `<field.fullNormalizedName>', nextValue)`. ويشير `TODO` في الرماز إلى أنّ القيمة لا تُحوَّل إلى سلسلة نصية في الحقول النصية.

التفاصيل الداخلية لوضع المعاينة

هذه الصفحة موجّهة إلى **المُكامِلين ومهندسي المنصة** الذين يحتاجون إلى معرفة كيفية عمل وضع المعاينة من وراء واجهة المصمّم: ترويسة الطلب، وكيف تتعامل معها الخدمات الخلفية، وكيف تُولَّد بيانات العيّنة، والعقد بين مصمّم الصفحات وعميل الجوال المضمَّن.

هل تعاين في المصمّم؟

إن كنت من بُناة التطبيقات، فابدأ بصفحة وضع المعاينة [بناء التطبيقات]؛ إذ تشرح المعاينتين كلتيهما باستخدام التسميات الظاهرة في المصمّمات.

آليتان

يشمل هذا الاسم آليتين. الأولى **وضع على مستوى الطلب**: يُوسَم الطلب بأنه طلب معاينة، فتعيد الخدمات التي تستجيب له بيانات مولَّدة تطابق بنية البيانات الحقيقية تمامًا بدلًا من القراءة من المخزن. والثانية، في مصمّم الصفحات، **معاينة جوال حيّة**: عميل الجوال الحقيقي مضمَّنًا في المصمّم، وتحرّكه الصفحة التي تحرّرها.

ويُولَّد محتوى العيّنة لواجهات البيانات الرئيسة: المُدخلات (القائمة، ولوحة المهام (Kanban)، والشكل الشجري، وتفاصيل المُدخل)، والمجموعات، والاستعلامات المحفوظة والمرشِّحات السريعة، وأزرار النظام، والسجل التاريخي للمُدخل وسجلات البريد، وحالة سير العمل، ولوحات المعلومات وإحصاءاتها، وخلاصات الشريط الزمني (المنشورات)، وسلاسل التعليقات.

المعاينات التي تطلب بيانات العيّنة

لا تطلب بيانات العيّنة إلا **لوحة المعاينة في مصمّم المعارض**. ففي الويب، توفّر اللوحة رمز الحقن `PREVIEW_MODE`، ولا تَسِم قراءاتها إلا المكوّنات المعروضة داخلها (`view-preview-panel.component.ts`، `dashboard-card.component.ts`، `entities-list.service.ts`)؛ ويطلب تبويب الجوال في مصمّم المعارض من العميل المضمَّن أن يفعل الشيء نفسه. أما **مصمّم الصفحات** (الصفحات / `GenericView`) فلا يفعل ذلك؛ إذ يعرض لوح التصميم فيه ومعاينته الحيّة على الجوال المُدخلات الحقيقية المحفوظة أصلًا. وفي عميل الجوال، يَسِم `PreviewModeService.isActive` نسخةَ المعاينة (حراسات الكتابة، واستثناءات الوقت الحقيقي)، بينما يحدّد `fakeDataHeaderEnabled`، ذو النطاق الأضيق، والذي لا تكون قيمته صحيحة إلا في معاينة DataView، ما إذا كان `PreviewModeInterceptor` يضيف `X-Preview-Mode: true` (`mobile-app/lib/core/services/preview/preview_mode_service.dart`).

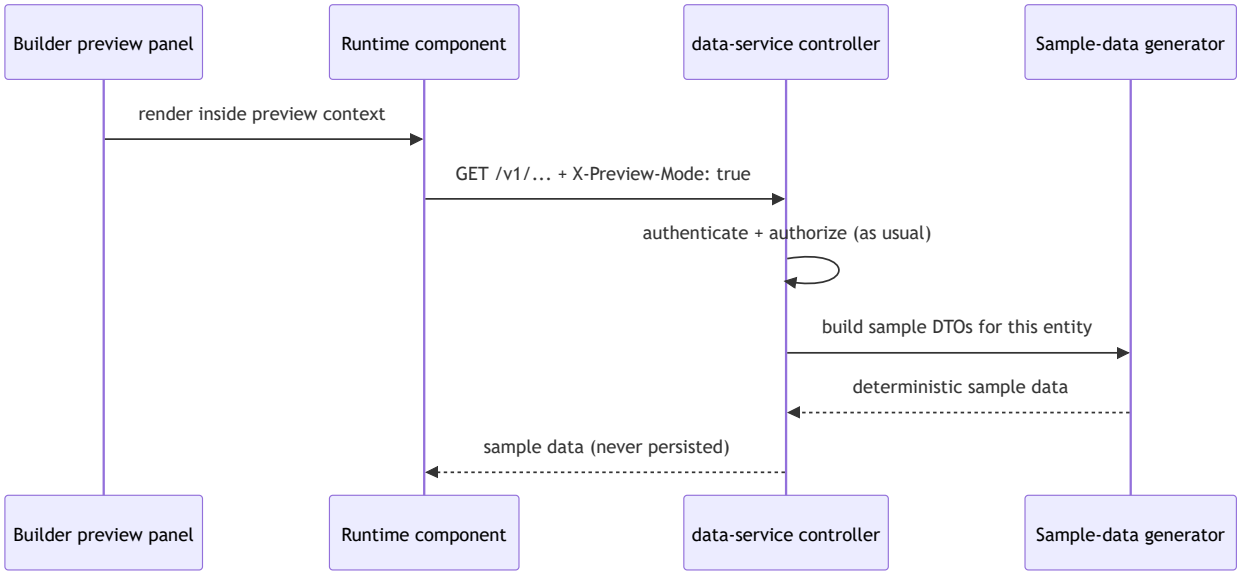
وسم الطلب بأنه معاينة

الترويسة `X-Preview-Mode`

المعاينة **مقيَّدة بنطاق الطلب**، وليست خاصية مخزَّنة في معرض أو صفحة. ويدخل الطلب في وضع المعاينة بحمل الترويسة `X-Preview-Mode: true` – ولا يتغيّر فيه شيء آخر: المسار نفسه، والمصادقة نفسها، ومعاملات الاستعلام نفسها. ولا تُعتَمد إلا القيمة `true` بالضبط (دون اعتبار لحالة الأحرف)؛ وأي قيمة أخرى، أو غياب الترويسة، يعني أن الطلب عادي.

ولا تُضبط الترويسة يدويًا في جانب الويب أبدًا. فلوحة المعاينة في المصمّم تَسِم نفسها بأنها سياق معاينة، وكل مكوّن تشغيلي يُعرض داخل تلك اللوحة يَسِم قراءاته، ويضيف معترِض HTTP واحد الترويسة إلى تلك الطلبات الموسومة وحدها. أما

حركة البوابة العادية فتبقى كما هي.



ما تخفّفه المعاينة – وما لا تخفّفه

وضع المعاينة **ليس** تجاوزًا للأذونات. فالطلب يظل مصادَقًا عليه، وتظل مصوِّتات التفويض نفسها التي تحرس نقاط النهاية الحقيقية فاعلة. وتبقى مستويات الوصول، والمعارض المقيَّدة، وقواعد الحقول الحسّاسة، كلها سارية.

ولا يوجد إلا تخفيف واحد محدود النطاق. فبعض المسارات تتحقّق من التفويض بتحميل المورد المستهدف بمعرّفه (معرض، أو استعلام محفوظ، أو مجموعة). وتُستخدم للإشارة إلى بيانات العيّنة **معرّفات مصطنعة** لا وجود لها في المخزن عمدًا، ويؤدّي تعذّر العثور على المورد إلى رفض الطلب قبل تشغيل المتحكّم أصلًا. لذا، في وضع المعاينة وحده، يُسمح بمرور حالة **عدم وجود حقيقية** في ذلك البحث، ليتمكّن المتحكّم من الاستجابة ببيانات العيّنة. أما أي إخفاق آخر – كخطأ في قاعدة البيانات – فلا يُخفى، ويظل إخفاقًا.

والنتيجة العملية أن المعاينة وسيلة لعرض التصميم، لا للتحقّق من البيانات؛ فلا يمكنها أن تعرض لك بيانات لا يحقّ لك رؤيتها، وليست وسيلة موثوقة للحكم على ما سيراه مستخدم نهائي مقيَّد الصلاحيات. تحقّق من ذلك باستخدام معرض عادي خارج وضع المعاينة بعد نشر التصميم؛ انظر الأذونات [الإدارة › الأذونات والوصول].

وثمة عنصر يُستثنى عمدًا من الاصطناع دائمًا: **تعريفات العلاقات**. فاسم العلاقة وكيانها المستهدف بيانات وصفية للمخطط، ولذلك يُعادان بقيمهما الحقيقية دائمًا. أما بيانات العيّنة فتقتصر على المُدخلات المعروضة تحت تبويب العلاقة.

بيانات العيّنة المولَّدة

يُنتج مولّد حتمي قيم العيّنة باستخدام بذرة مشتقّة من مفتاح ثابت – معرّف مُدخل العيّنة، أو الكيان والحقل، أو معرّف القطعة، وما إلى ذلك. فالمفتاح نفسه يعطي القيم نفسها في كل مرة. وهذا ما يحافظ على تطابق صف القائمة وصفحة تفاصيله، ويمنع تغيّر بيانات المعاينة بعد تعديل لا صلة له بها.

وحيثما توفّر للكيان إعداد حقيقي يمكن الاستناد إليه، يفضّله المولّد على الاصطناع: فـ**خيارات التصنيف** و**الوسوم** و**أعمدة لوحة المهام** تأتي من القيم الحقيقية المهيّأة في الكيان (ويقتصر الاصطناع على *أيّ* مُدخل عيّنة يحملها)، و**أزرار النظام** و**الاستعلامات المحفوظة** هي أزرار المعرض واستعلاماته الحقيقية مع استبدال معرّفاتها وحدها – فلا يمكن أبدًا تشغيل زر حقيقي على مُدخل عيّنة، ولا يترتّب على اختيار استعلام محفوظ من العيّنة أي إجراء.

القيم بحسب نوع الحقل

نوع الحقل	قيمة العيّنة
رقم	عدد صحيح عشوائي بين 1 و9999.
تاريخ	تاريخ في الماضي.
وقت	وقت حديث من اليوم.
تاريخ ووقت	طابع زمني حديث.
نعم/لا	قيمة منطقية عشوائية.
نص مترجَم	جملة لكل لغة مهيّأة.
موقع جغرافي (GeoJson)	معلَم GeoJSON حقيقي من نوع نقطة بإحداثيَّي طول وعرض عشوائيين، فتُعرض حقول الخرائط بدلًا من أن تتعطّل. `Since 1.8`
كائن / JSON	كائن فارغ – وهو الشكل الوحيد الذي يبقى متوافقًا مع مُصيِّر يكتب خصائص متداخلة في القيمة. `Since 1.8`
ملف	فارغ (بلا ملف).
نص (وأي نوع آخر)	نص يراعي المحتوى: عنوان بريد إلكتروني أو رقم هاتف أو URL أو اسم شخص حين يوحي اسم الحقل بذلك، وإلا فجملة من نص الحشو.

وثمة قاعدتان بنيويتان لا تقلّان أهمية عن القيم بحسب النوع:

- **تُصطنع الكتل حقلًا حقلًا**، مع الانتقال تعاوديًا إلى حقولها الفرعية. وتُملأ الكتلة القابلة للتكرار حتى الحدّ الأدنى المهيّأ لعدد تكراراتها، ويأخذ كل تكرار بذرته الخاصة كي لا تتطابق التكرارات. `Since 1.8`
- **حالة سير العمل** تُبنى من سير العمل الأول *الحقيقي* للكيان ومن معرّفات خطواته وترتيبها الحقيقية؛ ويقتصر الاصطناع على حالة التنفيذ، ويُترك مالك الخطوة فارغًا عمدًا كي لا يبدو أي عنصر في المعاينة قابلًا للتنفيذ.

منتقيات الحقول المرجعية

عند فتح منتقي علاقة أو منتقي مرجع مستخدم داخل المعاينة، يُعيد قائمة قصيرة من **مُدخلات العيّنة** للاختيار منها. `Since 1.8`

وكان ذلك يفشل سابقًا؛ إذ كان المنتقي يحاول حلّ المُدخل قيد التحرير بالرجوع إلى المخزن الحقيقي، في حين أن هذا المُدخل في المعاينة مصطنع وغير محفوظ – فتعود الاستجابة بالرسالة «entity not found» ويبقى المنتقي فارغًا. أما الآن فينتقل المنتقي مباشرةً إلى مُدخلات العيّنة بعد فحوص التحقّق من الحقل وتفويض المصدر المعتادة، وقبل أي بحث حقيقي.

لوحات المعلومات والإحصاءات وخلاصات الشريط الزمني

للكيان الجديد تمامًا لوحة معلومات تُنشأ تلقائيًا بلا **أي** قطعة، ولا منشورات له في الشريط الزمني؛ لذا كانت معاينة أيٍّ منهما تعرض شاشة فارغة دائمًا. أما الآن فيستخدم كلاهما محتوى عيّنة بديلًا. `Since 1.8`

- **لوحة المعلومات** – حين لا تحتوي لوحة المعلومات الحقيقية التي جرى حلّها للطلب على أي قطعة، تستبدل بها المعاينة لوحة عيّنة فيها قطعتان تجريبيتان. أما لوحة المعلومات التي *فيها* قطع مهيّأة فتحتفظ بتخطيطها الحقيقي؛ ويقتصر الاصطناع على الأرقام المجمَّعة.
- **الإحصاءات** – تُحلّ قطعة العيّنة إلى تعريف قطعة مصطنع (اسم، ونوع مخطّط، وتجميع)، مع صفوف تصنيف للعيّنة ومجموع. أما الاستعلام الإحصائي الحقيقي المهيّأ أصلًا فيحتفظ بمعايير مصدره الحقيقية بعد حلّها وبنطاق معرضه؛ ويقتصر الاصطناع على أرقامه.
- **الشريط الزمني** – تعرض المعاينة دائمًا خلاصة عيّنة صغيرة، سواء وُجدت منشورات حقيقية أم لا، مع تعبئة حالة النشر والتثبيت كي تعرض قطعة الشريط الزمني عناصر التحكّم فيها.

- **التعليقات** – سلسلة عيّنة قصيرة تتضمّن مرفق عيّنة واحدًا، وترتبط بالمُدخل قيد المعاينة. وتُعاد قائمة الإشارات والإكمال التلقائي فارغة.

معاينة الجوال

يتيح لك المصمّم أيضًا معاينة تصميمك كما يظهر على الهاتف، دون مغادرة المتصفح ودون الحاجة إلى جهاز. ففي مصمّم المعارض، تظهر المعاينة في تبويب **معاينة الجوال** بجانب معاينة الويب؛ وفي مصمّم الصفحات، تظهر في لوحة حيّة بجانب لوح التصميم.

> **ملاحظة:** تعرض معاينة الجوال عميل الجوال الحقيقي داخل المصمّم، لكنها تظل معاينة: فهي شاشة واحدة مضمَّنة، لا جلسة تطبيق كاملة. وبعض الواجهات متاحة على الويب فقط – **فالتقارير** مثلًا غير متاحة في عميل الجوال – لذا عاينها على الويب.

معاينة الجوال الحيّة في مصمّم الصفحات `Since 1.8`

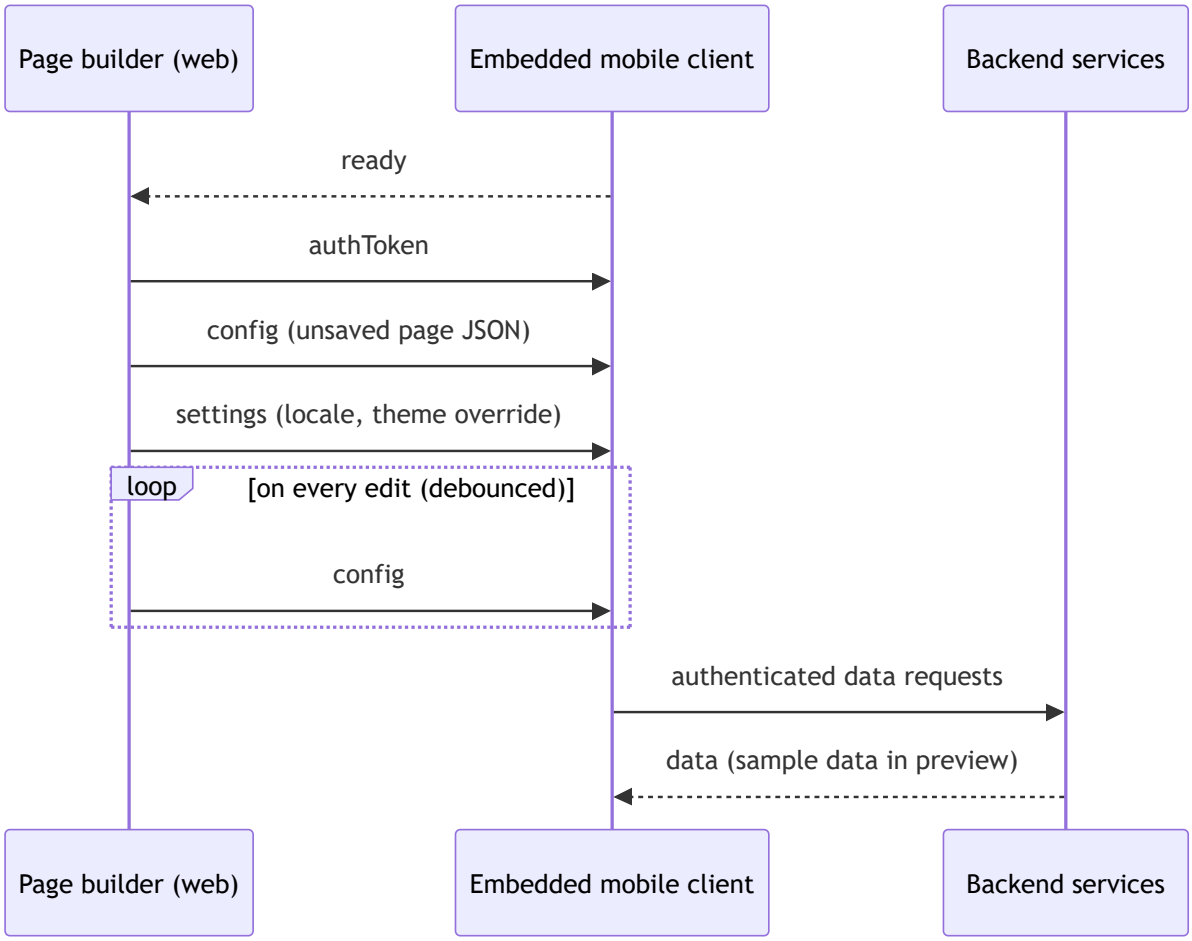
يستضيف تبويب **المحتوى** في مصمّم الصفحات معاينة جوال حيّة تضمّ عميل الجوال الفعلي مضمَّنًا في المصمّم، ويعرض الصفحة **التي تحرّرها الآن** – بما فيها التغييرات غير المحفوظة. ويُدفع إليه كل تعديل، فتواكب معاينة الجوال لوح التصميم دون حفظ أو إعادة تحميل.

ولأنه العميل الحقيقي، فما تراه هو التصيير الفعلي لقطعك على الجوال، لا محاكاة له في المتصفح: محرّك التخطيط نفسه، وتنفيذات القطع نفسها، والسمة نفسها. ويُجري العميل المضمَّن استدعاءاته الخاصة المصادَق عليها لجلب بيانات القطع. وفي معاينة الصفحة، **لا** تحمل هذه الاستدعاءات `X-Preview-Mode`، فتعرض القطع مُدخلات حقيقية محفوظة أصلًا (انظر المعاينات التي تطلب بيانات العيّنة)؛ ولا يطلب بيانات العيّنة إلا تبويب الجوال في مصمّم المعارض.

الجسر بين المصمّم والتطبيق

يتواصل المصمّم والعميل المضمَّن عبر عقد موجز لتبادل الرسائل ذي إصدار محدّد:

- **المصافحة** – يعلن العميل المضمَّن جاهزيته؛ فيردّ المصمّم برمز الوصول للجلسة، وإعداد الصفحة الحالي، وإعدادات العرض.
- **دفع الإعداد اللحظي** – بعد كل تعديل (مع تأخير زمني لتجنّب إغراق العميل بالتحديثات عند الكتابة المتتابعة)، يعيد المصمّم إرسال إعداد الصفحة. ويعيد العميل تطبيقه في مكانه بدلًا من إعادة التركيب، وبذلك يُحفَظ موضع التمرير وتبقى أي حالة محلية للنموذج بعد التعديل.
- **إعدادات العرض** – اللغة، إضافةً إلى تجاوز للسمة يتضمّن اللون الأساسي الحالي للبوابة، لأن نسخة المعاينة لا تجلب سمة خاصة بها؛ وإذا كان اللون غير صالح أو غائبًا، تبقى سمة الجوال كما هي.
- **إطار الجهاز** – إطار هاتف بأبعاد منطقية ثابتة، يُصغَّر بنسبة موحّدة ليلائم المساحة المتاحة ولا *يُكبَّر* أبدًا، فيظل واضحًا دون ضبابية.
- **عزل الأصل** – لا تُرسل الرسائل إلا إلى أصل مضيف المعاينة نفسه، لا إلى أي أصل عام، ويتحقّق العميل المضمَّن من أصل المرسل قبل قبول أي رسالة.



ما تعرضه معاينة الجوال

تركّب المعاينة **شاشة الصفحة الإنتاجية** في وضع المعاينة، فيطابق تصيير القطع تصييرها في التطبيق الحقيقي، مع ثلاثة فروق مقصودة: يُرسم شريط التطبيق (للحفاظ على التماثل البصري) لكن التنقّل عبره معطّل، ولا يوجد زر مساعدة، و**لا تُطلَق قواعد إعادة التوجيه أبدًا** – لأن إعادة التوجيه ستنقل العميل المضمَّن بعيدًا عن الصفحة نفسها التي تعاينها.

ويتسامح تحليل الإعداد مع الأخطاء على مستوى كل مكوّن، بدلًا من تطبيق مبدأ الكل أو لا شيء، لأن المصمّم يدفع باستمرار حالات لم تكتمل بعد:

- **تُسقط** القطعة ذات الإعداد المشوَّه، ويستمر عرض بقية الصفحة.
- تُستكمَل خيارات العرض الغائبة أو الجزئية بقيم افتراضية (عرض كامل، ودون فرض صف جديد، ومحاذاة تمديد).
- وإن تعذّر تحليل الإعداد المدفوع بأكمله، يُتجاهل و**يبقى آخر إعداد صالح على الشاشة** – فلا تصبح المعاينة فارغة أبدًا بسبب تعديل وسيط خاطئ.

صفوف الفرز أثناء التعديل الحيّ

لصفوف الفرز قاعدة خاصة: لا يُحتفَظ بالصف إلا إذا تضمّن حقلًا واتجاهًا **معًا**. أما الصف الذي يتضمّن أحدهما دون الآخر – وهي حالته مؤقتًا بعد إضافة صف فرز واختيار الحقل، وقبل تحديد الاتجاه التصاعدي أو التنازلي – **فيُسقط** وحده. `Since 1.8`

ولولا هذه القاعدة، لجعل ذلك الصف غير المكتمل تحديث الإعداد بأكمله غير قابل للتحليل، فيُهمَل التحديث المدفوع وتتجمّد المعاينة بصمت عند فرزها السابق بينما تواصل التحرير.

البقاء داخل المعاينة

المعاينة شاشة واحدة مضمَّنة بلا إطار تطبيق يحيط بها، لذا فإن أي تنقّل حقيقي سيتركها معلّقة بلا مخرج. وتُبقيها آليتا حراسة في مكانها: تُعترَض استدعاءات التنقّل دون تنفيذها، ويُعاد توجيه أي مسار ليس مسار معاينة إلى مضيف المعاينة.

وتطبّق آليتا الحراسة الاستثناء نفسه على **مسارات امتدادات المُدخل** – أي تبويبات النقاش وسير العمل والسجل التاريخي والمعلومات في شاشة تفاصيل المُدخل. `Since 1.8`

فهذه التبويبات مصمّمة لتُعرض *في مكانها*، متداخلةً داخل الصفحة المعايَنة. وقبل تطبيق الاستثناء على آليتي الحراسة كلتيهما، كان فتح أحدها داخل مُدخل معايَن يُطلق إعادة التوجيه، فيعود المسار إلى مضيف المعاينة وتصبح المعاينة كلها فارغة بدلًا من عرض التبويب.

إصلاحات دقّة المعاينة

تزيد عدة إصلاحات في 1.8 موثوقية عرض البيانات الحقيقية في وضع المعاينة – وهي تصحيحات للدقّة، لا قدرات جديدة. `Since 1.8`

في معاينة الجوال الحيّة في مصمّم الصفحات، كانت قطع **الخلاصة** و**الشريط الزمني** و**Timeline hero-slider** تواصل عرض البيانات التي جلبتها عند تركيبها أول مرة، بدلًا من مواكبة التعديلات اللاحقة. ولأن معاينة الجوال تُبقي شجرة القطع نفسها مركّبة عبر التعديلات الحيّة (انظر ما تعرضه معاينة الجوال)، فإن القطعة التي لا تُجلَب بياناتها إلا مرة واحدة عند التركيب لم تكن تكتشف أبدًا تغيير وجهة الصفحة إلى معرض أو كيان أو عنصر مختلف – فكانت قد تستمر، دون تنبيه، في عرض بيانات مرتبطة بمصدر استبدلته. أما الآن فتكتشف القطع الثلاث هذا النوع من التغيير وتفرض إعادة جلب البيانات.

وبصورة مستقلة عن ذلك، كانت معاينة الجوال الحيّة في مصمّم الصفحات قد تعرض بيانات مصطنعة فعلًا، لا بيانات قديمة فحسب؛ إذ كانت ترث سهوًا سلوك بيانات العيّنة الخاص بمصمّم المعارض، فيولّد الخادم لها صفوفًا اصطناعية تمامًا كما يفعل لمعرض غير محفوظ. أما الآن فيفصل عميل الجوال بين الحالتين عند تحديد نوع البيانات المطلوبة: فلا تطلب البيانات المصطنعة إلا معاينة الجوال في مصمّم المعارض – والغرض منها عرض صفوف اصطناعية لمعرض غير محفوظ ما زال قيد الإعداد – أما المعاينة الحيّة في مصمّم الصفحات فتعكس دائمًا المحتوى الحقيقي المحفوظ أصلًا الذي يعرضه لوح التصميم.

وشمل لوحةَ المعاينة في مصمّم الصفحات إصلاحان أصغر: فهي تضمّن الآن معرّف الصفحة الحقيقي في الإعداد الذي تدفعه إلى معاينة الجوال، فتُعالَج الطلبات المقيَّدة بالصفحة (كخلاصة قطعة الشريط الزمني) ضمن نطاق الصفحة الفعلية بدلًا من الرجوع إلى طلب غير مقيَّد بنطاق؛ وأصبحت تتضمّن أيضًا أيقونة التوسيع إلى نافذة حوار نفسها التي كانت متاحة لمعاينة الجوال في مصمّم المعارض.

التوفّر

القدرة	التوفّر
وضع المعاينة على مستوى الطلب (`X-Preview-Mode`)	Since 1.8
مُدخلات ومجموعات واستعلامات وأزرار وسجل تاريخي عيّنة حتمية	Since 1.8
قيم عيّنة للموقع الجغرافي وحقول الكائنات؛ واصطناع الكتل والكتل القابلة للتكرار	Since 1.8
منتقيات الحقول المرجعية تعيد مُدخلات عيّنة	Since 1.8
بيانات عيّنة للوحة المعلومات والإحصاءات والشريط الزمني حين لا توجد قطع أو منشورات حقيقية	Since 1.8
معاينة الجوال الحيّة في مصمّم الصفحات	Since 1.8
تصحيح صفوف الفرز أثناء التعديل الحيّ	Since 1.8
حارس إعادة التوجيه الذي يراعي مسارات امتدادات المُدخل	Since 1.8

مخازن البيانات: النموذج والتوجيه والواجهة البرمجية

هذه الصفحة موجّهة إلى **المُكامِلين ومهندسي المنصة** الذين ينشئون المخازن أو يقرؤونها عبر الواجهة البرمجية، أو يحتاجون إلى فهم دقيق لكيفية تحديد المنصة للمخزن الذي يُكتب فيه المُدخل والمساحات التي تستطيع قراءته.

تشارك المُدخلات بين المساحات من الواجهة؟

ينبغي لبُناة التطبيقات البدء بدليل مشاركة مُدخلات الكيان بين المساحات [بناء التطبيقات].

نظرة عامة

مخزن البيانات (data store) هو القسم الفيزيائي الذي تُكتب فيه *مُدخلات* الكيان. يحمل كل مُدخل مرجع `store` واحدًا يحدّد القسم الذي يُخزَّن فيه المُدخل، ومن ثَمّ المساحات التي يمكنها قراءته. المخازن **ليست** ذاكرة خبء عامة الأغراض من نوع مفتاح-قيمة لرماز المنشئ؛ بل هي طبقة توجيه التخزين التي تستند إليها مُدخلات الكيان وتحكمها **قواعد المخزن (store rules)** الخاصة بكل كيان. ومن حيث المفهوم، تمثّل المخازن وحدات مخصّصة تُصنَّف بحسب النوع لتحديد نطاق المشاركة والأذونات لمُدخلات الكيان.

يدعم أسس **التوجيه متعدد المخازن**: يُكتب المُدخل في **مخزن واحد** (المخزن المتاح للمساحة الحالية ذو الأولوية الأعلى)، بينما تُجمَّع نتائج عمليات **القراءة من جميع المخازن** المتاحة لتلك المساحة.

> **التوفّر:** Long-standing. يسبق نموذج قواعد المخزن وواجهة برمجة المخازن المرحلة 1.2؛ لذا تعامل معهما بوصفهما قدرة أساسية متاحة دائمًا، لا ميزة حديثة.

أنواع المخازن

هناك ثلاثة أنواع من المخازن، مُعرّفة في `DataStoreTypeEnum` (`backend/packages/osos-shared/src/models/enums.ts`):

النوع	القيمة الرقمية	النطاق	محفوظ كمستند `DataStore` ؟
`Public`	`0`	مخزن واحد شامل تتشاركه جميع المساحات	لا – ضمني/افتراضي
`Private`	`1`	مخزن واحد لكل مساحة	لا – ضمني/افتراضي
`Shared`	`2`	مخزن يمتد عبر مساحتين أو أكثر تُختار صراحةً	نعم

- **المخزن العام** – يُنشأ تلقائيًا مع الكيان؛ ويحفظ البيانات التي يمكن الوصول إليها من جميع المساحات. ولا يمكن تعديل سوى **وزنه**.
- **المخزن الخاص** – يُنشأ تلقائيًا مع الكيان؛ ويحفظ البيانات التي لا يمكن الوصول إليها إلا ضمن المساحة المالكة. ولا يمكن تعديل سوى **وزنه**.
- **المخزن المشترك** – يعرّفه المدير عند الحاجة؛ ويتيح مشاركة البيانات بين مساحات محدّدة تُختار عند الإنشاء. يختار المدير المساحات المشاركة ويضبط **وزنًا**.

المخازن من نوع **Shared** وحدها مستندات فعلية في مجموعة `DataStore` ، وهي النوع الوحيد الذي يمكن إنشاؤه أو قراءته أو تحديثه عبر واجهة برمجة المخازن (Stores API). أما `Public` و `Private` فيُولَّدان وقت التشغيل من معرّفات ثابتة، ولا يُخزَّنان أبدًا كصفوف في `DataStore` .

المعرّفات الثابتة

من `backend/packages/osos-loopback/src/store/types.ts` :

الثابت	القيمة
`PUBLIC_STORE_ID`	`"PUBLIC"`
`PRIVATE_STORE_ID`	`"PRIVATE"`
`PUBLIC_STORE`	`ReferenceProperty { dataEntity: "DataStore", dataEntry: "PUBLIC" }`
`PRIVATE_STORE(spaceId)`	`ReferenceProperty { dataEntity: "DataStore", dataEntry: "PRIVATE_<spaceId>" }`

للمخزن العام قيمة ثابتة واحدة للحقل `dataEntry` هي (`PUBLIC`). ولكل مساحة مخزنها الخاص، وتكون قيمة `dataEntry` فيه `PRIVATE_<spaceId>` .

قواعد المخزن

للكيان قائمة من **قواعد المخزن** (`dataEntity.storeRules: StoreRule[]`). وتضم قاعدة المخزن (`store-rule.model.ts`) ثلاثة حقول:

الحقل	النوع	ملاحظات
`type`	`DataStoreTypeEnum` (number)	`Shared` / `Private` / `Public`
`storeId`	`string | null`	يُضبَط **فقط** عندما يتحقق الشرط `type === Shared` ؛ وهو معرّف `DataStore`
`priority`	`number` (مطلوب)	تُختار القاعدة ذات الأولوية الأعلى عند توجيه الكتابة

عند إنشاء كيان، يُزوَّد بقاعدتين افتراضيتين (`StoreService.defaultStoreRule`):

القاعدة	النوع	storeId	priority
الافتراضي العام	`Public`	`PUBLIC_STORE_ID` (`"PUBLIC"`)	`100`
الافتراضي الخاص	`Private`	`PRIVATE_STORE_ID` (`"PRIVATE"`)	`50`

تُرفَق هاتان القاعدتان الافتراضيتان في معالجات أوامر إنشاء الكيان وإنشاء الكيان الوهمي. وعند إضافة مخزن **Shared** إلى كيان، تُلحَق قاعدة ثالثة تحمل `id` المخزن والوزن المطلوب.

تظهر قيمة `priority` لقاعدة المخزن في الواجهة البرمجية باسم `weight` . والاسمان يشيران إلى القيمة نفسها؛ إذ تستخدم كائنات نقل البيانات (DTOs) الاسم `weight` ، بينما يستخدم النموذج المحفوظ الاسم `priority` .

كيف توجّه قاعدة المخزن عملية كتابة

تحدّد `StoreService.getStoreForWrite(storeRules, spaceId)` المخزن الذي يُكتب فيه المُدخل الجديد:

1. استبعِد أي قاعدة **Shared** لا يتضمن مخزنها المساحة الحالية `spaceId` ضمن قائمة `spaces` الخاصة به. أما قاعدتا `Public` و `Private` فيُحتفَظ بهما دائمًا.
2. من القواعد المتبقية، اختَر القاعدة ذات `priority` **الأعلى**.
3. حوّلها إلى `ReferenceProperty` :
   - `{ dataEntity: "DataStore", dataEntry: <storeId> }` ← `Shared`
   - `PRIVATE_STORE(spaceId)` ← `Private`
   - أي شيء آخر ← `PUBLIC_STORE`

ولأن المخزن Shared المُنشأ حديثًا يُمنَح قيمة `priority` / `weight` ، فإن مقارنة هذه القيمة بقيمتَي المخزنين الافتراضيين Public (100) Privateg (50) تحدّد ما إذا كانت المدخلات الجديدة في المساحات المشاركة تُكتب في المخزن المشترك أم في المخزنين الافتراضيين العام والخاص.

كيف تحكم قواعد المخزن عملية قراءة

تُعيد `StoreService.getStoresForRead(spaceId)` مجموعة المخازن المرئية من مساحة ما: المخزن العام، والمخزن الخاص بتلك المساحة، وكل مخزن Shared تتضمن قائمة `spaces` الخاصة به تلك المساحة. وتتحقق مسارات قراءة المدخلات والتحقق منها (على سبيل المثال في `data-entry-writer.service.ts`) من وجود قيمة `store.dataEntry` الخاصة بالمُدخل ضمن هذه المجموعة؛ فأي مُدخل لا يكون مخزنه مرئيًا من المساحة الحالية يُعامَل كأنه *غير موجود* فيها. وبهذه الآلية تحدّد المخازن نطاق رؤية المدخلات عبر المساحات.

نموذج البيانات: `DataStore`

`data-store.model.ts` – يُحفَظ للمخازن من نوع **Shared** فقط. يوسّع `Identified<UUID>` ، لذا يكون `id` من نوع UUID.

الحقل	النوع	مطلوب	ملاحظات
`name`	`LocalizedValue`	نعم	اسم عرض مُوطَّن
`type`	`DataStoreTypeEnum` (number)	نعم	يكون عمليًا `Shared` (`2`) دائمًا للمخازن المحفوظة
`dataEntity`	`string`	نعم	معرّف الكيان المالك
`spaces`	`ReferenceProperty[]`	نعم	المساحات التي يمتد عبرها المخزن
`description`	`NullableLocalizedValue`	لا	
`creationInfo`	`ActionInfo`	لا	مَن أنشأ المخزن ومتى (مع مفوَّض اختياري)
`updateInfo`	`ActionInfo`	لا	مَن حدّث المخزن آخر مرة ومتى

ويجري الحفظ عبر مستودع بسيط من نوع `DefaultCrudRepository` في LoopBack (`DataStoreRepository`)، يستند إلى مصدر بيانات Mongo.

مرجع واجهة البرمجة

تقع جميع نقاط النهاية في `StoresController` ، وتتطلب المصادقة عبر استراتيجية أسس `Auth2` في LoopBack. وتُحدَّد جميعها ضمن نطاق كيان عبر `entityId` في المسار. وقبل تنفيذ أي عملية، يحمّل كل معالج الكيان ويشغّل `Helper.validateDataEntityExistsActive` (يجب أن يكون الكيان موجودًا ونشطًا).

الطريقة	المسار	الغرض
`POST`	`/v1/entities/{entityId}/stores`	إنشاء مخزن Shared للكيان
`GET`	`/v1/entities/{entityId}/stores`	سرد جميع مخازن الكيان (الافتراضية + المشتركة)
`GET`	`/v1/entities/{entityId}/stores/{storeId}`	جلب قاعدة مخزن / مخزن واحد
`PUT`	`/v1/entities/{entityId}/stores/{storeId}`	تحديث مخزن

شكل الاستجابة: `StoresResponseDto`

تُعيد كل نقطة نهاية استجابة بهذا الشكل (أو مصفوفة منه):

الحقل	النوع	ملاحظات
`storeId`	`string`	معرّف المخزن، أو `"PUBLIC"` / `"PRIVATE"` للمخازن الافتراضية
`storeName`	`LocalizedValue`	يُملأ لمخازن Shared فقط
`type`	`DataStoreTypeEnum` (number)	مطلوب
`description`	`NullableLocalizedValue`	اختياري؛ لمخازن Shared فقط
`spaces`	`ReferenceProperty[]`	اختياري؛ لمخازن Shared فقط
`weight`	`number`	أولوية قاعدة المخزن

أما بالنسبة إلى مخازن Public/Private، فتحمل الاستجابة `storeId` و`type` و`weight` فقط؛ وتُترَك حقول `storeName` و`description` و`spaces` دون ضبط، لأن تلك المخازن ليس لها مستند `DataStore`.

إنشاء مخزن – `POST /v1/entities/{entityId}/stores`

متن الطلب (`CreateDataStoreRequestDto`، الذي يوسّع `DataStore` ويضيف `weight`):

الحقل	النوع	مطلوب	ملاحظات
`name`	`LocalizedValue`	نعم	
`type`	`DataStoreTypeEnum` (number)	نعم	**يجب أن يكون `2` (Shared)** – انظر التحقق
`spaces`	`ReferenceProperty[]`	نعم	مساحتان على الأقل، بلا تكرار
`description`	`NullableLocalizedValue`	لا	
`weight`	`number`	نعم	أولوية قاعدة المخزن؛ يجب أن تكون فريدة داخل الكيان
`dataEntity`	`string`	–	يُتجاهَل في المتن؛ إذ يستبدله المتحكّم بقيمة `entityId` من المسار

تسلسل المعالجة في الخادم:

1. يُرفض الطلب إذا كانت أي قاعدة مخزن موجودة في الكيان تستوفي مسبقًا الشرط `priority === weight` → `StoreWithSameWeightExistsException`.
2. تُضبَط قيمة `storeDto.dataEntity` إلزاميًا على قيمة `entityId` في المسار.
3. تُجري `StoreService.createStore` التحقق (انظر أدناه) وتحفظ `DataStore`، مع تسجيل المستدعي في `updateInfo` / `creationInfo` (و`delegatee` عند التصرف نيابةً عن مستخدم آخر).
4. تُلحَق قاعدة جديدة `StoreRule { storeId, priority: weight, type }` بقائمة `storeRules` الخاصة بالكيان.
5. يُنشَر `UpdateDataEntityCommand` على ناقل الخدمة، ويُسجَّل رمز عبر `InvokerService.registerToken`. **ولهذا يكون إرفاق القاعدة بالكيان غير متزامن / متسقًا في نهاية المطاف** – إذ تُعاد استجابة HTTP قبل تأكيد تحديث الكيان.

الاستجابة: `StoresResponseDto` للمخزن المُنشأ (`storeName` و`description` و`spaces` و`type` و`weight` و`storeId`).

مثال طلب:

```
{
  "name": { "en": "Regional shared store" },
  "type": 2,
  "spaces": [
    { "dataEntity": "Space", "dataEntry": "space-a" },
    { "dataEntity": "Space", "dataEntry": "space-b" }
  ],
  "description": { "en": "Entries visible to both regional spaces" },
  "weight": 75
}
```

التحقق وقت الإنشاء (`validateCreateStore`)

الشرط	الاستثناء	رمز الخطأ
أقل من مساحتين	`NotEnoughStoreSpacesSelectedException`	`NOT_ENOUGH_STORE_SPACES_SELECTED`
معرّفات مساحات مكررة	`StoreSelectedSpacesDuplicationNotAllowedException`	`SELECTED_SPACES_SHOULD_NOT_HAVE_DUPLICATION`
`type !== Shared`	`OperationNotSupportedForThisType`	`OPERATION_NOT_SUPPORTED_FOR_THIS_STORE_TYPE`
تعارض الوزن مع قاعدة قائمة	`StoreWithSameWeightExistsException`	`STORE_WITH_SAME_WEIGHT_ALREADY_EXIST`

ويؤكّد تعليق في الرماز المصدري هذا القصد: *"At the moment of implementation only supports type: shared."* أي إن إنشاء مخازن Public أو Private عبر واجهة البرمجة غير مدعوم.

سرد المخازن – `GET /v1/entities/{entityId}/stores`

يُدمَج معامل الاستعلام الاختياري `filterCriteria` (وهو `FilterCriteria<DataStore>`) مع القيد الإلزامي `where.dataEntity = entityId` قبل تنفيذ الاستعلام، فلا تطابق المرشّحات إلا مخازن Shared الخاصة بهذا الكيان.

تُسلسِل الاستجابة ما يلي:

1. **القواعد الافتراضية** – كل قاعدة مخزن يتحقق فيها الشرط `type !== Shared` (أي قاعدتا Privateg Public الافتراضيتان)، وتُعاد كل واحدة على هيئة `{ storeId, type, weight }` .
2. **مخازن Shared** – تُجلَب من مجموعة `DataStore` لهذا الكيان، وتُعاد كل واحدة بكامل تفاصيلها، مع قيمة `weight` مأخوذة من `priority` لقاعدة المخزن المطابقة.

وبالنسبة إلى مخازن Shared، تحلّ الخدمة أيضًا كل مرجع مساحة؛ إذ تحمّل صفوف `DataEntryIndex` المطابقة وتُرفِق قيمة `meta` من فهرس المساحة بوصفها حقل `_resolved` بكل مرجع مساحة (ويُستخدَم لعرض البيانات الوصفية للمساحة دون استدعاء ثانٍ).

جلب مخزن واحد – `GET /v1/entities/{entityId}/stores/{storeId}`

يبحث المعالج عن قاعدة المخزن في الكيان باستخدام `storeId` .

- إذا لم يُعثر على قاعدة مطابقة ← `DataSourceStoreNotExist` .
- إذا تحقق الشرط `type !== Shared` للقاعدة (أي كانت قاعدة Public/Private افتراضية) ← يُعيد `{ type, weight, storeId }` فقط، دون الرجوع إلى مجموعة `DataStore` .
- إذا كانت القاعدة من نوع Shared ← يحمّل `DataStore` (عبر `getStoreById` ، الذي يرمي `DataSourceStoreNotExist` عند `ENTITY_NOT_FOUND`)، ويعيد التفاصيل كاملةً، إضافةً إلى `weight` الخاص بالقاعدة.

تحديث مخزن – `PUT /v1/entities/{entityId}/stores/{storeId}`

متن الطلب (`UpdateDataStoreRequestDto`) – جميع الحقول اختيارية، ويُعامَل المتن بوصفه تحديثًا جزئيًا:

الحقل	النوع	ملاحظات
`name`	`LocalizedValue`	مخازن Shared فقط
`description`	`NullableLocalizedValue`	مخازن Shared فقط
`weight`	`number`	الأولوية الجديدة لقاعدة المخزن
`spaces`	`ReferenceProperty[]`	مخازن Shared فقط؛ **يجب أن تحوي مدخلين على الأقل**

تسلسل المعالجة في الخادم:

1. يجب أن تكون قاعدة المخزن موجودة في الكيان → وإلا `DataSourceStoreNotExist` .
2. إذا مُرِّر `weight` وتعارض مع أولوية قاعدة **أخرى** → `StoreWithSameWeightExistsException` .
3. إذا كانت القاعدة **Shared**، تُحدِّث `StoreService.updateStore` مستند `DataStore` . وترفض مصفوفة `spaces` التي تحوي أقل من مدخلين (`NotEnoughStoreSpacesSelectedException`)، ثم تدمج الحقول المُمرَّرة مع القيم القائمة (`name` و `description` و `spaces`) وتُحدِّث `updateInfo` . وتبقى حقول `type` و `dataEntity` و `creationInfo` كما هي. أما في قواعد Public/Private فلا تُمسّ مجموعة `DataStore` .
4. تُضبَط قيمة `priority` لقاعدة المخزن على قيمة `weight` الجديدة (أو تُترَك دون تغيير)، وتُعاد كتابة مصفوفة `storeRules` الخاصة بالكيان. وكما في الإنشاء، يُنشَر `UpdateDataEntityCommand` ويُسجَّل رمز (غير متزامن / متسق في نهاية المطاف).

> **ملاحظة:** تستخدم `updateStore` الطريقة `updateById` من المستودع، وهي تعيد بناء النموذج قبل الكتابة. ولا يملأ مُنشئ `DataStore` حقول المصفوفات قسرًا، لذا تُدمج الحقول المحذوفة صراحةً داخل `updateStore` . فإذا وسّعت النموذج، فضع ذلك في الحسبان (راجع إرشادات ترحيل المستودع بشأن التحديثات الجزئية).

فهارس المخزن المرتَّبة (collated) لقراءات القوائم `Since 1.10`

تبدأ كل قراءة لقائمة مُدخلات كيان بمطابقة `store` الخاص بالمُدخل (ثم حالته `state` وعَلَم `virtual`)، وتُجرى وفق ترتيب (collation) الكيان `{locale: 'en', strength: 2}` . ولا يستخدم MongoDB إلا فهرسًا يطابق ترتيبُه ترتيبَ العملية. ولأن أيًّا من الفهارس الضمنية لم يكن له ترتيب، كانت مطابقة المخزن الأولية في الكيانات الكبيرة تستلزم مسحًا كاملًا للمجموعة.

ومنذ 1.10، يتضمّن كل كيان غير وهمي أيضًا:

- نسختين مرتَّبتين من فهرسَي `store` و `sharingLevel` الضمنيَّين، و
- فهرسًا للمخزن والترتيب باسم `IX_<entity>_store_metaOrder_state_ci` (بالمفاتيح `store.dataEntry` و `value._metaOrder.value` و `_id` ، ثم مرشِّح `state` / `virtual`)، فتُجرى عمليتا الترتيب والتخطّي لصفحة القائمة باستخدام مفاتيح الفهرس بدلًا من إجرائهما في الذاكرة.

تحصل الكيانات الجديدة على هذه الفهارس عند إنشائها، وتُضاف إلى الكيانات القائمة عبر ترحيلات data-service `1.10.0-a38` و `a41` و `a44` . وتبني خدمة الفهرسة هذه الفهارس بصورة غير متزامنة، لذا لا تُستخدم الفهارس الجديدة لقراءات القوائم لدى القاطن الذي رُقّي حديثًا إلا بعد أن يكتمل بناؤها عبر طابور الفهارس. ولا يتطلّب ذلك أي تهيئة، ولا تتغيّر قواعد المخزن أو واجهة Stores API. وتُتخطّى الكيانات الوهمية لأن كائن view في MongoDB لا يقبل فهرسًا.

مرجع الأخطاء

رمز الخطأ	المعنى
`NOT_ENOUGH_STORE_SPACES_SELECTED`	يحتاج المخزن إلى مساحتين على الأقل
`SELECTED_SPACES_SHOULD_NOT_HAVE_DUPLICATION`	تتضمّن المساحات المختارة تكرارات
`OPERATION_NOT_SUPPORTED_FOR_THIS_STORE_TYPE`	محاولة إنشاء مخزن من نوع غير Shared
`STORE_WITH_SAME_WEIGHT_ALREADY_EXIST`	تستخدم قاعدة مخزن أخرى هذا الوزن بالفعل
`DATA_SOURCE_STORE_...` (`DataSourceStoreNotExist`)	المخزن / قاعدة المخزن غير موجود لهذا الكيان

ملاحظات وقيود البناء

- **الوزن/الأولوية فريدان على مستوى الكيان.** لا يمكن أن توجد قاعدتا مخزن بالوزن نفسه على الكيان نفسه؛ ويُفرض هذا القيد عند الإنشاء والتحديث.
- **الوزن هو مفتاح ضبط التوجيه.** تُكتب المُدخلات الجديدة في مساحةٍ ما إلى قاعدة المخزن الأعلى أولويةً من بين القواعد التي تنطبق على تلك المساحة. ومع الأوزان الافتراضية، تتقدّم قاعدة Public (`100`) على قاعدة Private (`50`)، فتُكتب المُدخلات الجديدة إلى المخزن العام ما لم تُغيَّر الأوزان. اضبط وزن مخزن Shared على قيمة تتجاوز `100` ليتقدّم في المساحات التي يمتد عبرها؛ أما الوزن بين `50` و `100` فلا يمنحه أولوية إلا على المخزن الخاص الافتراضي، وهي أولوية يتمتّع بها المخزن العام الافتراضي أصلًا، لذا لا يؤثّر هذا الوزن ما لم يُخفَّض وزن Public إلى قيمة أقل منه.
- **يجب أن تمتد مخازن Shared عبر ≥ مساحتين بلا تكرار**، ويُفرض هذا الشرط عند الإنشاء وعند التحديث (إذا مُرِّرت `spaces`).
- **مخازن Shared وحدها قابلة للإدارة عبر واجهة البرمجة.** أما مخازن Public وPrivateg فتوجد ضمنيًا على كل كيان، ولا يمكن إنشاؤها أو تحريرها.
- **إرفاق القاعدة متسق في نهاية المطاف.** تُعيد عملية الإنشاء/التحديث الاستجابة قبل معالجة `UpdateDataEntityCommand` ، لذا قد لا يظهر التغيير بعدُ عند قراءة قواعد مخزن الكيان مباشرةً عقب العملية.
- **تُفرض رؤية المدخلات عبر المساحات عند القراءة.** لا يكون المُدخل مرئيًا إلا في مساحة تضمّ مجموعةُ قراءتها (Public + المخزن الخاص بتلك المساحة + مخازن Shared التي تغطّيها) مخزنَ ذلك المُدخل.

مرجع الواجهة البرمجية للخلاصات ولوحات المعلومات

تستهدف هذه الصفحة **المُكامِلين والمطوّرين** الذين ينشئون الخلاصات أو ينفّذونها عبر الواجهة البرمجية، أو يقرؤون لوحات المعلومات أو قطع العدّادات، أو يحتاجون إلى معرفة سلوك الخادم بدقة لكل بطاقة. والخلاصة في الواجهة البرمجية هي **استعلام إحصائي**.

> **تبني لوحات المعلومات من الواجهة؟**
>
> ينبغي لبُناة التطبيقات أن يبدؤوا بدليل الخلاصات ولوحات المعلومات والعدّادات [بناء التطبيقات]؛ إذ يغطّي الميزات نفسها بالتسميات التي يعرضها المصمّم.

تتناول هذه الصفحة لوحات أسس **الداخلية** فقط. وبدءًا من `Since 1.6`، أصبح بإمكان الاستعلامات الإحصائية نفسها تغذية لوحات **Grafana** عبر مصدر بيانات Apptomator، وهو الخيار المناسب حين تحتاج إلى إمكانات Grafana في التمثيل البصري والتحويلات، أو إلى لوحة معلومات واحدة تجمع بين أسس وأنظمة أخرى؛ راجع لوحات معلومات Grafana. اختر لوحة المنتج الداخلية حين يكون جمهورك من مستخدمي البوّابة أصلًا، واختر Grafana حين يكون جمهورك من مستخدمي Grafana.

نظرة عامة

تتيح لك ثلاث لبنات مترابطة في **data-service** ضمن أسس عرض أرقام مجمَّعة لبيانات كياناتك:

- **الاستعلامات الإحصائية (الخلاصات)** – تعريفات تجميع قابلة لإعادة الاستخدام، تُنفَّذ على الخادم (count / sum / min / max / average)، مع إمكانية التجميع، وتُنتج رقمًا واحدًا أو قائمة أو سلسلة بيانات لرسم بياني. وتظهر في أداة البناء على هيئة **بطاقات خلاصة**.
- **لوحات المعلومات** – مجموعات مرتّبة من عناصر الاستعلامات الإحصائية (مع إمكانية تنظيمها في مجموعات مُسمّاة)، ترتبط بالنظام أو بوحدة أو بكيان بيانات.
- **قطع العدّادات** – بطاقات خفيفة تعرض «عدد المُدخلات المطابقة لمعرض (مع استعلام اختياري)» للمستخدمين النهائيين.

توجد هذه اللبنات الثلاث جميعها في `backend/services/data-service`، وتتطلّب كل نقطة نهاية استراتيجية المصادقة `Auth2` (عبر `@authenticate` على مستوى صنف المتحكّم).

الاستعلامات الإحصائية (الخلاصات)

الاستعلام الإحصائي هو وحدة التجميع؛ ويُعرَّف **لكل كيان بيانات** ويُخزَّن في مجموعته الخاصة (`StatisticQueryRepository`). وتستند كل بطاقة في لوحة المعلومات إلى استعلام إحصائي واحد.

في المصمّم، يتحكّم **عرض في تبويب نظرة عامة** في الحقل `showInEntityOverview`؛ فتظهر الخلاصة في تبويب «نظرة عامة» المخصّص في المعرض، ويجب تفعيل هذا التبويب من إعدادات المعرض.

نموذج البيانات

المصدر: `models/statistics-queries/statistic-query.model.ts` .

الحقل	النوع	مطلوب	ملاحظات
`id`	UUID	–	يُسنَد عند الإنشاء
`specificationVersion`	number	نعم	القيمة الافتراضية `1`
`name`	LocalizedValue	نعم	الاسم المعروض المُترجَم
`normalizedName`	string	لا	يُولَّد تلقائيًا من `name` إن أُغفِل؛ فريد لكل كيان
`description`	NullableLocalizedValue	لا	
`dataEntity`	UUID	نعم	يضبطه الخادم استنادًا إلى كيان المسار – لا يمكن للعملاء تعيينه أو تجاوز قيمته
`icon`	Icon	نعم	
`source`	DataEntrySource	نعم	مصدر قراءة البيانات: جميع المُدخلات (Store)، أو مجموعة، أو استعلام
`sourceOptions`	StatisticQuerySourceOptions	لا	قيم المعامِلات لمصدر استعلام ذي معامِلات
`aggregation`	StatisticsQueryAggregation	نعم	التجميع + الدالة (انظر أدناه)
`orderBy`	number (`SortDirEnum`)	نعم	`0` = Desc، `1` = Asc – يُطبَّق على `category` في النتيجة
`showAs`	number (`StatisticsQueryDisplayEnum`)	نعم	`0` = Chart، `1` = List، Number
`chartType`	number (`ChartTypeEnum`)	لا	`0` Donut `1` Bar، `2` Line، `3` Pie، `4` Column
`options`	StatisticQueryOptions	لا	`{ colors?: string[] }`
`showInEntityOverview`	boolean	نعم	ما إذا كانت الخلاصة تظهر في تبويب «نظرة عامة» للكيان
`filterField`	string	لا	الحقل الوحيد الذي يُسمح للمستدعي بالترشيح وفقه وقت التنفيذ

كائن `aggregation` (`statistics-query-aggregation.model.ts`):

الحقل	النوع	ملاحظات
`field`	string	الحقل الذي تعمل عليه دالة التجميع
`groupByField`	[]string	الحقول المُراد التجميع حسبها؛ الأول هو الفئة الرئيسية، وتصبح البقية `subCategory`
`groupByOperator`	number | null (`GroupByOperatorEnum`)	معامل تجميع التواريخ في مجموعات (سنة، ربع سنة، شهر، أسبوع، يوم، ساعة، …)
`aggregationFunction`	number (`AggregationFunctionEnum`)	`0` Average `1` Sum، `2` Min، `3` Max، `4` Count

قواعد التحقّق

تفرض `StatisticQueryController.validateQuery` القواعد التالية، وتطرح `StatisticQueryValidationExcpetion` عند مخالفتها:

- `chartType` **يجب أن يكون null/undefined** ما لم يكن `showAs === Chart` .
- إذا وُجد `groupByField` **و** كان `showAs === Chart` ، ف**يجب** توفير `chartType` .
- إذا **لم** يوجد `groupByField` ، ف**يجب** أن يكون `showAs` هو `Number` .

- إذا **وُجد** `groupByField`، فـ**يجب ألّا** يكون `showAs` هو `Number`.
- يُتحقَّق من `name` في جميع اللغات المُهيّأة، ويُفحص للتأكد من عدم تكراره ضمن `dataEntity` نفسه.
- يُتحقَّق من `source` عبر `DataSourceService.validateDataSource`.

يُفرض تفرّد `normalizedName` لكل كيان عند الإنشاء والتحديث. وعند الإنشاء، إن أُغفِل، يُولَّد بالبادئة `statistic_query` عبر `generateUniqueNormalizedName`. **Since 1.6** – أُضيف الحقل `normalizedName` مع آلية توليده وضمان تفرّده في خط الإصدار 1.6.

مزالق الإنشاء عبر واجهة API: مصدر source.store ومسارات الحقول

تملأ أداة البناء `source` و`aggregation` نيابةً عنك؛ أما عند إنشاء الاستعلامات الإحصائية مباشرةً عبر واجهة HTTP، فهناك قاعدتان يسهل الخطأ في تطبيقهما، ولا تؤدي مخالفة أيٍّ منهما إلى رسالة خطأ مفيدة.

يجب أن يكون `source.store` هو الاسم النموذجي للكيان. عند استخدام مصدر `Store` (`type: 0`)، تبحث `DataSourceService.validateDataSourceStore` عن الكيان باستخدام `normalizedName` **فقط** – لا بمعرّف الكيان ولا باسمه المعروض. وتتسبّب أي قيمة أخرى في فشل الإنشاء/التحديث بالاستثناء `DataSourceEntityNotExist` – مع الرسالة `Entity <store> not found` ورمز الخطأ `ENTITY_NOT_FOUND` – وهي رسالة مضلِّلة عندما تكون القيمة معرّفًا لكيان موجود أو اسمًا معروضًا له: فالكيان موجود، لكنه غير مسجَّل بتلك السلسلة بوصفها اسمًا نموذجيًا. وإضافةً إلى ذلك، تخضع الخاصية `store` لنمط المخطط `(^[a-zA-Z_s][a-zA-Z0-9_]{2,127}$|^$)` الذي يرفض على مستوى المخطط معرّفات UUID (لاحتوائها على شرطات) والأسماء المعروضة التي تحتوي على مسافات. وعند التنفيذ **لا يوجد أي فحص وجود إطلاقًا** – إذ تُستخدم سلسلة store حرفيًا اسمًا لمجموعة MongoDB (تضبط `DataEntryRepositoryFactoryService.getDataEntryRepository` قيمة `settings.mongodb.collection` على هذه السلسلة)، ويكون ربط الكيان بالمجموعة في المنصة مطابقًا تمامًا لـ `entity.normalizedName`. أما مصدر `Query` (`type: 2`) فلا تُعيّن store فيه بنفسك – إذ تُؤخذ قيمتها من `dataSource.store` الخاص بالاستعلام المُشار إليه.

تحتاج مسارات حقول التجميع إلى البادئة `value.`. تُخزَّن بيانات حقول المُدخل تحت الخاصية `value` في كل `DataEntry`، لذا يجب كتابة `aggregation.field` وكل عنصر في `aggregation.groupByField` بالصيغة `value.<fieldName>` (أما مسارات البيانات الوصفية للمُدخل، مثل `creationInfo.doneAt`، فتُستخدم كما هي؛ وهذا هو العُرف الذي تتبعه الواجهة الأمامية دائمًا). ولا تفرض واجهة API ذلك: إذ تزيل `StatisticQueryService.getFieldParts` البادئة `value.` لأغراض البحث عن الحقل في البيانات الوصفية للكيان فقط، وتتجاهل أي فشل في البحث (يُسجَّل ولا يُطرح أبدًا)، وتُدرِج المسار **الخام** حرفيًا في مرحلة `$group` – فتأتي الفئة من `$<groupByField>` والقيمة المجمَّعة من `$<fullLookupPath>`. وعليه، فإن مسارًا مجرّدًا مثل `amount` يبني خط تجميع يستخدم `$amount`، وهو مفتاح غائب عن كل مستند، و**يفشل بصمت**: تُقيِّم MongoDB المسار الغائب على أنه `null`، فتندمج جميع الصفوف في خانة واحدة غير مجمَّعة فئتها `null`، وتُنتج `Sum` / `Average` عند تطبيقهما على المسار الغائب `0` / `null` – دون أي خطأ في أي موضع.

متن طلب إنشاء صحيح بالحد الأدنى من الحقول (`POST /v1/entities/{entity}/statistic-queries`)، باستخدام القيم الرقمية للتعدادات الواردة في جدول المرجع السريع في نهاية هذه الصفحة:

```
{
  "name": { "en": "Total amount by status" },
  "icon": { /* … */ },
  "source": { "type": 0, "store": "purchase_order" },  // 0 = Store; the entity's normalizedName
  "aggregation": {
    "field": "value.amount",                           // value. prefix required
    "groupByField": ["value.status"],                  // value. prefix required
    "aggregationFunction": 1                           // 1 = Sum
  },
  "orderBy": 0,                                        // 0 = Asc
  "showAs": 2,                                         // 2 = Chart (groupByField present)
  "chartType": 0,                                      // 0 = Column — required when showAs is Chart
  "showInEntityOverview": false
}
```

أشكال العرض وأنواع المخططات

يتحكّم إعداد `showAs` في طريقة عرض بطاقة الخلاصة، ويحدّد أحد الأشكال الثلاثة التالية:

1. **رقم** – النتيجة في صورة قيمة واحدة لمؤشر أداء.
2. **قائمة** – أعلى القيم في صورة قائمة.
3. **مخطط** – أحد أنواع المخططات التالية (`chartType`):

نوع المخطط	ملاحظات
العمود	أعمدة رأسية؛ يدعم التجميع المتعدد المتراكم.
الشريط	أعمدة أفقية.
الخط البياني	الاتجاهات بحسب مفتاح التجميع.
الدائري	النسب.
الحلقي (الدونات)	النسب مع تسمية في المنتصف.

تستخدم المخططات ألوانًا متوافقة مع سمة الواجهة (فاتحة/داكنة)، ومفتاحًا خارجيًا بصيغة HTML، وحدًّا لعدد الشرائح مع خانة **"قيم أخرى"** للتجميعات ذات العدد الكبير من القيم المتميّزة.

> ملاحظة: هناك ميزتان في المخططات **معطّلتان في الرماز حاليًا** – التعبئة بالتدرّج اللوني، وترتيب الشرائح حسب السطوع – لذا تظهر المخططات بألوان سمة مسطّحة.

ألوان الخلاصة: النسخ واللصق `Since 1.10`

يحمل حقل **اختر الألوان** في الخلاصة (المخزَّن في `options.colors`) **20** لونًا على الأكثر. وقد أُضيف لكل شريحة لون زرّا **نسخ** و**إزالة** (يظهران عند التمرير والتركيز بلوحة المفاتيح واللمس)، بينما ينسخ إجراء **نسخ الكل** القائمة كاملة بصيغة `#a, #b, …` بترتيبها. أمّا **اللصق** فيُلحق كل رمز يمثّل لونًا بصيغة `#RGB` أو `#RRGGBB` – بالتقسيم على الفواصل أو المسافات، مع الإبقاء على حالة الأحرف، ودون توسيع الصيغ المختصرة، ومع الإبقاء على المكرّر – حتى بلوغ حدّ العشرين لونًا؛ ويُهمَل ما سوى ذلك، مما يتيح لك نقل مجموعة ألوان من خلاصة إلى أخرى، أو من أداة تصميم، بخطوة واحدة. وتنطوي القائمة الطويلة إلى صفّين مع مفتاح **+N أخرى** / **عرض أقل**. وتُعدّ هذه ميزة تيسيرية في أداة التصميم فقط؛ ولم يتغيّر شكل الخلاصة المخزَّن.

التفاعل مع البطاقات

تدعم كل بطاقة خلاصة أيضًا ما يلي:

- **الانتقال بالنقر (Drill-down):** يفتح النقر على قيمة أو شريحة مخطط المُدخلات المطابقة.
- **التكبير:** فتح البطاقة في مربع حوار أكبر.
- **التحديث التلقائي:** تُحدِّث البطاقة بياناتها تلقائيًا (يُكتشف التغيير عبر مقارنة بصمة النتيجة `hash`).
- **مرشحات زمنية إضافية:** حصر بيانات البطاقة في *هذا الأسبوع* أو *هذا الشهر* أو *نطاق تاريخ مخصص*.

> في وضع المعاينة (راجع أدناه)، تُزوَّد لوحات المعلومات والخلاصات ببيانات **وهمية** محددة مسبقًا؛ لذا لن تطابق المعاينة في أداة البناء الأرقام الفعلية في الإنتاج.

نقاط النهاية

تخضع جميع المسارات المُقيَّدة بنطاق الكيان لفحص التخويل عبر `entityVoterFactory()` على مستوى الصنف.

الطريقة	المسار	الغرض
`POST`	`/v1/entities/{entity}/statistic-queries`	إنشاء استعلام لكيان
`GET`	`/v1/entities/{entity}/statistic-queries`	سرد استعلامات كيان
`GET`	`/v1/entities/{entity}/statistic-queries/count`	عدّ استعلامات كيان
`GET`	`/v1/entities/{entity}/statistic-queries/{id}`	جلب استعلام واحد (يراعي المعرض، راجع أدناه)
`PATCH`	`/v1/entities/{entity}/statistic-queries/{id}`	تحديث استعلام واحد
`DELETE`	`/v1/entities/{entity}/statistic-queries/{id}`	حذف استعلام واحد
`GET`	`/v1/statistic-queries`	سرد جميع الاستعلامات في النظام
`GET`	`/v1/statistic-queries/count`	عدّ جميع الاستعلامات في النظام
`GET`	`/v1/statistic-queries/{id}`	جلب استعلام واحد بصيغة `StatisticQueryDTO` دون كيان في المسار
`GET`	`/v1/statistic-queries/by-ids`	`Since 1.10` – الصيغة الدفعية للمسار السابق؛ راجع التحميل الدفعي بالمعرّف
`GET`	`/v2/entities/{entity}/statistic-queries/{id}/execute`	التنفيذ وإرجاع النتيجة المجمَّعة

يُحدَّد الكيان المشار إليه بـ `{entity}` باستخدام المعرّف **أو** الاسم النموذجي ضمن المساحة الحالية (`getEntityByIdOrNormalizedNameInSpace`).

المسار `GET /v1/statistic-queries/{id}` هو المسار الذي تستخدمه لوحات المعلومات لجلب عناصرها. ويُفحص التخويل لهذا المسار بالاستناد إلى `dataEntity` **الخاص بالاستعلام نفسه** (المُصوِّت `voteByLoadedEntityFactory('repositories.StatisticQueryRepository', 'dataEntity', 0)`)، لأن `entityVoterFactory` على مستوى الصنف قد يقرأ، لولا ذلك، معرّف الاستعلام على أنه معرّف كيان، فيرفض وصول مستخدم مُقيَّد (Restricted). ويُرجع المسار `StatisticQueryDTO` يحمل أيضًا `relatedData` (`{ dataEntity?, dataModule? }`) مع نسخة مخفّفة من الكيان/الوحدة الذي ينتمي إليه الاستعلام.

تَنشُر عمليتا الإنشاء والتحديث أحداثًا على ناقل الخدمة (`StatisticQueryCreatedEvent`، `StatisticQueryUpdatedEvent`، `StatisticQueryDeletedEvent`). وعند الإنشاء، يُربط الاستعلام أيضًا تلقائيًا بلوحات المعلومات ذات الصلة عبر `DashboardRepository.addStatisticQueryToDashboard`، وتُصدر كل لوحة تأثّرت حدث تحديث لوحة.

التحميل الدفعي بالمعرّف `Since 1.10`

كانت لوحة المعلومات تجلب تعريف كل بطاقة بطلب `GET /v1/statistic-queries/{id}` مستقل – طلب لكل بطاقة، مما يسبّب مشكلة N+1 في بوّابة وحدة مزدحمة. ويُعيد `GET /v1/statistic-queries/by-ids?ids=…&ids=…` الكائن

`StatisticQueryDTO` نفسه (مع `relatedData`) لعدّة معرّفات في طلب واحد، وأصبحت البوّابة تجمع جميع المعرّفات المطلوبة في الدورة نفسها ضمن استدعاء واحد لكل نطاق طلب (عادي، معرض عام، معاينة).

- **100 معرّف على الأكثر لكل طلب**، وتُحتسب المعرّفات المكرّرة؛ ويُردّ على ما يتجاوز ذلك بالرمز HTTP 400.
- **التخويل لكل استعلام لا لكل طلب.** لا يشترط المسار نفسه سوى دور مستخدم مسجّل الدخول (`SystemViewer` أو `Restricted`)؛ ثم يُفحص كل استعلام موجود بالاستناد إلى `dataEntity` **الخاص به** وفق القاعدة نفسها المتّبعة في مسار المعرّف الواحد. والاستعلام الذي لا يستطيع المُستدعي قراءته – أو المعرّف غير الموجود – **يُستبعد من المصفوفة** بدلًا من أن يُفشل الدفعة كلها. لذا لا تعتمد على تطابق ترتيب الردّ أو طوله مع الطلب، بل طابِق النتائج باستخدام الحقل `id`.
- وعند الطلب عبر **معرض بيانات**، لا يُحتفَظ باستعلام إلّا إذا كان ينتمي إلى كيان المعرض واجتاز فحوص المعرض نفسها المتّبعة في مساري القراءة والتنفيذ المُقيَّدين بالكيان (راجع التنفيذ المُقيَّد بنطاق المعرض)؛ وقائمة السماح الحالية في المعرض لـ `/v1/statistic-queries/{id}` تغطّي المسار أصلًا. وفي وضع المعاينة، ما زالت المعرّفات الوهمية تُعيد تعريفاتها المصطنعة.
- وفي البوّابة، يتسبّب المعرّف الذي يستبعده الخادم في فشل **تلك البطاقة وحدها**، بحالة الخطأ نفسها التي كان يُنتجها 403 / 404 في مسار المعرّف الواحد؛ وتُعرض بقية لوحة المعلومات.

تنفيذ استعلام إحصائي

يُرجع `GET /v2/entities/{entity}/statistic-queries/{id}/execute` ما يلي:

```
{
  "result": [ /* StatisticQueryResult[] */ ],
  "count":  123,          // total matching rows
  "query":  { /* FilterExpression | undefined */ },
  "hash":   "…"           // hash of result, for change detection
}
```

- يُحصر `where` الوارد في الطلب **إلى** `filterField` **الخاص بالاستعلام** فقط (`Utils.extractAllowedFilters`) – وتُسقَط أي مفاتيح ترشيح أخرى.
- يحمل الحقل `query` المعايير المُحلّلة للاستعلام المُخزَّن الذي يُشير إليه المصدر (أو `undefined` عندما لا يكون المصدر استعلامًا).
- `hash` هو `Utils.hashString(JSON.stringify(result))`.

يحمل كل صفّ من `StatisticQueryResult` (`statistic-query-result.model.ts`) ما يلي: `category`، و`originalCategory`، و`subCategory[]` اختياريًا (`{ field, result, originalCategory? }`)، و`result` (القيمة المجمَّعة).

داخليًا (`StatisticQueryService.execute`)، تبني الخدمة خط تجميع MongoDB: فهي تُرشِّح دائمًا بشرط `state = Active` وتستبعد `invalidReason = Obsolete`، وتطبّق معايير استعلام المصدر و`where` الوارد من المستدعي، وتفكّ الحقول المصفوفية/العلائقية، وتُجمِّع (`$group`) حسب الفئة/الفئات الفرعية المُحلّلة، وتُطابِق `aggregationFunction` مع مُعامل Mongo (`Count → {$sum: 1}`، `Sum → $sum`، `Min → $min`، `Max → $max`، `Average → $avg`). ثم تُحوَّل قيم الفئة وقيم switch/collection إلى تسمياتها، وتُقرَّب الفئات/النتائج الرقمية إلى منزلتين عشريتين، وتُرتَّب الصفوف حسب `category` وفق `orderBy` (تُقارَن القيم المُترجَمة بلغة المستخدم).

تقابل معاملات التجميع حسب التاريخ صيغ `$dateToString` أو عمليات حسابية (`GroupByOperatorEnum`): سنة، نصف سنة، ربع سنة، شهر، أسبوع (ISO 8601، يوم من الأسبوع/الشهر/السنة، ومجموعات الساعة/الدقيقة/الثانية. ويتضمّن `BiannualOfYear` و `QuarterOfYear` أيضًا `$year` بحيث تُسبَق النتائج بالبادئة `year.bucket` .

التنفيذ المُقيَّد بنطاق المعرض

عندما يُقدَّم الطلب عبر وسيط **معرض بيانات** (حيث يكون `view` مربوطًا)، يتغيّر السلوك بفعل `checkAgainstDataView` وإعداد `dashboardConfiguration` الخاص بالمعرض:

- يجب أن تكون قيمة `dashboardConfiguration.enabled` الخاص بالمعرض `true` ، وإلا فيُطرح `ViewDashboardConfigurationNotEnabledException` .
- إذا كان `statisticQueriesMatchingSource` مُعيَّنًا ولم يكن `overrideSourceForStatisticQuery` مُعيَّنًا، فيجب أن يُطابق `source` الخاص بالاستعلام `dataSource` الخاص بالمعرض، وإلا فيُطرح `StatisticQuerySourceNotSameAsViewException` .
- إذا كان `overrideSourceForStatisticQuery` بقيمة true وكان مصدر الاستعلام هو `Store` (جميع المُدخلات)، **يُستبدل** المصدر بـ `dataSource` الخاص بالمعرض قبل التنفيذ.
- إذا تحقّق الشرط `dataSource.type === Query` للمعرض وكان `mergeViewSourceQueryWithStatisticQuery` بقيمة true، يُدمج استعلام مصدر المعرض (بعملية AND) في التجميع بوصفه معيارًا إضافيًا. **Since 1.6.**

التسمية «N مُدخلًا» تَعُدّ مع استعلام المعرض `Since 1.4`

حين يدمج المعرض استعلام مصدره في خلاصاته (`mergeViewSourceQueryWithStatisticQuery` ، أعلاه)، أصبح العدّ الذي تستند إليه تسمية **«N مُدخلًا»** على البطاقة يطبّق الاستعلام المدموج نفسه، تمامًا كما يفعل التجميع وقائمة *انقر للاستعراض* – فيساوي العدد في التسمية عدد المُدخلات التي يراها المستخدم بعد النقر. وكانت التسمية قبل الإصلاح تَعُدّ دون تطبيق استعلام المعرض، لذا كان يمكن أن تُظهر عددًا أكبر. ويستمدّ استعلام المعرض **ذو المعاملات** قيم معامِلاته من خيارات مصدر المعرض نفسه؛ ويبقى حلّ معامِلات الخلاصة نفسها بديلًا احتياطيًا لأي اسم لا يحلّه المعرض. وقد أُتيح هذا التصحيح أيضًا في تحديث لخطوط 1.4 و1.6 و1.8.

وضع المعاينة

Since 1.8 – عندما يكون `PREVIEW_MODE_BINDING` مُعيَّنًا، يواصل مسار التنفيذ إجراء عمليات البحث الحقيقية عن الكيان/الاستعلام الإحصائي/المعرض وإنفاذ نطاق المعرض، لكن `count` / `result` المجمَّعين يُنتجهما `FakeStatisticQueryGeneratorService` بدلًا من التجميع الحقيقي. ويظلّ الحقل `query` مُستمدًّا من الاستعلام الحقيقي المُشار إليه.

الاستعلام الإحصائي مقابل محصّلة الكيان `Since 1.10`

الخلاصة (أي `StatisticQuery` – وهي موضوع هذه الصفحة كلّها) و**محصّلة الكيان** موردان مختلفان. ويسهل الخلط بينهما لأن كليهما يحسب في النهاية Count أو Sum أو Min أو Max أو Average على مُدخلات كيان، ولأن كليهما يستند إلى مجموعة الدوال الخمس نفسها. أمّا نموذج المحصّلات – قواعد أنواع الحقول، وكيف يفعّلها المعرض، وكيف يعمل التجميع، وكيف تتصرّف الحقول المحمية – فترد تفاصيله في تجميعات الكيان وتجميع المدخلات [بناء التطبيقات › المجاميع والتجميع] ولا تُكرَّر هنا. ويقتصر هذا القسم على بيان الفرق بينهما:

	الخلاصة (استعلام إحصائي)	محصّلة الكيان
أين تُنشأ	على لوحة معلومات، أو في تبويب لوحة المعلومات في الكيان	في تبويب **المحصلات** الخاص بالكيان نفسه

	الخلاصة (استعلام إحصائي)	محصّلة الكيان
كيف تُحسب	عبر نقطة نهاية التنفيذ الخاصة بها، استعلامًا واحدًا في كل مرة	عبر نقطة نهاية ملخّص قائمة المُدخلات، إلى جانب كل محصّلة أخرى فعّلها ذلك المعرض
ما الذي تنتجه	بطاقة قائمة بذاتها – رقم أو قائمة أو مخطط، بتفصيلها الخاص حسب الفئة	أرقامًا في قائمة مُدخلات: إجمالي كلي، ومجموع فرعي لكل مجموعة حين تكون القائمة مُجمَّعة
كيف تُجمَّع	بتجميعها الخاص حسب الفئة أو فترات التاريخ، مستقلًّا عن أيّ معرض	بتجميع قائمة المُدخلات في المعرض نفسه، وهو مشترك بين جميع محصّلات تلك القائمة
دورة الحياة	تتبع اللوحة التي تحملها؛ وقد تؤدي إزالة العنصر إلى إزالة الاستعلام معه	تتبع الكيان: تُحذف بصورة مستقلة، وتُسقَط دون التسبّب في خطأ من أيّ معرض كان قد فعّلها

اختر الخلاصة حين تريد بطاقة قائمة بذاتها يمكن وضعها على لوحة معلومات. واختر المحصّلة حين يكون الرقم جزءًا من قائمة مُدخلات يطّلع عليها القارئ أصلًا.

سياسة الوصول إلى البيانات الحسّاسة على الخلاصة `Since 1.10`

أصبح بإمكان الخلاصة أن تحمل **سياسة وصول إلى البيانات الحسّاسة**، وهي الإعداد نفسه المتاح في التفاعلات وقوالب التصدير: إذ تحدّد ما إذا كان يجوز لهذه الخلاصة قراءة القيم المحمية أصلًا، وما إذا كان ذلك مسموحًا دائمًا أم بحسب الشخص الذي يشغّلها. ويظهر عنصر التحكّم في نموذج الخلاصة نفسه، ولا يظهر إلا إذا كان الكيان يعلن فعلًا عن حقل حسّاس. راجع الحقول الحساسة [بناء التطبيقات › حماية حقل] لمعرفة معنى كل سياسة.

وثلاثة أمور خاصة بالخلاصات:

- **السياسة غير المضبوطة تتصرّف تمامًا كما كانت.** فكل خلاصة أُنشئت قبل وجود هذا الإعداد تحتفظ بسلوكها السابق؛ ولم يُرحَّل شيء ولم يُعَد تقييمه.
- **وخيار «تلقائي» غير معروض.** فليس للخلاصة تشغيل مجدوَل أو تلقائي يختلف عن فتح أحدهم لها، ولا شيء تميّزه تلك السياسة؛ وحفظها مرفوض.

تكرار خلاصة `Since 1.10`

تتضمّن كل بطاقة خلاصة في أداة بناء لوحات المعلومات أيقونة **تكرار** إلى جانب التكبير، على غرار أيقونة التكرار الموجودة أصلًا في قوائم التفاعلات والمعارض. ويفتح التكرار محرّر الخلاصة في وضع **الإنشاء** انطلاقًا من معرّف الخلاصة الأصلية – ولا يُعدّ ذلك تحديثًا للأصلية في أي حال، وتبقى الخلاصة الأصلية دون مساس سواء حفظت أم ألغيت.

- **كل شيء يُنقل باستثناء الاسم** – المصدر، والتجميع، والتجميع حسب الحقول، ونوع العرض، والألوان، و`showInEntityOverview`، والوصف، وسياسة الوصول إلى البيانات الحسّاسة أعلاه.
- **الاسم المقترَح** هو اسم الخلاصة الأصلية مع لاحقة «نسخة» مُترجَمة، باللغة التي تعمل بها حاليًا – وإذا لم يكن للأصلية نص بتلك اللغة، تُطبَّق اللاحقة على أيّ لغة يتوافر بها نص للأصلية، بدلًا من اقتراح اسم فارغ. ويبقى الاقتراح قابلًا للتعديل قبل الحفظ، ويسري فحص تفرّد الاسم المعتاد.
- **يتبع التكرار حاجز الأذونات نفسه الذي يتبعه التعديل** – فمن لا يستطيع تعديل خلاصة لا يستطيع تكرارها أيضًا.

لوحات المعلومات

لوحة المعلومات هي `DataEntry` مُخزَّن في مجموعة `dashboard`. وتتضمّن قيمتها `value` (`DashboardEntryValue`) مصفوفة `elements` المرتّبة، وتحدّد ما ترتبط به اللوحة. وتعرض اللوحة مجموعة من البطاقات (الخلاصات)، مُنظَّمة في مجموعات يمكن إعادة ترتيبها بالسحب والإفلات.

نموذج البيانات

`DashboardEntryValue` (`dashboard-entry-value.model.ts`):

الحقل	النوع	ملاحظات
`id`	UUID	
`name`	LocalizedValue	اسم اللوحة
`normalizedName`	string	يُولَّد تلقائيًا من الاسم؛ فريد ضمن نطاق `relatedTo`
`elements`	(DashboardElement | DashboardGroup)[]	المحتويات المرتّبة
`relatedTo`	ReferenceProperty	ما ترتبط به اللوحة (راجع أدناه)
`sharing`	ApplicationSharing	إعدادات المشاركة

يحدّد `relatedTo` نطاق اللوحة:

- **undefined** ← لوحة على مستوى **النظام**.
- `{ dataEntity: 'DataModule', dataEntry: '<module-uuid>' }` ← لوحة **وحدة**.
- `{ dataEntity: 'DataEntity', dataEntry: '<entity-uuid>' }` ← لوحة **كيان**.

يرفض `checkSupportedRelatedTo` أي قيمة أخرى لـ `relatedTo.dataEntity` ، ويطرح `DataNotSupportedException` ("Only related to DataModule or DataEntity is supported").

إصلاح – لم تعد حقول اللوحة تختفي عند الحفظ `Since 1.8` . يُعرِّف كيان `DASHBOARD` الآن `name` و `elements` / `relatedTo` و `sharing` بوصفها حقولًا فعلية في مخطّطه، بعد أن كان يقتصر على `normalizedName` . فقبل هذا الإصلاح، لم تكن لوحة النظام في أي تثبيت جديد تتضمّن تعريفات لأي حقول سوى `normalizedName` ، مما كان يجعل مسار الكتابة العام يحذف بصمت كل مفتاح آخر – بما فيه `name` – عند أول حفظ للوحة. ولأن العميل يُعيد إرسال آخر قيمة جلبها لـ `value` أيًّا كانت، كان أول تعديل على لوحة جديدة تمامًا (إضافة خلاصة، إضافة مجموعة) يفشل بخطأ تحقّق يطالب باسم (`name`) حُذف للتوّ. ويعالج إجراء ترحيل التثبيتات القائمة، فيستعيد الحقول في أي لوحة فقدتها دون المساس باللوحات التي لا تزال تحتفظ باسم.

يأتي كل عنصر بأحد شكلين (تُخزَّن عناصر `elements` بوصفها كائنات مرنة، ويُميَّز بينها بالحقل `elementType`):

`DashboardElement` (`elementType: 'Element'`) – `dashboard-element.model.ts` :

الحقل	النوع	ملاحظات
`id`	UUID	فريد ضمن اللوحة
`elementType`	`'Element'`	ثابت
`element`	ReferenceProperty	`{ dataEntity: 'StatisticQuery', dataEntry: '<query-uuid>' }`
`group`	UUID	مجموعة `DashboardGroup` التي ينتمي إليها هذا العنصر (اختياري)
`index`	number	الموضع في اللوحة

`DashboardGroup` (`elementType: 'DashboardGroup'`) – كتلة مُسمّاة يمكن تعشيش عناصر أخرى تحتها: `id` ، و `name` (مُترجَم، مطلوب)، و `description` ، و `icon` ، و `index` ، و `group` اختياري (يمكن تعشيش مجموعة تحت مجموعة أخرى). وفي أداة البناء، تُسحب الخلاصات المترابطة إلى مجموعة، وتُسمّى المجموعة باسم مميّز لتظهر كتلةً واحدة.

مستويات لوحة المعلومات

يمكن إنشاء لوحة المعلومات ضمن ثلاثة نطاقات:

- **مستوى الكيان** – يضم تبويب **لوحة المعلومات** في الكيان خلاصاته؛ ويمكنك فيه إضافة خلاصة أو مجموعة أو تعديلها أو حذفها، كما يمكن إظهارها من إعدادات المعرض.
- **مستوى الوحدة** – يجمع تبويب **لوحة المعلومات** في الوحدة خلاصاتٍ من كيانات الوحدة؛ وعند إضافة خلاصة، تختار الكيان الذي تُجلب منه خلاصة مُعدّة مسبقًا.
- **مستوى النظام** – تُحرَّر لوحة معلومات النظام (`relatedTo` غير معرَّف) من **الإدارة › التطبيقات › المبيانات**. ويمكن لصفحة أن تعرض خلاصات لوحة معلومات عبر قطعة **الخلاصة** فيها.

توفير لوحة المعلومات الافتراضية

لا تُضمَّن لوحات المعلومات مطلقًا على هيئة seed أو fixture JSON – فلا توجد ملفات seed تُنشئ كيانات النظام. بل تُوفَّر **تفاعليًا ومدفوعةً بالأحداث** عبر معالِجات ناقل الخدمة في node-ts (`data-service/src/handlers/events/`):

- **لكل كيان بيانات جديد** – يُنشئ `DefaultDashboardDataEntityCreatedHandler` لوحة افتراضية واحدة لكل `DataEntity` يُنشأ حديثًا، مع `relatedTo` يشير إلى الكيان، ومصفوفة `elements` فارغة، ومستوى مشاركة **Private**، و `spaces = ROOT_SPACE_ID` ، وحالة `Active` (`name.en = 'dashboard'`).
- **لكل وحدة جديدة** – يُنشئ `CreateDefaultDashboardOnModuleCreatedHandler` لوحة افتراضية مع `relatedTo` يشير إلى الوحدة عند وقوع `DataModuleCreatedEvent` (يُتخطّى الإنشاء إن وُجدت لوحة مسبقًا)، إلى جانب صفحة الوحدة / المعرض العام الافتراضي.

يُشغّل كلٌّ من المعالِجَين `CoreEntryNormalizedNameService.ensureForCreate` وينشر `DashboardCreatedEvent` . ولهذا لا يوفّر `DashboardController` أي نقطة نهاية للإنشاء (راجع أدناه) – إذ تكون اللوحة الأولية موجودة سلفًا حين يفتح المستخدم تبويب لوحة المعلومات.

نقاط النهاية

الطريقة	المسار	الغرض
`GET`	`/v1/dashboards`	سرد اللوحات (مُرشَّحة حسب حقوق الوصول أو مُقيَّدة بنطاق المعرض)
`GET`	`/v1/dashboards/{id}`	جلب لوحة واحدة
`PUT`	`/v1/dashboards/{id}`	استبدال محتويات لوحة

لا توجد نقطة نهاية **إنشاء أو حذف** في `DashboardController` – تُوفَّر اللوحات وتُدار في مكان آخر؛ ويقتصر عمل هذا المتحكّم على القراءة والتحديث. ويُصرَّح باستخدام `PUT` عبر `voteByRelatedToFactory('value.relatedTo', 1)` .

يختلف سلوك `GET /v1/dashboards` بحسب السياق:

- **بلا معرض** ← يُرجع `DashboardRepository.findWithAccessRight(userId, filter)` اللوحات التي يمكن للمستخدم المسجَّل الوصول إليها (وفق قواعد تجميع نصّي باللغة الإنجليزية).
- **عبر معرض بيانات** ← يجب أن يكون المعرض مثيلًا من `DataViews` (وإلا يُطرح `InvalidDataViewException`)، وأن يتحقّق الشرط `dashboardConfiguration.enabled === true` (وإلا يُطرح `DashboardConfigurationNotEnabledException`)، ثم يُرجع `DashboardService.getDashboardForEndUsers` اللوحات المسموح بها للمعرض فقط، مع إمكانية ترشيح العناصر حسب تطابق المصدر (راجع أدناه).

يتحقّق `GET /v1/dashboards/{id}` من أن المعرّف من نوع UUID (`InvalidDashboardId`)، ويُرجع اللوحة المطلوبة بعد ترشيحها حسب حقوق الوصول.

قواعد التحديث (PUT)

يستبدل `updateById` قيمة `value.elements` بالكامل، ويُجري تحقّقًا شاملًا:

- **صلاحية المعرّف** – يجب أن يكون معرّف المسار من نوع UUID.
- يُحتفَظ بقيمة `relatedTo` **كما هو** من اللوحة الحالية (فلا يمكن للعملاء ربط اللوحة بنطاق آخر)، ويُضبط `sharing` إلزاميًا على القيمة `Public` دون مساحات.
- `normalizedName` **مملوك بالكامل من الخادم أيضًا** `Since 1.8` – إلى جانب `relatedTo` و `sharing`، تُؤخذ قيمة `normalizedName` الآن عند حفظ اللوحة من السجلّ المخزَّن دون قيد أو شرط؛ فلا يحقّ لأي عميل إعادة كتابة المعرّف الذي يستند إليه تفرّد اللوحة. وكانت عملية الحفظ سابقًا تمحوه بصمت، لأن نموذج العميل لا يتضمّن هذا الحقل مطلقًا، ولأن التحديث يستبدل محتويات اللوحة بالكامل. أمّا `name` فيظل المستخدم هو من يزوّد قيمته، لكن تُستخدم الآن القيمة المحفوظة سابقًا عندما تُغفل عملية الحفظ هذا الحقل، وبذلك لم يعد بإمكان عميل لا يُعيد إرسال الاسم أن يمحوه.
- **معرّفات عناصر فريدة** (`checkElementsMostHaveUniqueId`) – يجب أن يكون لكل عنصر/مجموعة معرّف `id` صالح من نوع UUID، دون تكرار (`InvalidElementID`).
- **عدم تكرار الاستعلام الإحصائي** (`checkElementsMostHaveUniqueElement`) – لا يمكن أن يظهر `element.dataEntry` نفسه في عنصرين (`DashboardElementUsedMoreThanOneException`).
- **سلامة المجموعات** – يجب أن يشير `group` الخاص بكل عنصر إلى مجموعة موجودة في اللوحة (`InvalidDashboardGroup`)؛ ويجب أن تكون أسماء المجموعات غير فارغة في جميع اللغات وفريدة عبر اللغات (`EmptyLocalizedException` / `DashboardElementDuplicateException`).
- **نطاق الاستعلام الإحصائي** – في لوحات الوحدة/الكيان، يجب أن ينتمي كيان كل استعلام إحصائي مُشار إليه إلى وحدة/كيان اللوحة، وإلا يُطرح `DashboardStatisticQueryNotInModule` (تُجمَّع هذه الأخطاء وتُطرح ضمن `DashboardValidationException`).

فحوصات الأذونات (`checkUserPermission`):

نطاق اللوحة	من يجوز له التعديل
النظام (`relatedTo` = undefined)	مسؤول النظام أو مطوّر النظام فقط (`SystemDashboardEditNotAllowedException`)
الوحدة	مسؤول/مطوّر النظام، أو محرّر الوحدة (`EditModuleDashboardNotAllowedException`)
الكيان	مسؤول/مطوّر النظام، أو محرّر الوحدة، أو محرّر الكيان (`EntityDashboardEditNotAllowedException`)

يُرفض المستخدم الذي لا يملك أي حقّ وصول إلى النظام (`isSystemNone`)، ويُطرح `DataNotSupportedException` .

تنظيف الاستعلامات الإحصائية اليتيمة – في لوحات **الكيان**، عند إزالة عناصر أثناء التحديث، يحذف `handleUnusedStatisticQueries` الاستعلامات الإحصائية التي تستند إليها العناصر المُزالة، ولكن فقط إذا لم يعد أي عنصر آخر في اللوحة يشير إلى الاستعلام المعني. ويقتصر ذلك على المستخدمين التفاعليين؛ أما **الاستدعاءات الداخلية بين الخدمات** (مثل استيراد الحزم) فتتجاوز هذا التنظيف، لأن الاستيراد يدير الاستعلامات الإحصائية على نحو مستقل، وقد تظل معرّفات العناصر تشير إلى معرّفات بيئة المصدر.

ترشيح العناصر حسب مصدر المعرض

تقتصر `DashboardService.getAllowedDashboardSameSourceElements` (فقط عندما يكون `dashboardConfiguration.statisticQueriesMatchingSource` مُعيَّنًا) في ترشيح عناصر كل لوحة على الاستعلامات

الإحصائية التي يُطابق `source` الخاص بها `dataSource` الخاص بالمعرض – وتُبقي مجموعات اللوحة دائمًا، وإذا كان نوع مصدر بيانات المعرض هو `Store` (جميع المُدخلات)، فكل استعلام مسموح به. ويُضبط الوصول إلى لوحة عبر معرض باستخدام `getDashboardQueryFilter` بالرجوع إلى قائمة السماح `dashboardConfiguration.dashboards` الخاصة بالمعرض، مع طرح `DashboardAccessNotAllowedException` (أو `Error` بسيط) لأي معرّف غير مسموح به.

مرجع تهيئة لوحة المعلومات في المعرض

يتحكّم `DashboardConfiguration` (على `DataViews`) في كل ما سبق:

في مصمّم المعرض: يقابل `enabled` الخيار **لوحة المعلومات**، و `clickToBrowse` الخيار **انقر للاستعراض**، و `statisticQueriesMatchingSource` الخيار **الخلاصات المعروضة** (**عرض جميع الخلاصات** أو ما يطابق المصدر فقط)، و `overrideSourceForStatisticQuery` الخيار **تطبيق استعلام المعرض على الخلاصات التي مصدرها جميع المدخلات**، و `mergeViewSourceQueryWithStatisticQuery` الخيار **إلحاق استعلام مصدر الواجهة بالخلاصات التي لها استعلام مختلف كمصدر**. أصبح الحقل `isDefaultLandingPage` مهجورًا (Deprecated)، وليس له مفتاح تبديل في مصمّم المعرض: إذ تُحدَّد الشاشة التي يفتح عليها المعرض ضمن **التأثير** › **الشاشة الافتراضية للمعرض**. ويعرض المعرض عدّادًا عبر طريقة العرض **عرض العداد** (`MiniCard` في `availableDisplays`).

الخاصية	الافتراضي	المعنى
`enabled`	–	تفعيل اللوحات لهذا المعرض (يتطلّب `viewConfiguration.enabled`)
`clickToBrowse`	false	ما إذا كان النقر على استعلام إحصائي يتيح الانتقال إلى تفاصيل المُدخلات
`dashboards`	–	قائمة باللوحات المسموح بعرضها عبر المعرض
`statisticQueriesMatchingSource`	–	قصر عناصر اللوحة على الاستعلامات التي يُطابق مصدرها مصدر المعرض
`overrideSourceForStatisticQuery`	false	استبدال مصدر استعلام `Store` بمصدر المعرض عند التنفيذ
`mergeViewSourceQueryWithStatisticQuery`	false	دمج استعلام مصدر المعرض (بعملية AND) في الاستعلام الإحصائي (يُستخدم عندما يكون تطابق المصدر مُعطّلًا) – **Since 1.6**
`showEntityOverview`	–	إظهار نظرة عامة على الكيان في قائمة المُدخلات
`isDefaultLandingPage`	false	**Deprecated** – استخدام اللوحة صفحة هبوط للمعرض

قطع العدّادات

قطعة العدّاد بطاقة صغيرة من نوع «عدد المُدخلات» ترتبط بـ **معرض بيانات**، وبـ **استعلام** اختياريًا. وهي `DataEntry` مُخزَّن باسم الكيان `counter_widget` (`StringConstants.COUNTER_WIDGET_ENTITY_NAME`). **Long-standing** – يعود كيان قطعة العدّاد إلى ما قبل المرحلة 1.2.

نموذج البيانات

`CounterWidgetDataEntryValue` (`counter-widget-data-entry-value.model.ts`):

الحقل	النوع	ملاحظات
`name`	LocalizedValue	
`normalizedName`	string	يُولَّد تلقائيًا، وهو فريد لكل كيان مرتبط
`description`	NullableLocalizedValue	
`icon`	Icon	

الحقل	النوع	ملاحظات
`moduleId`	string	الوحدة المالكة
`dataEntity`	string	الكيان الذي تُعَدّ مُدخلاته
`view`	ReferenceProperty	`{ dataEntity: 'DataViews', dataEntry: '<view-uuid>' }`
`displayOrder`	number	ترتيب الفرز بين القطع
`query`	ReferenceProperty	استعلام اختياري لترشيح المُدخلات أو تمييزها

الإنشاء والتحقّق

تُدار العدّادات في الواجهة من **الإدارة › التطبيقات › العدادات** (**إضافة عداد**: **الاسم**، **الأيقونة**، **المعرض**، **شرط العد**، **الوصف**). أما عبر الواجهة البرمجية، فتُنشأ قطع العدّادات وتُحدَّث **عبر عمليات CRUD العامة للمدخلات** الخاصة بالكيان `counter_widget`؛ ولا يوفّر `CounterWidgetController` نفسه سوى نقاط نهاية للقراءة. ويعمل `CounterWidgetValidator` في مرحلتي `CREATE` و`UPDATE`، ويفرض ما يلي:

- يجب أن يوجد معرض يطابق `view.dataEntry` **و** `dataEntity` الخاص بالقطعة **و** `moduleId` (وإلا فيُطرح `DataViewNotExistException`).
- يجب أن تكون قيمة `viewConfiguration.enabled` الخاصة بالمعرض هي true (`EntriesConfigurationMustBeEnabled`).
- يجب أن يتضمّن `availableDisplays` في المعرض نمط العرض **MiniCard** (`CounterViewMissedMiniCardException`).
- إذا عُيِّن `query`، فيجب أن يوجد استعلام للكيان نفسه المحدّد في `dataEntity` (`DataQueryNotFound`).

نقاط النهاية

الطريقة	المسار	المصادقة	الغرض
`GET`	`v1/counter-widgets/end-user`	أي مستخدم مُصادَق عليه (`authorize: skip`)	سرد القطع المرئية للمستخدم في المساحة
`GET`	`v1/counter-widgets/{id}/count`	مُصوِّت وسيط المعرض	العدد الحالي للقطعة
`GET`	`v1/counter-widgets/{id}/entries`	مُصوِّت وسيط المعرض	المُدخلات المرتبطة بالقطعة (مع علامات التطابق)
`GET`	`v1/counter-widgets`	مُصوِّت مسؤول النظام	سرد إداري بصيغة `CounterWidgetDataEntryDto` (مع حلّ أسماء المعرض والوحدة)

> ملاحظة: تُسجَّل مسارات قطع العدّادات **بدون** شرطة مائلة في البداية (`v1/counter-widgets/...`)، على خلاف مسارات لوحة المعلومات والاستعلام الإحصائي.

يتطلّب المساران `{id}/count` و`{id}/entries` ورود الطلب عبر وسيط معرض (`onlyViewMiddlewareVoter`)، ويتحقّقان أيضًا من أن قيمة `value.view.dataEntry` الخاصة بالقطعة تساوي معرّف المعرض الذي يتوسّط الطلب الحالي، ويُطرح `CounterViewNotMatchedWithMiddlewareException` عند عدم التطابق.

السرد للمستخدم النهائي

يُرجع `getCounterWidgets` (عبر `/end-user`) القطع التي تستوفي الشروط التالية:

- تقع ضمن وحدة في المساحة الحالية،

- ترتبط بمعرض **مسموح** به للمستخدم **وموجود** في المساحة في آنٍ واحد (`intersection` بين المعارض المسموح بها للمستخدم ومعارض المساحة)،
- تستوفي `state === Active`.

وتُرتَّب حسب `value.displayOrder DESC`.

`Since 1.4` – لم تعد عملية السرد تحمّل جميع معارض القاطن لحساب ذلك التقاطع؛ بل تجلب القطع المرشّحة أولًا (وحدات المساحة × المعارض المسموح بها للمستخدم، `Active`، مرتّبة حسب `displayOrder`)، ثم تتحقّق من ظهور معرّفات معارض **تلك القطع وحدها** في المساحة؛ وإذا لم تكن لدى المستخدم معارض مسموح بها، تُرجع `[]` فورًا. وتبقى النتيجة كما هي، لكن الكلفة لم تعد تزداد مع عدد معارض القاطن (وقد أظهر قياس أثر الإصلاح انخفاض زمن البحث عن المعارض من نحو 1.9 ثانية إلى نحو 2 ميلي ثانية على 3,000 معرض). وأُتيح الإصلاح أيضًا كتحديث لخطوط الإصدارات 1.4 و1.6 و1.8.

العدّ والمدخلات

يحمّل `getCounterCount` معرض القطعة، ويتحقّق من إذن الوصول إلى المُدخلات (`checkCanAccessEntries`)، ويحلّ الكيان في المساحة، وإذا كان للقطعة `query`، فيغلّف `criteria` الخاصة بذلك الاستعلام ضمن `FilterExpression` من نوع `AND` قبل تفويض التنفيذ إلى `DataSourceService.countV2`. ويُرجع `{ count }`.

وبالمثل، يحمّل `getCounterEntries` المعرض والكيان، ويفرض `order = ['value.updateInfo.doneAt DESC']`، ويجلب المُدخلات عبر `DataSourceService.findV2`، ويُرجع `CounterWidgetDataEntryEndUserDto[]`. ويُضيف إلى كل DTO قيمة منطقية **`isIncludedInQueryResult`**: عندما يكون للقطعة استعلام، تُنشئ الخدمة مجموعة مؤقتة، وتُدرِج فيها المُدخلات المجلوبة، وتنفّذ معايير الاستعلام المُحلّلة باستخدام `$match`، وتضع علامة على المُدخلات المطابقة، ثم تحذف المجموعة المؤقتة. أما المُدخلات التي لا تطابق الاستعلام، فتُرجَع مع تعيين العلامة إلى `false`. **Long-standing** – تعود العلامة `isIncludedInQueryResult` إلى ما قبل المرحلة 1.2.

السرد الإداري DTO

يُنفّذ `GET v1/counter-widgets` (لمسؤول النظام فقط) عملية تجميع في MongoDB مباشرةً: فيطبّق البحث النصّي وتحليل الترشيح، وفرزًا افتراضيًا حسب `_id` (بالإضافة إلى أي ترتيب مطلوب)، وعمليات skip/offset/limit، ويحلّ `creationInfo.doneBy` و`updateInfo.doneBy` من فهرس المُدخلات، ويربط أسماء المعرض والوحدة، ويُرجع كائنات `CounterWidgetDataEntryDto` (القطعة + `viewName` + `moduleName`).

مرجع سريع: التعدادات

المصدر: `packages/osos-shared/src/models/enums.ts`.

التعداد	القيم (الفهرس)
`DataCollectionTypeEnum`	Store، 1 Collection، 2 Query 0
`AggregationFunctionEnum`	Count، 1 Sum، 2 Min، 3 Max، 4 Average 0
`StatisticsQueryDisplayEnum`	Number، 1 List، 2 Chart 0
`ChartTypeEnum`	Column، 1 Bar، 2 Line، 3 Pie، 4 Donut 0

التعداد	القيم (الفهرس)
`SortDirEnum`	Asc، 1 Desc 0
`GroupByOperatorEnum`	FullDate، 1 Year، 2 Biannual، 3 BiannualOfYear، 4 Quarterly، 5 QuarterOfYear، 6 Month، 7 MonthOfYear، 8 Week، 9 0 WeekWithMonth، 10 WeekWithYear، 11 WeekWithMonthAndYear، 12 DayOfWeek، 13 DayOfMonth، 14 DayOfYear، 15 Hours، 16 Minutes، 17 MinutesAndHours، 18 Seconds، 19 SecondsAndMinutes، 20 SecondsAndHours

مرجع المشاركة في سير العمل

هذه الصفحة موجّهة إلى مطوّري التكامل والمطوّرين الذين يُنفّذون خطوات سير العمل نيابةً عن المستخدمين، أو يحتاجون إلى تفسير دقيق لأسباب قبول إجراء ينفّذه مشارك أو رفضه. وهي المرجع التقني الذي يستند إليه دليل المستخدم النهائي تنفيذ إجراءات خطوات سير العمل [استخدام أسس]. أمّا نموذج تعريف سير العمل، وإعداد الإجراءات والمالكين، والتحقق الإضافي، والخطوات الخارجية، وحارس التزامن، فترد تفاصيلها في نموذج سير العمل ومرجع واجهته البرمجية، ولا تكرّرها هذه الصفحة. وترد رموز الأخطاء ونصائح إعادة المحاولة لكل خدمة في الأخطاء وإعادة المحاولة.

لوحة الحالة

تقرأ البوّابة الإلكترونية حالة سير العمل للمُدخل وتعرضها في التبويب الجانبي **سير العمل** الخاص بالمُدخل، ولا تطلب من المستخدم استدعاء الخادم مطلقًا.

حالات الخطوة (`MainFlowStatusStepTypeEnum`)

لكل خطوة ضمن الحالة نوع يحدّد موقعها بالنسبة إلى «الآن»:

الحالة	المعنى
`Executed`	اكتملت الخطوة؛ وتتضمّن بيانات من نفّذها ووقت التنفيذ وأي ملاحظات.
`CurrentMain`	الخطوة النشطة على المسار الرئيسي لسير العمل.
`CurrentAlternative`	الخطوة النشطة التي جرى الوصول إليها عبر تفرّع عن المسار الرئيسي.
`Pending`	خطوة مستقبلية لم يجرِ الوصول إليها بعد.
`Skipped`	خطوة جرى تخطّيها، مثلًا عبر «الانتقال إلى خطوة».
`ReExecuted`	خطوة جرى الوصول إليها وتنفيذ إجراء عليها أكثر من مرة.

وتُعرض لكل خطوة، حين تسمح الصفحة بذلك، البيانات التالية: **المالك** (يُملأ بعد الوصول إلى الخطوة)، و**نُفّذت بواسطة / المنفّذون** مع الطوابع الزمنية والملاحظات (عدّة أشخاص في الخطوة المملوكة لمجموعة)، و**استُلمت في / نُفّذت في**، و**الملاحظات**.

إظهار المالكين (`ShowStepOwnerModeEnum`)

يمكن للمعرض إخفاء مالكي الخطوات كليًا، أو إظهار **جميع** المالكين (`AllOwners`)، أو إظهار الخطوات **المملوكة للمستخدم الحالي** فقط (`OwnedByMe`). ولا تُرسل هويات المالكين أو المنفّذين إلى المشاهدين المجهولين مطلقًا.

النمط الرئيسي ونمط التنفيذ

تُطلَب الحالة بأحد نمطين (معامل الاستعلام `mode`):

- **`main`** – المسار الخطّي المتوقَّع: الخطوات المنفَّذة، ثم الخطوة الحالية، ثم الخطوات القادمة المتوقَّعة. وهذا ما يعرضه تبويب سير العمل افتراضيًا.
- **`execution`** `Since 1.4` – شجرة التنفيذ الكاملة، بما فيها عدد مرات العودة إلى الخطوات وكيفية تداخل كل تدفق فرعي داخل سير العمل الأصلي. ويطلبها مفتاح **عرض السجل التاريخي** في تبويب سير العمل.

`Since 1.8` أصبح إعداد المعرض يحدّد النمط الذي يُفتح عليه التبويب (**مسار سير العمل** أو **تاريخ سير العمل** في مصمِّم المعرض)؛ ولا يُعرَض المفتاح إلا عندما يفعّل المعرض شجرة التنفيذ. وإذا لم يُضبط نمط افتراضي، أو كانت شجرة التنفيذ معطّلة، يُفتح التبويب على المسار. ويتبع تطبيق الجوّال إعداد المعرض نفسه.

وتُبيّن الحالة أيضًا ما إذا كان المسار المتبقي **محدَّدًا**. فإذا كان الإجراء الرئيسي للخطوة الحالية يتفرّع بناءً على شرط لا يُقيَّم إلا في وقت التشغيل، تعذّر التنبؤ بالخطوات القادمة، وتُشير الحالة إلى ذلك.

الخطوة المملوكة لمجموعة تسرد كل عضو

في الخطوة المملوكة لمجموعة أو لقائمة من الأشخاص، تسرد الحالة والخط الزمني كل عضو مع حالته الخاصة: منفَّذة مع طابع زمني لمن نفّذ الإجراء، وغير منفَّذة للبقية. وتسترجع المنصة المجموعة المالكة من إعداد الخطوة حتى بعد اقتصار ملكية الخطوة على المنفّذ وحده. وهذا هو السلوك المتّبع في جميع خطوط الإصدارات المدعومة (1.4 و1.6 و1.8)، وليس قدرة جديدة. تفاصيل النموذج: ملكية المجموعة على خطوة مملوكة لمجموعة.

السجل التاريخي للتنفيذ بعد إعادة التعيين `Since 1.8`

بعد **إعادة تعيين سير العمل**، لا يضمّ السجل التاريخي للتنفيذ إلا التشغيلة الحالية. وقبل 1.8 كان عرض الخطوة الحالية وحده يراعي إعادة التعيين، لذا كان المُدخل المُعاد تعيينه قد يُظهر الخطوة نفسها مرتين في عرض التنفيذ: مرة بوصفها `Executed` من التشغيلة السابقة، ومرة أخرى بوصفها `CurrentMain` . ولم يعد من الممكن أن تحمل الخطوة الحالتين معًا.

ما يتحقّق منه المحرّك حين يتصرّف مشارك

يعالج محرّك سير العمل الإجراء ويطبّق القواعد التالية:

القاعدة	الرفض
يجب أن يكون المستدعي مالك الخطوة النشطة. وتقبل الخطوات المملوكة لمجموعة أي عضو فيها، وتُستثنى خطوة النهاية الأخيرة.	`InvalidWorkflowOwner`
في الخطوة المتوازية (حيث يجب أن ينفّذ عدّة مالكين الإجراء كلٌّ على حدة)، ينفّذ المالك الإجراء مرة واحدة.	`ParallelStepOwnerAlreadyExecutedStepException`
إذا كانت نافذة تأكيد الإجراء مفعّلة ونمط التعليق فيها `Required` ، وجب تقديم ملاحظة. وأنماط التعليق هي `Optional` و`Required` و`Disabled` (ويظهر للبُناة باسم **مخفي**).	`WorkFlowActionCommentRequired`
لا يمكن تنفيذ إجراء مخفي بناءً على شرط لهذا المُدخل. و`Since 1.8` تتضمّن استجابة الرفض سببه بدلًا من خطأ خادم غير مفسَّر، ويُعاد رمز الحالة 422.	`WorkFlowActionExecutionNotAllowed` (`WORKFLOW_ACTION_EXECUTION_NOT_ALLOWED`)
يُتحقَّق من استيفاء المُدخل متطلبات الخطوة قبل اعتماد الانتقال. وإذا كانت الحقول المطلوبة فارغة، توجّه البوّابة الإلكترونية المستخدم إلى تعديل المُدخل أولًا.	أخطاء التحقّق

وعند نجاح الإجراء، يسجّل المحرّك الخطوة في السجل التاريخي (المنفّذ ووقت التنفيذ والملاحظات)، وينقل المُدخل إلى الخطوة المستهدفة، ويحدّد المالك التالي، ويُخطر البوّابة الإلكترونية. وإذا كان الهدف تدفقًا فرعيًا، تتوقّف الخطوة الأصلية حتى يكتمل التدفق الفرعي.

التنفيذ المجمَّع

يطبّق المسار المجمَّع إجراءً واحدًا على عدّة مُدخلات. ويجب أن تكون جميع المُدخلات المحدّدة في الخطوة نفسها، وإلا رُفض الاستدعاء بأكمله (`WorkflowStepsNotSameException` ، وهو سلوك قديم ثابت). ثم يُعالَج كل مُدخل بصورة مستقلة، وتُبلَّغ نتيجته على حدة؛ وتطلب البوّابة الإلكترونية من المستخدم إعادة تنفيذ الإجراء على المُدخل الذي فشل من صفحة تفاصيله.

`Since 1.10` لم يعد تنفيذ إجراء خطوة على مُدخلات محدّدة في قائمة يفشل بخطأ وصول. فقبل 1.10 كان التنفيذ يُرفض في كل صفحة، أيًّا كان الإجراء. وإذا كان الإجراء يشترط التحقق الإضافي، فيكفي تحقّق واحد لجميع المُدخلات المحدّدة؛ وقبل 1.10 كان تحديد أكثر من مُدخل يشترط التحقق يؤدي إلى الرفض مع تحذير يطلب تنفيذ الإجراء على مُدخل واحد في كل مرة. المسار وجسم الطلب: التحقق الإضافي قبل تنفيذ إجراء.

السباقات الخاسرة والحفظ المتقادم

`Since 1.10` أصبحت كل عملية كتابة على سير العمل مشروطة بالحالة التي جرى التحقّق منها؛ فتتلقّى العملية الخاسرة في السباق استجابة `409 UPDATE_CONFLICT`، ولا يُسجَّل شيء، ويفتح عميل الويب نافذة **تم تغيير هذا السجل**. وقبل 1.10 كان يمكن أن يتزامن إجراء «موافقة» مع إجراء «رفض»، فيُبلَّغ بنجاح كليهما بينما تُنفَّذ الأتمتة المرتبطة بهما مرتين. وتبقى تعديلات الحقول التي أُجريت في أثناء ذلك خارج سير العمل محفوظة، كما يبقى إثبات التحقق الإضافي صالحًا بعد خسارة السباق. وتفشل محاولة أرشفة مُدخل سبق أن أرشفه مستخدم آخر بالرمز `ENTRY_ARCHIVED` (*المدخل مؤرشف بالفعل*) بدلًا من خطأ غير مفسَّر. العقد الكامل: حارس التزامن على كتابات سير العمل.

`Since 1.4` لم تعد آثار الأتمتة التي تنقل المُدخل إلى خطوة أخرى تُلغى بصمت. فقبل ذلك، كان حفظ المُدخل من نسخة حُمِّلت قبل تشغيل الأتمتة قد يعيده إلى خطوته القديمة، مع الإبلاغ بنجاح الأتمتة. وأصبح الحفظ لا يغيّر موضع المُدخل في سير العمل، وتنقل الأتمتة المُدخل انطلاقًا من خطوته الحالية.

الملكية

إعادة الإسناد (تغيير المالك)

يمكن إعادة إسناد الخطوة إلى شخص محدّد عبر **تغير المالك** (بمبادرة من المالك) أو **تغيير المالك** (إذن إداري على المعرض). ويُسمح بذلك للمستدعين الآتين:

- أي شخص لديه إذن **التعديل** على المُدخل، أو
- **المالك الحالي** حين يسمح تصميم الخطوة بتغيير المالك (`allowChangeOwner`)، أو عضو في مجموعة المالك الأصلية للخطوة.

الرفض	الرمز
الهدف يملك الخطوة أصلًا ولم يتغيّر أي شيء آخر	`UserAlreadyOwnException` (ويظهر بالنص *هذا المستخدم يملك هذه الخطوة بالفعل*)
لا يُسمح للمستدعي بتغيير مالك هذه الخطوة	`CanNotChangeOwnerException` / `ActionNotAllowed`

ويمكن أن تكون إعادة الإسناد **دائمة** (`permanentChange`)، فيظل المالك الجديد مالكًا للخطوة إذا عاد التنفيذ إليها لاحقًا؛ وإلا اقتصر التغيير على الوصول الحالي إلى الخطوة (وهو سلوك قديم ثابت). وفي البوّابة الإلكترونية يظهر خيار التغيير الدائم في نافذة **تغيير المالك** الإدارية (**نوع التغيير**: **مؤقت** / **دائم**)، بحسب ما يفعّله المعرض من إذنَي تغيير المالك: أمّا نافذة **تغير المالك** الخاصة بالمالك فتُجري تغييرًا مؤقتًا.

`Since 1.10` تظهر الملاحظة المرفقة بإعادة الإسناد أو تغيير المالك في نقاش المُدخل. انظر مرجع التعليقات والإشارات.

المالكون المؤهَّلون

يعرض منتقي المالك قائمة بالمالكين المؤهَّلين، قابلة للبحث ومقسّمة إلى صفحات. ويُستبعَد المرشّح إذا كان مؤرشفًا (أي ليس في الحالة Active، ويشمل ذلك المستخدمين والمجموعات والوحدات التنظيمية)، أو كان حساب مستخدم مُعطَّلًا، أو مُدخلًا متقادمًا أو غير صالح.

أخذ الملكية

يمكن المطالبة بخطوة مملوكة لـ**مجموعة** أو **قائمة** (باختيار مستخدم واحد وبنمط الاختيار *أخذ الملكية*) إذا كانت **ليس لها مالك فردي بعد**، وذلك عبر **أخذ الملكية**. وتتطلّب المطالبة:

- ألّا تكون الخطوة مملوكة أصلًا لفرد، وإلا أُعيد `CURRENT_STEP_ALREADY_OWNED` (برمز الحالة **422** `Since 1.8`)، و
- أن ينتمي المستدعي إلى مجموعة المالك الأصلية للخطوة أو تجميعتها، وإلا أُعيد `CanNotChangeOwnerException`.

تصحيحات `Since 1.8` على زر **أخذ الملكية** في البوّابة الإلكترونية:

- لا يظهر الزر أبدًا للخطوة التي لها مالك، بما في ذلك الخطوة المملوكة لمجموعة يُسلَّم العمل إلى أعضائها واحدًا تلو الآخر، ويتولّاها العضو الأول.
- يعتمد الفحص على الخطوة النشطة؛ ففي التدفق الفرعي يفحص الخطوة النشطة، لا الخطوة الخارجية العليا. وبذلك لم تعد العضوية في المجموعة المالكة لخطوة أصلية تُظهر خيار المطالبة بخطوة في تدفق فرعي لا صلة لتلك العضوية بها.
- تُحتسب العضوية في **مجموعة استعلام** كالعضوية في مجموعة عادية، وتظهر إجراءات الخطوة تبعًا لذلك.
- لم يعد الضغط على الزر ينتهي بخطأ خادم بلا رسالة توضيحية؛ فإمّا أن تنجح العملية، وإمّا أن يُبيَّن سبب فشلها.

الخطوات المملوكة لمجموعة بلا مالك `Since 1.8`

يمكن أن تُترك الخطوة المُسنَدة إلى مجموعة مع اختيار **مستخدم واحد** / **يدوي** بلا مالك عن قصد، وذلك بعد أن يعيد أحدهم إسنادها إلى المجموعة، أو لأن أتمتة تغيير المالك انتهت إلى أن «المجموعة مؤهَّلة، لكن لا يوجد شخص محدّد مؤهَّل»، فأصبحت القاعدة تترك الخطوة بلا مالك بدلًا من أن تفشل. ويظهر **أخذ الملكية** حين لا يكون للخطوة مالك، ويكون مالكها الأصلي المسجَّل مجموعةً أو مجموعة استعلام أو تجميعة بيانات (لا شخصًا واحدًا)، ويكون إعداد مالكها مستخدمًا واحدًا مع اختيار يدوي، ويكون المستدعي عضوًا. ويُعاد الفحص استنادًا إلى الإعداد الحالي للخطوة، فلم يعد تغيير المالك آليًا يؤدّي إلى إخفاء الإجراء عن جميع الأعضاء. النموذج: الخطوات المملوكة لمجموعة بلا مالك.

التصرّف دون أخذ الملكية `Since 1.10`

في معرض يُفعَّل فيه الإجراء الإداري **تنفيذ إجراء سير العمل** (`executeAsAdmin`)، لا يحتاج عضو المجموعة المالكة لخطوة من نمط *أخذ الملكية* إلى الضغط على **أخذ الملكية** أولًا:

- تعرض ترويسة المُدخل أزرار إجراءات الخطوة نفسها. ويؤدّي الضغط على أحدها إلى المطالبة بالخطوة تلقائيًا ودون تنبيه ثم تشغيل الإجراء؛ وإن فشلت المطالبة، فلا يُشغَّل الإجراء.
- وفي معرض يكون فيه هذا الخيار معطّلًا، يظهر **أخذ الملكية** زرًّا منقسمًا، وتُدرَج في قائمته المنسدلة جميع الإجراءات الأخرى التي يراها المستخدم في الترويسة (رجوع، وتعديل، وتكرار، والأزرار المخصّصة وأزرار القوالب)، مع الإجراءات الإدارية للمعرض في قائمة فرعية خاصة بها حين يفعّل المعرض أيًّا منها.
- وإن سبقه غيره إلى أخذ الخطوة (*الخطوة مملوكة بالفعل*)، يُعاد تحميل المُدخل؛ وتُحفَظ في صفحة التعديل القيم غير المحفوظة.

تصحيحات ذات صلة في الإصدار نفسه: أصبحت ترويسة مُدخل سير العمل تعرض مالك الخطوة بدلًا من المستخدم الذي يشاهده؛ وعاد مسؤول النظام يرى **أرشفة** في قائمة المُدخلات لمُدخل سير عمل لا يملك خطوته الحالية؛ ولم تعد الترويسة تظل في حالة التحميل حين لا ينطبق أي إجراء رئيسي (مثل مُدخل مؤرشف لا يستطيع المستخدم استعادته ولا حذفه).

التصرّف نيابةً عن المالك

يستطيع المدير التصرّف نيابةً عن المالك المُسنَدة إليه الخطوة دون إعادة إسنادها أولًا. ويُسجَّل تنفيذ الخطوة باسم المدير الذي تصرّف؛ لذا تعتمد الخطوةُ اللاحقة التي تحدّد مالكها من هذه الخطوة *بالتنفيذ* المديرَ مالكًا لها، لا المالكَ المُسنَدة إليه الخطوة أصلًا.

التفويض

حين يتصرّف المفوَّض، تُرفَق معلومات التفويض بالأمر؛ إذ يُمرَّر `delegateeId` و `delegationId` بوصفهما سمتين لاصقتين على كل أمر من أوامر سير العمل، فيُسجَّل مع كل إجراء مَن تصرّف بصفة مفوَّض والتفويض الذي تصرّف بموجبه. وتحدّد واجهة المُدخل القيم المعروضة للمفوَّض إليه، فتُظهر تفاصيل الخطوة الحالية والسجل المفوَّضَ بوصفه المنفّذ (شارة **بالنيابة عن**).

الخطوات الإضافية

يستطيع مالك الخطوة إدراج خطوة خارج المسار دون تغيير التصميم، إذا سمحت الخطوة بذلك (`allowAddAdHoc`). وهناك نوعان (`AdhocStepTypeEnum`):

النوع	الغرض
`AdditionalApproval`	توجيه المُدخل إلى شخص آخر للحصول على اعتماد إضافي.
`AdditionalInformation`	إرساله إلى شخص لجمع مزيد من المعلومات.

تطلب نافذة البوّابة الإلكترونية تحديد النوع ومالك من نوع **مستخدم** أو **مجموعة** (عبر منتقي المالكين المؤهَّلين القابل للبحث، `Since 1.8`)، مع **ملاحظة** اختيارية. ولا يوجد مالك احتياطي. وتُضاف الخطوة الإضافية بوصفها خطوةً فرعيةً تابعة للخطوة الحالية؛ ويتوقّف سير العمل عندها حتى يتصرّف مالكها، ثم يعود إلى حيث توقّف. ولا يستطيع فتح المنتقي من بوّابة المستخدم النهائي إلا مالك الخطوة الحالي. و `Since 1.8` تُحفظ الملاحظة وصفًا للخطوة المُنشأة، وتظهر لاحقًا في تلميح بجانبها. ويستند الإدراج إلى لقطة الحالة التي قرأها، فيفشل بسبب تعارض بدلًا من أن يُلغي تغييرًا متزامنًا.

حالة اتفاقية مستوى الخدمة

يمكن أن يُحدَّد للخطوة وقت مستهدف للتنفيذ. ويُتتبَّع الالتزام باتفاقية مستوى الخدمة لكل خطوة من خلال حالة (`SLAStatus`):

الحالة	المعنى
`Pending`	ضمن الوقت المسموح به.
`Achieved`	اكتملت ضمن الوقت المستهدف.
`Overdue`	انقضى الوقت المستهدف قبل الاكتمال.

ويستند التصعيد والتذكيرات إلى تتبّع التأخّر. ويُجرى فحص التأخّر مرة كل دقيقة، لذا قد يتأخّر اكتشافه حتى دقيقة واحدة (وهو سلوك قديم ثابت). وتضع البوّابة الإلكترونية علامة **تجاوز الوقت** على الخطوة المتأخّرة في تبويب السجل للمُدخل. لضبط الأهداف، انظر: سير العمل والاعتمادات [بناء التطبيقات].

إجراءات تغيير الخطوة (الإدارية)

لا تُتاح إجراءات تغيير الخطوة إلا إذا فعّلها المعرض، ولكلٍّ منها رايتها الخاصة. فإذا كانت الراية معطّلة، يُرفض الاستدعاء بـ `ForbiddenWorkflowAdminActionException` . والأنماط هي (`ChangeStepModeEnum`):

تسمية البوّابة الإلكترونية	النمط	راية المعرض
الرجوع إلى خطوة سابقة	`ReturnToStep`	`returnToStep`
الانتقال لخطوة	`MoveToStep`	`moveToStep`
إعادة تعيين سير العمل	`ResetWorkflow`	`resetWorkflow`
إنهاء سير العمل	`EndWorkflow`	`closeWorkflow`

يعيد إجراء *الرجوع إلى خطوة* المُدخل إلى خطوة سابقة منفَّذة؛ وينقله إجراء *الانتقال إلى خطوة* إلى الأمام ويَسِم الخطوات المتخطّاة بـ `Skipped`؛ ويمسح إجراء *إعادة تعيين سير العمل* الخطوات المنفَّذة ويبدأ من جديد (ولا يُعرض حين يكون المُدخل في خطوته الأولى: *لا يمكن إعادة تعيين سير العمل إذا كانت الخطوة الحالية هي الخطوة الأولية*)؛ ويغلق إجراء *إنهاء سير العمل* سير العمل. ويقبل كلٌّ منها ملاحظة اختيارية. وتبقى قواعد التنقّل في المحرّك سارية: النقل اليدوي إلى الأمام فقط، والعودة إلى الخطوات المنفَّذة فقط، وعدم النقل المباشر إلى خطوة النهاية، والتنقّل داخل ما هو مفتوح من سير العمل للمُدخل فقط.

ومن الأذونات الموسَّعة المتعلقة بالمالك التي يستطيع المعرض منحها: **تغيير المالك كمسؤول** (مؤقتًا)، و**تغيير المالك كمسؤول بشكل دائم**، و**التنفيذ كمسؤول** (تنفيذ إجراء على خطوة لا يملكها المستدعي). ومن دون هذه الأذونات، يخضع المستدعي لقواعد المالك المعتادة. ولا يُطلب التحقق الإضافي من المسؤول عند التنفيذ كمسؤول.

`Since 1.8` يُعرَض كل إجراء لتغيير الخطوة، وكذلك **تغيير المالك**، معطَّلًا ما دامت الخطوة الحالية خارجية، بحيث لا تتيح هذه الإجراءات تخطّي التبليغ الإلزامي من النظام الخارجي؛ ويرفض الخادم الإكمال اليدوي للخطوة الخارجية في كل الأحوال. انظر الخطوة الخارجية.

تجاوز المعرض الإداري لا يسري إلا على غير المشاركين `Since 1.6`

يرفع المعرض الذي يُفعَّل فيه الإجراء الإداري **تعديل المدخل كمدير** أو **عرض المدخل كمدير** القيود التي تفرضها قواعد حقول الخطوة (مخفي، ومقفل، وإلزامي). ولأن هذا الإذن يشمل المعرض كله، كان يرفع القيود أيضًا عن الشخص الذي يعمل على الخطوة في ذلك المعرض. أمّا الآن فلا تسري إلا على من لا يشارك في الخطوة الحالية: أي من ليس مالكها، ولا عضوًا في المجموعة أو التجميعة أو الوحدة التنظيمية المالكة، ولا أحد المالكين الأصليين لخطوة لم يُطالَب بها. وتسري على المشارك قواعد الخطوة في جميع مواضع ذلك المعرض (التفاصيل، ونموذج التعديل، والحقول القابلة للتعديل، والسجل، والخلايا المخفية في القائمة) وعند الحفظ؛ ويتبع المُدخل الجديد المُنشأ هناك قواعد الخطوة الأولى. وقد أُتيح هذا التغيير أيضًا في فرعَي الإصدارين 1.6 و1.8 ضمن تحديث. النموذج: تجاوز المعرض الإداري لا يسري إلا على غير المشاركين.

المُدخلات المجمَّعة بعد انتهاء سير العمل المصدري `Since 1.8`

تُشتقّ المُدخلات المجمَّعة (الوهمية) من مُدخل مصدري. ومتى بلغ سير عمل المُدخل المصدري خطوةً لا يُسمح فيها بالتعديل (خطوته النهائية)، يختفي إجراء **تعديل** للصف المجمَّع من ترويسة المُدخل ومن إجراءات الصف في القائمة المجمَّعة معًا. وقبل 1.8، كان الإجراء يُعرض ويقتصر عمله على إعادة التوجيه إلى صفحة تعديل المُدخل المصدري، التي أصبحت للقراءة فقط بالفعل. وكانت الحقول المفردة مقفلة من قبل؛ أمّا الإجراء وحده فلم يكن قد خضع لهذا القيد بعد. للاطّلاع على جانب التصميم: الكيانات الوهمية [بناء التطبيقات].

الرسائل والرموز

ما يراه المشارك	الرمز
لست المالك	`InvalidWorkflowOwner`
سبق تنفيذ إجراء على خطوة متوازية	`ParallelStepOwnerAlreadyExecutedStepException`

ما يراه المشارك	الرمز
التعليق مطلوب	`WorkFlowActionCommentRequired`
الإجراء غير مسموح لهذا المُدخل	`WorkFlowActionExecutionNotAllowed`
الخطوة مملوكة بالفعل	`CURRENT_STEP_ALREADY_OWNED`
هذا المستخدم يملك هذه الخطوة بالفعل	`UserAlreadyOwnException`
غير مخوّل بتغيير المالك	`CanNotChangeOwnerException` / `ActionNotAllowed`
الإجراء الإداري ممنوع في المعرض	`ForbiddenWorkflowAdminActionException`
الإكمال اليدوي غير مسموح (خطوة خارجية) `Since 1.8`	انظر الخطوة الخارجية
رابط العودة غير صالح أو منتهي الصلاحية (خطوة خارجية) `Since 1.8`	انظر عقد العودة
هذا الإجراء يتطلب تحققًا إضافيًا من هويتك قبل التنفيذ. `Since 1.10`	التحقق الإضافي مطلوب
تم تغيير هذا السجل `Since 1.10`	`409 UPDATE_CONFLICT`
المُدخلات المحدّدة في خطوات مختلفة	`WorkflowStepsNotSameException`

أُزيل في `Since 1.10` تحذير «غير متاح لعدّة مُدخلات دفعة واحدة» عند التحديد المجمَّع الخاضع للتحقق. فقد أصبح تحقق واحد يغطّي التحديد: أمّا إذا كان التحديد كبيرًا جدًا، فتُعرض الرسالة: *هذا الإجراء يتطلب تحققًا إضافيًا، ويمكنه تغطية N مدخل كحد أقصى في المرة الواحدة. اختر عددًا أقل من المدخلات وحاول مرة أخرى.*

مرجع التعليقات والإشارات

تُوجَّه هذه الصفحة إلى مطوّري التكامل والمطوّرين الذين يقرؤون التعليقات والتفاعلات أو يكتبونها عبر الواجهة البرمجية، أو يحتاجون إلى شرح دقيق لما يراه المستخدم في تبويب **المناقشة** الخاص بالمُدخل. وهي توثّق الطبقة التقنية التي يقوم عليها دليل المستخدم النهائي التعليقات والتعاون [استخدام أسس]. أما المرجع المولَّد لنقاط النهاية فيرد في Comment – comments-service وReaction – comments-service، وتُوثَّق خدمة رموز المكالمات في الزمن الحقيقي (المقابس وRTC).

التخزين والترتيب

يرتبط التعليق دائمًا بمُدخل واحد؛ إذ يخزّن مرجع `relatedTo` المكوّن من كيان المُدخل (`dataEntity`) ومعرّفه (`dataEntry`). وتُحمَّل سلاسل التعليقات لكل مُدخل على حدة، فلا يظهر التعليق أبدًا في مُدخل آخر. وتُعاد التعليقات **من الأقدم إلى الأحدث** بحسب وقت إنشاء كل منها، ويتضمن كل تعليق النص، والكاتب (`commenter`)، والمرفقات، والجمهور إن وُجد، والوقت.

تُستخدم الخدمة في موضعين:

- **تبويب المناقشة في المُدخل**، إذا كانت المناقشات مفعّلة في المعرض؛ وهذه هي الحالة العامة لأي كيان.
- **منشورات الشريط الزمني.** في الكيان المدمج `POSTS` ، تمر التعليقات والتفاعلات عبر الخدمة نفسها، ويتوقف ذلك على تفعيل عرض الشريط الزمني في المعرض.

وإلى جانب السلسلة، يستطيع العميل طلب **قائمة المرفقات** (الملفات المرفقة بتعليقات المُدخل فقط) و**عدد التعليقات** (للشارات والعدّادات).

`Since 1.8` تحلّ الخدمة الكاتبَ والجمهورَ الموجَّه إليه، وكذلك المفوَّض له في التعليقات المنشورة نيابةً عن شخص آخر، قبل إعادة السلسلة، فتظهر الأسماء والصور دون حاجة إلى بحث إضافي.

`Since 1.10` لا يرتبط التعليق الذي ينشره زائر لم يسجّل الدخول (في معرض عام) بحساب كاتب، فيظهر باسم **مجهول** ولا يستطيع أحد تغيير جمهوره. ويظهر تعليق كهذا إلى جانب بقية صفحة المُدخل العامة، وكذلك نافذة إجراء سير العمل الخاصة بالزائر غير المسجّل في تلك الصفحة.

ملاحظات سير العمل في السلسلة

حين يكون إعداد **إظهار تعليقات سير العمل** (`showWorkflowComments`) في المعرض مفعّلًا، تُدمج كل ملاحظة من ملاحظات خطوات سير العمل (مَن نفّذ الخطوة ومتى) في السلسلة بوصفها تعليقًا للقراءة فقط، موسومًا بالخطوة التي صدرت عنها (سلوك قائم منذ زمن).

ملاحظات تغيير المالك `Since 1.10`

تُدمج في السلسلة نفسها، وبالإعداد نفسه، الملاحظةُ التي تُترك عند **إعادة إسناد** خطوة أو **تغيير مالكها**؛ ولم يكن ذلك يحدث قبل 1.10. فتغيير المالك ليس تنفيذًا لخطوة، ولذلك لم يكن قط جزءًا من السجل التاريخي للخطوات الذي تُبنى منه السلسلة. وتقرأ المنصة هذه الملاحظة من مصدر محدود النطاق، هو `GET …/entries/{id}/owner-change-notes` ، الذي لا يُعيد إلا سجلات تغيير المالك التي تتضمن ملاحظة، ثم تعرضها كما تعرض ملاحظة الخطوة.

ولأن الملاحظة تُجمَّع عند القراءة ولا تُخزَّن بوصفها تعليقًا، فإن نشرها **لا** يُطلق إشارة «تحديث التعليقات» الفورية التي يطلقها نشر التعليق العادي، ولا يتلقّاها تبويب المناقشة المفتوح مسبقًا في نافذة أخرى إلا حين يعيد ذلك التبويب جلب السلسلة (بإعادة فتح المُدخل، أو بالانتقال إلى تبويب آخر ثم العودة، أو بالتحديث الخاص بتلك الواجهة).

الكتابة والتحرير

يقبل التعليق الجديد ثلاثة عناصر:

ما يقدّمه العميل	الحقل	مطلوب
نص الرسالة	`commentText`	لا (قد يكون فارغًا أو null)
ملفات للإرفاق	`attachments` (لكل منها `id` و`name` و`mimeType` و`size`، و`thumbnail` اختياريًا)	لا
الأشخاص أو المجموعات الموجَّه إليهم	`audience` (مراجع تُقسَّم داخليًا إلى مستخدمين ومجموعات)	لا

تسجّل الخدمة الكاتب والوقت تلقائيًا، ولا يحددهما العميل أبدًا، ويُربط التعليق بالمُدخل المحدد في المسار.

التحرير مقصور على كاتب التعليق نفسه، وتُرفض طلبات أي مستدعٍ آخر. ولا يتغير إلا **الجمهور**، فعملية التحرير لا تغيّر النص. وإفراغ الجمهور يعيد إتاحة التعليق للجميع (مع مراعاة قاعدة التعليقات الخاصة أدناه).

الحذف. لا توفّر الواجهة البرمجية للتعليقات عملية حذف للمستخدم النهائي؛ فالتعليقات تُدار عبر دورة حياة البيانات في المنصة، حيث يكون التعليق المُزال في حالة مؤرشفة ويُستبعد من السلاسل، أما التفاعلات فيمكن إزالتها مباشرة.

المرفقات

يُتحقَّق من المرفقات وفق إعدادات المرفقات في المعرض؛ فلا توجد قائمة ثابتة بالأنواع المقبولة، بل تُهيَّأ الأنواع المسموح بها لكل معرض.

- **الحجم.** يجب ألا يتجاوز كل ملف الحد الأقصى المهيَّأ. فإن لم يحدد المعرض حدًا، كان الحد الأقصى الافتراضي **1 MB (1,048,576 بايت)** لكل ملف.
- **الأنواع.** تُفحص الملفات وفق قائمة الامتدادات المسموح بها في المعرض؛ فإذا فعّل المعرض المرفقات **دون** أن يهيّئ أي امتداد مسموح به، رُفضت الملفات. ويمكن أن يكون عنصر القائمة امتداد ملف (`.pdf`)، أو نوع MIME محددًا (`image/png`)، أو نوع MIME يتضمن حرف بدل (`image/*`).
- وإذا لم يفعّل المعرض المرفقات، رُفض أي ملف.

وإذا أخفق التحقق من ملف واحد، رُفض النشر كله، مع خطأ يحدد المرفق بحسب موضعه. وتعرض البوابة الرسالة *تعذّر رفع الملف. تحقق من متطلبات رفع الملف وأعد المحاولة.*

`Since 1.10` يُحفظ الملف المرفق في تخزين مؤقت، ولا يُحتفظ به إلا بعد نشر التعليق؛ أما الملف المرفق بتعليق لم يُنشر قط فلا يُحتفظ به. انظر الملفات: قواعد التخزين.

الإشارات

حين يسمح المعرض بالإشارات، يُظهر مربع التعليق في البوابة اقتراحات الإكمال التلقائي عند كتابة **@**. ويبحث الإكمال التلقائي للإشارات بين المستخدمين والمجموعات **النشطين** باستخدام اسم المجموعة والاسم الكامل للشخص (بكل لغة مهيَّأة)، وعنوان البريد الإلكتروني للشخص أو اسم المستخدم، وتقتصر النتائج على عدد قليل من الاقتراحات.

ويتوقف من يمكن اقتراحه على **نطاق الإشارة** في المعرض، الذي يحدده باني التطبيق:

الوضع	من يمكن الإشارة إليه
جميع المستخدمين	أي مستخدم أو مجموعة نشطة.
حسب إذن المعرض	المستخدمون والمجموعات الذين يملكون إذن الوصول إلى المعرض فقط (يُستخدم حين يكون المعرض مقيَّدًا).
إشارة مخصصة – ثابت	قائمة ثابتة يختارها باني التطبيق.
إشارة مخصصة – حقل	مستخدمون ومجموعات مستمدّون من حقول في المُدخل، كحقل المكلَّفين مثلًا.

لا يستطيع الزوار المجهولون (الذين لم يسجّلوا الدخول) الإشارة إلى أحد، حتى في المعارض التي تكون فيها الإشارات مفعّلة.

`Since 1.8` يستخدم بُناة التطبيقات والمديرون الذين يضيفون تعليقًا من لوحة الإدارة (back office: الشريط الجانبي للمُدخل في الإدارة، لا البوابة) البحثَ نفسه عن الإشارات والجمهور. وقبل هذا الإصلاح، كان البحث من لوحة الإدارة يُعيد دائمًا الخطأ «ليس لديك إذن وصول»، لأن الفحص الوحيد المتاح كان فحص البوابة القائم على المعرض، وهو لا ينطبق خارج المعرض. أما الآن فيشترط البحث من لوحة الإدارة أن يملك المستدعي صلاحية **عرض** على الأقل في موضع ما من لوحة الإدارة، عبر حسابه أو إحدى مجموعاته.

كيف تُخزَّن الإشارة

عند النشر، تفحص الخدمة النص بحثًا عن علامات إشارة بالصيغة `@<uuid>` (معرّف شخص أو مجموعة محاط بـ `@<…>`)، وتربط كل علامة بمستخدم أو مجموعة فعليين، ثم تخزّن مع التعليق ثلاث قوائم: **الإشارات** مجتمعةً، و**المستخدمون المشار إليهم**، و**المجموعات المشار إليها**. ويستند الإشعار إلى القائمتين الأخيرتين.

`Since 1.10` تظهر الإشارة التي لم يعد ممكنًا ربطها بحساب أو مجموعة (حُذفت أو عُطّلت لاحقًا) بالصيغة `@Unknown` (وبالعربية "*غير معروف*") بدلًا من أن تعطّل الصفحة. وكانت إشارة كهذه تتسبب سابقًا في ظهور صفحة المُدخل كلها فارغة إلى أن يُعاد تحميل الصفحة. كما تُحلّ أسماء الإشارات لحظة نشر التعليق، فيظهر اسم من أُشير إليه للتوّ فورًا.

إشعار الإشارة

تتضمن المنصة مشغّلًا مدمجًا للتعليقات ينفّذ الإجراء **إرسال إشعار** إلى المستخدمين والمجموعات المشار إليهم عبر قناة الإشعارات داخل التطبيق، بعنوان **إشارة** ونص **قام {{person}} بالإشارة إليك في تعليق**، حيث `{{person}}` هو اسم الكاتب. أما الإشارات والإكمال التلقائي وأوضاع القائمة فمتاحة منذ زمن.

`Since 1.10` في التثبيت الذي يستضيف عدة مؤسسات، كان من الممكن سابقًا ألا تُشعر الإشارة **أحدًا**، لأن كيانات التعليقات ومشغّل الإشارة المدمج لم تكن تُهيَّأ لكل مؤسسة. أما الآن فتُنشأ لكل مؤسسة، سواء أكانت موجودة أم تُضاف لاحقًا، ولم يتغير شيء في طريقة الكتابة أو الإشارة لدى المستخدمين.

الجمهور (التعليقات الموجَّهة)

يمكن توجيه التعليق إلى **جمهور**: مستخدمين أو مجموعات أو كليهما، تُخزَّن مراجعهم في قائمتين منفصلتين للمستخدمين والمجموعات. ولا يرى التعليق الموجَّه إلا كاتبه والمستخدمون المستهدفون وأعضاء المجموعات المستهدفة، أما التعليق الخالي من الجمهور فيراه كل من يستطيع رؤية السلسلة. ولا يعرض مربع التعليق في البوابة منتقي الجمهور إلا إذا سمح المعرض بذلك.

التعليقات الخاصة (مقصورة على المعرض الحالي)

يستطيع المعرض إبقاء تعليقاته خاصة (`limitCommentsToCurrentView` ، ويظهر لباني التطبيق بوصفه خيارًا يقصر الرسائل المنشورة على المعرض). فتُوسَم التعليقات المنشورة فيه بأنها خاصة (`commentIsPrivate`)، ويُسجَّل معها المعرض الذي أُنشئت فيه. وعند قراءة السلسلة، تُعاد التعليقات التي نُشرت بصفة خاصة في المعرض نفسه، إضافةً إلى كل التعليقات غير الخاصة من أي معرض، مع استمرار تطبيق قواعد الجمهور. ومن ثَمّ يمكن أن يتضمن المُدخل الواحد مناقشات منفصلة، كل منها مقصور على معرضه (سلوك قائم منذ زمن).

التفاعلات

في منشورات الشريط الزمني وغيرها من المُدخلات التي تكون التفاعلات مفعّلة فيها، يكون نوع التفاعل الوحيد المعرَّف هو **Like** (الإعجاب).

- تحلّ إضافة تفاعل أو تغييره محل تفاعل المستدعي السابق؛ فلكل مستخدم تفاعل واحد على الأكثر في كل مُدخل، وتسجّل الخدمة الهوية والوقت تلقائيًا.
- تحذف إزالة التفاعل تفاعلَ المستدعي نفسه فقط.
- يستطيع العميل سرد تفاعلات المُدخل والحصول على عددها.

ويُسجَّل عند التفاعل مع منشور في الشريط الزمني أيضًا ظهورٌ (impression) للمنشور، بشرط أن يكون المنشور قد نُشر وأن يكون الشريط الزمني نشطًا في المعرض.

المكالمات على المُدخل

يفتح الإجراء **انضم** في البوابة (ويظهر في وضع العرض على المُدخلات غير المسودّة التي فُعّل الزمن الحقيقي في كيانها) مكالمةً يكون معرّف قناتها هو معرّف المُدخل، فيشترك في المكالمة نفسها كل من ينضم من المُدخل ذاته. وتطلب البوابة نيابةً عن المستخدم رمزًا للقناة من خدمة RTC التي تُصدر رموز Agora. ويحدد طلب الرمز **القناة**، و**الدور** (**publisher** الذي يرسل الصوت والفيديو، أو **audience** الذي يستقبل فقط)، و**مدة الصلاحية** التي تكون افتراضيًا **3,600 ثانية (ساعة واحدة)**. ويُربط الرمز بمعرّف المستخدم والقناة، ويجدّده العميل قبل انتهاء صلاحيته. وتُرفض الطلبات التي لا تحدد قناة، أو لا تصدر عن مستخدم معرَّف، أو تتضمن دورًا غير معروف. والمكالمات متاحة على الويب فقط. تفاصيل الخدمة: الزمن الحقيقي (المقابس وRTC).

رسائل البريد الإلكتروني للمُدخل

تُجمَّع في المُدخل رسائل البريد الإلكتروني التي أرسلتها المنصة بشأنه. ويسرد تبويب **بريد الكتروني** في المُدخل (حين يُظهره المعرض) هذه الرسائل لكل مُدخل، ويتيح فتح كلٍّ منها بمحتواها ومرفقاتها؛ وهو سجل تاريخي للقراءة فقط لا يمكن إنشاء رسائل منه.

ضوابط الميزات والوصول

تحدد **تهيئة المعرض** (التي يضبطها باني التطبيق) الميزات المتاحة:

- **إظهار المناقشة** – شرط لرؤية التعليقات أو إضافتها أصلًا في المُدخلات غير التابعة للشريط الزمني؛ أما في الشريط الزمني `POSTS` فيجب بدلًا من ذلك تفعيل عرض الشريط الزمني.
- **السماح بالتعليقات** – شرط لكتابة التعليقات النصية.

- **السماح بإرسال المرفقات** – شرط لإرفاق الملفات، وله حدوده الخاصة للحجم والنوع.
- **السماح بالإشارة إلى الأشخاص** – شرط للإشارة، ويجب أن يكون المستخدم مسجّل الدخول.

يحدد **وصول المستدعي إلى المُدخل** إمكان القراءة والكتابة:

- تتطلب قراءة تعليقات المُدخل صلاحية **عرض** عليه على الأقل، ممنوحة على مستوى النظام أو الوحدة أو الكيان.
- وتتطلب إضافة تعليق أو تفاعل صلاحية **تحرير**، على المستويات نفسها.

`Since 1.8` تراعي قراءة التعليقات المنح على **مستوى الوحدة ومستوى الكيان**، بدلًا من الاقتصار على وصول المستعرض على مستوى النظام، فيستطيع المستخدم المقيَّد الذي يملك منحة على الوحدة أو الكيان رؤية المناقشة دون دور شامل على مستوى النظام.

وضع المعاينة `Since 1.8`

حين يعاين باني التطبيق معرضًا ليست فيه بيانات حقيقية، تعرض سلاسل التعليقات محتوى بديلًا: سلسلة واقعية على المُدخل الحقيقي، بمعلّقين ونصوص تجريبية، ليتسنى تقييم التخطيط. ولا يمس ذلك التعليقات الحقيقية أبدًا.

نقاط النهاية

تتطلب جميع نقاط نهاية التعليقات والتفاعلات المصادقة، وتخضع لفحوص المعرض والوصول المذكورة أعلاه. وتوفّر خدمة التعليقات مسارات التعليقات والتفاعلات ضمن المسار الأساسي لواجهة التعليقات البرمجية (مثل `/api/comments`)، بينما توفّر خدمة RTC مسار الرمز.

الغرض	الطريقة والمسار
سرد تعليقات المُدخل	`GET /v1/{entity}/{relatedToId}`
سرد مرفقات التعليقات	`GET /v1/{entity}/{relatedToId}/attachments`
عدّ تعليقات المُدخل	`GET /v1/{entity}/{relatedToId}/count`
إضافة تعليق	`POST /v1/{entity}/{relatedToId}`
تحرير جمهور التعليق	`PATCH /v1/{entity}/{relatedToId}/{commentId}`
الإكمال التلقائي للإشارات	`GET /v1/comments/{entityId}/{entryId}/mentions-list?keyword=…`
الإكمال التلقائي للجمهور	`GET /v1/comments/audience-list?keyword=…`
سرد التفاعلات على المُدخل	`GET /v1/reaction/{entity}/{relatedToId}`
عدّ التفاعلات	`GET /v1/reaction/{entity}/{relatedToId}/count`
إضافة تفاعل المستدعي أو تغييره	`PUT /v1/reaction/{entity}/{relatedToId}`
إزالة تفاعل المستدعي	`DELETE /v1/reaction/{entity}/{relatedToId}`
الحصول على رمز مكالمة (Agora)	`GET /api/rtc/{channel}/{role}?expiry=…`

الحقول الحسّاسة والمشفَّرة: النموذج والواجهة البرمجية والإنفاذ

هذه الصفحة موجّهة إلى **المُكامِلين ومراجعي الأمن ومهندسي المنصة**. وتوثّق كيفية تخزين الحقول الحسّاسة والمشفَّرة، ومنح الوصول إليها، وتقنيعها، وكشف قيمها، وإنفاذ قيودها، إلى جانب نقاط النهاية ذات الصلة وحالات الرفض الدقيقة التي يجب أن يعالجها العميل.

تحمي حقلًا من الواجهة؟

ينبغي لبُناة التطبيقات البدء من حماية حقل [بناء التطبيقات]. ويمنح مسؤولو القاطن الوصول من الأذونات الحساسة [الإدارة]. ويضبط المشغّلون مفاتيح التشفير في مفاتيح تشفير الحقول ومزوّدوها.

Since 1.8 – أُطلقت ميزة الحقول الحسّاسة (وضع علامة على قيم الحقول الفردية، وإخفاؤها، والتحكّم في الوصول إليها لكل مستخدم على حدة) ضمن تغيير واحد في سلسلة إصدارات 1.8. **Since 1.10** – الحقول المشفَّرة، Decrypt، وآلية التحقق الإضافي ذات الأغراض الثلاثة، وسياسة الوصول إلى البيانات الحسّاسة.

تحتوي بعض الحقول على معلومات لا ينبغي أن يطّلع عليها سوى عدد محدود من الأشخاص، مثل الرواتب والأرقام الوطنية والحسابات البنكية وما شابهها. تتيح أسس لمصمّم كيان البيانات وضع علامة على هذا الحقل بوصفه **حساسًا**؛ وبمجرد أن يصبح الحقل حسّاسًا، تتولّى أسس **بإخفاء** قيمته حجبها عن كل مستخدم لم يُمنح صراحةً حق الوصول إليه، سواء في المُدخلات الحيّة أو في السجل التاريخي للتدقيق (audit history)، ولا يستطيع قراءة القيمة الحقيقية إلا المستخدمون الذين يملكون **منحة وصول صريحة إلى الحقل الحساس**.

يمثّل هذا **بُعدًا منفصلًا** عن الأذونات الاعتيادية؛ فقد يملك المستخدم وصولًا كاملًا إلى كيان ما وإلى مُدخلاته، ومع ذلك تظلّ قيمة الحقل الحسّاس مُخفاة عنه إلى أن يُمنح الوصول إلى ذلك الحقل تحديدًا. ويُتحكَّم في الوصول عبر منح خاصة بكل حقل ولكل مستخدم/مجموعة على حدة، تُخزَّن في مجموعة `access_right`، وليس عبر أذونات المعرض/الصف الخاصة بالكيان نفسه، **ولا** تستطيع أدوار المسؤول (admin) تجاوزها.

`Since 1.10` يمكن بدلًا من ذلك وضع علامة **مشفَّر (Encrypt)** على الحقل، وهي آلية مختلفة تستند إلى نموذج المنح نفسه: تُخزَّن القيمة في قاعدة البيانات على هيئة نص مشفَّر بدلًا من إخفائها عند الإخراج، وتُكشف قيمة حقل واحد في مُدخل واحد في كل مرة عبر استدعاء فك التشفير المخصَّص، الذي يتطلّب، `Since 1.10`، إضافةً إلى المنحة، تحققًا إضافيًا (step-up) أحادي الاستخدام. ويكون الحقل إمّا حسّاسًا **أو** مشفَّرًا، ولا يجمع بين الحالتين. انظر الحقول المشفَّرة.

المفهوم والمصطلحات

- **الحقل الحسّاس** – هو `DataEntityField` تكون قيمة `options.sensitive === true` فيه. تتأهّل الحقول ذات النوع القِيَمي (scalar) فقط؛ أما الحقول المرجعية (Entity) وحقول حاويات الكتل (Block) فتُرفَض (انظر القيود).
- **النطاق (Scope)** – السلسلة النصية المرجعية التي تُعرّف حقلًا حسّاسًا واحدًا عبر المساحة (space)، وبنيها `buildSensitiveFieldScope(moduleNn, entityNn, fieldFullNn)`. انظر نطاقات الحقول الحساسة.
- **المنحة (Grant)** – مُدخل `AccessRight` من نوع الإذن `SENSITIVE_FIELD` (`PermissionTypeEnum.SensitiveField =`) تساوي قيمة `value.scope` فيه نطاقَ حقل، وتخوّل مستخدمًا أو مجموعةً واحدة قراءة ذلك `'SENSITIVE_FIELD'`

الحقل.

- **المنحة المباشرة مقابل المنحة الموروثة** – المنحة على النطاق الدقيق للحقل نفسه *مباشرة*؛ أما المنحة على نطاق بالرمز البديل يشمله فهي *موروثة*. انظر المنح المباشرة والموروثة.
- **القناع (Mask)** – القيمة التي تُستبدَل بها قيمةٌ لا يجوز للمستخدم الطالب قراءتها. تستبدل القراءة الحيّة حارسًا (sentinel) غير قابل للتخمين من 26 محرفًا (`SENSITIVE_DISPLAY_MASK`)، لا السلسلة البسيطة `****` التي وصفتها نسخة سابقة من هذه الصفحة – انظر الأقنعة لسبب أهمية هذا الفارق. أما حمولات التدقيق فتستخدم حارسًا منفصلًا وأبسط هو `***` .
- **جلسة الكشف (Unmask session)** – جلسة قصيرة العمر يعاد فيها التحقق ("step-up")، وتتحكّم إضافةً إلى ذلك في ما إذا كانت القيمة *الممنوحة* ستُكشف فعلًا عند القراءة.
- **الحقل المشفَّر (Encrypted field)** `Since 1.10` – هو `DataEntityField` تكون قيمة `options.encrypt === true` فيه. تُخزَّن قيمته نصًّا مشفَّرًا (AES-256-GCM) لا مجرّد مُقنَّعة عند القراءة، وهو متبادل الحصرية مع `sensitive` على الحقل نفسه. انظر الحقول المشفَّرة.
- **فك التشفير (Decrypt)** `Since 1.10` – مسار الكشف الوحيد للحقل المشفَّر: استدعاء واحد، لحقل واحد، على مدخل واحد، مقيَّد بمنحة الحقل **و**إثبات تحقق إضافي (step-up) لمرة استخدام واحدة (`Since 1.10`). انظر فك تشفير قيمة.
- `StepUpPurpose` `Since 1.10` – المُميِّز ثلاثي الأغراض (`Unmask` / `Decrypt` / `WorkflowAction`) الذي يُحدَّد عند طلب التحقق الإضافي؛ تتشارك الأغراض الثلاثة نقاط النهاية نفسها، لكن إثباتًا أو جلسة صادرة لغرض واحد لا يمكن أن تفي بغرض آخر أبدًا. انظر كشف القناع بالتحقق الإضافي، وDecrypt، وWorkflowAction.
- **سياسة الوصول إلى البيانات الحساسة** `Since 1.10` – إعداد على مستوى العنصر يُضبَط على تفاعل أو قالب تصدير أو خلاصة، ويقرّر ما الذي يجوز *لذلك العنصر* كشفه عند تشغيله، بصرف النظر عمّن يقرأ ناتجه. انظر سياسة الوصول إلى البيانات الحساسة.

نطاقات الحقول الحساسة

يُعرَّف كل حقل حسّاس عبر المساحة (space) بأكملها بسلسلة **نطاق (scope)** مرجعية واحدة. ويُبنى النطاق من الأسماء المُنمّطة (أسماء الآلة)، مفصولةً بالرمز `/` :

```
<module>/<entity>/<field>
```

وبالنسبة إلى حقل يقع داخل **كتلة (block)**، يكون المقطع الأخير هو الاسم المُنمّط ذا الصيغة المنقوطة `block.subfield` – مثل `hr/employee/payroll.iban` .

تحمل المنحة نطاقًا كما يحمل الحقل نطاقًا، وتُقارَن السلسلتان **مقطعًا بمقطع**. ويجب أن يكون لكليهما العدد نفسه من المقاطع المفصولة بالرمز `/` كي تحصل المطابقة أصلًا. ويجوز أن يكون أي مقطع هو الرمز البديل (wildcard) `*` ، فيطابق أيًّا كان ما يقع في ذلك الموضع:

نطاق المنحة	ما الذي يشمله
`hr/employee/salary`	ذلك الحقل وحده تحديدًا – منحة **مباشرة**.
`hr/employee/payroll.iban`	الحقل الفرعي `iban` داخل الكتلة `payroll` في الكيان `employee` .
`hr/employee/*`	كل حقل حسّاس في الكيان `employee` – منحة **بالرمز البديل على مستوى الكيان**.
`hr/*/*`	كل حقل حسّاس في أي موضع ضمن الوحدة `hr` – منحة **بالرمز البديل على مستوى الوحدة**.
`hr/*/salary`	الحقل `salary` في كل كيان ضمن الوحدة `hr` .

المنحة التي يسمّي نطاقُها حقلًا بعينه هي منحة **مباشرة** على ذلك الحقل؛ أما المنحة التي تصل إلى الحقل عبر مقطع بالرمز البديل فهي منحة **موروثة**. وكلتاهما تمنحان حق القراءة نفسه تمامًا – والفارق هو *أين* تقيم المنحة، ومن ثمّ أين تذهب لتعديلها أو إلغائها.

ولا تُكتب النطاقات يدويًا. إذ يعرض مُحدِّد النطاق المساحةَ على هيئة شجرة **وحدة › كيان › حقل**: فاختيار حقل طرفي يؤلّف النطاق الملموس ذا المقاطع الثلاثة، بينما يؤلّف اختيار عقدة كيان أو عقدة وحدة النطاقَ المقابل بالرمز البديل الذي يشمل كل ما تحتها.

تُنفَّذ المطابقة مقطعًا بمقطع عبر `matchesSensitiveScope` (`libs/sensitive-field.ts`)، ويمرّ بها كل فحص أثناء التشغيل.

تعليم حقل: نقاط نهاية تعريف الحقل

لا توجد نقطة نهاية مخصّصة *لتبديل* حالة الحساسية في حقل؛ إذ تُوضَع العلامة عبر محرّرات تعريف الحقل/الكيان ونقاط النهاية المعتادة لها.

ملاحظة: تدعم الحقول الحسّاسة الحقولَ ذات المستوى الأعلى والحقول الفرعية داخل الكتل القابلة للتكرار. أما حقل الكتلة *الحاوية* نفسه فلا يمكن وضع علامة عليه بوصفه حسّاسًا (انظر القيود).

تتضمّن نقاط نهاية التعريف في مُتحكّم الحقل (`data-entity-fields.controller.ts`) منطق الحساسية:

الطريقة	المسار	السلوك بالنسبة للحقول الحسّاسة
`POST`	`/v1/entities/{entity}/fields`	`validateAndNormalizeSensitive` ؛ إذا كانت `options.sensitive` مفعَّلة، يُستدعى `syncSensitiveFieldAccess(scope, users, groups)` ، وإلا فيُستدعى `removeSensitiveFieldAccess(scope)` ؛ ثم يُنفَّذ `delete field.sensitiveAccess` قبل الحفظ.
`PATCH`	`/v1/entities/{entity}/fields/{fieldId}`	منطق المزامنة/الإزالة وحذف الخاصية نفسه.
`DELETE`	`/v1/entities/{entity}/fields/{fieldId}`	`removeSensitiveFieldAccess(scope)` للحقل المحذوف.
`GET`	`/v1/entities/{entity}/fields` و `/fields/{fieldId}`	يعيد `attachSensitiveFieldsAccess` ملء `sensitiveAccess` لكل حقل حسّاس من `access_right` قبل إرجاع النتيجة.

منح الوصول إلى حقل حساس

يُمنح الوصول إلى الحقل الحسّاس **على مستوى كل نطاق**، بصورة مستقلة عن مستويات الأذونات الاعتيادية. فهو نوع مستقل من المنح؛ لذا فإن منح المستخدم إذن التعديل على الكيان **لا** يمنحه وصولًا إلى حقوله الحسّاسة.

توجد طريقتان للكتابة إلى مُدخلات `AccessRight` نفسها:

1. **محرّر الحقل** – افتح الكيان، وحدّد في لوحة الوصول/خيارات الحقل المستخدمين أو المجموعات الذين ينبغي أن يتمكنوا من قراءة القيمة الحقيقية للحقل. وعلى مستوى التنفيذ، أرسِل الحقل مع `options.sensitive = true` ومصفوفتَي المراجع المطلوبتين `sensitiveAccess.users` / `sensitiveAccess.groups` . ومجموعة المنح **تصريحية** (declarative): إذ يقارن `syncSensitiveFieldAccess` المستخدمين والمجموعات المطلوبين بالمنح النشِطة (Active) الموجودة للنطاق، فيحذف القديمة منها وينشئ الناقصة، ثم ينشر `AccessRightChangedEvent` . ويؤدّي إرسال تحديد فارغ إلى إبطال جميع المنح لذلك الحقل، ولا تحمل المنح المنشأة بهذه الطريقة أي `expiry` .

2. **واجهة إدارة المنح** – `/v1/sensitive-field-grants*`، المضافة في الإصدار 1.8، وهي الواجهة التي تستند إليها صفحة **الأذونات الحساسة** في الإدارة وتبويبا المستخدم والمجموعة. وتتيح إنشاء نطاقات تتضمّن رمزًا بديلًا وتحديد فترات صلاحية، وهو ما لا يتيحه محرّر الحقل. انظر واجهة API لمنح الحقول الحساسة، وانظر الأذونات الحساسة [الإدارة] للاطلاع على الإجراء الموجّه للمسؤول.

تُشغَّل المزامنة التصريحية نفسها أيضًا على مستوى الكيان كاملًا عبر `DataEntityService.syncSensitiveFieldsAccess` (الذي تستخدمه معالِجات أوامر إنشاء/تحديث كيان البيانات). ويزيل هذا أيضًا المنح الخاصة بالحقول التي لم تعد حسّاسة أو التي أُزيلت كليًا (باستخدام لقطة الكيان السابقة)، بحيث لا تتبقّى أي منح يتيمة.

المنح المباشرة والموروثة `Since 1.8`

المنحة التي يكون نطاقها المسار الدقيق `module/entity/field` للحقل هي منحة **مباشرة**، أما المنحة على النطاق `module/entity/*` أو `module/*/*` فهي **موروثة**: إذ تشمل كل حقل حسّاس ضمن ذلك الكيان أو تلك الوحدة، بما في ذلك الحقول التي أُنشئت بعد كتابة المنحة.

يختلف هذا المحور عن *كيفية اكتساب* المستخدم للمنحة: إذ يعرض `GET /v1/sensitive-field-grants/by-user/{userId}` طريقة اكتسابها بصورة منفصلة عبر `assignmentType` (`Direct` أو `group`). وبذلك يمكن أن تكون المنحة الواحدة مباشرةً من حيث النطاق ومُكتسَبةً عبر مجموعة في آنٍ واحد. ويجيب المحوران عن سؤالين مختلفين:

المحور	السؤال	أين يُبلَّغ عنه
شكل النطاق (مباشر مقابل موروث)	*ما مدى اتساع نطاق هذه المنحة؟*	سلسلة النطاق نفسها؛ ويسرد `inherited-holders` أصحاب المنح ذات الرمز البديل لحقل واحد.
نوع الإسناد (`Direct` مقابل `group`)	*لمن تعود هذه المنحة؟*	`UserSensitiveFieldGrantDto.assignmentType` / `acquiredThrough`.

ويُختزَل كلا المحورين إلى سلاسل نطاق بسيطة عند الإنفاذ: فلا يرى `canAccessSensitiveField` سوى `permissions.sensitiveFields`، ولا يستطيع التمييز بين منحة مباشرة وأخرى موروثة.

واجهة API لمنح الحقول الحساسة `Since 1.8`

`SensitiveFieldGrantsController` (`controllers/sensitive-field-grants.controller.ts`) طبقة تفويض خفيفة فوق `AccessRightsService`، وهي **لا تُدخِل تخزينًا جديدًا**: إذ يظلّ `AccessRight` المرجع الوحيد للحقيقة، ولا تعدو هذه المسارات كونها صيغًا جديدة لقراءة المُدخلات نفسها التي يكتبها مسار مزامنة محرّر الحقل والكتابة فيها.

التحقق من الصلاحية (في كل مسار): `@authorize({voters: [systemAdminVoterFactory()]})`، ويُجري التصويت بحسب فعل HTTP: إذ يتطلّب `GET` صلاحية **Viewer** على مستوى النظام أو أعلى، بينما تتطلّب `DELETE` / `PATCH` / `POST` استيفاء `isSystemDeveloper()` (Developer أو Editor أو Admin على مستوى النظام). أما صفحة الإدارة نفسها، فتفرض الواجهة الأمامية عليها قيدًا أضيق، إذ تقتصر على مسؤولي النظام (Admin).

الطريقة	المسار	ما تُرجعه
`GET`	`/v1/sensitive-field-grants`	`SensitiveFieldGrantDto[]` – كل منحة نشطة على مستوى المساحة.
`GET`	`/v1/sensitive-field-grants/by-user/{userId}`	`UserSensitiveFieldGrantDto[]` – منح المستخدم المباشرة والموروثة عبر مجموعاته.
`GET`	`/v1/sensitive-field-grants/by-group/{groupId}`	`SensitiveFieldGrantDto[]` – منح المجموعة نفسها (المباشرة).

الطريقة	المسار	ما تُرجعه
`GET`	`/v1/sensitive-field-grants/inherited-holders`	`InheritedSensitiveFieldHoldersDto` – الأطراف التي تحوز منحة برمز بديل تشمل حقلًا واحدًا.
`POST`	`/v1/sensitive-field-grants`	`CreateSensitiveFieldGrantsResultDto` – المنح المُنشأة + المكرّرات المُتخطّاة.
`PATCH`	`/v1/sensitive-field-grants/{id}`	`SensitiveFieldGrantDto` – تحديث المدة فقط.
`DELETE`	`/v1/sensitive-field-grants/{id}`	`204`، حذف نهائي.
`PATCH`	`/v1/sensitive-field-grants/{id}/archive`	`204`، أرشفة ناعمة.

`GET /v1/sensitive-field-grants` يسرد كل منحة `Active` (غير مؤرشفة) في المساحة. وتطابق عملية تجميع واحدة منح `SENSITIVE_FIELD`، وتُجري `$lookup` على المستخدم أو المجموعة المستحِقّة لاستخراج تسمية للعرض، وتُثري كل صفّ بقيم `scopeLabels` مُعرَّبة. وهو **غير** مُرشَّح عمدًا بحسب نافذة الصلاحية؛ فالمنحة التي لم تبدأ بعد يجب أن تظلّ ظاهرة للمدير كي يراها ويعدّلها. ويحلّ `SensitiveFieldGrantScopeLabelsDto` كل مقطع من النطاق على حدة: فإذا كان المقطع `*`، أو اسمًا مُنمّطًا غير موجود في الكتالوج (لحقل محذوف مثلًا)، كانت نتيجة ذلك المقطع `null` بدلًا من التسبّب في فشل القائمة بأكملها.

`GET .../by-user/{userId}` يُرجع كل منحة سارية تشمل المستخدم فعليًا – أي المنح المباشرة والمنح الموروثة من كل مجموعة ينتمي إليها. و**لا توجد إزالة للتكرار**: فالنطاق الذي يملكه المستخدم بالطريقتين يُنتج صفَّين، يميّزهما `assignmentType` (`UserPermissionAssignmentTypeEnum.Direct = 0` / `.group = 1`). وفي الصف غير المباشر، يحدّد `acquiredThrough` (مرجع مجموعة جرى حلّه) و`acquiredThroughGroupType` (`GroupTypeEnum`) المجموعة التي اكتُسبت المنحة عبرها؛ وتكون قيمة كلٍّ منهما `null` في المنحة المباشرة.

`GET .../by-group/{groupId}` يُرجع منح المجموعة المباشرة فقط – فالمجموعة لا ترث من مجموعة أخرى – ولذلك يعيد استخدام `SensitiveFieldGrantDto` البسيط دون بيانات وصفية عن نوع الإسناد.

`GET .../inherited-holders?moduleNn=&entityNn=&fieldNn=` يسرد المستخدمين والمجموعات القادرين على قراءة حقل واحد عبر منحة برمز بديل، مقابل منحة تطابق نطاق الحقل نفسه بدقة (وتلك المجموعة هي مُحدِّد `sensitiveAccess` الخاص بالحقل، وهي مستبعَدة هنا عمدًا).

- معاملات الاستعلام الثلاثة **إجبارية** جميعها؛ فقد كانت نسخة سابقة تسمح، عند إغفال معامل، ببناء نطاق مثل `undefined/employee/*`، فلا يطابق شيئًا ويُرجع `200 {holders: []}` – ويفسّره مربع الحوار على أنه «لا أحد يرث هذا الحقل»، بدلًا من فشل الطلب بالرمز `400`.
- ويُقبَل `fieldNn` لتوضيح العقد، لكنه **لا** يُستخدم في المطابقة؛ إذ يقتصر الاستعلام على `buildSensitiveFieldScope(moduleNn, entityNn, '*')` و`buildSensitiveFieldScope(moduleNn, '*', '*')`.
- ولا تُحتسب إلا المنح السارية (`state != Archived`، وضمن نافذة الصلاحية النشطة). وإذا شمل النطاقان الطرف نفسه، أُزيل التكرار ليظهر في صف واحد.
- ويحمل كل `InheritedSensitiveFieldHolderDto` القيم `type` (`USER` / `GROUP`) و`id` واسمًا جرى حلّه في `name`، ويحمل – للمستخدمين فقط – `email` و`avatar`.

`Since 1.8` يعتمد مربع حوار **الأذونات الموروثة** في مصمّم الحقل (الذي يُفتح من الإعدادات المتقدمة لحقل حسّاس، بجوار مفتاح التبديل «حسّاس») على نقطة النهاية هذه. وأصبح كل صفّ لطرف حائز يسرده مربع الحوار قابلًا للنقر، ليفتح لوحة ثانية للقراءة فقط تُظهر منحة ذلك الطرف بعد حلّها لهذا الحقل تحديدًا – وذلك بإعادة الاستعلام عبر `GET .../by-user/{userId}` أو `GET .../by-group/{groupId}`. وعندما يستحقّ الطرف أكثر من نطاق، تُفضَّل مطابقة نطاق الكيان على نطاق برمز بديل على مستوى الوحدة (بعد استبعاد أي منحة انتهت صلاحيتها أولًا). وتبقى اللوحة للقراءة فقط، شأنها شأن مربع الحوار نفسه؛

فتغيير المنحة لا يزال يتطلّب تعديلها عند نطاقها الفعلي ذي الرمز البديل على مستوى الكيان أو الوحدة. وترد التفاصيل الكاملة الخاصة بصفحة الإدارة – الربط المباشر بتبويب الحقول الحسّاسة الخاص بالطرف الحائز، وحالتا الفراغ والخطأ – في الأذونات الحساسة [الإدارة]، ولا تُكرَّر هنا.

`POST /v1/sensitive-field-grants` ينشئ `AccessRight` واحدًا لكل مستخدم ولكل مجموعة في `dto.users` / `dto.groups`، وجميعها للنطاق `dto.scope` نفسه.

- يُتحقَّق أولًا من أن `dto.scope` غير فارغ (وإلا أُرجع `400`)؛ ويُقبَل المقطع ذو الرمز البديل – فلا يُفرَض سوى شرط «غير فارغ».
- و `dto.expiry` اختياري: فغيابه أو كونه `[]` يعني عدم انتهاء الصلاحية. وخلاف ذلك، يجب أن يكون `[start, end]` بالضبط، وأن يكون كل عنصر إما `null` (حدّ مفتوح) أو سلسلة يقبلها `Date.parse`؛ ويجب ألّا يأتي `start` بعد `end`؛ كما يجب أن يكون `end`، إن وُجد، في **المستقبل** – فلا يمكن إنشاء منحة منتهية الصلاحية أصلًا. وتؤدي كل مخالفة إلى إرجاع `400`.
- والطرف الذي يملك أصلًا منحة نشطة ضمن نافذة الصلاحية للنطاق **يُتخطّى** ولا تُكرَّر منحته، ويُبلَّغ عنه في `skippedDuplicates`.
- ويُنشَر حدث `AccessRightChangedEvent` واحد للاستدعاء كله، ولا يُنشَر إلا إذا أُنشئت منحة واحدة على الأقل فعلًا.

`PATCH /v1/sensitive-field-grants/{id}` يعدّل **نافذة الصلاحية وحدها**. أما النطاق والطرف المستحِقّ فلا يمكن تغييرهما – إذ قد تؤدي إعادة توجيه المنحة إلى طرف آخر إلى إنشاء منحة نشطة ثانية للنطاق نفسه دون تنبيه. ويُرجع `404` إذا لم يوجد المُدخل أو لم يكن منحة `SENSITIVE_FIELD`.

`dto.expiry`	الأثر
غائب (المفتاح غير مُرسَل)	دون تغيير.
`[]`	**مسح** النافذة – يُحذَف المفتاح من `value`.
`[start, end]`	**استبدال** النافذة كما هي.

ويجب *حذف* المفتاح، لا ضبطه على `undefined`: فموصل MongoDB غير مُهيّأ باستخدام `ignoreUndefined`، وسيحوّل `undefined` إلى قيمة `null` مخزَّنة. وتتعامل جميع مسارات القراءة مع المفتاح الغائب و `null` الصريح بالطريقة نفسها، فهما متكافئان عند القراءة – غير أن المخطط لا يقبل `null` على المستوى الأعلى، ومن ثمّ يجب على العميل إرسال `[]` لمسح النافذة.

`DELETE /v1/sensitive-field-grants/{id}` يحذف المُدخل حذفًا نهائيًا دائمًا (ويُرجع `404` إن لم يوجد المُدخل أو لم يكن منحة `SENSITIVE_FIELD`)، وينشر `AccessRightChangedEvent`.

`PATCH /v1/sensitive-field-grants/{id}/archive` يؤرشف المُدخل أرشفةً ناعمة (`state → Archived`) عبر مسار `ArchiveDataEntryCommand` نفسه الذي يستخدمه تفاعل الأرشفة التلقائية عند انتهاء الصلاحية. وتُستبعَد المنح المؤرشفة من جميع مسارات القراءة تمامًا كما لو حُذفت نهائيًا – فيُفقَد الوصول فورًا – لكن المُدخل يبقى لأغراض التدقيق، ويصبح الزوج (النطاق، الطرف) متاحًا مجددًا لإنشاء منحة جديدة. ويظلّ مسارا الإزالة متاحَين؛ ويحدّد العميل أيّهما تستدعيه شاشة بعينها (انظر الأذونات الحساسة [الإدارة]).

استبعاد الخيارات: `excludeForUserId` / `excludeForGroupId`

يقرأ `SensitiveFieldSelectorComponentFetcher`، وهو الجالِب الذي تعتمد عليه نقطة نهاية بيانات المرجع الخاصة بمُحدِّد النطاق، المعامل `excludeForUserId` أو `excludeForGroupId` من حاوية `parameters` في جسم الطلب. وعند وجوده، يحلّ

النطاقات التي يملكها الطرف المستهدف حاليًا ويستبعدها من الكتالوج المُرجَع، فلا يعرض مربع حوار «إضافة منحة» لمستخدم أو مجموعة محدّدة حقولًا يملكها ذلك الطرف أصلًا. وتكون الأولوية لـ `excludeForUserId` إن وُجد المعاملان معًا.

والاستبعاد **مقيَّد بمستوى system-viewer**: إذ يمكن الوصول إلى الجالِب عبر نقطة النهاية العامة `POST /v2/entities/{entity}/fields/{field}/data`، التي لا تتطلّب سوى `canViewEntity` – مما يتيح استخدامها لشريحة أوسع من مستخدمي نقاط نهاية المنح المخصّصة. ولذلك لا يُنفَّذ طلب الكشف عن النطاقات التي يملكها *هدف بعينه* إلا إذا كان **المستدعي** نفسه مستوفيًا لمستوى `isSystemViewer()` على الأقل؛ أما لغيره، فيُتخطّى الاستبعاد دون تنبيه ويُرجَع الكتالوج الكامل دون ترشيح. ولا يُرجع خطأً في أي حال، ولا يسرّب منح الهدف إلى مستدعٍ غير مخوَّل.

كيفية سلوك الإخفاء

للمستخدم الذي **لا يملك** المنحة – أو يملكها دون جلسة كشف نشطة – تُستبدل القيمة الحسّاسة بالقناع `****` على نحو متّسق في المنتج كله:

الموضع	السلوك تجاه مستخدم لا يستطيع قراءة الحقل حاليًا
عروض القوائم	يعرض العمود القناع.
عرض المدخل (التفاصيل)	يعرض الحقل القناع.
التعديل	يظلّ تعديل الحقل «على العمياء» ممكنًا؛ إذ يُدخِل المستخدم قيمة جديدة في حقل مُقنَّع دون أن يرى القيمة الحالية إطلاقًا. أما اعتماد القيمة التي يرسلها أو تجاهلها فيتوقف على الحماية في جانب الكتابة، الموصوفة في مسار الكتابة.
رسائل التأكيد بعد الحفظ	تبقى أي قيمة مُرتجَعة مُخفاة؛ إذ يُطبَّق القناع على نسخة من القيمة التي يتلقّاها العميل، فيحتفظ المُدخل المرجعي وأحداثه بالقيمة الحقيقية.
التاريخ	لا تُعرض التعديلات على الحقل الحسّاس إلا إذا كان المستخدم يملك إذن الوصول إلى ذلك الحقل؛ وإلا فتُخفى القيمتان قبل التعديل وبعده.
سجل التدقيق	يُخفي سجل التدقيق القيم الحسّاسة بصورة مستقلة (تظهر على هيئة `***`).
التقارير والتصدير	تُقرأ قيم التقارير (المتاحة على الويب فقط) والتصدير (بصيغة XLSX أو PDF) عبر مسار الإخفاء نفسه؛ فيحمل التقرير أو التصدير الخاص بالمستخدم غير المصرّح له القيمة المُخفاة بدلًا من الحقيقية.

يُطبَّق الإخفاء عند قراءة البيانات، ومن ثمّ يرافق القيمة تلقائيًا في كل موضع من هذه المواضع.

وعلى مستوى التنفيذ:

- `DataEntryService.sensitiveMaskApplies(dataEntity, user)` يحدّد ما إذا كان الإخفاء سيُطبَّق أصلًا في عملية قراءة معيّنة؛ فلا بد من وجود كيان يعرّف حقلًا حسّاسًا واحدًا على الأقل، وألّا تُجرى القراءة عبر استدعاء خدمة داخلي (نظامي).
- `DataEntryService.applyCallerSensitiveMask(entries, dataEntity, user)` هو نقطة الدخول في جانب العرض، ويُستدعى من `handleResolvedValuesAfterAggregation` ما لم تُمرَّر `disableHideSensitiveFields`؛ وهو يحلّ الاسم المعياري للوحدة التي ينتمي إليها الكيان، ويقرأ نطاقات الكشف النشطة للمستدعي، ثم يستدعي `maskSensitiveFields` – وإذا كُشفت أي قيمة فعلًا، ينشر حدث `SensitiveDataUnmaskedEvent` واحدًا مُجمَّعًا للطلب.
- `DataEntryService.maskSensitiveFields(entries, dataEntity, moduleNn, permissions, unmaskScopes?)` يستبدل قيمة كل حقل غير قابل للقراءة بـ `SENSITIVE_DISPLAY_MASK` في موضعها، ويتوغّل إلى الحقول الفرعية داخل الكتل القابلة للتكرار. ولا يُكشف الحقل إلا حين يكون **كلٌّ من** الشرطين متحقّقًا: تكون نتيجة `canAccessSensitiveField(permissions, scope)` صحيحة **و** يطابق الحقلَ نطاقُ كشف نشط. وقد حُصر نوع

المعامل `permissions` فيه في `Pick<IAllUserPermissions, 'sensitiveFields'>`، بحيث يستطيع المستدعي الذي يعتمد الفشل المُغلَق تمرير `{sensitiveFields: []}` بما يطابق النوع فعلًا، بدلًا من استخدام cast.

- **الفشل المُغلَق عند تعذّر جلب الوحدة.** يحلّ `applyCallerSensitiveMask` الوحدة عبر `dataModuleRepository.findById`، ضمن مستدعٍ تُرجع كتلة `catch` فيه مخرجات التجميع الخام عند وقوع *أي* استثناء. ولذلك يُلتقَط فشل الجلب محليًا، ويُقنَّع كل حقل حسّاس باستخدام قائمة منح فارغة، بدلًا من السماح بانتشار الاستثناء وإرجاع بيانات غير مُقنَّعة دون تنبيه.
- **الفشل المُغلَق عند تعذّر جلب جلسة الكشف.** تُحاط قراءة نطاقات المستدعي النشطة بمعالجة أخطاء بالطريقة نفسها؛ فأي خطأ (انقطاع في المخزن المؤقت مثلًا) يُسجَّل ويُعامَل على أنه «لا جلسة نشطة»، فيبقى كل شيء مُقنَّعًا.

لا يمكن لمستخدمي الإدارة تجاوز الإخفاء

لا يمكن تجاوز إخفاء الحقل الحسّاس بمجرد امتلاك صلاحية مسؤول النظام أو صلاحية التعديل. فحتى مسؤول النظام ذو الصلاحيات الكاملة يرى القناع ما لم يملك منحة للحقل الحسّاس المعني *و* جلسة كشف نشطة. والاستثناء الوحيد هو الاستدعاءات الداخلية التي تجريها الخدمات الخلفية للمنصة (أو عند تعطيل الإخفاء صراحةً عبر `disableHideSensitiveFields`)؛ وتستخدمها التكاملات الخلفية الموثوقة، لا المستخدمون التفاعليون.

ويُبلَّغ العميل **بأيّ** الحقول مُقنَّعة على نحو منفصل عبر `sensitiveMaskedFields` الخاص بتعريف الكيان، الذي يُحسَب لكل طلب بواسطة `computeSensitiveMaskedFields` / `resolveSensitiveMaskedFields`، بدلًا من استنتاج ذلك بفحص القيمة المُرجَعة. ولأن القيمة الحارسة للقناع مُصمَّمة بحيث لا يمكن تمييزها عن قيمة أدخلها المستخدم بنفسه على نحو مشروع، فإن هذه القائمة هي الإشارة الموثوقة الوحيدة. وهي خاصة بكل طلب، ويجب ألّا تُخزَّن مؤقتًا (cached) مع تعريف الكيان الأساسي مطلقًا؛ ويؤدّي غياب المستخدم إلى إخفاء **جميع** الحقول الحسّاسة.

Since 1.8، أصبحت هذه الإشارة ممثَّلة فعليًا بقائمتين منفصلتين، بحيث يستطيع العميل التمييز بين «لا منحة إطلاقًا» و«منحة موجودة، لكن الحقل غير مكشوف حاليًا»: تشمل `sensitiveMaskedFields` الحقول التي لا يملك المستخدم أي منحة عليها إطلاقًا؛ وتشمل `sensitiveMaskedFieldsWithUserAccess` الحقول التي *يملك* المستخدم منحة عليها، لكن لا تشملها حاليًا أي جلسة نشطة لكشف القناع. أما الحقل الذي تكشفه جلسة نشطة فلا يظهر في أيٍّ من القائمتين. وهذا ما يتيح للواجهة عرض قفل بسيط لحقل لن يتمكّن المستخدم من رؤيته أبدًا، مقابل أداة كشف («عين») لحقل يملك منحة عليه، لكنه لم يُجرِ بعد التحقق الإضافي من هويته.

`Since 1.10` أصبحت القائمتان تشملان أيضًا الحقول **المشفَّرة فقط** (`encrypt: true`، `sensitive: false`)، التي لم تكن تظهر سابقًا في أيٍّ منهما؛ فلم تكن لدى العميل أي إشارة بشأنها خاصة بكل مُطّلِع، وكان يعرض أداة الكشف دائمًا، سواء أكان المُطّلِع يملك منحة فعلًا أم لا. والحقل المشفَّر لا يخضع لجلسة الكشف إطلاقًا؛ فهو إمّا في `sensitiveMaskedFields` (بلا منحة ← يعرض العميل **قفلًا**) أو، عند وجود منحة، في `sensitiveMaskedFieldsWithUserAccess` (← **العين** التي تُطلِق فك التشفير)؛ ولا يغيب أبدًا عن القائمتين معًا كما يحدث لحقل حسّاس كشفته جلسة. والنتيجة التي يراها المستخدم هي ظهور أيقونة القفل أيضًا على حقل مشفَّر لا يستطيع فك تشفيره؛ إذ **تُكبَح الأداة بدلًا من عرضها ثم رفضها بالرمز 403**. ويعكس تطبيق الهاتف المحمول التمييز الثلاثي نفسه بثلاث أيقونات: أيقونة تشفير للحقل المشفَّر، وعين مشطوبة للحقل الحسّاس القابل للكشف، وقفل للحقل الحسّاس بلا منحة؛ كما أصبح القناع يُطبَّق أيضًا على المواضع النصية في البطاقات ولوحة المهام والتقويم. أما عنصر التحرير المضمَّن للحقل المشفَّر فيُفرَّغ **دون شرط، بمنحة أو بدونها**، لأن القيمة المشفَّرة لا يمكن قراءتها من جانب العميل في الحالتين؛ ولولا ذلك لملأ المحرّرُ الحقلَ مسبقًا بالقيمة الحارسة للقناع كأنها محتوى حقيقي.

رندرة القيمة المُقنَّعة على العميل `Since 1.8`

تُرافق `sensitiveMaskedFields` استجابةَ **تفاصيل** المُدخل. أما استجابة **قائمة** المُدخلات العادية فلا تتضمّن قائمة مماثلة خاصة بكل مستخدم؛ فالإشارة الوحيدة المتاحة للعميل فيها هي القيمة الحارسة للقناع التي حلّت محل قيمة الحقل نفسه. ومن ثمّ، يتعيّن على العميل الذي يعرض قائمة أن يستنتج الإخفاء من القيمة ذاتها، مع اشتراط وجود علامة `options.sensitive` على الحقل نفسه، حتى لا يخلط أبدًا بين حقل غير حسّاس يحمل مصادفةً سلسلةً تطابق القيمة الحارسة على نحو مشروع وحقل مقيَّد.

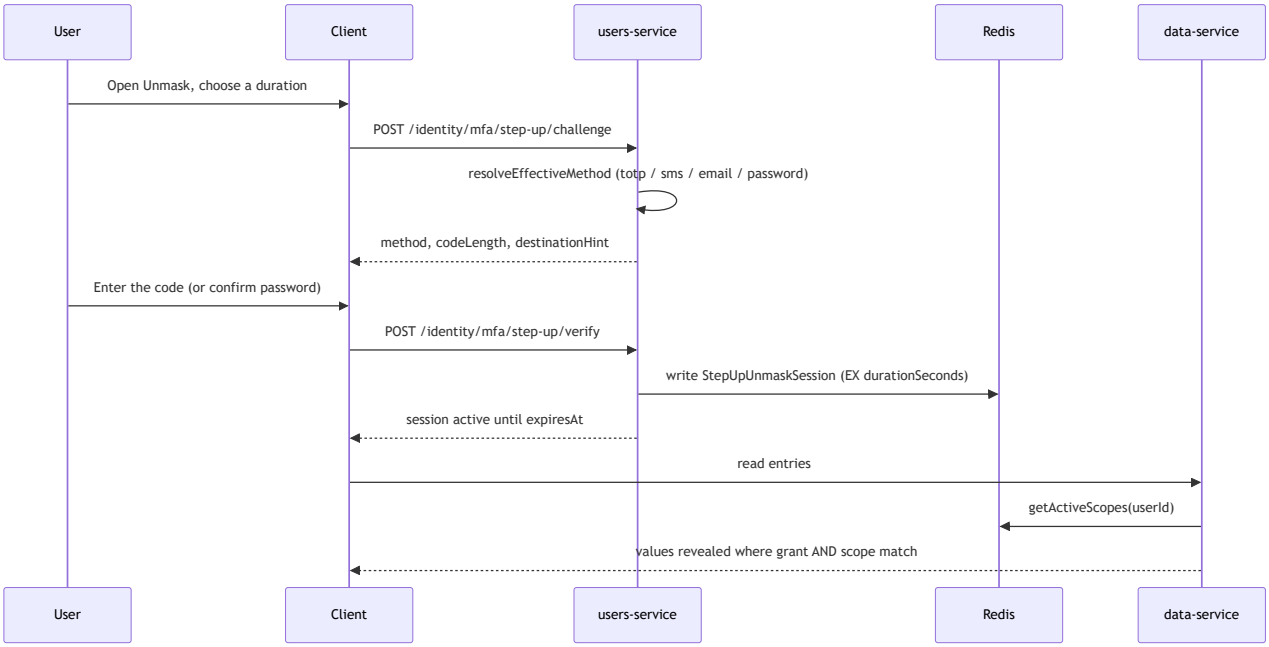
ويترتّب على ذلك نتيجتان، يلتزم بهما الآن عميل الهاتف المحمول وعميل الويب على السواء:

- **الحقل المقيَّد قراءةً يعرض قناعه لا عنصرًا نائبًا فارغًا.** كانت القيمة المُقنَّعة التي يتعذّر تحليلها وفق نوع الحقل نفسه (رقم، تاريخ) تُعامَل على أنها «بلا قيمة» في واجهات القائمة والجدول، فيبدو الحقل المحمي مطابقًا لحقل لم يُملأ. وأصبح فرعا الرندرة – الفارغ وغير الفارغ – يستخدمان مكوّن القيمة المُقنَّعة نفسه، فلا يكشف رمز القيمة الفارغة أبدًا عن أيّ الحالتين تنطبق.
- **الحقل المُقنَّع قابل للتعديل المضمَّن من شاشة التفاصيل ومن القائمة معًا.** وهذا يتوافق مع عقد الخادم في مسار الكتابة، الذي لم يشترط قط قدرة المستدعي على *قراءة* الحقل الذي يكتب فيه؛ فالكتابة العمياء آمنة لأن الخادم يُلغي ما يرسله المستدعي غير المصرّح له، لا لأن العميل أخفى عنصر التحكّم. كما تقيّم بوّابة التعديل المضمَّن شرطَ سير العمل بالطريقة الثلاثية نفسها التي يتّبعها نموذج التعديل الكامل؛ فلم يعد التعديل المضمَّن على مُدخل يُرفَض لمستدعٍ يملك **حقوق تعديل المسؤول** على المعرض لمجرد أنه ليس شخصيًا مالك الخطوة النشطة لذلك المُدخل.

ولا يُعطَّل الإخفاء إلا داخل عنصر التعديل المضمَّن نفسه وعند الإنشاء، إذ تتطلّب الكتابة العمياء حقل إدخال فارغًا قابلًا للتحرير.

جلسة الكشف بالتحقق الإضافي `Since 1.8`

امتلاك المنحة **ضروري لكنه غير كافٍ** لرؤية القيمة الفعلية. فالحقل المشمول بالمنحة يبقى مُقنَّعًا افتراضيًا، ولا يُكشف إلا ما دام المستخدم الذي يطلبه يملك **جلسة كشف** نشطة محدودة زمنيًا، فتحها بإعادة المصادقة. وهذه آلية إضافية تُطبَّق فوق نموذج المنح، ولا تحلّ محلّه.



دورة حياة الجلسة

يسجّل `step-up-router.ts` أربعة مسارات تحت `/identity/mfa/step-up` (مع تطبيق تحليل جسم JSON وتحديد النطاق (domain) والمصادقة برمز الوصول عليها جميعًا): `POST /challenge` و `POST /verify` و `POST /disable` و `GET /status`، ويقع التنفيذ في `step-up-unmask.service.ts`.

- **`challengeUnmask(accountId, {durationSeconds, scope?, method?}, domain?)`** يختار عامل التحقق ويجهّزه.
  - يجب أن تكون `durationSeconds` إحدى القيم المسموح بها في `STEP_UP_UNMASK_DURATIONS_SECONDS` – 120 / 300 / 600 / 1800 / 3600 / 10800 / 14400 / 21600 / 28800 (2 / 5 / 10 / 30 / 60 دقيقة، أو 3 / 4 / 6 / 8 ساعات) – وإلا يُطلَق `StepUpError('INVALID_DURATION')`. وقد حلّت هذه القائمة محلّ قائمة سماح سابقة تضمّ المدد 15/30/45/60/120 بالدقائق؛ ويستمدّ مُنتقيا المدّة على الويب والجوّال قيمهما من الثابت نفسه، فلا يمكن أن يختلفا عنه.
  - ويُرفض الطلب مباشرةً بالخطأ `USER_NOT_FOUND` أو `EXTERNAL_USER_FORBIDDEN`؛ إذ لا يمكن لمستخدمي SSO أو المستخدمين الخارجيين إجراء التحقق الإضافي إطلاقًا.
  - ويختار `resolveEffectiveMethod(available, preferred, requested?)` الطريقة المطلوبة إن كانت صالحة، وإلا فالطريقة المفضّلة للمستخدم، وإلا فأول طريقة متاحة ضمن `['totp', 'sms', 'email']`. أما `passkey` فـ**غير** صالح صراحةً لهذا النمط القائم على إدخال رمز، ويُتخطّى حتى لو طُلب.
  - **البديل بكلمة المرور.** إن لم تتوافر أي طريقة MFA صالحة (`MFA_NO_METHOD`)، وكان النطاق (domain) يحمل `allowPasswordStepUp === true` (تفعيل اختياري، والقيمة الافتراضية `false`)، وكان للمستخدم كلمة مرور محلية، تُختار الطريقة `'password'` بدلًا من إطلاق خطأ. ويكون مستخدمو SSO والمستخدمون الخارجيون قد رُفضوا قبل ذلك، فلا ينطبق هذا إلا على الحسابات المحلية. و`allowPasswordStepUp` حقل في كيان Domain الأساسي لدى data-service، يُزامَن إلى إسقاط `Domain` في users-service عبر معالِجَي إنشاء مُدخل البيانات وتحديثه.
  - وتُقيَّد معدّلات الطرق القابلة للإرسال `sendable` (`sms` و`email`) عبر `otpRateLimiter` *قبل* إرسال رمز OTP (`MFA_RATE_LIMITED`، `429`)، على نحو منفصل عن حدّ معدّل التحقق؛ وإلا لأمكن لمستدعٍ مُصادَق عليه تكرار `/challenge` وإغراق بريد المستخدم بالرسائل.
  - ويُخزَّن سياق معلّق (الطريقة والمدة والنطاق) تحت المفتاح `step-up-pending:<accountId>`، بمدة حياة (TTL) تساوي مدة صلاحية OTP.
  - ويُرجِع `{method, sendable, codeLength, resendAfterSeconds, destinationHint?, availableMethods}`. وتساوي `codeLength` القيمة `0` لكلمة المرور و`6` لـ TOTP؛ أما `destinationHint` فهو البريد/الهاتف المُقنَّع، ولا يوجد إلا حين تكون الطريقة `sendable`.
- **`verifyUnmask(accountId, {code?, password?, method?})`** يتحقّق من التحدّي المعلّق ضمن حدّ معدّل خاص به. ويُجري بديل كلمة المرور التحقق عبر `PasswordHelper.isSame`، ويُرجع `INVALID_PASSWORD` عند عدم التطابق؛ أما التحقق بطريقة MFA فيُفوَّض إلى المُتحقِّق المسجَّل لها. و**تُقرأ المدة والنطاق من السياق المعلّق لا من طلب التحقق**، فلا يستطيع العميل توسيع النافذة أو النطاق عند التحقق. وعند النجاح، يكتب الجلسة ويحذف السياق المعلّق ويعيد ضبط حدّ معدّل التحقق وينشر `SensitiveDataUnmaskEnabledEvent`.
- **`disableUnmask(accountId)`** ينفّذ `GETDEL` ذريًّا على مفتاح الجلسة، فلا يمكن لعمليتَي إلغاء متزامنتين أن ترصدا معًا جلسةً نشطة وتنشرا الحدث مرتين. ولا يُنشَر `SensitiveDataUnmaskDisabledEvent` إلا عند إزالة جلسة فعلًا؛ أما الإلغاء المكرّر أو إلغاء جلسة منتهية فهو عملية صامتة بلا أثر.

- `getUnmaskStatus(accountId)` يُرجع `{active, scopes?, expiresAt?}` لاستخدامه في مُبدِّل الشريط العلوي وعدّاده التنازلي.

تخزين الجلسة (عقد بين الخدمات)

الجلسة كائن JSON من النوع `StepUpUnmaskSession` (`@osos/shared/step-up-unmask.ts`) – `{scopes: string[], enabledAt: ISOstring, expiresAt: ISOstring}` – يُكتب تحت المفتاح `stepUpUnmaskRedisKey(envPrefix, userId)` . والمفتاح **نظيف** عمدًا (بلا مقطع مجال أسماء `oidc:`)، لأن users-service يكتبه عبر العميل الخام، لا عبر المُهايئ الذي يضيف مجال الأسماء. وتحتوي `scopes` إما على النطاق `scope` الذي قدّمه المستدعي في التحدّي، أو على `STEP_UP_UNMASK_ALL_SCOPE` (`'*/*/*'`)، ويطابق كل حقل عبر مُطابِق الرمز البديل المعتاد) حين لا يُقدَّم نطاق.

ويكتب users-service على **اتصال مُضاعَف مخصّص** مثبَّت على `stepUpSessionDb()` (`CACHE_REDIS_DB ?? 1`)؛ إذ قد يتصل عميله الافتراضي بقاعدة منطقية أخرى (مخزن جلسات/رموز OIDC)، وكتابة الجلسة هناك ستضعها في موضع لا يقرأ منه data-service أبدًا. ويُكتب المفتاح مع `EX ttlSeconds = durationSeconds` ، فتنتهي صلاحيته تلقائيًا، ولا توجد مهمة تنظيف.

ويتولّى `UnmaskSessionService` في `data-service` (`SINGLETON`) القراءة؛ إذ يقرأ `getActiveScopes(userId)` المفتاحَ نفسه حرفًا بحرف (دون إعادة إضافة بادئة)، ويحلّله بصيغة JSON، ويُرجع `session?.scopes` أو `undefined` إن كان غائبًا أو منتهي الصلاحية.

جلسات الكشف المحدودة بالجلسة `Since 1.8`

أصبحت جلسة الكشف مرتبطة **لجلسة الدخول (sign-in session)** التي فتحتها، لا بالحساب ككلّ. فعند تسجيل الخروج ثم الدخول مجدّدًا، أو فتح جلسة متزامنة ثانية للمستخدم نفسه (متصفّح مختلف، جهاز ثانٍ)، تبدأ الجلسة الجديدة مع تفعيل الإخفاء مجدّدًا، ولا ترث نافذة كشف فُتحت في مكان آخر.

وهذا يسدّ ثغرة حقيقية: فقد كانت الجلسة سابقًا مفهرَسة بمعرّف الحساب وحده، مما كان يتيح *لأي* جلسة لذلك الحساب – بما فيها جلسة فتحها لاحقًا أي شخص استطاع المصادقة بصفة ذلك المستخدم، أو جلسة في ملف تعريف متصفّح نسي المستخدم الأصلي تسجيل الخروج منه – أن ترث أي نافذة كشف لا تزال مفتوحة، دون أن تفعّل تلك الجلسة الجديدة الكشفَ بنفسها إطلاقًا.

وعلى مستوى التنفيذ، أصبح كلٌّ من مفتاح الجلسة ومفتاح التحدّي المعلَّق مُحدَّدًا بالزوج `(userId, sessionId)` ، ويمرّر كل قارئ يستعلم عن النطاقات النشطة معرّف جلسة المستدعي صراحةً، مع **أي تراجع إلى المفتاح على مستوى الحساب عند عدم العثور على تطابق** مستبعَد عمدًا: فمعرّف الجلسة الغائب أو الخاطئ يعني «لا جلسة نشطة»، ولا شيء غير ذلك. والاستثناء الوحيد هو المستدعي بلا جلسة (رمز خدمة-إلى-خدمة لا توجد له جلسة لتحديدها)، إذ يظلّ يستخدم المفتاح على مستوى الحساب بدلًا من أن يُرفَض كليًا. أما تحديد معدّل إرسال OTP والتحقّق منه فيبقيان مفهرَسَين بالحساب وحده عمدًا: فالهاتف والبريد ومحاولات القوة الغاشمة التي تشملها الحماية تخصّ الحساب، ولا يجوز أن يؤدّي فتح جلسات إضافية إلى مضاعفة الحدود المسموح بها.

مفاتيح التحقق الإضافي مقسّمة بحسب القاطن `Since 1.10`

تكتب خدمة الهوية منحةَ الكشف وإثباتَ التحقق الإضافي أحاديَّ الاستخدام، وتقرؤهما data-service عبر ريديس. ولم يكن أيٌّ منهما يتضمّن القاطن، فكانت المنحة أو الإثبات المُكتسَب ضمن قاطن واحد قابلًا للقراءة ضمن قاطن آخر في النشر متعدّد القاطنين، ولم يكن بوسع أي عملية تنظيف عند إنهاء خدمة قاطن العثور عليهما. أما الآن، فيُشكَّل كلا المفتاحين وفق وضع القاطنين:

النمط	منحة الكشف	الإثبات
وحيد القاطن	`<ENV prefix>step-up:unmask:<uid>[:sid:<sid>]` (دون تغيير)	`<ENV prefix>step-up:proof:<id>` (دون تغيير)
متعدّد القاطنين	`<ENV prefix>app:<appId>:step-up:unmask:…`	`<ENV prefix>app:<appId>:step-up:proof:<id>`

لا تظهر المنحة أو الإثبات إلا ضمن القاطن الذي أصدرهما؛ وإذا لم يوجد قاطن ضمن النطاق في نشر متعدّد القاطنين، يرفض الطرفان الطلب قبل الوصول إلى ريديس؛ ويُزال كلاهما عند إنهاء خدمة القاطن. وتظلّ قراءة الإثبات ذرّية ومقصورة على استخدام واحد. ولا يطرأ أي تغيير على التثبيتات وحيدة القاطن، فلا تضيع جلسات الكشف عند الترقية، ولا يتغيّر شيء في منشئ الكيانات أو معالج الكشف.

الأحداث المُدقَّقة

ثلاثة أحداث، لكل منها معالِج مقابل في `audit-service` يحفظ سجلّ تدقيق:

الحدث	يَنشره	التفاصيل المحفوظة
`SensitiveDataUnmaskEnabledEvent(userId)`	users-service، عند نجاح `verify`	– (إجراء التفعيل فقط)
`SensitiveDataUnmaskDisabledEvent(userId)`	users-service، عند إلغاء فعلي (لا عملية صامتة)	– (إجراء الإلغاء فقط)
`SensitiveDataUnmaskedEvent(userId, …)`	data-service، حين يكشف `maskSensitiveFields` حقلًا واحدًا على الأقل فعلًا	`{module, entity, fields, entryIds}`

ويُنشَر `SensitiveDataUnmaskedEvent` **مرة واحدة لكل طلب**، ويجمع كل الحقول التي كُشفت في جميع المُدخلات ضمن تلك الاستجابة (باتّحاد `entryIds` عبر الحقول)، ولا يُنشَر مرةً لكل حقل. ويجري النشر بأسلوب «أرسِل وانسَ» مع احتواء الأخطاء: فإذا أخفق، يُسجَّل تحذير دون عرقلة القراءة أو إفشالها أبدًا.

الاستعلام والترشيح على الحقول الحساسة

لمنع المستخدم من استنتاج قيمة مُخفاة بالترشيح عليها، **يُرفَض** أي استعلام يقدّمه ويشير إلى حقل حسّاس لا يستطيع قراءته (قرار تصميمي مقصود: ارفُض، لا تُسقِط بصمت). فلا يستطيع المستخدم بناء شرط مثل «الراتب أكبر من س» على حقل لا يُسمح له بقراءته. وبذلك تُغلَق القناة الجانبية التي قد تتيح تسريب قيمة من خلال نتائج البحث، حتى مع بقائها مُخفاة على الشاشة.

ويؤدّي الرفض إلى إطلاق `SensitiveFieldQueryNotAllowedException` (رمز الخطأ `SENSITIVE_FIELD_QUERY_NOT_ALLOWED`، مع التفصيل `{field}`). ويتولّى الإنفاذ مساران يقابلان صيغتَي الترشيح:

- `assertUserCanQueryCriteria` – يعيد استخدام `QueryEngineService.parseCriteria` ضمن سياق حسّاس، بحيث تُجتاز الفروع المتداخلة `ElemMatch` / `Or` / `And` بالطريقة نفسها التي تُنفَّذ بها.
- `assertUserCanQueryWhere` – يجتاز شجرة `where` من نوع loopback، ويزيل البادئة الأولية `value.` ومقاطع فهارس المصفوفة الرقمية، ويطابق الحقل إما مطابقة تامة أو بوصفه أبًا لمسار ذي لاحقة.

ولا ينفّذ أيٌّ منهما أي إجراء (no-op) عند غياب المستخدم، أو غياب المعايير/الـ where، أو عندما لا يحتوي الكيان على حقول حسّاسة. أما معايير المعرض/الترشيح السريع (quick-filter) التي يُعرِّفها المسؤول، فتُعامَل بوصفها سياق نظام موثوقًا **ولا** تخضع للإنفاذ هنا؛ إذ تُفحَص معايير المستخدم الارتجالية (ad-hoc) فقط. وهذا الفحص **قائم على المنحة وحدها**: فهو لا يشترط أيضًا وجود جلسة كشف، ومن ثمّ يستطيع حامل المنحة الترشيح على حقل لا تزال قيمته مُقنَّعة على الشاشة.

`Since 1.10` **لم تعد تسمية الحقل المحمي في استعلام مساويةً لحق قراءته.** ففي منطقة **الإدارة**، أصبح بإمكان محرِّر النظام أن *يسمّي* حقلًا حسّاسًا في استعلام أو مرشِّح أو شرط تفاعل **دون أن يملك منحة قراءة عليه**. وكان الحقل المُقنَّع قبل ذلك يُستبعَد من منتقي الحقول بلا شرط وفي كل المواضع: فلم يكن المدير الذي لا يملك منحة خاصة به يرى أيًّا من حقول الكيان الحسّاسة عند تصميم استعلام، حتى في شاشات الإدارة التي يصمّم فيها الاستعلام لغيره. ومع ذلك، فإن ثلاثة أمور **لم** تتغيّر:

- **لا يزال المستخدم النهائي العادي بحاجة إلى المنحة** كي يشير إلى الحقل؛ إذ يحتفظ باني استعلامات المستخدم النهائي بالقاعدة القديمة.
- **لا يتغيّر شيء ممّا يراه أي أحد من القيم.** فمسار الإخفاء، وقائمتا الحقول المُقنَّعة، وجلسة الكشف بالتحقق الإضافي، وكل سياسة على مستوى العنصر، جميعها بقيت كما هي؛ إذ يقتصر هذا التغيير على توسيع نطاق من يجوز لهم *الإشارة* إلى الحقل في المعايير.
- **يبقى الحقل المشفَّر مرفوضًا في الحالتين**، في كلتا المنطقتين، وقبل إجراء هذا الفحص أصلًا؛ انظر ما الذي يرفضه الحقل المشفَّر.

ويتحدّد «سياق الإدارة» بحسب **المنطقة التي جاء منها الطلب، لا دور المُستدعي**؛ فمحرِّر النظام الذي يتصفّح جانب المستخدم النهائي (عبر معرض بيانات مثلًا) يخضع للرفض نفسه الذي يخضع له المستخدم النهائي عند تنفيذ الاستعلام نفسه. وبذلك تتّفق قائمة الحقول لدى العميل وحارس الخادم دون أن يُعيد أيٌّ منهما صياغة قاعدة الآخر، ويكون الرفض هو السلوك عند الإخفاق؛ فغياب مُخوِّل مربوط، أو غياب الأذونات، أو وقوع أي خطأ، يؤدّي إلى اعتبار السياق «ليس إدارة».

منح ذات صلاحية زمنية (منتهية الصلاحية)

يمكن أن تكون لمنحة الحقل الحسّاس **نافذة صلاحية زمنية محدودة** بدلًا من أن تظلّ دائمة حتى إلغائها يدويًا. ويمكن عند منح الوصول تحديد تاريخ بداية و/أو تاريخ نهاية:

- ترك كِلا التاريخين فارغًا يجعل المنحة **بلا انتهاء** (السلوك الافتراضي).
- تحديد **تاريخ نهاية فقط** يجعل المنحة سارية من الآن وحتى ذلك التاريخ.
- تحديد **التاريخين معًا** يحدّد فترة الصلاحية الكاملة؛ وتكون المنحة قبل تاريخ البداية موجودة، لكنها لا تخوّل أي وصول.

عند بلوغ المنحة تاريخ نهايتها، تتولّى أسس تلقائيًا **بأرشفة** مُدخل المنحة عبر تفاعل مؤقّت (Timer) مُهيّأ مسبقًا لهذا الغرض، ويسري أثر الأرشفة فور حدوثها: تُمسح ذاكرة الخبء (Cache) لأذونات المستخدم، وتُعاد حساب حالة «حقل مقيّد» للحقل، ويُستبعد الإذن المؤرشف فورًا من كل استعلام لاحق عن الأذونات؛ أي إن المستخدم يفقد الوصول بمجرد انتهاء الصلاحية دون أي تدخل يدوي.

ويمكن أيضًا أرشفة المنحة أو حذفها يدويًا في أي وقت قبل موعد انتهائها؛ ولكلا الإجراءين الأثر الفوري نفسه على الوصول. وتؤدّي إعادة إضافة طرف (principal) كانت منحته مؤرشَفة أو منتهية الصلاحية إلى إنشاء مُدخل Active جديد، بدلًا من أن تمرّ العملية بلا أثر (silent no-op).

نموذج البيانات

علامة الحقل: `options.sensitive`

تُعرَّف على `FieldOptions` (`field-options.model.ts`):

الحقل	النوع	الافتراضي	المعنى
`sensitive`	`boolean`	`false`	عندما تكون القيمة true، يُقصَر حق قراءة قيمة هذا الحقل على المستخدمين/المجموعات الذين يملكون منحة مطابقة.

التحديد العابر: `sensitiveAccess`

الحقل `DataEntityField.sensitiveAccess` من النوع `SensitiveFieldAccess` (`sensitive-field-access.model.ts`):

```
class SensitiveFieldAccess {
  users: ReferenceProperty[];   // users granted read access
  groups: ReferenceProperty[];  // groups granted read access
}
```

هذا التحديد **خاص بالطلب فقط**، ولا يُخزَّن أبدًا في مستند الكيان:

- عند **الكتابة** (إنشاء/تحديث حقل أو كيان)، تُزامَن قيمته مع مجموعة `access_right` ثم تُزال من الحقل قبل الحفظ.
- عند **القراءة** (جلب تعريف الحقل/الكيان)، تُعاد حوسبة قيمته من `access_right` ويُعاد إرفاقها مباشرةً (on the fly).

تخزين المنح: مجموعة `access_right`

يُمثَّل كل زوج ممنوح (user|group، scope) بمستند `AccessRight` واحد يتضمّن:

- `SENSITIVE_FIELD` = `value.permission_type.id`
- `value.scope` = سلسلة النطاق
- `{dataEntity: USER|GROUP, dataEntry: <principalId>}` = `value.entitled_entity`
- `value.expiry` = نافذة صلاحية اختيارية `[startIso, endIso]` ؛ وتشير `null` أو غياب القيمة إلى أن المنحة لا تنتهي أبدًا
- `state` = `Active` ، إضافةً إلى معلومات الإنشاء/التحديث القياسية و `store = PUBLIC` .

يثبّت مسار الكتابة ثابتَي إعداد (configuration constants) عند حفظ المنحة: يُضبَط حقل `store` في المُدخل على مرجع `PUBLIC_STORE` (المعرّف `'PUBLIC'`)، ويُنشَر حدث `AccessRightChangedEvent` المصاحب ضمن المساحة الجذرية `ROOT_SPACE_ID` (`'ROOT'`)، بحيث تظلّ المنحة وحدث إبطال ذاكرة الخبء (cache) المرتبط بها محصورَين دائمًا في المخزن العام (public store) عند جذر المساحة، لا في مساحة فرعية خاصة بقاطن.

تتجاهل عمليات قراءة المنح المُدخلات التي تكون حالتها `Archived` أو التي تقع خارج نافذة صلاحيتها؛ ومن ثمّ فإن إعادة إضافة طرف كانت منحته مؤرشفة أو منتهية الصلاحية تُنشئ مُدخلًا Active جديدًا بدلًا من أن تمرّ بلا أثر أو تنبيه. وتستند استعلامات المنح إلى فهرس مركّب على `(value.permission_type.id, value.scope)` (الترحيل `1.8.0-a25`)، وتستخدم تعبيرًا نمطيًا (regex) مُثبَّتًا بالرمز `^` لمطابقة البادئات (prefix) بأسلوب ملائم للفهرسة.

ويعيد ترحيل لاحق مُتكافئ (idempotent)، هو `1.8.0-a31` ، تأكيد البنية التراكمية لـ `AccessRight` (`permission_type` / `scope` / `expiry`) في عملية واحدة للقراءة والتعديل والإرسال. فقد كان سابقاه يحفظان تغيير الكيان مع الإرسال على ناقل الرسائل بأسلوب «أرسِل وانسَ»؛ ولذلك كانت البيئة التي شغّلتهما دون وجود data-service يستهلك الطابور تُسجّل الترحيلين بوصفهما مُطبَّقَين، بينما لا يكتسب `access_right` بنية `scope` / `expiry` مطلقًا، فيظلّ مُحدِّد النطاق فارغًا بلا أي تعافٍ ذاتي. ويحمل `a31` رقم إصدار أعلى، لذا يُشغَّل في عملية الترحيل التالية، ولا يُحدث أثرًا في قاعدة بيانات متقاربة أصلًا.

الأذونات المُحلَّلة: `sensitiveFields`

عند تحليل أذونات المستخدم (`access-rights.service.ts`)، تُسطَّح جميع نطاقات منح `SENSITIVE_FIELD` المتاحة عبر منح المستخدم نفسه ومنح مجموعاته، وتُزال منها التكرارات، وتُتاح في صورة المصفوفة النصية `IAllUserPermissions.sensitiveFields` . ويقرأ كل فحص وصول أثناء التشغيل هذه المصفوفة حصرًا؛ انظر فحوص الوصول. وامتلاك نطاق مطابق هنا **ضروري لكنه لم يعد كافيًا** لرؤية قيمة حيّة؛ إذ تمثّل جلسة الكشف بالتحقق الإضافي البوابة الثانية.

الأقنعة

توجد أربعة ثوابت في `libs/sensitive-field.ts` ، ولا تزال ثلاثة منها مستخدَمة:

الثابت	القيمة	الاستخدام	منذ
`SENSITIVE_DISPLAY_MASK`	`****__DISPLAY_SENSITIVE__****`	القيمة التي تُستبدَل فعليًا في `value` الخاص بحقل مُقنَّع عند القراءة. **ليست** السلسلة البسيطة `****` التي وصفتها نسخة سابقة من هذه الصفحة.	`1.8`
`SENSITIVE_STARS`	`****`	الشكل الذي تبدو عليه القيمة المُقنَّعة *للإنسان* – تُستبدَل فقط حين تُرندَر القيمة داخل نثر (رسالة ما-بعد-الحفظ، حوار تأكيد زرّ، عنوان إعادة توجيه)، لا في حمولة `value` الخاصة بالمدخل أبدًا.	`1.8`
`AUDIT_SENSITIVE_MASK`	`***`	قناع دائم في حمولات التدقيق/السجلّ.	`1.8`
`SENSITIVE_MASK` (قديم)	`****__MAKE_SENSITIVE__****`	**لا يُصدَر أبدًا عند القراءة.** يُحتفظ به فقط ككاشف للبيانات التي تلوّثت حين كان لا يزال مستخدمًا.	`1.8`

يتعمّد مسار القراءة كتابة حارس غير قابل للتخمين بدلًا من قيمة يمكن لمستخدم كتابتها فعليًا، كي يستطيع مسار الكتابة التمييز بين حقل مُقنَّع أعاده العميل كما هو وقيمة غيّرها العميل فعلًا؛ انظر مسار الكتابة. ولأن الحارس يتكوّن من 26 محرفًا، لا من أربع نجوم، يجب على أي مسار رماز يستبدل قيمة حقل مُقنَّع داخل نصّ موجّه للبشر، بدلًا من حمولة `value` بصيغة JSON، تحويله إلى نجوم بسيطة أولًا. وتتولّى `renderSensitiveMaskAsStars` هذا التحويل، وتستدعيها `TokenService.replaceTokenInText` مركزيًا لتغطية جميع مواضع استبدال النصوص (رسائل ما بعد الحفظ، حوارات تأكيد الأزرار، عناوين إعادة التوجيه) في مكان واحد.

ملاحظة تاريخية. في مرحلة سابقة من سلسلة إصدارات 1.8، كانت قراءة الحقل المُقنَّع تُعيد الحارس الداخلي `SENSITIVE_MASK` حرفيًا، وكانت الحماية الوحيدة عند الكتابة تتمثّل في استعادة المُتحقِّق للقيمة القديمة عند تلقّي السلسلة نفسها مجددًا؛ أي إن أمان الكتابة كان يعتمد كليًا على حُسن سلوك العميل. وقد استُبدل ذلك الحارس بـ `SENSITIVE_DISPLAY_MASK` أعلاه؛ ولا يزال المُتحقِّق يستعيد القيمة عند تلقّي أيٍّ من السلسلتين، فتظلّ الحماية قائمة للعميل الذي لا يزال يُعيد إرسال الحارس القديم.

يبقى الحارس القديم في المصدر لسببين محدّدين فقط، ولا يكون أبدًا قيمة يُنتجها النظام عند القراءة:

1. يستعيد `data-entry-validator.service.ts` القيمة المخزّنة إذا رأى `value === SENSITIVE_MASK` **أو** `value === SENSITIVE_DISPLAY_MASK`؛ وهذا هو فرع الاستعادة في المُتحقِّق (انظر مسار الكتابة).
2. يُستخدَم للكشف **للبيانات الملوَّثة**: إذ يطابق كلٌّ من الترحيل `1.8.0-a34` وحارس `containsLegacySensitiveMask` في indexing-service السلسلة القديمة للعثور على البيانات التي تسرّبت حين كان الحارس القديم مستخدمًا أو لرفضها. ولا يطابق أيٌّ منهما `SENSITIVE_DISPLAY_MASK` الحالي، الذي يكفي تمايزه وصعوبة تخمينه لجعله كاشفًا آمنًا بذاته إن احتاج إليه ترحيل مستقبلي.

علّة تلويث الفهرس وإصلاحها

تساعد معرفة هذه العلّة على فهم الرماز: كان `DataEntryWriterService` يعيد قراءة المُدخل المحفوظ للتوّ عبر مسار القراءة **المُقنَّع**، وينشر ذلك الكائن المُقنَّع بوصفه `DataEntryUpdatedEvent` / `DataEntryCreatedEvent`. ولأن indexing-service يشتقّ `DataEntryIndex.meta.mainTitle` من تلك الحمولة، كان حفظ مُدخل ذي حقل عنوان حسّاس على يد مستخدم لا يملك المنحة يُثبِّت القناع في الفهرس المشترك. ولأن الارتباط يستمدّ تسميته مباشرةً من ذلك الفهرس (الذي لا يمكن إعادة تقنيعه لكل قارئ)، كان القناع يُعاد إلى *كل* قارئ، بمن فيهم حاملو المنح. ويتكوّن الإصلاح من ثلاثة أجزاء:

- `DataEntryWriterService.resolveSavedEntryUnmasked()` – أصبحت كل إعادة قراءة بعد الحفظ (الإنشاء، والتحديث، والتعديل الجزئي، والترقية، وتحديث الكيانات الوهمية، والحلّ بعد المعاملة، والإنشاء الدفعي مع المُدخلات المرتبطة) تُنفَّذ مع `disabledStages.hideSensitiveFields`، فتحمل حمولات الأحداث وعمليات الكتابة المشتقّة القيم المرجعية دائمًا.
- `DataEntryService.maskEntryForResponse()` – يُقنِّع **نسخة عميقة** من المُدخل المحفوظ لاستجابة HTTP، ويُبقي المُدخل في حمولة الحدث كما هو؛ فالتقنيع يخصّ طريقة العرض لكل قارئ، ويجب ألّا يُطبَّق إلا عند إرسال الاستجابة إلى ذلك القارئ تحديدًا.
- ويصلح الترحيل `1.8.0-a34` الصفوف التي انحرفت بالفعل؛ إذ يُفرِغ تزامنيًا كل `DataEntryIndex.meta.mainTitle` يطابق نمط الحارس **القديم**، ثم ينشر لا تزامنيًا `DataEntityFieldMetaUpdatedEvent` لكل كيان متأثّر كي يعيد indexing-service بناء العنوان الحقيقي. وتُنفَّذ الخطوة الثانية على أساس بذل أفضل جهد، ولا يؤكّد الترحيل نفسه اكتمالها؛ وتُسجَّل معرّفات الكيانات المتأثّرة كلٌّ على حدة لإتاحة إعادة إطلاق البناء.

فحوص الوصول

يوفّر `SensitiveFieldsService` (المصدر الأساسي) دوالًا مساعدة ثابتة (static) خالصة، إضافةً إلى دالة نسخة (instance) واحدة:

`canAccessSensitiveField(permissions, fieldScope): boolean`

فحص ثنائي، **دون تجاوز للمسؤول (admin)**: يُرجع true إذا وفقط إذا طابقت منحة واحدة على الأقل في `permissions.sensitiveFields` القيمةَ `fieldScope` باستخدام `matchesSensitiveScope` (الذي يدعم الرمز البديل wildcard). ويُعامَل المستخدم الذي لا يملك مصفوفة `sensitiveFields` (أو الطلب غير المُوثَّق) على أنه لا يملك أي منح، ومن ثمّ لا يملك حق الوصول.

يفحص هذا *المنحة* وحدها؛ فقراءة مُدخل حيّ تتطلب، إضافةً إليه، التحقق من وجود جلسة كشف نشطة. أما حارس الكتابة وحارس الاستعلام فيعتمدان على `canAccessSensitiveField` وحده، وكلاهما غير مقيَّد بالجلسة. وتُستثنى من ذلك قراءة السجل التاريخي للتدقيق؛ إذ تشترط Since 1.8 وجود المنحة وجلسة نشطة معًا، تمامًا كالقراءة الحيّة؛ انظر قراءة السجل التاريخي للتدقيق.

`filterQueryableFieldsForUser(fields, permissions, moduleNn, entityNn)`

يُرشِّح قائمة مسطّحة من الحقول، فيستبعد الحقول الحسّاسة التي لا يستطيع المستخدم الوصول إليها، ويُبقي الحقول غير الحسّاسة دائمًا. والغرض منه تزويد قائمة حقول باني الاستعلام (query-builder) بالحقول التي يجوز للمستخدم الترشيح بناءً عليها، بحيث لا تعرض الواجهة سواها. (جرى التحقق من أنه دالة مساعدة ثابتة ومُصدَّرة، ولم يُعثر على أي موضع يستدعيها في بيئة الإنتاج؛ انظر فجوات معروفة.)

`getSensitiveFieldOptions(search?, excludeScopes?)` – كتالوج الخيارات

دالة نسخة (instance) تُرجع `SensitiveFieldOptionDto` واحدًا لكل حقل حسّاس في المساحة (space) بأكملها (جميع الكيانات/الوحدات)، لاستخدامه في مُحدِّد النطاق. وتُجري عملية تجميع (aggregation) في Mongo على `DataEntity` تنفّذ ما يلي:

1. تنفّذ `$lookup` على `DataModule` الأب لجلب سياق الوحدة المُعرَّب.
2. تجمع الحقول الحسّاسة **من المستوى الأعلى** (مع استبعاد حقول حاويات الكتل Block).
3. تجمع **الحقول الفرعية للكتل** (block sub-fields) المُعلَّمة بأنها حسّاسة، وتُنتج اسمًا مُنمّطًا بالصيغة المنقوطة `block.subfield`؛ وتتخطّى الحقول الفرعية اليتيمة للكتل (التي لا يشير أي حقل Block إلى كيانها الداخلي)، على غرار `getAllFields()`.
4. تُرشِّح اختياريًا بحسب `search`، بمطابقته مع **أسماء** الحقول بجميع رموز اللغات المسجَّلة. وتُهرَّب محارف المصطلح الخاصة بالتعبيرات النمطية (`\\$&`) قبل الاستيفاء (interpolation)، لمنع الأعطال الناجمة عن تعبير نمطي غير صالح وهجمات ReDoS.
5. تستبعد اختياريًا أي نطاق وارد في `excludeScopes`، وتستخدم مربعات حوار «إضافة منحة» هذا الخيار لإخفاء الحقول التي يملك المستخدم أو المجموعة المستهدفة منحها بالفعل؛ انظر استبعاد الخيارات. `Since 1.8`

يحمل كل `SensitiveFieldOptionDto` القيمَ `id` و`fieldName` / `fieldNormalizedName` و`entityName` / `entityNormalizedName` و`moduleName` / `moduleNormalizedName` (الأسماء المُعرَّبة من النوع `LocalizedValue`).

يُتاح الـ DTO عبر نقطة النهاية العامة لبيانات الحقل المرجعي، من خلال `SensitiveFieldSelectorComponentFetcher` (اسم المكوّن `SensitiveFieldSelectorComponent`)، المرتبط بالحقل `scope` في الكيان `AccessRight` (الترحيل `1.8.0-a24`). ويُرجع الجالب (fetcher) الكتالوج المسطّح كما هو، ويأخذ مصطلح `search` من جسم الطلب (request body).

قراءة السجل التاريخي للتدقيق

تستبدل `DataEntryHistoryService.maskSensitiveHistoryFields(history, dataEntity, user)` قيمتَي `old` / `new` للحقول الحسّاسة في `additionalData` الخاص بكل سجلّ تعديل (`Update`) بالقيمة `AUDIT_SENSITIVE_MASK`. **Since 1.8**، لا تكشف الدالة قيمة حقل إلا عند استيفاء **كلٌّ من** المنحة (`canAccessSensitiveField`) وجلسة كشف قناع نشطة؛ وهي قاعدة اشتراط المنحة والجلسة نفسها التي تنطبق على القراءة الحيّة. وكانت القاعدة السابقة هنا تعتمد على المنحة وحدها، أي إن أي حامل منحة كان يستطيع قراءة القيم السابقة لحقل في سجلّه دون فتح جلسة كشف قناع إطلاقًا. وكان ذلك يتيح الالتفاف على قاعدة القراءة الحيّة، إذ تبقى القيمة مُخفاة في المُدخل نفسه، لكنها تظهر على شاشة أخرى في سجلّه. ولا تستبدل الدالة `old` / `new` إلا إذا كانت الخاصية موجودة فعلًا، حتى لا تُضاف إلى سجلّ الكتابة الأولى (الذي لا يحمل سوى `new`) قيمة `old` مُقنَّعة زائفة. ويُعيد مسار سريع السجل التاريخي دون تغيير عندما لا يحتوي الكيان على حقول حسّاسة.

وبصورة مستقلة، يُخفي `audit-base-handler.ts` في `audit-service` القيم الحسّاسة **قبل حفظ** سجلّ التدقيق (باستخدام نسخته المحلية `AUDIT_SENSITIVE_MASK = '***'`)، **ويفشل مُغلَقًا** (fails closed): فإذا تعذّر تحميل البيانات الوصفية للحقل الحسّاس، تُقنَّع كل قيمة في الحمولة، بدلًا من المخاطرة بتسريب قيمة حسّاسة إلى سجلّ التدقيق غير القابل للتغيير.

مسار الكتابة (الحفظ المُنفَّذ بالأذونات) `Since 1.8`

بما أن قراءة الحقل غير المسموح بقراءته تُرجع حارس قناع، فقد تتيح إعادة كتابته دون مراعاة ذلك للمستخدم إعادة إرسال الحارس وإتلاف القيمة الحقيقية دون قصد، أو إدراج قيمة اعتباطية. وقد صُمِّمت آلية سدّ هذه الثغرة لتُفرَض **بالأذونات**، دون الاعتماد على سلوك العميل:

- `DataEntryService.preserveSensitiveFieldsOnWrite(newValue, oldValue, dataEntity, permissions)` – يتجاهل القيمة المُرسَلة لكل حقل حسّاس لا يستطيع المستدعي قراءته، ويستعيد القيمة المخزّنة. وعند **الإنشاء** (حيث لا توجد `oldValue`) يُستبعَد الحقل كليًا بدلًا من إدراج قيمته.
- أما الحقل الفرعي الحسّاس داخل **كتلة قابلة للتكرار**، فيُزال عند الإنشاء من كل صف مُرسَل؛ وعند التحديث تُستعاد القيم **حسب فهرس المصفوفة**، لأن الصفوف لا تحمل معرّفًا ثابتًا. وإذا اختلف عدد الصفوف المُرسَلة عن العدد المخزَّن – بما يعني إضافة صفوف أو حذفها، وهي عملية لا يمكن التوفيق بين قيمها بأمان لمستدعٍ لا يملك المنحة – **تُرفَض** الكتابة بـ `SensitiveFieldWriteNotAllowedException` (بتفصيل `{field}`)، بدلًا من المخاطرة بإسقاط قيمة محمية أو إلحاقها بالصف الخطأ. أما إعادة الترتيب مع بقاء الطول نفسه فلا يمكن كشفها دون معرّف صف، لكنها حالة حدّية تتعلق بسلامة البيانات، ولا تؤدي إلى كشفها.
- **ولم يكن التجاوز للمسؤول جزءًا من التصميم أبدًا.** فالوصول إلى الحقل الحسّاس قائم على المنح، وقناع القراءة لا يستثني المسؤولين أيضًا؛ ومن ثمّ، كان استثناء الكتابة سيعيد فتح الثغرة التي وُضع الحارس لإغلاقها. وكان يُفترَض أن تُنفَّذ كتابات النظام غير المشروطة عبر الاستدعاءات الداخلية بين الخدمات بدلًا من ذلك.

القيود (ما الذي يمكن وضع علامة حسّاس عليه)

يفرض `FieldService.validateAndNormalizeSensitive` ، عندما تكون `options.sensitive === true` ، ما يلي:

- **الرفض** للحقول المرجعية (`DataTypeEnum.Entity`) وحقول حاويات الكتل (`DataTypeEnum.Block`) ← يُطلِق `SensitiveFieldNotAllowedException` (رمز الخطأ `SENSITIVE_FIELD_NOT_ALLOWED`).
- **الإجبار على الإيقاف**: `options.searchable = false` و `options.showAsTitle = false` (يجب ألّا تُفهرَس القيمة الحسّاسة للبحث أو تُستخدَم عنوانًا للمُدخل).

ومع ذلك، يظل بالإمكان وضع علامة حسّاس على **الحقول الفرعية** للكتل، كلٌّ على حدة؛ فالمنع يقتصر على حقل الكتلة *الحاوية* نفسه.

لا يمكن وضع علامة حسّاس على حقل العنوان الرئيسي للكيان `Since 1.8`

`Since 1.8` أصبح المنع مطبَّقًا في كلا الاتجاهين:

- وضعُ علامة حسّاس على حقل **يُجبِر** `options.showAsTitle` **على الإيقاف**، كما تقدّم؛ فلا يمكن للحقل أن يكون حسّاسًا *وأن يكون* عنوان المُدخل في الوقت نفسه.
- وأصبح منتقي **View Options › Main Title** الخاص بالكيان **يستبعد كل حقل تكون** `options.sensitive` **فيه صحيحة**، بما في ذلك الحقول الفرعية للكتل، إذ يراعي مُرشِّح الاستبعاد البنية الشجرية. وكان المنتقي سابقًا يستبعد فقط حقل العنوان الفرعي المُهيَّأ (وكل ما يتداخل تحته)، دون مراعاة الحساسية إطلاقًا؛ لذا كان بالإمكان *اختيار* حقل حسّاس بالفعل عنوانًا رئيسيًا، رغم أن قاعدة الواجهة الخلفية تهدف إلى منع ذلك.

ويتعلق السبب بكشف البيانات، لا بترتيبها: فالعنوان الرئيسي يُعرَض **غير مُقنَّع** في واجهات القوائم والتفاصيل، لأنه يُستخدَم في تسميات الصفوف والمراجع. ومن ثمّ، كان استخدام حقل حسّاس عنوانًا سيسرّب قيمته إلى كل مستخدم يستطيع رؤية

المُدخل في قائمة.

العنوان الفرعي، على النقيض، مسموحٌ له أن يكون حسّاسًا `Since 1.8`

يسهل الخلط بين هذه القاعدة وقاعدة العنوان الرئيسي: لذا راجع السلوك نفسه بدلًا من افتراض مطابقته للقاعدة أعلاه: **يجوز أن يكون الحقل الحسّاس عنوانًا فرعيًا.** كان ذلك مرفوضًا في مرحلة سابقة من سلسلة الإصدار 1.8 في كلا الاتجاهين، تمامًا كقيد العنوان الرئيسي: إذ كان يفشل كلٌّ من اختيار حقل حسّاس بالفعل عنوانًا فرعيًا، ووضع علامة حسّاس على الحقل المُهيَّأ عنوانًا فرعيًا، على أساس أن واجهات البطاقة والشارة والتقويم والبطاقة المصغّرة والشجرة تعرض جميعها العنوان الفرعي كنصّ خام أيضًا. وأُزيل هذا القيد قبل طرح سلسلة الإصدار هذه، فأصبحت تلك الواجهات تُخفي العنوان الفرعي الحسّاس بنفسها (باستخدام `****` نفسها التي يطبّقها كل مكوّن آخر لعرض الحقول على قيمة حسّاسة بالفعل)، بدلًا من منع استخدام الحقل في هذا الدور. وإذا وجدت وصفًا أقدم يذكر أن العناوين الفرعية محظورة، فهو يعكس تلك الحالة الوسيطة التي لم تعُد سارية، ولا يصف ما تفرضه المنصّة اليوم.

الحقول المُخطَّطة (mapped) ترث الحساسية `Since 1.8`

الحقل المُخطَّط (mapped field) – حقل في كيان، مُهيَّأ ليعكس حقلًا في كيان آخر عبر علاقة (`mapping.enabled` ، `mapping.sourceFieldId`) – يرث تلقائيًا حساسية الحقل المصدر وحاملي منحته. فعند وضع علامة حسّاس على الحقل المصدر أو تغيير من يستطيع قراءته، ينتقل التغيير إلى كل حقل مُخطَّط منه، فتُقنَّع قيمته وتُضبط منحه بالآلية نفسها المطبَّقة على أي حقل حسّاس آخر، دون حاجة إلى معالجة خاصة في مسار القراءة.

وتنتشر التغييرات في كلا الاتجاهين، لأن أيًّا من الطرفين قد يتغيّر أولًا:

- **عند حفظ الكيان المُخطَّط نفسه**، يُعاد اشتقاق حساسية حقوله المُخطَّطة وحاملي منحها من الحقول المصدر قبل اكتمال الحفظ.
- **وعند تغيّر الحقل المصدر دون أن يُمَسّ الكيان المُخطَّط**، يُمرَّر التغيير إلى كل كيان يُخطِّط منه، بالتتابع عبر تخطيطات متعدّدة القفزات (فإذا كان الكيان `C` يعكس `B` الذي يعكس بدوره `A` ، فإن تحديث `B` في عملية التمرير نفسها يجعله المصدر للقفزة التالية إلى `C`)، مع حارس للدورات يمنع حلقة التخطيط من الاستمرار إلى ما لا نهاية. ويؤدي حذف حقل خطّطته كيانات أخرى إلى نشر التغييرات أيضًا، بحيث يفقد الحقل المُخطَّط الذي فقد مصدره صفة الحساسية، وتُزال منحُه المنعكسة من المصدر.

ويُسجَّل فشل النشر دون تمرير الخطأ، كي لا يفشل التعديل الذي أدّى إليه، ويمكن تدارك الفشل بإعادة حفظ أيٍّ من الكيانين.

الحقول المشفَّرة `Since 1.10`

يغيّر وضع علامة **حسّاس** على حقل ما *يُعرَض* للقارئ، أما القيمة المخزَّنة فتبقى نصًّا صريحًا في قاعدة البيانات. أما **التشفير (Encrypt)** فهو خيار ثانٍ مستقل يغيّر ما *يُخزَّن*: إذ تُكتب القيمة نصًّا مشفَّرًا (AES-256-GCM)، وتبقى مشفَّرة لكل مُستدعٍ. فلا يوجد «تجاوز للنظام» يتيح تخطّي التشفير.

التشفير خيار قائم بذاته، والاثنان حصريّان متبادلان

يقع `options.encrypt` إلى جانب `options.sensitive` على الحقل. وفي محرّر الحقل، يظهر مفتاح **التشفير** في كتلة مستقلة موازية لمفتاح **حسّاس**، مباشرةً بعده وقبل **قابل للبحث**، لا متداخلًا فيه. ويجوز للحقل أن يحمل **واحدًا منهما على الأكثر**. ويرفض `FieldService.validateAndNormalizeEncrypt` التركيبة `{sensitive: true, encrypt: true}` بإطلاق `SensitiveAndEncryptMutuallyExclusiveException` (الرمز `SENSITIVE_AND_ENCRYPT_MUTUALLY_EXCLUSIVE` ، والرسالة "*A field cannot have both sensitive and encrypt enabled at the same time.*"، والتفصيل `{field}`).

وثمّة ملاحظتان عمليتان على هذه القاعدة:

- **لا نسخة مترجمة لرمز الخطأ هذا** – فهو غير موجود في قاموس الأخطاء، ومن ثمّ يصل إلى المستخدم دون ترجمة في الحالات النادرة التي يصل فيها إلى هذا المسار.
- **في المحرّر تُنفَّذ القاعدة كإلغاء تفعيل تلقائي صامت ومتناظر، لا كخطأ تحقّق.** فتفعيل «حسّاس» يُلغي تفعيل «التشفير»، وتفعيل «التشفير» يُلغي تفعيل «حسّاس»، وتكون الأولوية للخيار الذي فعّله المصمّم للتوّ. ولا يُعطَّل أي من المفتاحين بسبب الآخر أبدًا، ولا يوجد في جانب العميل أي مُتحقِّق أو رسالة لهذه القاعدة؛ لذا تظلّ رسالة الخادم غير المترجمة الاستجابة الوحيدة إن أرسل عميلٌ الخيارين معًا بطريقة ما. أما الحقل القديم المخزَّن بالعلامتين معًا، فيعتمد المحرّر له **حسّاس** عند تحميله.

كما أن تفعيل التشفير **يمسح "قابل للبحث"** – إذ يفرض الخادم `options.searchable = false` عند الحفظ، فيُبقي النص المشفَّر خارج فهرس البحث النصي الكامل المشترك – ويُفرِغ مُحدِّد الحائزين الموروثين.

ولا تشكّل المنح آليةً ثانيةً؛ فهي **ليست** كذلك. إذ يُخوَّل الوصول إلى الحقل المشفَّر عبر مُحدِّد `sensitiveAccess` العابر نفسه ومُدخلات منح `SENSITIVE_FIELD` نفسها الموصوفة أعلاه، وعلى النطاق `module/entity/field` نفسه. ويُحرَّر منتقي المستخدمين/المجموعات في محرّر الحقل تحت مفتاح التشفير لتظلّ المنح متاحة حين يكون التشفير مفعَّلًا دون «حسّاس»؛ فهو يكتب **المنحة ذاتها**، لا منحة ثانية. أما الأثر في جانب الإدارة – أي إتاحة الحقل المفعَّل فيه التشفير وحده ضمن مربّعات حوار إضافة المنحة أصلًا – فمُوثَّق في الأذونات الحساسة [الإدارة].

أي أنواع الحقول يمكن تشفيرها

ثمة أربعة أنواع لا يمكن تشفيرها أبدًا، ولا يرفضها الخادم، بل **يُعيد بصمت** `encrypt` **إلى** `false`:

لا يمكن تشفيره	كيف يُميَّز
حقل الارتباط / العلاقة	`dataType === DataTypeEnum.Entity`
حاوية الكتلة (Block)	`dataType === DataTypeEnum.Block`
المُحوّل (Switch)	`dataType === Json` مع مكوّن تحرير `SwitchComponent`
التصنيف (Category)	`dataType === Json` مع مكوّن تحرير `CategoryComponent`

وكل ما عدا ذلك قابل للتشفير: النص، والرقم، والتاريخ، والوقت، والتاريخ والوقت، والقيمة المنطقية، و `blob` (الملفات والصور)، والأيقونة، وJSON المترجَم، وGeoJSON، وJSON العادي، ObjectJsong، وأيقونة SVG.

وقائمتا الاستبعاد مختلفتا الشكل عمدًا – في الاتجاهين معًا:

- يستبعد `sensitive` حقلَي الارتباط والكتلة فقط؛ ومن ثمّ **يمكن وضع علامة حسّاس على حقل المُحوّل أو التصنيف، لكن لا يمكن تشفيرهما أبدًا**.
- وفي محرّر الحقل، يُخفى الحقل **المُخطَّط (mapped)** من مفتاح «حسّاس»، لكنه **لا** يُخفى من مفتاح «التشفير». وهذا الاختلاف مقصود ومنصوص عليه في المكوّن نفسه؛ كما لا تتضمّن الواجهة الخلفية قاعدة خاصة بتشفير الحقول المُخطَّطة، وإنما تطبّق قاعدة تعارض أضيق نطاقًا: لا يجوز أن يكون الحقل المُخطَّط ذو المزامنة المستمرة (`ParentOnly` / `TwoWay`) مُخطَّطًا **من** مصدر مشفَّر أو قيد الانتقال (`FIELD_MAPPED_FROM_ENCRYPTED_SOURCE`).

ولأن التطبيع أعلاه يجري **قبل** فحص الحصر المتبادل، فإن إرسال أحد تلك الأنواع الأربعة بالعلامتين معًا يؤدّي إلى تطبيعه دون رفضه؛ إذ تكون قيمة `encrypt` فيه قد أصبحت `false` عند إجراء الفحص، فلا يُطلَق الاستثناء في هذه الحالة أبدًا.

وثمة قيدان إضافيان يقابلان قيدَي «حسّاس»: لا يجوز أن يكون الحقل المشفَّر **أو الذي في منتصف الانتقال** العنوانَ الرئيسي للكيان (`ENCRYPTED_FIELD_CANNOT_BE_MAIN_TITLE` – وخلافًا لرمز الحصر المتبادل، فإن هذا الرمز *له* نسخة مترجمة)، بينما يظلّ تشفير **العنوان الفرعي** مسموحًا، لأن الشاشات التي تعرضه تُقنِّعه بنفسها. ويعتمد فحص العنوان الرئيسي على حالة الانتقال *المحفوظة*، لا على ما يورده جسم الطلب.

كيف يعمل التشفير

تكفي التفاصيل التالية لتخطيط النشر وتمييز القيمة المخزَّنة، دون أن تشكّل مواصفة تعمية:

- **الخوارزمية: AES-256-GCM.** تُشفَّر كل قيمة تحت متجه تهيئة (IV) جديد بطول 12 بايت مع بطاقة استيثاق بطول 16 بايت؛ والمغلَّف هو `iv:authTag:ciphertext`، كل مقطع بصيغة hex ومفصولة بـ `:` . ويرفض فك التشفير أي مغلَّف لا ينقسم إلى ثلاثة أجزاء بالضبط، أو لا يكون متجه التهيئة فيه 12 بايت، أو لا تكون بطاقته 16 بايت، ويثبّت طول البطاقة صراحةً – تحصينٌ مقصود ضد قبول بطاقة قصيرة، ومن ثمّ قابلة للتزوير.
- **طبقتان، وبلا دالة اشتقاق مفاتيح (KDF).** يُقرأ **المفتاح الرئيسي** مباشرةً من متغيّر البيئة `FIELD_ENCRYPTION_MASTER_KEY` – 64 محرفًا hex، أو base64، على أن يُفكّ إلى 32 بايت بالضبط. ولا توجد خطوة PBKDF2 / HKDF / scrypt؛ فقيمة متغيّر البيئة *هي* المفتاح الرئيسي. وهو يغلّف **مفتاح تشفير بيانات (DEK)** بطول 32 بايت مولَّدًا عشوائيًا، والDEK هو ما يشفّر قيم الحقول.
- **مفتاح نشِط واحد لكل قاعدة بيانات لـ data-service؛ ومفتاح رئيسي واحد لكل نشر (مزوِّد `local`).** المفتاح النشِط هو المستند الوحيد الذي تحمل عليه العلامة `isDefault: true` في مجموعة `EncryptionKeys` (علامة يفرضها فهرس، `Since 1.10` – حلّت محلّ اتفاقية سابقة هي «الوثيقة المسمّاة `default` حرفيًا هي النشِطة») – لا يُخزَّن المفتاح الخام أبدًا، بل مفتاح بيانات (DEK) مغلَّف أو، في الوضع `direct`، لا DEK إطلاقًا. ولا يوجد مفتاح لكل قاطن أو كيان أو حقل أو مستخدم، وتدوير المفاتيح *ضمن المزوِّد نفسه* لا يزال خارج النطاق. أمّا ما تضيفه `Since 1.10` فهو قدرة مختلفة – إمكانية *تبديل المزوِّد/الوضع النشِط* – انظر مزوِّدو التشفير أدناه؛ وهذا ليس تدويرًا، ولا يُعيد تشفير أي شيء مخزَّن سلفًا. وغياب المفتاح الافتراضي إخفاقٌ صريح (`DEFAULT_ENCRYPTION_KEY_NOT_FOUND`) لا كتابةً صامتة لقيمة غير مشفَّرة تبدو مشفَّرة، ويُحلّ المزوّد من سجلّ المفتاح نفسه، فلا يمكن أبدًا معالجة مفتاح أُنشئ تحت مزوّد بمزوّد آخر. وإحالة مفتاح إلى التقاعد لا تجعل بياناته غير قابلة للقراءة أبدًا؛ ففكّ التشفير يحلّ المفاتيح بمعرِّفها من النصّ المشفَّر نفسه، فيجب ألّا يُحذَف مفتاح متقاعد (غير افتراضي) ما دام يحمل نصًّا مشفَّرًا ناجيًا.
- **غياب المفتاح الرئيسي إخفاق صريح** عند الإقلاع أو أول استخدام – إلا إذا ضُبط *كلٌّ* من `NODE_ENV=development` و`ALLOW_INSECURE_FIELD_ENCRYPTION_KEY=true`، فيُستخدَم عندئذٍ مفتاح تطوير أصفاره كلها. وهذا البديل غير قابل للبلوغ في بيئة الإنتاج أبدًا.
- **القيمة عند التخزين هي** `<keyId>:<iv>:<tag>:<ciphertext>` – أربعة مقاطع مفصولة بنقطتين، ومعرّف المفتاح هو كل ما يسبق النقطتين *الأوليَين*. و**الحقل متعدّد القيم يُشفَّر عنصرًا عنصرًا**: سلسلة مركّبة لكل عنصر في المصفوفة، لا نصًّا مشفَّرًا واحدًا للمصفوفة كلها. وتُطلِق القيمةُ المخزَّنة المشوَّهة `MALFORMED_ENCRYPTED_VALUE`، وحمولتها تحمل الطول الخام فقط – لا السلسلة الخام أبدًا.
- **لا شيء يكتشف النص المشفَّر بالنظر إليه.** فمعرفة ما إذا كانت القيمة المخزَّنة لمدخل ما تعكس إعداد `encrypt` الحالي للحقل هي مقارنة عددية بسيطة – `entry.entityVersion >= options.encryptTransitionVersion` – فتتصرّف السلسلة والمصفوفة وحقل الكتلة الفرعي تصرّفًا واحدًا، ولا يمكن أبدًا أن تُقرأ قيمة صريحة على أنها نص مشفَّر مصادفةً. كما يمنع حارسٌ على مسار الكتابة تشفير قيمة مشفَّرة سلفًا مرةً ثانية عند إعادة حفظ مدخل لم يتغيّر.

ما الذي يرفضه الحقل المشفَّر

لا تستطيع قاعدة البيانات قراءة محتوى القيمة المشفَّرة؛ ولذلك تُرفَض صراحةً كل عملية تتطلّب القراءة *داخلها*، بدلًا من إنتاج نتائج بلا معنى بصمت:

العملية	على حقل مشفَّر
البحث الكامل في النص	معطَّل – يُفرَض تعطيل `searchable` عند حفظ التشفير.
معايير الترشيح/الاستعلام، و `where` من نوع loopback	**مرفوضة دون شرط** (`SENSITIVE_FIELD_QUERY_NOT_ALLOWED`)، قبل أي فحص للمنحة أو منطقة الإدارة. ولا يغيّر امتلاك المنحة ذلك.
شروط التفاعلات ومعاييرها	**مرفوضة**، برمزها الخاص `ENCRYPTED_FIELD_NOT_ALLOWED` – *"This field is Encrypted and cannot be used in a reaction condition."* كما تُستبعَد الحقول المشفَّرة من جميع منتقيات التفاعلات، في المستوى الأعلى والكيانات الداخلية على السواء.
التجميعات الحسابية في الخلاصة	**مرفوضة تمامًا** – فلا يظهر التجميع إطلاقًا، حتى في صورة مُقنَّعة؛ لأن التجميع الحسابي على نص مشفَّر لا معنى له، وليس مجرد قيمة سرّية. ويُخفي منتقي التجميعات الحقول المشفَّرة للسبب نفسه.
العنوان الرئيسي للكيان	غير مسموح (`ENCRYPTED_FIELD_CANNOT_BE_MAIN_TITLE`).
التقارير	تُستبعَد الحقول المشفَّرة من كتالوج حقول التقرير ومجموعة أعمدة التصدير معًا.
رموز الإشعارات والبريد الإلكتروني	يُستبدَل برمز الحقل أو التعبير أو المؤشِّر الذي يشير إلى حقل مشفَّر سلسلة فارغة. (أما رمز `Fixed` فنصّ حرفي لا يتأثّر.)
قوالب تصدير المستندات، ونسخ بيانات الحقول المُخطَّطة	تُقنَّع دون شرط، قبل تقييم أي تعبير وقبل عرض أي شيء.
التجميع (Grouping)	**لا يفشل – بل ينهار في مجموعة واحدة.** انظر أدناه.

والتجميع هو ما يتصرّف تصرّفًا مختلفًا، وعمدًا. فالمعرض المجمَّع بحسب حقل أصبح لاحقًا مشفَّرًا (أو حسّاسًا) لا يتعطّل، بل يُختزَل المستوى إلى **مجموعة واحدة عنوانها** `****`، تضمّ العدد الإجمالي الصحيح وكل مُدخل مطابق. ولا تختلف النتيجة باختلاف المُطّلِع: فحامل المنحة الذي لديه جلسة كشف نشطة يرى المجموعة الواحدة نفسها التي يراها زائر مجهول، لأن مستوى التجميع المحفوظ إعدادٌ *مشترك* لا خيارٌ خاص بكل طلب، ولأن التجميع بحسب قيمة محميّة لا يفيد أحدًا. ولا يتضمّن مفتاح المجموعة المُقنَّع أي مسار حقل إطلاقًا، فلا تستطيع أي مرحلة في المسار فكّه أو البحث فيه أو تحويله أو الترتيب به. وقد حلّ هذا محلّ تصميم سابق كان يعيد `400` يصف الحقل بأنه حسّاس، وهو ما يكشف معلومات أكثر من مجموعة واحدة بعنوان `****`، ولم يكن بوسع المستخدم فعل شيء حياله. وللسبب نفسه، لم يعُد *منتقي* التجميع يستبعد الحقول الحسّاسة إطلاقًا.

فك تشفير قيمة

لا يُكشف الحقل المشفَّر **أبدًا** عبر مسار الإخفاء التقليدي: إذ يرفضه `maskSensitiveFields` صراحةً، حتى مع وجود منحة أو جلسة. وله آلية كشف واحدة لا غير:

```
POST /v2/entities/{entity}/entries/{entryId}/fields/{field}/decrypt
Body: { "stepUpProof"?: string, "rowIndex"?: number }
200 → { "value": <decrypted value> }
```

لا متطلّبات لجسم الطلب سوى الإثبات. وهذه **POST بوصفها قراءة**: إذ يحلّ مُصوِّت على مستوى الدالة محلّ بوابة التحرير على مستوى الصنف، ويفحص `canViewEntity` لا `canEditEntity`، لأن الاستدعاء يجلب قيمة ويكشفها، ولا يكتب شيئًا أبدًا.

ويعمل المُعالِج بالترتيب الآتي:

1. استرجاع الكيان والحقل، وإطلاق `EntityFieldNotExistException` إن غابا.
2. التحقق من جواز قراءة المُستدعي **هذا المدخل تحديدًا** (`ForbiddenToReadEntryException`)؛ إذ كان الاكتفاء بالمُصوِّت على مستوى الصنف سيسمح لمُطّلِع على مستوى النظام بقراءة بيانات أي كيان.

3. رفض الحقل القابل للتكرار ما لم يُقدَّم `rowIndex` – `DecryptNotSupportedForRepeatableFieldException` / `DECRYPT_NOT_SUPPORTED_FOR_REPEATABLE_FIELD`.

4. رفض الحقل المعروض بوصفه كلمة مرور قبل جلب المُدخل أصلًا – `DecryptNotSupportedForPasswordFieldException` (`Since 1.10`) – لأن القيمة المخزَّنة لحقل كلمة المرور تجزئة (hash) أحادية الاتجاه، ولا تتضمّن شيئًا يمكن فكّ تشفيره.

5. رفض الحقل الذي لا يحتوي على ما يمكن فكّ تشفيره – `FIELD_NOT_ENCRYPTED`. أما الحقل الذي لا يزال قيد **إطفاء** التشفير ولم تُرقَّ مُدخلاته بعد، فيبقى **قابلًا للبلوغ عمدًا**: فهذا هو مسار الاسترداد الموثَّق، انظر تبديل التشفير.

6. **منحة الحقل.** فحص `canAccessSensitiveField` على نطاق الحقل `module/entity/field`؛ ويكون الإخفاق بالرمز `403` (`SensitiveFieldDecryptForbiddenException` / `SENSITIVE_FIELD_DECRYPT_FORBIDDEN`، "*Permission denied. You do not have access to decrypt this field.*").

7. `Since 1.10` – **إثبات تحقق إضافي جديد، مرتبط ارتباطًا صحيحًا، أحادي الاستخدام.** يستهلك `StepUpProofService.consumeDecrypt(body?.stepUpProof, {userId, sessionId, entityId, entryId, fieldId, rowIndex})` الإثبات ذرّيًا (أمر ريديس `GETDEL`)، ويُعيد مقارنة كل حقل من حقول الارتباط. ويؤدّي الإثبات الغائب أو المنتهي الصلاحية أو المُستهلَك سلفًا أو غير المطابق إلى إطلاق `SensitiveFieldDecryptStepUpRequiredException` – وهو الاستثناء نفسه سواء لم يُقدَّم الإثبات أصلًا أم كان خاطئًا، فلا تميّز الاستجابة أبدًا بين «نسيت التحقّق» و«تحقّقك لم يطابق هذا الطلب بالضبط». انظر كشف القناع بالتحقق الإضافي، Decryptg، WorkflowActiong أدناه لمعرفة كيفية إصدار الإثبات.

8. فكّ التشفير عبر `readEncryptedFieldValue`. وتُمرَّر القيمة `null` / `undefined`، أو القيمة التي يتعذّر تحليلها على النحو المعتاد، بالطريقة نفسها المتّبعة مع أي نوع آخر من الحقول المشفَّرة؛ فلا يُنشئ فكّ التشفير قيمة من عنده أبدًا.

9. نشر `SensitiveDataDecryptedEvent`.

والمنحة والإثبات كلاهما مطلوب – لا يكفي أحدهما وحده، ولا يوجد **أي استثناء للمسؤول (admin)**: فخلافًا لِـ `WorkflowAction` (أدناه)، التي تتخطّى التحقق الإضافي عند التنفيذ الإداري، ينطبق Decrypt على كل مُستدعٍ، حتى لا يُفرَض الفحص على حامل المنحة بينما يُسمح للمسؤول بتخطّيه بصمت. ويمكن للمُستدعي المرفوض (الذي لا يملك منحة) إعادة الطلب نفسه بلا حدّ، وسيحصل دائمًا على `403` نفسه؛ فلا شيء هنا يمكن تخمينه، ومن ثمّ لا حظر مؤقت لفحص المنحة نفسه. أما المُستدعي الذي *يملك* المنحة لكنه يفشل مرارًا في تحدّي التحقق الإضافي (بإدخال رمز/كلمة مرور خاطئة)، فيخضع لحدّ معدّل OTP/MFA نفسه على مستوى الحساب، الذي يشترك فيه مسار Unmask التقليدي – انظر جلسة الكشف بالتحقق الإضافي أعلاه.

وتُسجَّل كل عملية فكّ تشفير ناجحة في سجلّ التدقيق تحت **اسم حدث خاص به**، لا بوصفها سجلّ كشف إضافيًا، مع بيان من نفّذها، والوحدة والكيان والحقول، والمُدخلات المعنيّة. و**القيمة المفكوكة نفسها لا تُسجَّل أبدًا**. وهذا سجلّ تدقيق منفصل عن فحص التحقق الإضافي نفسه، الذي يُسجَّل أيضًا – انظر كشف القناع بالتحقق الإضافي، Decryptg، WorkflowActiong أدناه. انظر المراقبة: السجلات والتدقيق [الإدارة] وسجلّ التحقق الإضافي المنفصل [الإدارة › المراقبة: السجلات والتدقيق] لمعرفة ما يراه المسؤول، وأمان الحساب وكشف البيانات الحساسة [استخدام أسس › أمان الحساب وكشف القيم المحمية] لمعرفة ما يظهر للشخص الذي ينقر الأيقونة.

كشف القناع بالتحقق الإضافي، Decrypt، WorkflowAction: ثلاثة أغراض، آلية واحدة Since 1.10

و Since 1.10 عمّمت المنصة آلية التحقق الإضافي التي كانت ذات غرضَين (Unmask ، الموجود منذ 1.8؛ و WorkflowAction ، المضاف في الإصدار نفسه – انظر مرجع نموذج سير العمل وواجهة API: التحقق الإضافي قبل تنفيذ إجراء) إلى ثلاثة أغراض، بإضافة Decrypt . والأغراض الثلاثة أعضاء في تعداد واحد هو StepUpPurpose (packages/osos-shared/src/step-up-proof.ts؛ وانظر أيضًا core-reference-enums.md)، وتتشارك نقاط النهاية الأربع ذاتها في users-service (POST /identity/mfa/step-up/{challenge,verify,disable}، و GET .../status) وفتحة ريديس نفسها للتحدّي المُعلَّق، بمفتاح من الحساب وجلسة OIDC – لكن الغرض يُحدَّد عند التحدّي، و**لا يمكن أبدًا** تبادل الأغراض الثلاثة: إذ تتفرّع verifyUnmask بحسب pending.purpose ، والإثبات أو الجلسة الصادرة لغرض ما لا تفي بغرض آخر أبدًا.

الغرض	الحالة	الآلية	الارتباط	مدة الصلاحية (TTL)
Unmask ('unmask')	موجود مسبقًا، **Since 1.8**	**جلسة** قابلة لإعادة الاستخدام في ريديس، تُفتَح لمدة يختارها المُستدعي، ويدعم نطاقها المطابقة بالمحرف البديل (wildcard) وصولًا إلى النطاق الشامل */*/*	الحساب + جلسة OIDC + النطاق	يختارها المُستدعي من القائمة المسموحة في مدد الكشف

Decrypt ('decrypt')	**جديد، Since 1.10**	**إثبات أحادي الاستخدام** – يكشف حقلًا واحدًا بالضبط، ثم يُتلَف عند الاستهلاك	{entityId, entryId, fieldId, rowIndex?} – يُعاد فحص كل حقل عند الاستهلاك	120 ثانية	
WorkflowAction ('workflow-action')	**جديد، Since 1.10**	**إثبات أحادي الاستخدام لكل مدخل** – قابل للتجميع: يمكن لفحص عامل واحد إصدار خريطة proofs: {entryId: proof} تغطّي جميع المُدخلات المحدَّدة	{entityId, entryId, actionId} × N، ارتباط واحد لكل مُدخل	مدة الصلاحية نفسها، 120 ثانية لكل إثبات	

تنطبق تفاصيل التصميم التالية على الأغراض الثلاثة جميعًا:

- **الإثبات هو السرّ نفسه.** لِـ WorkflowAction / Decrypt ، يُنشئ الإصدار معرِّفًا عبر crypto.randomUUID() ويخزِّنه مع ارتباطه في مفتاح ريديس مخصَّص للإثبات. ويستخدم الاستهلاك أمر GETDEL – ذرّي، بحيث لا يفوز إلا مستدعٍ واحد – و**تقديم إثبات غير مُطابِق يُتلِف الإثبات رغم ذلك** بدل تركه متاحًا لإعادة المحاولة.
- **يرفض الإصدار مستدعيًا بلا جلسة.** يُرفَض مستدعٍ من نوع client_credentials/ROPC (بلا جلسة OIDC) تحدّي Decrypt أو WorkflowAction مباشرةً – إذ لا يمكن استهلاك مثل هذا الإثبات بأمان أو تدقيقه أبدًا. لا يحمل Unmask هذا القيد، إذ يتراجع مستدعٍ بلا جلسة فيه ببساطة إلى منحة على مستوى الحساب بأكمله.
- **Decrypt وحده يعرض دومًا بديل كلمة المرور، بمعزل عن إعداد allowPasswordStepUp الخاص بالقاطن.** يتراجع Unmask و WorkflowAction إلى كلمة مرور الحساب فقط حين لا يملك المستدعي طريقة MFA قابلة للاستخدام **و**كان النطاق قد اشترك صراحةً. يتجاهل Decrypt هذا الإعداد ويسمح دومًا بالبديل (مع بقاء الشرط أن يفتقر المستخدم إلى طريقة MFA قابلة للاستخدام ويملك كلمة مرور محلية) – لأن حامل المنحة يجب ألّا يُترَك أبدًا بلا أي طريقة لكشف حقل يملك منحته، على نطاق مُهيَّأ افتراضيًا لم يُفعِّل قط البديل الخالي من MFA لأيّ شيء آخر.
- **يمكن لتحدّي WorkflowAction أن يُجمَّع دفعيًا؛ Decrypt لا يُجمَّع أبدًا.** تُحدّ دفعة WorkflowAction بـ100 ارتباط، وكل ارتباط يشارك entityId / actionId نفسهما (إجراء واحد، مداخل كثيرة – لا دفعة مختلطة أبدًا)، ولا entryId مكرَّر؛ وانتهاء صلاحية الدفعة الفعلي هو قيمة expiresAt في **الإثبات الأول**، إذ لا تبقى الدفعة قابلة للاستخدام إلا ما دام أقدم عضو فيها كذلك. يقبل Decrypt ارتباطًا واحدًا فقط – كشف واحد لكل إثبات، في كل مرة.
- **التدقيق.** يسجِّل كلا الغرضَين الجديدَين نتيجة فحص الهوية على سجلّ تدقيق MFA الحالي عبر تابعَين مخصَّصَين، **منفصلَين عن، وإضافةً إلى** حدث تدقيق مدخل البيانات الذي يُصدره كشف ناجح أو تنفيذ إجراء سير عمل على نحو مستقل. انظر المراقبة: السجلات والتدقيق [الإدارة].

إعادة توليد/عرض الرموز الاحتياطية آلية رابعة منفصلة وأبسط – وليست `StepUpPurpose` . تتلقّى `POST` `/identity/mfa/totp/recovery-codes/regenerate` رمزًا أو كلمة مرور مباشرةً في جسم الطلب، وتتحقّق منه ضمنيًا، وفق النمط نفسه الذي تستخدمه نقطة نهاية تعطيل TOTP الموجودة مسبقًا. ولا تشترك مع الأغراض الثلاثة أعلاه في أي جلسة ريديس، أو إثبات أحادي الاستخدام، أو قيمة `StepUpPurpose` ، ولا تُوصَف بمصطلحات step-up. انظر أمان الحساب وكشف البيانات الحساسة [استخدام أسس › أمان الحساب وكشف القيم المحمية] للاطّلاع على التدفّق الموجَّه للمستخدم النهائي.

ترد التفاصيل الكاملة لإجراء سير العمل – خاصية `stepUp` في تهيئة إجراء سير العمل، ونقاط النهاية، وعقد التجميع الدفعي من طرف إلى طرف، وشكل الرفض – في مرجع نموذج سير العمل وواجهة API: التحقق الإضافي قبل تنفيذ إجراء. وتغطّي المقارنة أدناه `Decrypt` تحديدًا، مقابل جلسة `Unmask` التقليدية:

	حسّاس: الكشف بالتحقق الإضافي	مشفَّر: فك التشفير `Since 1.10`
الشرط المسبق	منحة **و** جلسة تحقق إضافي نشطة تغطّي النطاق	منحة **و** إثبات جديد غير مُستهلَك، مرتبط ارتباطًا صحيحًا، وأحادي الاستخدام
التحقق الإضافي (OTP/MFA/كلمة مرور)	نعم – يفتح جلسة	**نعم،** `Since 1.10` – يُصدر إثباتًا صالحًا لكشف واحد بالضبط
الجلسة مقابل الإثبات	جلسة ريديس بمدّة يختارها المُستدعي، مرتبطة بجلسة OIDC، وقابلة لإعادة الاستخدام لكل حقل مطابق حتى انتهاء صلاحيتها	إثبات ريديس أحادي الاستخدام، مدة صلاحيته 120 ثانية، مرتبط بتركيبة واحدة بالضبط من `{entity, entry, field, rowIndex?}` ، ويُتلَف عند أول استخدام سواء طابق أم لا
مدى الكشف الواحد	كل النطاقات التي تغطّيها الجلسة، وقد تشمل عمليًا جميع النطاقات (عبر المطابقة بالمحرف البديل وصولًا إلى النطاق الشامل)	حقل واحد بالضبط في مُدخل واحد (+ صفّ، لحقل قابل للتكرار داخل كتلة)
كيف تصل القيمة	تُعيد القراءة التالية القيمة غير مُقنَّعة	تُعاد ضمنيًا من الاستدعاء نفسه
الحقول القابلة للتكرار	مدعومة	**مرفوضة** ما لم يُقدَّم `rowIndex`
حدث التدقيق	`SENSITIVE_DATA_UNMASKED` : لا صفّ منفصل للتحقق الإضافي لهذا الغرض	`SENSITIVE_DATA_DECRYPTED` **و** صفّ تحقق إضافي موسوم بـ `Decrypt` (سجلّ تدقيق MFA)
استثناء المسؤول	لا ينطبق – Unmask هي *نفسها* الجلسة لكل مستخدم	**لا يوجد.** خلافًا لِـ `WorkflowAction` ، التي تتخطّى التحقق الإضافي عند التنفيذ الإداري، ينطبق Decrypt على كل مُستدعٍ.

ماذا يفعل العميل، `Since 1.10` **.** يعرض كلٌّ من الويب والجوّال الآن مطالبة تحقق إضافي قبل استدعاء Decrypt عند كل نقرة، لا عند النقرة الأولى فقط. ويستخدم الويب غلاف مربّع حوار موحَّدًا لمطالبتَي Decrypt وWorkflowAction، لكنه يتعمَّد **عدم** توجيه Decrypt عبر المساعد نفسه الذي تستخدمه `WorkflowAction` ، لأن ذلك المساعد يطبِّق استثناء المسؤول، الذي يجب ألّا ينطبق على Decrypt أبدًا. ويعرض الجوّال مطالبة التحقّق المُعمَّمة نفسها التي تستخدمها بوّابة إجراء سير العمل. ولا يعرض أيّ من العميلَين أداة الكشف إطلاقًا لحقل معروض بوصفه كلمة مرور، بل يظهر القفل الدائم بدلًا منها، اتّساقًا مع ضابط الواجهة الخلفية في الخطوة 4 أعلاه.

تبديل التشفير: تغيير كاسر مُرقَّم

لا يوجد **أي ترحيل بيانات (backfill)** مرتبط بمفتاح التشفير. فتفعيله أو إلغاء تفعيله يُعامَل بوصفه تغييرًا كاسرًا في المخطَّط، مرتبطًا برقم إصدار، وتُعالَج آثاره عبر تدفّق **ترقية المدخلات (Upgrade Records)** المعتاد:

1. يضيف تغيير `encrypt` السبب `Field encryption changed` إلى قائمة التغييرات الكاسرة لإصدار الكيان الجديد. ويُضاف **حتى إن كان سبب آخر ينطبق بالفعل** (كتغيّر النوع مثلًا)، لضمان إجراء الوسم في الخطوة 2 دائمًا.
2. يُوسَم كل حقل متأثّر بـ `encryptTransitionVersion` (إصدار الكيان الجديد) و `pendingEncryptUpgrade = true` . ولا يُعرَض أيٌّ منهما للمسؤول، وتُعاد استعادة وسمَيهما من الحقل المحفوظ عند كل تحديث للحقل، بصرف النظر عمّا يرسله جسم الطلب؛ فلا يستطيع عميل يُغفلهما محو انتقال جارٍ بصمت.

3. تصبح المُدخلات ذات الإصدار الأقدم من ذلك الإصدار **متقادمة (Obsolete)**، ويعالجها المسؤول عبر ترقية المُدخلات كالمعتاد. وطوال هذه الفترة، يُستبعَد الحقل من جميع واجهات البناء.
4. بعد كل ترقية ناجحة، تَعُدّ المنصة المُدخلات التي لا تزال أقدم من إصدار الانتقال، **متجاهلةً المدخلات المؤرشفة** – وإلا لما بلغ العدّ صفرًا أبدًا ولبقيت العلامة عالقة إلى الأبد – ثم تمسح `pendingEncryptUpgrade` حين لا يبقى أي مُدخل.
5. **يُرفَض "الإصلاح التلقائي" (Auto Repair) عند إطفاء المفتاح.**

`SYSTEM_BEHAVIOR_NOT_ALLOWED_FOR_ENCRYPTION_TOGGLE_OFF`: "*Auto Repair cannot be used to resolve this field while its Encrypt toggle-off is still in progress. Choose Fixed, Expression, Dynamic, or Not Set instead.*" فالقيمة المخزَّنة تظلّ نصًّا مشفَّرًا فعليًا، وما إن تصبح قيمة `encrypt` هي `false` حتى لا يعود هناك ما يشفّرها أو يقنّعها؛ وعندئذٍ يُرقّي الإصلاح التلقائي النص المشفَّر الخام ليصبح القيمة الفعلية للمُدخل من ذلك الحين. و**الاسترداد الموثَّق هو استدعاء فك التشفير على المدخل – الذي يبقى قابلًا للبلوغ لهذه الحالة تحديدًا – ثم تقديم النص الصريح بوصفه الحلّ `Fixed` في مربّع الحوار.** ويظلّ الإصلاح التلقائي متاحًا في اتجاه **التشغيل**، وهو الخيار المقصود بعبارة «شفّره فحسب».

وإعادة تبديل التشفير بينما لا يزال الانتقال معلَّقًا **مسموحة**؛ فقد أُزيل بعد الإطلاق ضابط سابق كان يرفضها.

مزوِّدو التشفير ومفاتيحه `Since 1.10`

اختيار خدمة المفاتيح التي تشفّر قيم الحقول (`local`، أو HashiCorp Vault Transit، أو Azure Key Vault)، ونمطا المفاتيح `wrap` و`direct`، وتهيئة المزوِّد وتفعيله، وتطبيق تغيير المزوِّد على تثبيت قائم، وتوصيلات المخطط – كلّها من اختصاص المشغّلين، وموثّقة في مفاتيح تشفير الحقول ومزوّدوها. ولا يتغيّر شيء ممّا توضّحه هذه الصفحة بتغيّر المزوِّد؛ فالمنح والإخفاء وDecrypt وحالات الرفض تظلّ على السلوك نفسه.

سياسة الوصول إلى البيانات الحساسة `Since 1.10`

تجيب المنحة عن سؤال: «هل يجوز *لهذا القارئ* أن يرى هذا الحقل؟». لكن بعض العناصر تُشغَّل أو تُولَّد نيابةً عن شخص – كتفاعل أو قالب تصدير أو خلاصة – والسؤال هنا مختلف: «ما الذي يجوز *لهذا العنصر* كشفه عند تشغيله؟». ويحدّد ذلك إعداد منفصل على مستوى العنصر: **سياسة الوصول إلى البيانات الحساسة**.

السياسة	معناها
محظور (Prohibited)	لا تكشف أبدًا، أيًّا كان من ينفّذ العنصر أو يفتحه أو يولّده.
صلاحية كاملة (Privileged)	اكشف دائمًا، بصرف النظر عن منح المنفِّذ نفسه. **وهي الافتراضية.**
حسب إذن المستخدم (User-Based)	يتحدّد القرار في كل تنفيذ وفق منح الحقول الحسّاسة الخاصة بالمنفِّذ الفعلي.
تلقائي (Auto)	يعامل التشغيل النظامي وفق سياسة «صلاحية كاملة»؛ والتشغيل الذي يبدأه مستخدم وفق سياسة «حسب إذن المستخدم».

وتُخزَّن السياسة على العنصر باسم `sensitiveDataAccessPolicy`. و**القيمة الغائبة تُحلّ إلى `Privileged`**، فيحتفظ كل تفاعل وقالب وخلاصة سبقت هذه الميزة بسلوكه السابق – فلم يتغيّر شيء ممّا كان قائمًا، ولم يلزم أي ترحيل للبيانات. أما القيمة المخزَّنة *غير المعروفة* فتُحلّ إلى الإخفاء؛ إذ يمنع المُحلِّل الكشف عند تعذّر تحديد السياسة.

أين تنطبق – وأين لا تنطبق عمدًا:

العنصر	هل السياسة متاحة؟
التفاعل	نعم – بالقيم الأربع كلها.

العنصر	هل السياسة متاحة؟
قالب تصدير المستندات	نعم – بالقيم الأربع كلها.
الخلاصة (بطاقة لوحة المعلومات / الاستعلام الإحصائي)	نعم، **باستثناء** `Auto` – إذ لا تُشغَّل الخلاصة تلقائيًا أو وفق جدول مستقلّ عن عرضها الحيّ، لذا لا يُتاح هذا الخيار إطلاقًا.
التقرير	**لا، عمدًا.** فالتقرير يظلّ خاضعًا لإذن الحقول الحسّاسة الخاص بكل مُطّلِع.
المعارض والاستعلامات العادية وقوائم المُدخلات	لا تخضع لهذه السياسة إطلاقًا.

ولا يظهر عنصر التحكّم في نماذج التفاعل والقالب والخلاصة **إلا حين يعلن الكيان فعلًا عن حقل حسّاس واحد على الأقل** (بما في ذلك حقول الكتل الفرعية)؛ وعند إخفائه يُعطَّل كي لا يمنع الحفظ دون تنبيه.

وتُطبَّق ثلاث قواعد عند حفظ العنصر أو تشغيله:

- `Auto` **على خلاصة** مرفوضة – *«خيار «تلقائي» غير متاح على الخلاصة»*.
- `User-Based` **مع مُطلِق تفاعل تلقائي** – منفِّذ نظام، أو مؤقّت، أو حدث تجاوز اتفاقية مستوى الخدمة – مرفوضة: *«لا يمكن اختيار سياسة «حسب المستخدم» مع نوع مُطلِق تلقائي»*؛ إذ لا يوجد منفِّذ بشري يُرجَع إلى منحه.
- والتفاعل الذي تُخفي سياستُه قيمةً يحتاج لاحقًا إلى **العمل عليها** *يفشل* (`SENSITIVE_DATA_ACCESS_PROHIBITED`) بدلًا من حفظ `****`. ولا تسمح بتمرير قيمة مُقنَّعة سوى إجراءات العرض الخمسة – إرسال إشعار، وإرسال بريد إلكتروني، وإرسال رسالة نصية، وتوليد ملف من قالب، والإعلان عن حدث.

والخلاصة التي ترفضها السياسة تُخفى بالكامل، لا تُقنَّع. فلا يظهر مبيان ولا بطاقة ولا `****`؛ بل يُرجع الطلب `403`، وتعرض لوحة المعلومات *«لا تملك إذنًا لعرض هذه الخلاصة»* مكان البطاقة. وعند تطبيق `User-Based` يجب أن يملك المنفِّذ منحةً لـ**كل** حقل حسّاس تقرؤه الخلاصة – فالتجميع المكشوف جزئيًا تسريبٌ أيضًا – ويشمل ذلك الآن الحقولَ التي تُطبَّق عليها شروط الترشيح في **استعلام المصدر** الخاص بالخلاصة، لا الحقول المجمَّعة والمصنَّفة فحسب. (قارن ذلك بقاعدة الحقل الواحد في تجميع اعتيادي لا يملك *القارئ* منحةً عليه؛ إذ يظلّ ظاهرًا باسم العرض الخاص به، ويعرض `****`، لأن إخفاء التجميع كان سيكشف بحدّ ذاته عن معلومات.)

ملخّص المسار من أوله إلى آخره

1. يضع المصمّم علامة `sensitive` على حقل ويختار المستخدمين/المجموعات في محرّر الحقل، أو يمنح المسؤولُ الوصولَ لاحقًا عبر واجهة API للمنح. ويكون النطاق `module/entity/field`، أو نطاقًا بالرمز البديل لمنحة موروثة.
2. عند الحفظ، يكتب `syncSensitiveFieldAccess` (محرّر الحقل) أو `createSensitiveFieldGrants` (واجهة الإدارة) مُدخلات `SENSITIVE_FIELD` من `AccessRight` للنطاق وينشر `AccessRightChangedEvent`.
3. عند تحليل الأذونات، تتدفّق نطاقات المنح المطابقة إلى `permissions.sensitiveFields`.
4. ولكي *يرى* المستخدم القيمة فعلًا، عليه إضافةً إلى ذلك أن يفتح جلسة كشف بالتحقق الإضافي تشمل النطاق؛ وتلك الجلسة تابعة لجلسة دخوله المحدَّدة، لا لحسابه فحسب.
5. عند قراءة المدخل، لا يكشف `maskSensitiveFields` الحقلَ إلا حين تتطابق المنحة ونطاق كشف نشط معًا؛ وإلا فيستبدل به حارس القناع. وتؤدّي الحقول المكشوفة إلى نشر حدث `SensitiveDataUnmaskedEvent` مُجمَّع لأغراض التدقيق. ويُبلِّغ تعريف الكيان عن أسماء الحقول المُقنَّعة في `sensitiveMaskedFields` (بلا منحة) و`sensitiveMaskedFieldsWithUserAccess` (منحة موجودة، لكن الجلسة لا تغطّيها).
6. عند الاستعلام، تتسبّب الحقول الحسّاسة غير القابلة للقراءة في معايير/where المستخدم بإطلاق `SensitiveFieldQueryNotAllowedException` (فحص المنحة وحدها، دون بوابة الجلسة).

7. عند قراءة التدقيق، تُستبدَل القيم الحسّاسة `old` / `new` بالقيمة `***`، ويشترط ذلك الآن كلًّا من المنحة **و**جلسة كشف قناع نشطة تغطّي النطاق؛ ويُقنِّع كاتِب التدقيق، بأسلوب الفشل المُغلَق، قبل الحفظ.
8. وإذا كان الحقل يعكس حقل كيان آخر (**حقل مُخطَّط**)، تُبقى حساسيته وحامِلو منحته متزامنَين مع الحقل المصدر تلقائيًا، في الاتجاهين معًا – انظر الحقول المُخطَّطة ترث الحساسية.
9. `Since 1.10` وإذا كان الحقل **مشفَّرًا** لا حسّاسًا، فلا تنطبق الخطوات 4-6 إطلاقًا: فالقيمة نصّ مشفَّر عند التخزين، وترفضه كل واجهة بناء (والتجميع ينهار في مجموعة واحدة بدلًا من ذلك)، والسبيل الوحيد لرؤيته هو استدعاء فك التشفير لكل مدخل، مقيَّدًا بالمنحة **و**إثبات تحقق إضافي طازج لمرة استخدام واحدة (`Since 1.10`) – دون جلسة قابلة لإعادة الاستخدام، لكن لم يعُد الأمر بلا تحقّق أيضًا – ومُدقَّقًا تحت اسم حدث خاص به مع سجلّ منفصل للتحقق الإضافي.
10. `Since 1.10` ويحمل التفاعل أو قالب التصدير أو الخلاصة إضافةً إلى ذلك سياسة الوصول إلى البيانات الحساسة الخاصة به، وهي تقرّر ما الذي يجوز *لذلك العنصر* كشفه عند تشغيله. والافتراضية `Privileged` هي بالضبط ما كانت الثلاثة تفعله قبل وجود الإعداد.
11. `Since 1.10` ويمرّ كلٌّ من كشف Decrypt وتنفيذ `WorkflowAction` المُقيَّد عبر الآلية نفسها المُعمَّمة ثلاثية الأغراض للتحقق الإضافي – انظر كشف القناع بالتحقق الإضافي، وDecrypt، وWorkflowAction.

مرجع: الرموز الأساسية

الرمز	الملف	الدور	منذ
`buildSensitiveFieldScope`	`libs/sensitive-field.ts`	بناء نطاق `module/entity/field`.	1.8
`matchesSensitiveScope`	`libs/sensitive-field.ts`	مطابقة المقاطع مع الرمز البديل (wildcard).	1.8
`SENSITIVE_DISPLAY_MASK`	`libs/sensitive-field.ts`	الحارس المكوَّن من 26 محرفًا الذي يُرجَع فعليًا عند القراءة.	1.8
`renderSensitiveMaskAsStars` / `SENSITIVE_STARS`	`libs/sensitive-field.ts`	نجوم `****` البشرية ومُعقِّم النصوص المُستبدَلة الذي يُنتجها.	1.8
`SENSITIVE_MASK` (قديم) / `AUDIT_SENSITIVE_MASK`	`libs/sensitive-field.ts`	كاشف الحارس القديم / قناع التدقيق.	1.8
`SensitiveFieldsService.canAccessSensitiveField`	`services/sensitive-fields.service.ts`	فحص المنحة الثنائي (دون تجاوز للمسؤول).	1.8
`SensitiveFieldsService.filterQueryableFieldsForUser`	`services/sensitive-fields.service.ts`	إسقاط الحقول الحسّاسة غير القابلة للوصول من قائمة.	1.8
`SensitiveFieldsService.getSensitiveFieldOptions`	`services/sensitive-fields.service.ts`	الكتالوج على مستوى المساحة (مع `excludeScopes` اختياري).	1.8
`AccessRightsService.syncSensitiveFieldAccess`	`services/access-rights.service.ts`	مقارنة المنح التصريحية + الحدث (مسار محرّر الحقل).	1.8
`AccessRightsService.listSensitiveFieldGrants`	`services/access-rights.service.ts`	قائمة المنح النشِطة على مستوى المساحة.	1.8
`AccessRightsService.listUserSensitiveFieldGrants`	`services/access-rights.service.ts`	منح المستخدم المباشرة والموروثة عبر مجموعاته.	1.8
`AccessRightsService.listGroupSensitiveFieldGrants`	`services/access-rights.service.ts`	منح المجموعة المباشرة.	1.8
`AccessRightsService.getInheritedSensitiveFieldHolders`	`services/access-rights.service.ts`	حائزو النطاق بالرمز البديل لحقل واحد.	1.8

الرمز	الملف	الدور	منذ
`AccessRightsService.createSensitiveFieldGrants` / `updateSensitiveFieldGrant` / `deleteSensitiveFieldGrant` `archiveSensitiveFieldGrant`	`services/access-rights.service.ts`	عمليات التعديل في واجهة إدارة المنح.	`1.8`
`AccessRightsService.getSensitiveFieldGrants`	`services/access-rights.service.ts`	قراءة المنح تحت بادئة نطاق كيان.	`1.8`
`AccessRightsService.removeSensitiveFieldAccess`	`services/access-rights.service.ts`	تفكيك المنح عند نطاق أو تحته.	`1.8`
`DataEntityService.attachSensitiveFieldsAccess`	`services/data-entity.service.ts`	إعادة ملء `sensitiveAccess` عند القراءة.	`1.8`
`DataEntityService.computeSensitiveMaskedFields`	`services/data-entity.service.ts`	قائمتا أسماء الحقول المُقنَّعة لكل طلب (`sensitiveMaskedFields` + `sensitiveMaskedFieldsWithUserAccess`).	`1.8`
`DataEntityService.propagateMappedFieldSensitivity` / `propagateSensitivityToDependentMappedFields`	`services/data-entity.service.ts`	وراثة حساسية الحقول المُخطَّطة، في الاتجاهين.	`1.8`
`DataEntryService.maskSensitiveFields` / `maskEntryForResponse` / `applyCallerSensitiveMask`	`services/data-entry.service.ts`	التقنيع عند القراءة بالمنحة والجلسة.	`1.8`
`DataEntryWriterService.resolveSavedEntryUnmasked`	`services/data-entry-writer.service.ts`	إعادة القراءة غير المُقنَّعة بعد الحفظ.	`1.8`
`UnmaskSessionService.getActiveScopes`	`services/unmask-session.service.ts`	قراءة نطاقات الكشف النشطة للمستخدم، المحدودة بالجلسة (تفشل مُغلَقة).	`1.8`
`challengeUnmask` / `verifyUnmask` / `disableUnmask` / `getUnmaskStatus`	`users-service/src/services/step-up-unmask.service.ts`	دورة حياة جلسة التحقق الإضافي.	`1.8`
`stepUpUnmaskRedisKey(envPrefix, userId, sessionId)` / `StepUpUnmaskSession` / `STEP_UP_UNMASK_ALL_SCOPE` / `STEP_UP_UNMASK_DURATIONS_SECONDS`	`packages/osos-shared/src/step-up-unmask.ts`	عقد الجلسة بين الخدمات: الفهرسة المحدودة بالجلسة ومجموعة المدد الحالية كلتاهما من هذا الإصدار.	`1.8`
`PermissionTypeEnum.SensitiveField`	`packages/osos-shared/src/models/enums.ts`	`'SENSITIVE_FIELD'` .	`1.8`
`FieldService.validateAndNormalizeEncrypt`	`services/field.service.ts`	تطبيع الأنواع الأربعة المستبعَدة، ورفض `sensitive + encrypt` ، وفرض `searchable = false`.	`1.10`
`options.encrypt` / `encryptTransitionVersion` / `pendingEncryptUpgrade`	`models/data-entity/field-options.model.ts`	مفتاح التشفير ورفيقاه اللذان يملكهما الخادم لإدارة الانتقال.	`1.10`
`aesGcmEncrypt` / `aesGcmDecrypt`	`libs/aes-gcm.ts`	مغلِّف AES-256-GCM (`iv:tag:ciphertext`) مع تثبيت بطاقة بطول 16 بايت؛ يتشاركه مع أسرار TOTP.	`1.10`
`getFieldEncryptionMasterKey`	`libs/field-encryption-master-key.ts`	قراءة `FIELD_ENCRYPTION_MASTER_KEY` والتحقق منه (32 بايت، hex أو base64).	`1.10`
`writeEncryptedFieldValue` / `readEncryptedFieldValue` / `isFieldValueMigrated`	`libs/encrypted-field-value.ts`	تركيب `<keyId>:<iv>:<tag>:<ciphertext>` عند التخزين، واختبار الترحيل بمقارنة الإصدار.	`1.10`
`DataEntriesController.decryptField`	`controllers/data-entries.controller.ts`	`POST …/fields/{field}/decrypt` — منحة **و**إثبات تحقق إضافي لمرة استخدام واحدة، لحقل واحد على مدخل واحد.	`1.10`

الرمز	الملف	الدور	منذ
`SensitiveDataDecryptedEvent`	`packages/osos-shared/src/messages/events/sensitive-data/`	حدث تدقيق فك التشفير المخصَّص؛ لا يحمل القيمة أبدًا.	`1.10`
`DataEntity.getFieldsExcludedFromBuilders`	`models/data-entity/data-entity.model.ts`	– `encrypt || pendingEncryptUpgrade` التعريف الواحد الذي يعود إليه كل رفض تقريبًا.	`1.10`
`SensitiveDataAccessPolicyEnum` / `resolveSensitiveDataAccessDecision`	`packages/osos-shared/src/sensitive-data-access-policy.ts`	السياسات الأربع، والافتراضية `Privileged` ، والمُحلِّل الذي يفشل مُغلَقًا.	`1.10`
`SensitiveFieldsService.canQuerySensitiveField`	`services/sensitive-fields.service.ts`	قدرة الاستعلام = منطقة الإدارة **أو** المنحة؛ أضعف عمدًا من قاعدة القراءة.	`1.10`
`StepUpPurpose` (`WorkflowAction` / `Decrypt` / `Unmask`)	`packages/osos-shared/src/step-up-proof.ts`	يُميِّز الأغراض الثلاثة للتحقق الإضافي؛ يُحدَّد عند التحدّي ولا يمكن تبادله أبدًا.	`1.10`
`mintDecryptProof` / `mintWorkflowActionProof(s)` / `auditDecryptStepUp` / `auditWorkflowActionStepUp`	`users-service/src/services/step-up-unmask.service.ts`	إصدار الإثبات لمرة استخدام واحدة + تسجيله على سجلّ تدقيق MFA لكل غرض.	`1.10`
`validateStepUpBatch` / `STEP_UP_PROOF_MAX_BATCH` / `stepUpProofRedisKey` / `STEP_UP_PROOF_TTL_SECONDS` / `stepUpProofMatchesBinding`	`packages/osos-shared/src/step-up-proof.ts`	عقد الإثبات لمرة استخدام واحدة بين الخدمات؛ مدة صلاحية 120 ثانية، وحدّ دفعة 100 عنصر.	`1.10`
`StepUpProofService.consume` / `consumeDecrypt`	`data-service/src/services/step-up-proof.service.ts`	استهلاك ذرّي (`GETDEL`) للإثبات + مقارنة الارتباط.	`1.10`
`DecryptNotSupportedForPasswordFieldException`	`data-service/src/exceptions/`	يرفض Decrypt لحقل مُعرَّض ككلمة مرور قبل جلب المدخل.	`1.10`
`VaultEncryptionProvider` / `AzureKeyVaultProvider`	`data-service/src/encryption-providers/`	مزوِّدا HashiCorp Vault Transit / Azure Key Vault، بوضعَي `wrap` و `direct` .	`1.10`

صفحات ذات صلة

- حماية حقل [بناء التطبيقات] – دليل بُناة التطبيقات لوسم الحقول بأنها حسّاسة أو مشفَّرة.
- الأذونات الحساسة [الإدارة] – الإجراء الموجّه للمسؤول لمنح هذه الأذونات وتعديلها وإلغائها.
- الأذونات والتحكم في الوصول [الإدارة › الأذونات والوصول] – سلّم الصلاحيات على مستوى الكائن، الذي يعمل هذا البُعد إلى جانبه.
- مرجع نموذج الكيان وواجهته البرمجية – تعريفات الكيانات والحقول.
- مرجع نموذج سير العمل وواجهة API – غرض `WorkflowAction` للتحقق الإضافي، والتجميع الدفعي، وصيغة الرفض.
- مرجع: التعدادات – تعريف تعداد `StepUpPurpose` .
- المراقبة: السجلات والتدقيق [الإدارة] – ما يظهر للمسؤول بشأن الكشف عبر Decrypt وفحص الهوية المرتبط به باستخدام التحقق الإضافي.
- مفاتيح تشفير الحقول ومزوّدوها ومرجع الإعدادات – مفاتيح التشفير ومتغيّرات البيئة.
- أمان الحساب وكشف البيانات الحسّاسة [استخدام أسس › أمان الحساب وكشف القيم المحمية] – الوصف الموجَّه للمستخدم النهائي لمعالج الكشف ومطالبة Decrypt.

مرجع نموذج الوحدات وواجهتها البرمجية

هذه الصفحة موجّهة إلى **المُكامِلين ومهندسي المنصة** الذين ينشئون الوحدات أو يقرؤونها أو يحذفونها عبر الواجهة البرمجية، أو يبنون آليات التنقّل بالاعتماد على نقطة نهاية الشريط الجانبي، أو يحتاجون إلى معرفة قواعد التفويض والحذف بدقة.

> **هل تدير الوحدات من الواجهة؟**
>
> إن كنت من بُناة التطبيقات، فابدأ بصفحة الوحدات [بناء التطبيقات].

توجد جميع واجهات الوحدات البرمجية في **data-service**، وتُتاح تحت المسار `/v1/data-module` (إضافةً إلى الشريط الجانبي تحت `/v2/side-bar`). وتتطلّب كل نقطة نهاية المصادقة عبر استراتيجية Auth2 في أسس.

نظرة عامة

الوحدة كائن تجميع خفيف؛ فهي **لا** تضمّ كياناتها داخلها، بل يحمل كل `DataEntity` مرجع `module` يشير إلى الوحدة التي تحتويه. ولذلك يُمنع حذف الوحدة ما دام أي كيان يشير إليها.

وللوحدات نوعان:

- **الوحدات العادية** – يُنشئها المطوّرون، وهي قابلة للتحرير والحذف بالكامل. وتجمع كلٌّ منها وظائف مترابطة في وحدة متكاملة (كوحدة مخصّصة لإدارة المهام، أو أخرى للموارد البشرية).
- **الوحدات الضمنية** (`isImplicit: true`) – حاويات يديرها النظام، ولا يمكن تحريرها أو حذفها عبر الواجهة البرمجية. وتُعامَل الوحدة الأساسية (وحدة النظام، `00000000-0000-0000-0000-000000000000`) بوصفها وحدةً مقيَّدة، وتُستبعد من القوائم العادية ما لم يُطلب إدراجها صراحةً.

عند إنشاء وحدة جديدة، تزوّدها أسس تلقائيًا بـ**لوحة معلومات** افتراضية و**صفحة هبوط** افتراضية. وتحمل الوحدة كذلك **مستوى مشاركة** (خاص/عام/مشترك) و**ترتيبًا** يحدّد موضعها في الشريط الجانبي.

الاسم الموحّد

يخضع الاسم الموحّد لفهرس فريد، ويُتحقَّق من مطابقته لنمط محدّد. وترسل البوابة الاسم الموحّد المخزَّن مع كل تعديل، فلا يمكن تغييره عبرها بعد الإنشاء؛ غير أن `PATCH /v1/data-module/{id}` يحفظ قيمة جديدة للحقل `normalizedName` (انظر التحديث بالمعرّف)، وقد يؤدّي ذلك إلى تعارض بسبب تكرار الاسم، كما يُعطِّل كل ما يرتبط بالاسم القديم – فلا تغيّره إلا عن قصد.

الاعتماديات بين الوحدات

الوحدة كائن مستقل قائم بذاته، ينفّذ وظائفه دون أن يعتمد مباشرةً على وحدات أخرى. وعندما يلزم التفاعل بين الوحدات، تتواصل عبر **كيان وهمي (وكيل)** بدلًا من الاعتماد المباشر.

الوحدة الأساسية

الوحدة الأساسية وحدة افتراضية يُنشئها النظام، وتضمّ الكيانات الأساسية التي تحتاجها أسس لتهيئة بيئتها. وتُنشأ تلقائيًا، ولا يستطيع المستخدمون تعديل معلوماتها أو إنشاء كيانات فيها كما يفعلون في الوحدات الأخرى. وتُعرَف داخليًا بمعرّف الوحدة

المقيَّدة `00000000-0000-0000-0000-000000000000`، وتُستبعد من القوائم افتراضيًا، ويحميها مسار تفويض الوحدات المقيَّدة.

نموذج البيانات

يوسّع `DataModule` الصنف `Identified<UUID>` (فيكتسب `id` من نوع string)، وينفّذ `IName` و `ISpecificationVersion<1>`.

الحقل	النوع	إلزامي	ملاحظات
`id`	`UUID` (string)	مُسنَد	UUID v4؛ يولّده الخادم عند الإنشاء إن لم يُرسَل.
`specificationVersion`	`1`	نعم	قيمة حرفية ثابتة `1`، وقيمتها الافتراضية `1`.
`normalizedName`	`string`	نعم	اسم آلي؛ **فهرس فريد**. يُتحقَّق منه وفق `NormalizedNameValidation` (الحدّان الأدنى والأقصى للطول + النمط).
`name`	`LocalizedValue`	نعم	اسم العرض المترجَم (سلسلة لكل لغة).
`description`	`NullableLocalizedValue`	لا	مترجَم، ويقبل null.
`icon`	`Icon`	نعم	كائن يحتوي على سلسلة `name` إلزامية غير فارغة.
`isImplicit`	`boolean`	نعم	يدلّ على أن النظام هو الذي يدير الوحدة.
`creationInfo`	`ActionInfo`	لا*	يختمه الخادم عند الإنشاء (مَن نفّذ الإجراء ومتى، مع المفوَّض إليه اختياريًا).
`updateInfo`	`ActionInfo`	لا*	يعيد الخادم ختمه عند كل تحديث.
`sharing`	`ApplicationSharing`	نعم	يتحكّم في الظهور عبر المساحات (انظر المشاركة).

* الحقلان `creationInfo` و `updateInfo` معلَنان بالقيمة `required: false` في النموذج، لكن الخدمة تملؤهما دائمًا؛ ويُستبعدان من أجسام الطلبات.

الأيقونة

`Icon` كائن مضمَّن صغير (`idInjection: false`)، ويقبل null على مستوى المخطط)، وله حقل إلزامي واحد هو `name:` `string`. وإضافةً إلى ذلك، يرفض المتحكّم عند الكتابة اسم الأيقونة إذا كان فارغًا أو مؤلَّفًا من مسافات فقط.

ختم الإجراءات

عند الإنشاء، تبني الخدمة كائن `ActionInfo` يحتوي على `doneAt` (وقت الخادم)، و `doneBy` (مرجع `ReferenceProperty` إلى الكيان `USER` بمعرّف المستخدم الذي ينفّذ الإجراء)، ومرجع `delegatee` اختياري حين يُقدَّم الطلب نيابةً عن مستخدم آخر (`user.delegatee.dataEntry`). ويُضبط كلٌّ من `creationInfo` و `updateInfo` على هذه القيمة عند الإنشاء؛ ثم يعيد `updateModule` كتابة `updateInfo` بختم جديد (بلا مفوَّض إليه) عند كل تحديث.

المشاركة

`sharing` كائن من النوع `ApplicationSharing`:

الحقل	النوع	ملاحظات
`level`	`CategoryComponentEntryValue<ApplicationSharingLevelEnum>`	مستوى المشاركة.
`spaces`	`ReferenceProperty[]`	المساحات التي تُشارَك معها الوحدة (تُستخدم مع `SHARED / PRIVATE`).

`ApplicationSharingLevelEnum` (من `@osos/shared`):

القيمة	المعنى
`PUBLIC`	ظاهرة لكل المساحات.
`PRIVATE`	ظاهرة للمساحات المُدرجة في `spaces` وحدها.
`SHARED`	ظاهرة للمساحة المالكة وللمساحات المُدرجة في `spaces`.

تحدّد `DataModuleService.getSpaceModules(spaceId)` الوحدات الظاهرة لمساحة ما وفق هذه القواعد بالضبط: أي وحدة `PUBLIC`، إضافةً إلى وحدات `PRIVATE` أو `SHARED` التي يتضمّن فيها `sharing.spaces.dataEntry` معرّف المساحة.

Long-standing – المشاركة على مستوى الوحدة (مشاركة الوحدات والكيانات) تسبق سلسلة الإصدارات 1.2، وهي من الخصائص الأساسية في كل الإصدارات المدعومة؛ ولا تُعرض لها شارة توفّر.

كائنات DTO

`DataModuleCreateDto`

يوسّع `DataModule` ويضيف حقلَين مساعدَين للكتابة فقط، يستخدمهما طلب الإنشاء:

الحقل	النوع	ملاحظات
`entities`	`UUID[]`	اختياري، وقيمته الافتراضية `[]`.
`permissions`	`Permission[]`	اختياري. يُزال من مستند الوحدة ويُخزَّن منفصلًا بوصفه حقوق وصول (انظر أدناه).

`LiteModuleWithViews` / `LiteModule`

إسقاطات مختصرة تستخدمها حمولات التنقّل: `id`، و`specificationVersion`، و`name`، و`icon` – ويتضمّن `LiteModuleWithViews` أيضًا مصفوفة من `LiteDataViews`. ويتألّف `SidebarDto` (الإصدار v1) من هذه الإسقاطات، إضافةً إلى معارض النظام وكتلة حالة الخدمات.

`ExportingDataModulesDto` / `ExportingDataModuleDto`

كائنات DTO خاصة بالتحزيم، تُستخدم عند تصدير وحدة لمشاركتها عبر المساحات: تُختزل الوحدة إلى معرّفها `id` مع مصفوفة من `ExportingEntityDto` (`dataEntities`)؛ ويغلّف `ExportingDataModulesDto` قائمة منها. ويُحمى مسار التصدير بالاستثناء `ForbiddenToExportModule`.

واجهة REST

المسار الأساسي: `/v1/data-module`. تتطلّب جميع المسارات المصادقة، وتُذكر الأدوار والمصوِّتات الخاصة بكل مسار.

الطريقة والمسار	الغرض	التفويض
`POST /v1/data-module`	إنشاء وحدة	الدور `SystemDeveloper`
`GET /v1/data-module/count`	عدّ الوحدات	الدور `SystemViewer` أو `Restricted`
`GET /v1/data-module`	سرد الوحدات	الدور `SystemViewer` أو `Restricted`
`GET /v1/data-module/{id}`	جلب وحدة واحدة	`moduleVoterFactory(0)`
`PATCH /v1/data-module/{id}`	تحديث وحدة	`moduleVoterFactory(0)`
`DELETE /v1/data-module/{id}`	حذف وحدة	`moduleVoterFactory(0)`

الطريقة والمسار	الغرض	التفويض
`DELETE /v1/data-module/{id}/permanent`	حذف وحدة وكل ما فيها (`Since 1.10`، مهمة في الخلفية)	الدور `SystemAdmin` (ويجب أن تشمل الرخصة الإدارة)

الإنشاء (`POST /v1/data-module`)

جسم الطلب هو `DataModuleCreateDto` مع استبعاد `id` و`creationInfo` و`updateInfo`، إذ يتولّى الخادم إسنادها.

التسلسل (`create` ← `moduleChecker` ← `DataModuleService.createModule`):

1. **التحقق** (`moduleChecker`)، بالترتيب:
   - `languagesService.validateName(name)` – قد يرمي `NameEnIsRequired` أو `NameEnIsShort` أو `NameEnIsLong`.
   - `checkIcon` – يرمي `DataModuleIconIsRequired` إن كان `icon.name` فارغًا.
   - `checkDuplicateNormalizedName` – يرمي `ModuleDuplicateException` إن كانت وحدة أخرى تستخدم `normalizedName` نفسه.
   - `checkDuplicateNamesFromDatabase` – يرمي `ModelWithGivenNameExistException` إن تعارض اسم مترجَم.
2. يُزال `permissions` من كائن DTO ويُمرَّر منفصلًا (ولا يُخزَّن في مستند الوحدة).
3. تُسند الخدمة `id` (UUID v4 إن غاب)، وتضبط `specificationVersion: 1`، وتضع ختمَي `updateInfo` / `creationInfo`، وتحفظ المستند عبر `DataModuleRepository`.
4. تنشر `DataModuleCreatedEvent` على ناقل الرسائل.
5. إن قُدِّمت `permissions`، تُخزَّن بوصفها حقوق وصول (`EntitlementEnum.Module`، بمفتاح معرّف الوحدة الجديدة). ويُسجَّل أي إخفاق هنا ثم يُعاد رميه.

وتعيد `DataModule` المُنشأة.

```
POST /v1/data-module
Content-Type: application/json

{
  "normalizedName": "hr_module",
  "name": { "en": "Human Resources", "ar": "الموارد البشرية" },
  "description": { "en": "HR entities and views" },
  "icon": { "name": "users" },
  "isImplicit": false,
  "sharing": {
    "level": { "id": "PUBLIC", "name": { "en": "Public" } },
    "spaces": []
  },
  "permissions": []
}
```

العدّ (`GET /v1/data-module/count`)

معاملات الاستعلام:

- `where` – مرشِّح `where` القياسي في LoopBack على `DataModule`.

- `includeCoreModule` (boolean) – إذا كانت قيمته false أو ما يعادلها (وهو الافتراضي)، يُستبعد معرّف الوحدة الأساسية `00000000-0000-0000-0000-000000000000` عبر `neq('id', ...)`.

ويمرّ العدّ عبر `countWithAccessRight`، الذي يحصر النتيجة فيما يستطيع المستدعي رؤيته فعلًا (انظر حصر حقوق الوصول).

السرد (`GET /v1/data-module`)

معاملات الاستعلام:

- `filter` – مرشِّح LoopBack القياسي على `DataModule`.
- `includeCoreModule` (boolean) – يُطبَّق استبعاد الوحدة الأساسية نفسه المتّبع في `count`.

تأتي النتائج من `findWithAccessRightInSpace`، وتُرتَّب وفق قواعد المقارنة الإنجليزية (`{collation: {locale: 'en'}}`). وكما في العدّ، تُحصر النتائج وفق حقوق وصول المستدعي.

الجلب بالمعرّف (`GET /v1/data-module/{id}`)

يطبّق `getModule` مرشِّح مستوى المشاركة الخاص بالمستدعي (`spaceService.getFilterBySharingLevel`) إضافةً إلى شرط مطابقة `id`، ويقرأ عبر `findOneInSpace`. وإن لم توجد نتيجة مطابقة، يرمي `EntityNotFoundError` (404). ويُتجاهل الجزء `where` من أي `filter` يُمرَّر.

التحديث بالمعرّف (`PATCH /v1/data-module/{id}`)

جسم طلب جزئي (`DataModule`، `partial: true`) مع استبعاد `id` و `creationInfo` و `updateInfo`. التسلسل:

1. `getModule(id)` – 404 إن لم تكن الوحدة ظاهرة.
2. `moduleChecker(..., id)` – التحقق نفسه المتّبع في الإنشاء، لكن فحوص التكرار تتجاهل معرّف الوحدة نفسها (`neq: selfId`).
3. يكتب `DataModuleService.updateModule` الحقول `name` و `description` و `normalizedName` و `icon` و `sharing` وقيمة جديدة لـ `updateInfo`، ثم يعيد قراءة المستند.
4. ينشر `DataModuleUpdatedEvent` حاملًا فرق `changes`. ويحسب `identifyChanges` هذا الفرق، ولا يتتبّع حاليًا **إلا** تغييرات `sharing` (القيم القديمة والجديدة).

ويعيد `DataModule` المحدَّثة.

> تحفظ الخدمة `normalizedName` عند التحديث، لكنه يظلّ المفتاح الثابت للوحدة. وتعرضه البوابة بوصفه قيمة ثابتة بعد الإنشاء، وقد تؤدّي إعادة تسميته على مستوى الواجهة البرمجية إلى تعارض بسبب التكرار؛ فلا تغيّره إلا عن قصد.

الحذف بالمعرّف (`DELETE /v1/data-module/{id}`)

الضوابط بالترتيب:

1. `getModule(id)` – 404 إن لم تكن الوحدة ظاهرة.
2. إن كان `isImplicit`، يرمي `ImplicitModuleException`؛ فلا يمكن تعديل الوحدات الضمنية أو حذفها.
3. إن كان أي `DataEntity` لا يزال يشير إلى الوحدة (`findInSpace` حيث `module == id`)، يرمي `ModuleHasEntities`؛ فانقل كياناتها أو احذفها أولًا.
4. يحذف `DataModuleService.deleteModule` المستند وينشر `DataModuleDeletedEvent`.

ويعيد `204` (بلا محتوى). ولحذف وحدة لا تزال تحتوي على محتوى، استخدم الحذف النهائي أدناه.

الحذف النهائي (`DELETE /v1/data-module/{id}/permanent`) `Since 1.10`

لا يحذف المسار السابق إلا وحدة **فارغة**. أما الحذف النهائي فيحذف الوحدة **وكل ما فيها** ضمن مهمة تعمل في الخلفية. ويقتصر على مسؤول النظام (`allowedRoles: [SystemAdmin]` ؛ ويجب أن تشمل الرخصة الإدارة).

الضوابط، بحسب ترتيبها في الرماز (`ModulePurgeService.assertPurgeable`):

الفحص	الرفض
الوحدة ظاهرة للمستدعي	`404`
ليست الوحدة الأساسية وليست ضمنية	`400 MODULE_PURGE_NOT_ALLOWED`
لا تجري عملية حذف أخرى (أما التشغيل الفاشل أو المتوقّف فيُستأنف بدلًا من ذلك)	`409 MODULE_BEING_DELETED`
كل كيان فيها قابل للحذف عبر الحذف النهائي للكيان	`400 MODULE_PURGE_NOT_ALLOWED` مع `details.entityIds`
لم تُثبَّت الوحدة، ولا أي كيان أو صفحة فيها، من حزمة	`403`
لا يوجد فيها كيان مقفل لدى مستخدم آخر	`409 LOCKED_OBJECT`
لا شيء **خارج** الوحدة لا يزال يشير إليها أو إلى أي شيء فيها (`GET` `/v1/external-dependents/DataModule/{id}` في dependency-service)	`409 MODULE_HAS_DEPENDENTS` (`details.count`)، أو `503` `DEPENDENCY_CHECK_UNAVAILABLE` حين يتعذّر إتمام الفحص

ويجسّد الفحص الأخير قاعدة «افصل أولًا»: إذ تسرد نافذة البوابة تلك المراجع، مع رابط **انتقال** لكلٍّ منها، ولا تفعّل الحذف إلا حين لا يبقى منها شيء.

الاستجابة. يستجيب المسار **قبل** إنجاز العمل بـ `200 {success: true, result: null}` (ويصرّح مستند OpenAPI بـ `202`). فهو يَسِم الوحدة بـ `purgeInfo` و `purgeStatus: {step: 'queued'}` عبر كتابة شرطية (`409 UPDATE_CONFLICT` إن سبقه طلب آخر إلى وسمها)، ويؤرشف رابط الوحدة في الشريط الجانبي والروابط التي تحته، ويرسل `PurgeDataModuleCommand` وينشر `DataModulePurgeStateChangedEvent` (`deleting`). ولا ينشر `DataModuleUpdatedEvent` .

المهمة. تستحوذ `ModulePurgeService.run` على الوحدة بمعرّف تشغيل جديد، وتنفّذ خطوات متساوية الأثر عند التكرار، وتسجّل كلًّا منها في `purgeStatus.step` :



- تحذف `entities` جميع كيانات الوحدة (بما فيها المؤرشفة والموجودة في أي مساحة) عبر الحذف النهائي للكيان، مع عدّ تنازلي في `entitiesLeft` .
- تؤرشف `functions` وظائف الوحدة، فتُلغي خدمة الوظائف نشرها؛ وتزيل `grants` منح الوحدة وحقوق الحقول الحسّاسة المسجّلة باسم الوحدة، ثم تحدّث حقب الأذونات والملفات الشخصية.
- تعدّ `verify` ما تبقّى، وتفشل بـ `MODULE_PURGE_INCOMPLETE` إن أُضيف شيء في الأثناء؛ وتنشر `event` الحدث `DataModuleDeletedEvent` حاملًا المعرّفات المحذوفة؛ وتحذف `module` المستند.
- تُسجَّل الخطوة الفاشلة في `purgeStatus.error {step, code, message}` وتُعلَن الحالة `failed` . **وتبقى الوحدة موسومة، واستدعاء الحذف مجددًا يستأنفه** (ويستأنف الحذفُ كذلك تشغيلًا لم تتغيّر حالته منذ 10 دقائق). ولا تتوفّر إمكانية الإلغاء.

ما دامت الوحدة قيد الحذف يرفض data-service تحريرها، وإنشاء كيان أو كيان وهمي فيها، ونقل كيان إليها، وإنشاء صفحة فيها (`409 MODULE_BEING_DELETED`)؛ ولا تُقبل `purgeInfo` و `purgeStatus` في أجسام طلبات الإنشاء والتحديث. وتستطلع

البوابة `GET /v1/data-module/{id}` كل 3 ثوانٍ (و `404` يعني انتهاء الحذف)، وتدفع خدمة web-socket الرسالة `modules/delete-state {moduleId, state}` إلى كل جلسة مفتوحة للقاطن، فتعرض صفحات الوحدات **جارٍ الحذف…** أو **فشل الحذف** فور تغيّر الحالة.

> **قيد معروف:** يعمل الحذف داخل معالج لناقل الرسائل، ويستهلك data-service افتراضيًا رسالة واحدة من الناقل في كل مرة، لذا تؤخّر عملية الحذف الطويلة بقية عمل الناقل في تلك الحاوية (pod).

ترتيب قائمتَي الوحدات والكيانات `Since 1.10`

تتضمّن صفحة **الوحدات** الإدارية، وصفحة **كيانات النظام**، وتبويب **الكيانات** في كل وحدة، عنصر التحكّم **الترتيب** متعدد المستويات نفسه الموجود في تبويب الاستعلامات في الكيان. وحقول الترتيب هي **الأحدث إنشاءً أولاً** و**الأحدث تعديلاً أولاً** و**الاسم أ › ي**؛ والترتيب الافتراضي هو الأحدث إنشاءً أولًا، ويُستعاد عبر **إعادة تعيين**.

وتُقسَّم القائمتان إلى صفحات على الخادم، لذا يُرسَل الترتيب المختار إلى مسارات السرد الحالية بوصفه `order` في LoopBack مع `id` لفضّ التساوي؛ فلا توجد نقطة نهاية جديدة. وتحفظ كل قائمة ترتيبها الخاص (صفحة الوحدات، وصفحة كيانات النظام، وتبويب الكيانات في كل وحدة، كلٌّ على حدة) طوال جلسة المتصفح؛ فيبقى بعد التحديث، ويُمسح عند تسجيل الخروج، ويُعاد ضبطه حين تنتقل إلى القائمة. ويُبقي تغيير الترتيب المرشِّح المطبَّق ويعيدك إلى الصفحة 1.

التفويض

الأدوار

يتطلّب `create` الدور `SystemDeveloper`. ويتطلّب `find` / `count` الدور `SystemViewer` أو `Restricted`. أما المسارات على مستوى العنصر (`findById` و `updateById` و `deleteById`) فتستخدم مصوِّت الوحدة بدلًا من دور ثابت.

مصوِّت الوحدة

يبني `moduleVoterFactory(moduleParamIndex = 0, bodyParamKey?, ignoreRestricted?)` مُفوِّضًا لمسار ما:

- يُسمح بالاستدعاءات الداخلية بين الخدمات مباشرةً؛ وإذا وُجد `view` في السياق أصلًا، يمتنع المصوِّت عن التصويت.
- يقرأ معرّف الوحدة من المعامل الموجود عند `moduleParamIndex` (أو من `bodyParamKey` داخل جسم كائن). وإن كان المعرّف غائبًا أو null، يمتنع ويترك القرار للمزوّد الافتراضي.
- إن كان المعرّف **وحدة مقيَّدة** (الوحدة الأساسية)، تمرّ القرارات عبر `canViewRestrictedModule` / `canEditRestrictedModule` (ما لم يُضبط `ignoreRestricted`).
- وفيما عدا ذلك، يُمنح **العرض** عبر `canViewModuleIncludingEntityGrants(moduleId)` و**التحرير** عبر `canEditModule(moduleId)`. ونطاق وصول القراءة أوسع عمدًا من نطاق وصول التحرير: فالمستخدم الذي لا يملك إلا منحة على *كيانٍ داخل* الوحدة يستطيع رؤية الوحدة بوصفها حاوية تنقّل، أما التحرير فيظل يتطلّب منحة فعلية على الوحدة.

Since 1.8 – أُتيح أول مرة في خط 1.8 فتح الوحدة الأم استنادًا إلى منحة على كيان وحده (`canViewModuleIncludingEntityGrants` وتمرير `entityParentModuleIds`).

حصر حقوق الوصول

يفوّض مسارا السرد والعدّ كلاهما العمل إلى `DefaultCrudWithSharingAndAccessRightRepository` (الذي لا يستخدمه إلا مستودعا الوحدات والكيانات)، فيركّب **مرشّحين مستقلّين أحدهما عن الآخر** في `where` الخاص بـ Mongo:

1. **مرشّح المساحة والمشاركة** (`getWhereSharing`) – لا يظهر الصف إلا إذا كان `sharing.level == Public` ، **أو** كان مستواه `Shared` / `Private` **و** تضمّن `sharing.spaces.dataEntry` معرّف `spaceId` الحالي. ويقابل هذا الفحصُ فحصَ `getFilterBySharingLevel` على مستوى العنصر في مسار الجلب بالمعرّف، لكنه يُطبَّق عند السرد والعدّ.

2. **مرشّح نطاق الوصول** – يستدعي المستودع `accessRightsService.getAllModuleIdsUserCanView()` ويتفرّع بحسب النتيجة:

- `AdminAccess` ← بلا ترشيح (المساحة كاملة، مرتَّبة `_id DESC` في السرد).
- `EditAccess` / `DeveloperAccess` / `ViewAccess` ← كل شيء **عدا** الوحدات المقيَّدة (`nin('id',` `restrictedModuleIds)`).
- نتيجة محصورة بالموارد `{moduleIds, entityIds, entityParentModuleIds}` ← تُحصر الوحدات في `moduleIds` ∪ `entityParentModuleIds` (الوحدات الممنوحة، إضافةً إلى الوحدات الأم للكيانات الممنوحة، كي تظهر هذه الوحدات الأم بوصفها حاويات تنقّل؛ وتبقى الكيانات الشقيقة مخفية). ويُفرض شرط `where` الخاص بهذا النطاق على المرشِّح، فلا يتلقّى المستخدمون المحصورون بالموارد (مثل `Restricted`) إلا ما مُنحوه.

أحداث دورة الحياة

تنشر الخدمة حدثًا على ناقل الرسائل (`@node-ts/bus-core`) عند كل تعديل، مع ضبط `correlationId` على معرّف تتبّع الطلب، وإرفاق سمات لاصقة تحمل `userId` و `spaceId` :

الحدث	ينشره	الحمولة
`DataModuleCreatedEvent`	`createModule`	الوحدة المُنشأة + معرّف المستخدم الفاعل.
`DataModuleUpdatedEvent`	`updateModule`	الوحدة المحدَّثة + معرّف المستخدم + `changes` (فرق المشاركة).
`DataModuleDeletedEvent`	`deleteModule`	الوحدة المحذوفة + معرّف المستخدم.

وتُستهلك هذه الأحداث في مواضع أخرى لمزامنة **روابط** التنقّل (إذ يُحتفظ بكيان رابط لكل وحدة وصفحة ومعرض وتقرير).

التنقّل والشريط الجانبي

الوحدات هي العمود الفقري لتنقّل البوابة. وتعيد نقطة نهاية الشريط الجانبي الروابط التي يستطيع المستخدم رؤيتها، مجمَّعةً بحيث تكون كل وحدة حاوية لمعارضها وصفحاتها.

`GET /v2/side-bar`

`SideBarController.getViewsForSidebarV2` (يُتخطّى التفويض على مستوى المسار، ويجري الحصر داخل الخدمات).

المُدخلات:

- الترويسة `x-domain-id` ، والترويسة `x-space-id` .
- معامل الاستعلام `include` (`SidebarIncludeDataEnum` ، الافتراضي `All`):

القيمة	الاسم	يعيد
`0`	`All`	`links` + `serviceViewsStatus`.
`1`	`Links`	`links` فقط (`serviceViewsStatus: null`).
`2`	`OnlyService`	`serviceViewsStatus` فقط (`links: []`).

والاستجابة من النوع `SidebarDtoV2`:

الحقل	النوع	ملاحظات
`links`	`GenericLink[]`	روابط التنقّل (الوحدات وما تحتها).
`serviceViewsStatus`	`SidebarServiceStatus | null`	حالة دليل الخدمات والطلبات وصندوق الوارد، مع العدّادات.
`isRestricted`	`boolean`	هل يحمل المستدعي الحدّ الأدنى `Restricted` على مستوى النظام.
`restrictedAccess`	`{hasModuleAccess, hasEntityAccess}`	لا معنى له إلا حين يكون `isRestricted`؛ ويحدّد سلوك الواجهة المقيَّدة.

يُحسب `restrictedAccess` من أذونات `module` / `entity` للمستخدم، وتكون قيمة كل علَم صحيحة متى كانت منحة واحدة على الأقل `>= ViewAccess`.

Since 1.4 – أُتيحت خيارات `include` في الشريط الجانبي v2 ونموذج حالة الخدمات أول مرة في خط 1.4. أما بنية `SidebarDto` / `LiteModuleWithViews` في v1 وآلية الروابط فتسبقان خط 1.2، وتشكّلان الأساس.

`SidebarServiceStatus`

القيم المنطقية `hasServiceCatalog` و`hasRequests` و`hasInbox` و`hasObserverRequests`، والعدّادات `onGoingRequests` و`onGoingInbox` و`onGoingObserverRequests`.

اكتشاف المستخدم النهائي للمعارض `Since 1.8`

`GET /v1/views/end-user/modules`

تسدّ `DataViewsController.findModulesForEndUser` فجوة فعلية: فكل عملية سرد أخرى للوحدات أو المعارض تجيب عن سؤال **تأليفي**، وتخضع لقيود تتوافق معه: يتطلّب `GET /v1/data-module` (أعلاه) الدور `SystemViewer` أو `Restricted`، وتتطلّب نقطة نهاية المعارض بحسب الوحدة في data-service منحة على الوحدة أو الكيان، ومع ذلك تعيد **كل معرض منشور** في تلك الوحدة، سواء أكان المستدعي قادرًا على فتحه أم لا. أما المستخدم الذي لا يملك إلا منحًا على **المعارض** – دون أي منحة على وحدة أو كيان – فلم يكن يستطيع اكتشاف معارضه عبر أيٍّ من نقاط النهاية هذه، مما جعل منتقي المعارض المنظَّم بحسب الوحدات حكرًا على المديرين عمليًا.

أما نقطة النهاية هذه فتجيب عن سؤال المستخدم النهائي – أي المعارض أستطيع فتحها فعلًا، وفي أي وحدة يوجد كلٌّ منها – وتعيد `LiteModuleWithViews[]` (انظر `LiteModule` / `LiteModuleWithViews` أعلاه): الوحدات التي تحتوي على معرض واحد على الأقل يستطيع المستدعي فتحه، مع إرفاق تلك المعارض بكل وحدة. وهي أيضًا **أول** نقطة نهاية تتيح هذا الشكل فعليًا عبر HTTP – إذ يضمّه `SidebarDto` في v1 أيضًا، لكن لم يعد أي مسار للشريط الجانبي v1 مُركَّبًا؛ ولا يعمل إلا `GET /v2/side-bar` (انظر أعلاه).

وقد ضُبط التفويض هنا على `@authorize({skip: true})` عمدًا: فالترشيح داخل `DataViewService.getModulesWithAccessibleViews` *هو* التفويض، وهو يمنع الوصول عند الفشل – فلا يحصل المستدعي المجهول إلا على المعارض التي فُعِّل فيها الوصول المجهول. ويُجري الترشيح فحص إتاحة الوصول للمستخدم النهائي نفسه الذي تستخدمه نقطة نهاية المعرض الواحد (المجهول، ثم المصادَق عليه، ثم منحة معرض صريحة أو مطوّر نظام فأعلى

للمعرض المقيَّد)، مع اختلاف واحد مقصود: يُستبعد هنا المعرض الذي لم يُفعَّل فيه **أيٌّ** من وضعَي الوصول، بدلًا من أن يتخطّى الفحص ويُعاد رغم ذلك، لأن إدراج معرض لا يستطيع أي مسار مهيّأ فتحه سيكون مضلِّلًا في المنتقي.

وتقتصر المعارض المرشّحة على المعارض **المنشورة غير الضمنية** في مساحة الطلب، وتُجمَّع تحت الوحدة المستخلصة من **كيان** المعرض (لا من لقطة `module` الخاصة بالمعرض، التي قد تكون قديمة)، وتُرتَّب بالاسم لضمان ثبات ترتيبها في المنتقي.

الاستثناءات

الاستثناء	رمز الخطأ	متى
`DataModuleIconIsRequired`	`DATA_MODULE_ICON_REQUIRED`	اسم الأيقونة فارغ عند الإنشاء أو التحديث.
`ModuleDuplicateException`	`MODULE_DUPLICATE`	`normalizedName` مستخدَم بالفعل.
`ModuleHasEntities`	`MODULE_HAS_ENTITIES`	محاولة حذف الوحدة مع وجود كيانات تشير إليها.
`ImplicitModuleException`	`MODULE_IS_IMPLICIT`	محاولة تعديل وحدة ضمنية أو حذفها.
`InvalidModuleId`	`INVALID_MODULE_ID`	معرّف الوحدة المقدَّم غير صالح (404).
`EntitiesNotFound`	`ENTITY_NOT_FOUND`	كيان واحد أو أكثر من الكيانات المُشار إليها غير موجود.
`ForbiddenToExportModule`	`CANNOT_EXPORT_MODULE`	التصدير غير مسموح (403).
`ModulePurgeNotAllowedException`	`MODULE_PURGE_NOT_ALLOWED`	`Since 1.10` . حذف نهائي للوحدة الأساسية أو لوحدة ضمنية، أو لوحدة تحتوي على كيان يرفض الحذفُ النهائي للكيانات حذفَه (400، `details.entityIds`).
`ModuleHasDependentsException`	`MODULE_HAS_DEPENDENTS`	`Since 1.10` . محاولة حذف الوحدة نهائيًا مع وجود عناصر من خارجها تشير إليها (409، `details.count`).
`ModuleBeingDeletedException`	`MODULE_BEING_DELETED`	`Since 1.10` . تحرير وحدة يجري حذفها، أو الإنشاء فيها، أو النقل إليها، أو حذفها نهائيًا (409).
`DependencyCheckUnavailableException`	`DEPENDENCY_CHECK_UNAVAILABLE`	`Since 1.10` . تعذّر الحصول على نتيجة من فحص الاعتماديات (503، دون إعادة محاولة).

وتصدر أخطاء التحقق من الاسم (`NameEnIsRequired` و `NameEnIsShort` و `NameEnIsLong` و `ModelWithGivenNameExistException`) عن `LanguagesService` في أثناء `moduleChecker` .

ملاحظات عملية للبُناة

- **الإنشاء يتطلّب `SystemDeveloper` .** أما العرض والسرد فلا يحتاجان إلا إلى `Restricted` / `SystemViewer` ، لكن النتائج تظل دائمًا مقيَّدة بحقوق الوصول.
- **الاسم الموحّد هو مفتاحك الثابت** – فهو مفهرس بفهرس فريد ويُتحقَّق من مطابقته لنمط؛ لذا اختره بعناية، لأن إعادة التسمية قد تؤدي إلى تعارض بسبب التكرار.
- **أفرغ الوحدة قبل حذفها – أو احذفها نهائيًا.** يرفض مسار الحذف العادي حذف الوحدة غير الفارغة ويُرجع `ModuleHasEntities` ؛ و `Since 1.10` يستطيع مسؤول النظام بدلًا من ذلك حذفها نهائيًا مع كل ما فيها، بعد فصل كل مرجع إليها من خارجها.
- **لا تمسّ الوحدات الضمنية والوحدة الأساسية** – فمعرّف الوحدة الأساسية `00000000-0000-0000-0000-000000000000` يُستبعد من القوائم افتراضيًا ويحميه مسار مصوِّت الوحدات المقيَّدة؛ والوحدات الضمنية ترفض التحرير

والحذف مباشرةً.

- **فروق التحديث تقتصر على المشاركة** – فحمولة `changes` في `DataModuleUpdatedEvent` لا تعكس حاليًا إلا تغييرات `sharing`، مع أن الحقول الأخرى تُحفظ.

سجل السلوك

- كان التبديل بين الوحدات في أثناء تصفّح تبويب **الكيانات** في وحدة ما يُبقي مرشِّح الوحدة المعروضة سابقًا؛ أما الآن فتقتصر القائمة على الوحدة التي انتقلت إليها.

مرجع إعدادات الحقول

توجَّه هذه الصفحة إلى **مطوّري التكامل والمطوّرين** الذين يقرؤون تعريفات الكيانات أو المُدخلات أو يكتبونها عبر الـ API، أو يحتاجون إلى معرفة كيفية تخزين المنصة لإعدادات الحقل وفرضها بدقّة. وتغطّي نموذج التخزين الذي تقوم عليه أنواع الحقول، والصيغة المخزَّنة للإعدادات التي يعرضها الباني، والملاحظات التقنية الخاصة بكل نوع من الحقول.

> **تصمّم الحقول من الواجهة؟**
>
> ينبغي للبُناة البدء بـ اختر نوع الحقل [بناء التطبيقات] و الإعدادات المشتركة بين الحقول [بناء التطبيقات] – فهما يغطّيان الميزات نفسها بالتسميات التي يعرضها مصمِّم الكيان. أمّا أنواع الحقول التي لا يعرضها المصمِّم، فترد في أنواع الحقول الداخلية.

أنواع التخزين وأصناف الحقول

من المفيد التمييز بين مفهومين مختلفين:

- **نوع بيانات التخزين (`dataType`)** – تعداد صغير مغلق يصف كيفية *تخزين* القيمة *وتحويلها* (نص، رقم، تاريخ، منطقي، blob، مرجع كيان، JSON، وغيرها). وهو عقد التخزين الخاص بالحقل.
- **صنف الحقل** – الحقل الدلالي الذي يختاره الباني فعليًّا (نص عادي، نص طويل، كلمة مرور، هاتف، صورة، مرجع، تصنيف، محوِّل، خريطة، …). وتحدّد المنصة الصنف بفحص `dataType` **و**مكوّن التحرير المهيَّأ معًا (وأحيانًا خيارات إضافية).

وقد تشترك أصناف كثيرة في نوع تخزين واحد؛ فمثلًا، يشكّل `Text` نوع التخزين للنص العادي والنص الطويل وكلمة المرور والهاتف، ويتميّز كلٌّ منها بمكوّن التحرير الخاص به. ويوضّح التمييز بين الطبقتين سبب الاختلاف الكبير الذي قد يظهر في سلوك حقلين وطريقة عرضهما، مع أنهما يخزّنان القيمة الأساسية نفسها.

طبقة أدوار وسيطة

بين أنواع التخزين الأولية التي يقارب عددها خمسة عشر (`DataTypeEnum`) وأصناف حقول الواجهة التي يقارب عددها خمسة وثلاثين، تقع طبقة محدودة من **رموز الأدوار الدلالية** (مثل `MAIN_TITLE` و`SUB_TITLE` و`PICTURE` و`CATEGORY`) تَسِم الحقل بدور العرض الذي يؤدّيه، لا بطريقة تخزين قيمته أو تحريرها. وهذه الطبقة موسومة في الرماز بأنها مؤقتة، لذا تعامَل معها على أنها تفصيل تنفيذي قد يُدمج لاحقًا، واعتمد على نوع التخزين وصنف الحقل بوصفهما العقد الثابت.

أنواع بيانات التخزين

تتكوّن طبقة التخزين من تعداد مغلق لأنواع البيانات، يشمل النص، والرقم، والتاريخ، والوقت، والتاريخ والوقت، والمنطقي، وblob (الملفات والصور)، ومراجع الكيانات، والأيقونات، والكتل المتداخلة، وJSON متعدد اللغات، وGeoJSON (الخرائط)، وJSON. وأُضيف SVG المضمَّن (`SVGIcon`) **Since 1.8**. وعند الحفظ، تُحوَّل القيم المرسلة إلى نوع تخزينها باستخدام أداة تحويل خاصة بكل نوع؛ أمّا الأنواع التي لا تتوفّر لها أداة تحويل مسجَّلة، فتمرّ دون تغيير.

أصناف الحقول وأنواع تخزينها

فيما يلي الأصناف التي يستطيع الباني اختيارها في مصمِّم الكيان، مع التسمية التي يعرضها المصمِّم ونوع التخزين الذي يقوم عليه كلٌّ منها:

تسمية المصمِّم	نوع التخزين (dataType)	ملاحظات
نص	`Text`	يُعرض أيضًا بصيغة للعرض فقط
نص طويل	`Text`	صيغة للعرض فقط؛ لأنماط النص المنسَّق، انظر النص المنسَّق
نص متعدد اللغات	`LocalizedJson`	صيغة للعرض فقط
نص طويل متعدد اللغات	`LocalizedJson`	Markdown فقط؛ صيغة للعرض فقط
كلمة المرور	`Text`	يُخزَّن في صورة تجزئة أحادية الاتجاه؛ انظر كلمة المرور
هاتف	`Text`	انظر صيغ الهاتف
رقم	`Number`	
حقل متسلسل	`Number`	تزايد تلقائي؛ انظر التسلسل
سنة	`Number`	
تاريخ	`Date`	
شهر	`Date`	يُحدَّد الحدّان الأدنى والأقصى بأرقام الأشهر؛ انظر الشهر
سنة وشهر	`Date`	
وقت	`Time`	
تاريخ ووقت	`DateTime`	
منطقي	`Boolean`	
محوِّل	`Boolean`	
تصنيف	`Json`	انظر التصنيف
ملف، صورة، مستند، فيديو	`blob`	انظر قيود الملفات
أيقونة	`Icon`	
خريطة	`GeoJson`	انظر الخريطة
مرجع	`Entity`	انظر مصادر بيانات المرجع
مستخدم	`Entity`	يشير إلى كيان المستخدم؛ انظر مؤشِّرات المستخدم
مرجع التطبيق	`Entity`	انظر مرجع التطبيق
حقل ربط	نوع الحقل المصدر	انظر أنماط تحديث حقل الربط
كتلة	`Block`	انظر صفوف الكتلة

أمّا الأصناف المسجَّلة على أنها مخفية – JSON، وأيقونة SVG، ومرجع الصفحة، والخيار الرقمي، ومحدِّد الحقل الحسّاس – وعنصر التحكّم «محدد الاستعلام»، فلا يعرضها المصمِّم؛ وترد تفاصيلها في أنواع الحقول الداخلية.

رايات القدرات

عند تسجيل الصنف، يُعلَن عن مجموعة من **رايات القدرات** التي تحكم كيف تسمح المنصة باستخدام ذلك الصنف – بمعزل عن إعدادات أي حقل بعينه:

- **`supportUniqueOption`** – يمكن وسم الصنف بأنه فريد (ويُفرض ذلك بفهرس فريد في قاعدة البيانات).
- **`supportSearchable`** – يمكن للصنف المشاركة في البحث النصي الكامل (`searchIndex.fullTextSearch`).
- **`supportAdvancedDefaultValue`** – يقبل الصنف قيمًا افتراضية قائمة على تعبير أو مؤشِّر، ولا يقتصر على قيمة حرفية.
- **`fieldQueryOptions`** – المرشِّحات والمؤشِّرات التي يتيحها الصنف في منشئ الاستعلامات.

- `overwriteFieldOption` – تجاوزات خاصة بكل صنف لسلوك قابلية البحث / قابلية الاستعلام / العرض فقط / التزايد التلقائي.

وتفسّر هذه الرايات سلوكًا قد يبدو غريبًا لولاها؛ فحقل **JSON** غير قابل للبحث ولا للاستعلام (لذا اجعل البيانات التي تحتاج إلى الترشيح بحسبها حقولًا فعلية)، وحقل **التسلسل** يتزايد تلقائيًّا ويظلّ مقفلًا للقراءة فقط، بينما حقلا **الرقم** و**المنطقي** قابلان للاستعلام.

الأصناف التي يمكن وسمها بـ **فريد** (`supportUniqueOption`): نص، ونص طويل، ونص متعدد اللغات، ونص طويل متعدد اللغات، وهاتف، ورقم، وتاريخ، ووقت، وتاريخ ووقت، وشهر، وسنة وشهر، وسنة، ومرجع، ومستخدم (والصنفان المخفيان مرجع الصفحة والخيار الرقمي). أمّا الأصناف التي تدعم **السماح بتحديدات متعددة** (`supportMultipleOptions`) فهي: ملف، وصورة، ومستند، وفيديو، وتصنيف، ومرجع، مع حدّين أدنى وأقصى قابلين للضبط؛ ومستخدم ومرجع التطبيق دون هذين الحدّين.

التحقّق وتعدّد القيم

تسري اعتبارات عدّة على جميع الأصناف؛ وتحيل إليها صفحات الأصناف بدلًا من تكرارها:

- **التحقّق** – يحمل كل حقل نموذج تحقّق (مطلوب، وحدّ أدنى/أقصى، وتعبير نمطي، وامتدادات الملفات، وقواعد القيم متعددة اللغات). وأُضيفت أنماط التحقّق من القيم متعددة اللغات **Since 1.4**. وتُطبَّق عمليات التحقّق المخصّصة القائمة على التعابير بالإضافة إلى القواعد المدمجة.
- **تعدّد القيم** – يمكن جعل أي حقل *متعدد القيم* (مصفوفة من قيمه) و/أو *قابلًا للتكرار* (يتكرّر الحقل أو الكتلة). وأُضيفت المعالجة الآمنة للفهارس عند إزالة العناصر المتكرّرة **Since 1.6**. و `Since 1.10` أصبح حقل **التصنيف** أيضًا قابلًا لتعدّد القيم، وهو الصنف الوحيد الذي يُفرض فيه الحدّ الأدنى/الأقصى لعدد القيم عند الحفظ – انظر التصنيف: القيم المتعددة. و `Since 1.10` أيضًا، يحمل كل صفّ في الكتلة القابلة للتكرار معرّف `__id` ثابتًا، فتظلّ القيم المقفلة والحسّاسة مرتبطة بالصفّ نفسه، لا بموضعه – انظر الكتلة: معرّفات صفوف ثابتة. ويحافظ الحفظ عبر معرض على أعمدة الكتلة التي لا يمنحها المعرض (`Since 1.4`)، وابتداءً من الإصدار 1.10.1 لا تُستعاد هذه الأعمدة إلا للصفوف المطابقة بحسب `__id` – انظر الكتلة: حفظ الصفوف عبر معرض.
- **القيم المحسوبة** – يمكن حساب قيمة الحقل من تعبير، وتتحكّم في ذلك مشغّلات الحساب ونطاق التنفيذ؛ ويسري النطاق (عند الإنشاء فقط، أو عند التحديث فقط، أو دائمًا) على كل مشغّل، بما فيه مشغّل تغيّر الحقل (`OnFieldsChanged`)، ولا يقتصر على مشغّلات وقت الحفظ.
- **الحقول الحسّاسة** – يمكن قصر إذن قراءة الحقل على مستخدمين أو مجموعات مُنحت هذا الإذن، وإرجاع قناع لسائر المستخدمين. **Since 1.8.** ولا يمكن وسم حقول المرجع والكتل بأنها حسّاسة.
- **الحقول المشفّرة** – يمكن بدلًا من ذلك تخزين الحقل في صورة نصّ مشفّر في حالة السكون، وكشف قيمة واحدة في كل مرة بإجراء فكّ التشفير. **Since 1.10.** والتشفير خيار *منفصل* عن الحساسية، ولا يمكن الجمع بينهما في الحقل نفسه. انظر الحقول المشفّرة.
- **التفرّد وأدوار العرض** – أسماء الحقول فريدة داخل الكيان، ولا يُسنَد كلّ دور من أدوار العنوان/الملخّص (العنوان، والرقم الرئيسي، والتاريخ الرئيسي، وما يشبهها) إلا إلى حقل واحد على الأكثر.

القيم المحسوبة

في صفحات الأصناف، يُخزَّن ما يُضبط باستخدام مفتاح **القيمة الافتراضية › متقدّم** – أي التعبير البرمجي ومشغّلاته – ضمن خيارات الحقل في `options.calculatedValueOptions` :

```
{
  "expression": "value.price * value.quantity",
  "calculationTriggers": [
    {
      "type": 3,
      "triggerOptions": { "watchedFields": ["price", "quantity"] }
    }
  ],
  "executionScope": 2
}
```

الخاصّية	إلزامية	النوع	المعنى
`expression`	لا	string	التعبير الذي يُقيَّم لإنتاج قيمة الحقل
`calculationTriggers`	**نعم**	array	يحدّد متى يُعاد الحساب؛ ولا يجوز أن يتكرّر أي `type` من أنواع المشغّلات أكثر من مرة واحدة
`calculationTriggers[].type`	**نعم**	number	قيمة من `CalculationTriggerEnum` (الجدول أدناه)
`calculationTriggers[].triggerOptions`	لا (يقبل null)	object	`{ "watchedFields": [...] }` – الأسماء النموذجية للحقول التي تراقبها المشغّلات القائمة على الحقول
`executionScope`	لا	number	قيمة من `ExecutionScopeEnum` (الجدول أدناه)

أنواع المشغّلات هي الخيارات الأربعة التي تصفها الواجهة بتحميل الصفحة، وإرسال النموذج، وتغيّر المُدخل، وتغيّر الحقل (`CalculationTriggerEnum`):

القيمة	الاسم	ينطلق
`0`	`OnFormOpened`	عند فتح النموذج؛ وعلى الخادم لا يسري إلا عند الإنشاء
`1`	`OnSave`	عند حفظ المُدخل
`2`	`OnEntryChanged`	عند تغيّر المُدخل
`3`	`OnFieldsChanged`	حين تتغيّر فعلًا قيمة حقل مدرج في `triggerOptions.watchedFields`

ويحدّد نطاق التنفيذ عمليات الحفظ التي تعيد حساب القيمة (`ExecutionScopeEnum`):

القيمة	الاسم	يعيد الحساب
`0`	`NewEntries`	عند الإنشاء فقط
`1`	`ExistingEntries`	عند التحديث فقط
`2`	`AllEntries`	عند كل حفظ

وعند غياب `executionScope`، لا يتحقّق شرط توقيت الحفظ على الخادم مطلقًا، فلا يُعاد حساب الحقل عند الحفظ. لذا اضبطه صراحةً لكل حقل تتوقّع أن يحافظ الخادم على تحديث قيمته.

أين يُقيَّم التعبير

يُقيَّم التعبير المخزَّن نفسه في موضعين: يقيّمه النموذج مباشرةً كلما انطلقت المشغّلات، **ويقيّمه** الخادم مرة أخرى أثناء التحقّق عند الحفظ. وفي عمليات الكتابة الواردة عبر الـ API، تُطبَّق النتيجة التي يحسبها الخادم في حالتين تحديدًا:

- أن يكون الحقل **مقفلًا أو مخفيًا** – فتُستبدل القيمة المرسلة بالنتيجة المحسوبة (فقفل الحقل لا يمنع حساب قيمته مطلقًا؛ إذ صُمّمت الخاصّيتان للعمل معًا)؛
- أن يكون الحقل **محذوفًا** من الحمولة وألّا تكون له قيمة افتراضية – فتُملأ قيمته بالنتيجة المحسوبة.

أمّا القيمة الصريحة المرسلة لحقل محسوب غير مقفل وغير مخفي، فتأخذ الأولوية على النتيجة المحسوبة. ولإعادة الحساب على الخادم شروط أيضًا: يجب أن يكون للحقل مشغّل واحد على الأقل غير `OnFormOpened`، وأن تكون له قيمة `executionScope` تطابق العملية (وغيابها يعني عدم إعادة الحساب مطلقًا). ويتطلّب مشغّل `OnFieldsChanged`، إضافةً إلى ذلك، أن تكون قيمة حقل مراقَب قد تغيّرت فعلًا. ولا ينطلق `OnFormOpened` مطلقًا عند التحديث؛ أمّا عند الإنشاء، فيُطلق التقييم على الخادم (حتى دون `executionScope`) لملء القيم المحذوفة، بينما تُحفظ في الحقول المقفلة أو المخفية القيمة التي حسبها النموذج مسبقًا.

والقاعدة العملية لمستدعي API هي: يمكنك حذف الحقل المحسوب متى كان مقفلًا أو مخفيًا، أو كانت مشغّلاته ونطاق تنفيذه تسمح للخادم بملء قيمته؛ أمّا الحقل الذي مشغّله الوحيد `OnFormOpened`، أو الذي لم يُضبط `executionScope` فيه، فلا يُعاد حسابه عند الحفظ – لذا أرسل قيمته بنفسك.

وتقابل عناصر المصمِّم هذه الحمولة على النحو التالي: **متقدّم** يفعّل `options.isAdvancedDefaultValue`؛ و**أدخل الإجرائية** يقابل `expression`؛ و**محفزات التنفيذ** (لا يظهر إلا ما دام **قفل الحقل** مفعّلًا) يقابل العنصر الأول في `calculationTriggers`، حيث **عند تحميل الصفحة** = `0`، و**عند الإرسال** = `1`، و**عند تغير المدخل** = `2`، و**عند تغير الحقل** = `3` (ويحدّد **الحقول القادحة للحساب** قيم `triggerOptions.watchedFields`)؛ و**نطاق التنفيذ** يقابل `executionScope`، حيث **المدخلات الجديدة** = `0`، و**المدخلات الموجودة** = `1`، و**كل المدخلات** = `2`. ويعيد اختيار مشغّل ضبط النطاق إلى `0`، ومع **عند تحميل الصفحة** لا يُعرض إلا `0`.

الحقول المقفلة على الخادم

يُفرض القفل على الخادم، وكذلك في النموذج. فعملية الحفظ التي ترسل قيمة جديدة لحقل مقفل – سواء أرسلها مستخدم أو تفاعل أو استدعاء مباشر لواجهة API – تنجح مع ذلك، لكنّ القيمة المرسلة تُهمَل بصمت، ويحتفظ الحقل بقيمته المخزَّنة (أو بقيمته الافتراضية عند إنشاء المُدخل). ولا يميّز هذا الفحص بين الجهات التي تكتب القيمة، فيسري على حفظ النماذج والاستدعاءات المباشرة لواجهة API والأتمتة على حدّ سواء؛ ولا يُتجاوز القفل إلا حين يتخطّى المستدعي التحقّق صراحةً، أو حين يُرفع قفل الحقل باستخدام الحقول المقفلة القابلة للتجاوز في زرّ نظام. أمّا القيمة المحسوبة للحقل نفسه فلا تتأثّر: إذ تُحسب وتُطبَّق والحقل مقفل، ولهذا يُعدّ إقران «قفل الحقل» بقيمة محسوبة الطريقة المعيارية لنمذجة حقل يتولّى النظام تحديث قيمته.

وفي المصمِّم، يجب أن تكون للحقل المقفل والإلزامي دون شرط قيمة افتراضية، ومع تفعيل **متقدّم** يصبح التعبير إلزاميًا. ولا يمكن قفل حقول الكتلة.

الحقول المشفّرة

تشفير (`options.encrypt`) خيار مستقل للحقل، وليس درجة أقوى من **حساس**. فالحساسية تغيّر ما *يُعرض* للقارئ، بينما تبقى القيمة المخزَّنة نصًّا صريحًا؛ أمّا التشفير فيغيّر ما *يُخزَّن*، إذ تُكتب القيمة نصًّا مشفّرًا. ولا يجوز أن يحمل الحقل **إلا أحد الاثنين على الأكثر** – فحفظ الخيارين معًا يُطلق `SENSITIVE_AND_ENCRYPT_MUTUALLY_EXCLUSIVE`، وفي محرّر الحقل يؤدّي تفعيل أحدهما إلى تعطيل الآخر بصمت.

الأصناف التي يمكن تشفيرها. لا يمكن تشفير أربعة أصناف مطلقًا، ولا يرفضها الخادم، بل يعيد `encrypt` بصمت إلى `false`:

قابل للتشفير؟	أصناف الحقول
أبدًا	المرجع / العلاقة، وحاوية الكتلة، و**المحوِّل**، و**التصنيف**.

قابل للتشفير؟	أصناف الحقول
نعم	كل ما عداها – النص وما يشبهه، والرقم، والتاريخ/الوقت/التاريخ والوقت، والمنطقي، والملف، والصورة، والفيديو، والمستند، والأيقونة، وأيقونة SVG، والنص متعدد اللغات، وJSON، وJSON الكائن، والخريطة (GeoJSON).

لاحظ أنّ هذه القائمة **ليست** مطابقة لقائمة الحساسية في أيّ من الاتجاهين. فالحساسية لا تستثني إلا حقول المرجع والكتل، لذا **يمكن لحقل المحوِّل أو التصنيف أن يكون حسّاسًا لكن لا يمكن تشفيره أبدًا**.

تفعيل التشفير يُلغي قابلية البحث. يفرض الخادم `options.searchable = false` عند الحفظ، فلا تصل القيمة المشفّرة مطلقًا إلى فهرس البحث النصي الكامل المشترك – وهو ما تفعله الحساسية أصلًا، ولهذا لا تكفي `supportSearchable` (انظر رايات القدرات) وحدها لوصف حالة الحقل المشفّر. ويمنع التشفير أيضًا استخدام الحقل عنوانًا رئيسيًا للكيان. ولأنّ القيمة المخزَّنة مبهمة، يمنع استخدامه في المرشّحات، ومعايير الاستعلام، وشروط التفاعلات، وتجميعات الخلاصة، والتقارير.

وترافق `encrypt` في خيارات الحقل خاصّيتان أخريان، هما `encryptTransitionVersion` و`pendingEncryptUpgrade`. ولا يضبط الباني أيًّا منهما، إذ تشكّلان سجلًّا يديره الخادم لعملية الانتقال «ترقية السجلّات» التي تبدأ عند تبديل حالة التشفير. وترد الآليات كاملةً – التشفير نفسه، وإجراء فكّ التشفير، والانتقال، وما يرفضه الحقل المشفّر – في الحقول الحساسة [بناء التطبيقات ‹ حماية حقل].

حقول الربط. تُنسخ رايتا `sensitive` و`encrypt` إلى حقل الربط من حقله المصدر في جميع أنماط التحديث، لذا لا يعرض المصمِّم أيًّا من المفتاحين لحقل الربط؛ وأي قيمة تُضبط على حقل الربط نفسه تُستبدل بقيمة المصدر عند الحفظ.

العرض للقراءة فقط والتحرير المباشر

لكل صنف قابل للتحرير **مكوّن عرض** مقابل يعرض قيمته المخزَّنة للقراءة فقط في السياقات التي تظهر فيها المُدخلات – **التفاصيل** و**الجدول** و**البطاقة** و**الشارة (Badge)**. فلكل حقل مكوّنان: مكوّن تحرير للنماذج ومكوّن عرض لإظهار القيمة (فللرقم مثلًا عرض بتنسيق العملة، وللتاريخ عرض فترة نسبية «منذ / بعد»، وللمنطقي عرض على هيئة شارة).

ويجمع **عارض محتوى المُدخل** مكوّنات العرض هذه في التخطيط النهائي للمُدخل، مراعيًا خيار العرض للكيان ونمط المعرض. ويدعم عدّة تخطيطات: حاوية بسيطة، وحاوية كتلة، وأعمدة مُدخلات، وتبويبات.

وعندما يكون الحقل قابلًا للتحرير، يتيح العارض أيضًا **التحرير المباشر** في موضعه: انقر (أو اضغط Enter) للتحرير، واضغط Esc للإلغاء، وCtrl-Sg للحفظ. وتُحفظ القيمة تلقائيًا عند النقر خارج الحقل. ويبقى الخادم صاحب القرار في قابلية التحرير – ولا سيما في حقول الكيانات التجميعية وحقول خطوات سير العمل، إذ له الكلمة الأخيرة في السماح بالتحرير المباشر.

ملاحظات بحسب نوع الحقل

النص المنسَّق: النص الطويل والنص الطويل متعدد اللغات

يقابل إعداد **النوع** لحقل النص الطويل في المصمِّم الخيارات المخزَّنة على النحو التالي: **عادي** (دون أيّ من الرايات)، و**غير منمط** = `isRichText`، و**مزين** = `isRichTextV2`، و**مزين مع رماز** = `isRichTextV2` مع `allowCodeEditor`. ولا يتيح حقل النص الطويل متعدد اللغات إلا **غير منمط**.

وللمحتوى الأطول متعدد اللغات، يمكن تحرير حقل النص متعدد اللغات بتنسيق منسَّق بدلًا من النص العادي. ويتوفّر نمطان للتحرير المنسَّق:

- `isRichText` – نمط **markdown** خفيف.

- `isRichTextV2` – محرّر منسَّق متقدّم بشريط أدوات تنسيق أشمل.
- `allowCodeEditor` – يتيح للمستخدم أيضًا التبديل إلى تحرير المصدر (الرماز) الأساسي مباشرةً.

ويستخدم التحرير المنسَّق محرّرًا مبنيًا على roosterjs. ويضمّ شريط أدواته خيارات الغامق، والمائل، والتسطير، والشطب، والعنوان، والقوائم المرقّمة وغير المرقّمة، والاقتباس، والرماز، والجدول، ومسح التنسيق (ولا يتيح نمط markdown التسطير). وعند عرض المُدخل، يُحلَّل markdown المخزَّن ويُنقّى HTML الناتج على مستوى DOM قبل عرضه.

صيغ الهاتف

يتضمّن حقل إدخال الهاتف قائمة منسدلة بالدول تتيح البحث فيها، ليختار المستخدم الدولة ورمز الاتصال بها قبل إدخال الرقم. وتُستمدّ بيانات الدول (الأسماء والأعلام ورموز الاتصال) من معلومات الدول المعيارية وفق ISO، ويُعرض رمز الاتصال منفصلًا عن الرقم. وعند عدم اختيار دولة، تُستخدم الإمارات العربية المتحدة (`ae`) افتراضيًا؛ ويمكن ضبط دولة افتراضية مختلفة للحقل باستخدام رمز ISO.

ويمكن عرض قيم الهاتف المخزَّنة بأيٍّ من الصيغ المعيارية التالية:

الصيغة	الوصف
`E164`	المعيار الدولي: علامة `+` في البداية، ثم رمز الدولة، ثم الرقم دون مسافات أو فواصل (مثل `+97150XXXXXXX`).
`INTERNATIONAL`	رمز الدولة مع مسافات تسهّل القراءة (مثل `+971 50 XXX XXXX`).
`NATIONAL`	الرقم كما يُكتب داخل بلده، دون رمز الدولة (مثل `050 XXX XXXX`).
`RFC3966`	صيغة على هيئة URI مسبوقة بـ `tel:` (مثل `tel:+971-50-XXX-XXXX`).

وافتراضيًا، تُعرض قيم الهاتف بصيغة `INTERNATIONAL` في سياقات القراءة فقط (التفاصيل والجداول والبطاقات).

الخريطة

حقل الخريطة محدِّد مبني على Leaflet. وإلى جانب النقر مباشرةً على الخريطة لوضع علامة، يمكنك البحث عن مكان باسمه؛ إذ يُجري الحقل الترميز الجغرافي والإكمال التلقائي عبر Nominatim ويعرض قائمة بالنتائج، وعند اختيار نتيجة، تنتقل العلامة إلى ذلك الموقع. وتُخزَّن القيمة بصيغة GeoJSON، ويتحقّق منها الخادم عند الإقلاع باستخدام مدقِّق GeoJSON الخاص به.

عرض القيمة المفكوك تشفيرها بالعرض الكامل · Since 1.10

إذا كان حقل الخريطة موسومًا أيضًا بـ **تشفير** (انظر الحقول المشفّرة) وكشفه مستخدم بإجراء فكّ التشفير، فإنّ الخريطة المكشوفة – حقل العنوان مع المحدِّد التفاعلي – تُعرض الآن بعرضها الكامل المعتاد في تفاصيل المُدخل. وقبل هذا الإصلاح، كانت تُعرض مضغوطة إلى نحو نصف العرض، وفق التخطيط الضيّق المخصّص للنص النائب القصير لقيمة مقنّعة أو مقفلة؛ أمّا الخريطة المقنّعة أو المقفلة، فما زالت تستخدم ذلك التخطيط الضيّق دون تغيير.

قيود الملفات

يتيح لك هذا القسم تقييد ما يمكن رفعه إلى حقل الملف:

1. **أنواع الملفات المقبولة:** قصر الحقل على أنواع ملفات محدّدة بإدراج الامتدادات المسموح بها (الإعداد `accept`). وعندئذٍ لا يستطيع المستخدمون تصفّح سوى الملفات المطابقة واختيارها.
2. **الحجم الأقصى للملف:** ضبط أكبر حجم (`maximumSize`) مسموح به لملف واحد مرفوع. ويُفحص الحجم في المتصفح قبل الرفع، فتُرفض الملفات التي تتجاوز الحدّ فورًا، ثم يُحوَّل الملف ويُنقّى أيضًا على الخادم بعد استلامه.

3. **الحدّ الأدنى / الأقصى للعدد:** حين يقبل الحقل أكثر من ملف، يمكنك اشتراط عدد أدنى وتحديد عدد أقصى للملفات في كل مُدخل.

ملاحظة: تُعرض الملفات المرفوعة بأيقونة تطابق نوع MIME الخاص بالملف، إلى جانب اسم الملف وحجمه بصيغة تسهّل قراءته.

وفي المصمِّم تظهر هذه الإعدادات باسم **الأنواع المقبولة للملف** (في حقول الملف)، و**الحجم الأقصى**، وحين يكون الحقل **السماح بتحديدات متعددة** أو **قابل للتكرار**، **حجم الملفات الكلي**. والحجمان افتراضيًا 20 MB، ولا يمكن ضبط **الحجم الأقصى** فوق 20 MB.

كلمة المرور

تُخزَّن قيمة حقل كلمة المرور كتجزئة أحادية الاتجاه، لا كنصّ مشفّر يمكن استعادته بفكّ التشفير. ويسري ذلك حتى إذا كان **تشفير** مفعّلًا على الحقل، لأنّ تجزئة كلمة المرور لها الأولوية دائمًا على التشفير في هذا النوع من الحقول. ولأنّه لا توجد قيمة يمكن كشفها، لا يعرض الحقل أبدًا إجراء فكّ التشفير أو أيقونة العين، بل يعرض دائمًا عنصرًا نائبًا بسيطًا `********`، دون أيقونة قفل أيضًا (فأيقونة القفل توحي بإمكان منح الوصول، وهو ما لا ينطبق أبدًا على قيمة كلمة المرور). ويشمل التقنيع أيضًا التلميح الذي يظهر عند التمرير بالفأرة فوق الحقل في عروض الجدول والبطاقة والشارة، ولا يقتصر على النص الظاهر.

مؤشّرات التاريخ والعرض النسبي

مؤشّرات الترشيح

بدلًا من تاريخ تقويمي ثابت، يمكن لحقل التاريخ استخدام **مؤشّرات الترشيح**، وهي رموز لتواريخ نسبية تُحَلّ وقت التشغيل، فتبقى الاستعلامات والقيم الافتراضية ديناميكية؛ إذ تتبع التاريخ الحالي دائمًا بدلًا من قيمة ثُبّتت عند تهيئة الحقل.

ومؤشّرات التاريخ المتاحة هي:

- **Now** – التاريخ الحالي (والوقت أيضًا، إذا كان الحقل يتضمّن وقتًا).
- **Today** – اليوم الحالي.
- **StartOfDay**، **StartOfWeek**، **StartOfMonth**، **StartOfYear** – بداية اليوم أو الأسبوع أو الشهر أو السنة الحالية.

ويمكن استخدام هذه المؤشّرات في موضعين:

1. **الاستعلامات / المرشّحات** – فمثلًا، يُبقي الشرط «المُدخلات التي يقع تاريخها في **StartOfMonth** أو بعده» المرشّح مقتصرًا على الشهر الحالي دون الحاجة إلى تعديله كل شهر.
2. **القيم الافتراضية** – يمكن أن تكون القيمة الافتراضية لحقل التاريخ في مُدخل جديد **Today** أو **Now** بدلًا من تاريخ ثابت يُكتب صراحةً.

عرض الفترة الزمنية النسبية

وفي سياقات القراءة فقط، يتيح حقل التاريخ أيضًا عارض **فترة زمنية** نسبية يُظهر الفاصل بين التاريخ المخزَّن والوقت الحالي بتعبير *fromNow* / *toNow* (مثل «منذ 3 أيام» أو «بعد أسبوعين») بدلًا من تاريخ مطلق.

وفي منشئ الاستعلامات، تُعرض المؤشّرات بالصيغ `@now` و`@today` و`@StartOfDay` و`@StartOfWeek` و`@StartOfMonth` و`@StartOfYear`. وفي التعبير، تُعيد الدالتان المساعدتان `scope.today()` و`scope.StartOfMonth()` تواريخ.

الشهر: الحدّ الأدنى والأقصى

يقبل حقل الشهر أيضًا قيدًا للحدّين الأدنى والأقصى، تُدخَل قيمتاه ويُتحقَّق منهما بوصفهما رقمَي شهر بسيطين (1-12) لا تاريخين؛ فإعداد مثل `{min: 1, max: 12}` صالح لحقل الشهر كما هو صالح لحقل الرقم. وإذا لم يُضبط حدّ أدنى أو أقصى، يُقبل أي شهر.

مصادر بيانات المرجع

إلى جانب **الكيانات** و**القوائم**، يمكن أيضًا ربط مصدر بيانات حقل المرجع بـ **معرض** أو **وحدة** أو **استعلام** محفوظ أو **مجموعة**. ويتيح ذلك للقائمة المنسدلة استمداد خياراتها من معرض مصفّى، أو من كائنات وحدة، أو من نتيجة استعلام محفوظ، أو من مجموعة مستقلة بدلًا من كيان كامل.

و**مؤشّرات الترشيح** رموز نسبية يمكنك تطبيقها لإبقاء قائمة الخيارات محصورة في سياق المستخدم الحالي بدلًا من كتابة قيم ثابتة. وتعتمد المؤشّرات المتاحة على المصدر المختار، وتشمل:

- **Me** – المستخدم الحالي.
- **MyGroups** – المجموعات التي ينتمي إليها المستخدم الحالي.
- **MyRoles** – أدوار المستخدم الحالي.
- **MyOrgUnit** – الوحدة التنظيمية للمستخدم الحالي.

الاستعلامات ذات المعاملات: حين يكون المصدر استعلامًا، يمكنك تمرير معاملات إليه كي تتكيّف الخيارات المجلوبة مع قيم حقول أخرى أو مع سياق وقت التشغيل.

الحفاظ على القيمة عند التحرير: يمكن لحقل المرجع الاحتفاظ بقيمته المخزَّنة عند تحرير المُدخل، حتى لو لم تعد تلك القيمة تطابق مرشّح المصدر الحالي، فلا تُمسح المُدخلات المرتبطة سابقًا دون تنبيه.

وفي المصمِّم، يتيح قسم **مصدر البيانات** خيار **مصدر الكيان** (كيان واحد أو أكثر، يُضاف باستخدام **إضافة مصدر**، ويمكن اختياريًا تضييق نطاق كلٍّ منها باستعلام محفوظ في **مصدر المدخلات**) وخيار **القوائم** (**اختر عنصر من القوائم**). وفي منشئ الاستعلامات، تُعرض المؤشّرات أعلاه بالصيغ `@me` و`@myGroups` و`@myRoles` و`@myOrgUnit`.

مؤشّرات المستخدم

لأنّ حقل المستخدم يشير إلى كيان USER، فإنه يتيح أيضًا مؤشّرات نسبية تُحَلّ ديناميكيًا بحسب المستخدم الحالي بدلًا من استخدام قائمة ثابتة:

- **Me** (`@me`): يُحَلّ إلى المستخدم المسجَّل دخوله حاليًا.
- **MyGroupsUsers** (`@myGroupsUsers`): يُحَلّ إلى المستخدمين المنتمين إلى مجموعات المستخدم الحالي.

وتُبقي هذه المؤشّرات الحقل ديناميكيًا لكل مستخدم، ويمكن استخدامها في القيم الافتراضية وشروط الترشيح.

مرجع التطبيق

عند عرض قيمة الحقل، تسترجع المنصة معلومات إضافية عن الكائن المشار إليه وترفقها، بما فيها أصله الأعلى (التطبيق المالك)، واسم العرض الخاص به، وما إذا كان الكائن المستهدف تطبيقًا أو خدمة. ويتيح ذلك للمعارض إظهار عنوان ورابط يوضّحان الكائن المختار.

التصنيف: خيارات يديرها المستخدمون `Since 1.10`

في العادة، لا يحرّر قائمةَ خيارات حقل التصنيف إلا مطوّر، من إعدادات الحقل المذكورة أعلاه. ويتيح إعدادان للمستخدمين النهائيين إضافة الخيارات أو إعادة تسميتها بأنفسهم من بطاقة إنشاء المُدخل أو تحريره، دون تعديل إعدادات الحقل. ويجب تفعيل كليهما:

الإعداد	أين تضبطه	الأثر
السماح للمستخدمين بإدارة الخيارات (`options.allowUsersToManageOptions`)	إعدادات حقل التصنيف	مطلوب لأي تغيير يجريه مستخدم نهائي على الخيارات. ولا يُسمح به إلا لحقل تصنيف تكون خياراته **غير** مربوطة بحقل آخر؛ ويُرفض حفظه على حقل مربوط (`INVALID_USER_OPTION_PERMISSIONS`).
إضافة خيارات التصنيف / **تعديل خيارات التصنيف** (`allowAddCategoryOptions` / `allowEditCategoryOptions`)	إعدادات البطاقة في المعرض، لكل عملية (بطاقة الإنشاء، وبطاقة التعديل)، في `editConfiguration.supportedOperations[]`	يحدّد ما إذا كان يجوز للمستخدمين النهائيين إضافة الخيارات أو تعديلها في بطاقة ذلك المعرض. ويُبقي **ربط إعدادات بطاقتي الإنشاء والتعديل** العمليتين متطابقتين.

وحين يحفظ المستخدم مُدخلًا يتضمّن تغييرات على الخيارات، يرسلها العميل في جسم المُدخل (عند الإنشاء والتحديث والإنشاء الدفعي) بصيغة `categoryOptionChanges`: كائن مفاتيحه الأسماء النموذجية الكاملة للحقول (وحقول الكتلة الداخلية بالشكل `block.field`)، وتكون كل قيمة فيه بالشكل `{baseOptionIds?, create?: [{id, label, color?}], update?: [{id, label, color?, base: {label, color?}}]}`. والتسميات سلاسل نصية لكل لغة، واللون الافتراضي `#81C784`، ويختار العميل معرّفات الخيارات الجديدة.

ويفحص الخادم كل حقل **قبل** كتابة المُدخل. فإذا جرى الحفظ عبر معرض، وجب أن ينتمي المعرض إلى هذا الكيان، وأن تسمح بطاقته بالعملية، وأن يسمح الحقل بإدارة المستخدمين للخيارات وأن يكون قابلًا للتحرير في ذلك المعرض؛ أمّا دون معرض، فيجب أن يكون المستدعي قادرًا على تعديل الكيان. ويُرفض التغيير إذا كان الكيان مقفلًا من مستخدم آخر، أو مثبَّتًا من حزمة. ثم تُضاف الخيارات إلى تعريف الكيان بتحديثات محروسة، ويُنشر حدث `DataEntityUpdatedEvent` واحد.

الخطأ	متى
`403 CATEGORY_OPTION_CHANGE_NOT_ALLOWED`	لا تسمح إعدادات الحقل أو بطاقة المعرض أو أذونات المستدعي بهذا التغيير.
`400 INVALID_CATEGORY_OPTION_CHANGE`	حقل غير معروف، أو تسمية فارغة، أو تغيير آخر غير سليم البنية.
`409 CATEGORY_OPTION_CONFLICT`	غيّر مستخدم آخر الخيار أو حذفه في الأثناء (`update.base` هو رمز التزامن المتفائل).
`409 CATEGORY_OPTIONS_CHANGED`	لم يعد `baseOptionIds` يطابق مجموعة الخيارات المخزَّنة.

التصنيف: القيم المتعددة `Since 1.10`

يمكن لحقل التصنيف أن يحمل **عدّة خيارات في مُدخل واحد**. فعّل **السماح بتحديدات متعددة** في إعدادات قيمة الحقل (الراية العامة في المنصة `options.multiValue`)؛ ويمكنك أيضًا ضبط حدّين أدنى وأقصى اختياريين لعدد الاختيارات. وعندئذٍ تعرض محرّرات المُدخلات الاختيارات على هيئة رقاقات قابلة للإزالة، وتعرض خلايا الجداول جميع القيم. ويدعم تطبيق الجوّال السلوك نفسه.

- **البنية المخزَّنة.** يخزّن حقل التصنيف المفرد `{id}`؛ أمّا متعدد القيم، فيخزّن **مصفوفة** `[{id}, …]` بترتيب الاختيار (والتصنيف متعدد القيم القابل للتكرار مصفوفة من المصفوفات). وتتبع القيمة الافتراضية البنية نفسها: فالقيمة الافتراضية متعددة القيم مصفوفة من `{id}`، تكون معرّفاتها ضمن خيارات الحقل وتلتزم بالحدّين الأدنى والأقصى.
- **قواعد العدّ.** يجب أن يكون كل `id` مختار معرّفًا لأحد خيارات الحقل. ويُفرض الحدّ الأدنى (`multiValue.min.value`) والأقصى (`multiValue.max.value`) في حقول التصنيف: فالاختيار الفارغ لا يخالف الحدّ الأدنى أبدًا، لأنّ السماح بالفراغ

يحدّده **مطلوب** (فلا يجعل حدّ أدنى قدره 1 الحقلَ مطلوبًا ضمنيًا)؛ وغياب الحدّ الأدنى يعني 0، وغياب الحدّ الأقصى، أو كونه 0 أو أقل، يعني عدم وجود حدّ.

- **تفعيل التعدّد تغيير كاسر للتوافق.** يُرفع إصدار الكيان وتصبح المُدخلات الموجودة متقادمة حتى تُرقّى؛ ويضع الإصلاح التلقائي كل قيمة مفردة مخزَّنة في مصفوفة من عنصر واحد. وإلى أن يُرقّى المُدخل، يظلّ حفظه لأي سبب آخر ناجحًا، لأنّ كل عملية حفظ تضع القيمة المفردة المتبقية في مصفوفة. انظر الإصدارات والتغييرات الكاسرة [بناء التطبيقات › إنشاء كيان وتصميمه].
- **تعطيل التعدّد** يُبقي القيمة الأولى المختارة فقط ويُسقط البقية؛ ويُحوَّل الاختيار الفارغ إلى قيمة فارغة (`null`).
- **يُرفض التبديل ما دام الحقل معتمَدًا عليه.** إذا استخدم معرض أو استعلام إحصائي للكيان نفسه الحقلَ في لوحة مهام، أو تجميع، أو فرز أو لوحة معلومات، أو شريط زمني، يفشل الحفظ بـ `400` `CATEGORY_FIELD_MULTI_VALUE_IN_USE`، ويسرد `details.usages` كل استخدام (`{type, id, name}`) لتزيل تلك الاستخدامات أولًا. ولا تُفحص المعارض التي تستند إلى كيان *مرتبط* وتصل إلى الحقل عبر علاقة.
- **أين لا يمكن استخدام التصنيف متعدد القيم.** ترفضه جميع الاستخدامات التي تفترض قيمة واحدة لكل مُدخل بالخطأ `400 MULTI_VALUE_CATEGORY_FIELD_NOT_ALLOWED` (`details.usage`): **الفئة الرئيسية** للكيان أو **الفئة الفرعية**، وحقل أعمدة لوحة المهام، والفرز الافتراضي، والتجميع في لوحة المعلومات، وتعيين سمة الشريط الزمني. ويُستبعد أيضًا من خيارات التجميع في المعرض.
- **التصدير والمرشّحات.** تطبع عمليات التصدير بالقوالب التصنيفَ متعدد القيم بتسمياته مجمَّعةً لكل لغة، ويقبل المرشِّحان `In` / `NotIn` قيمة تصنيف مفردة كما يقبلان قائمة.

التصنيف: أدوار عرض الخيارات

يمكن إسناد دور عرض لكل خيار يُضاف إلى حقل التصنيف، فيُعرض **فئةً** أو **فئةً فرعية** أو **عنوانًا** عند عرض المُدخلات. وتستند خيارات حقل التصنيف إلى مجموعة تصنيفات داخلية تُدار لكل كيان. وحين يكون حقل التصنيف مربوطًا بحقل آخر، تُحدَّث خياراته المتاحة تلقائيًا كلما تغيّر الحقل المصدر.

أنماط تحديث حقل الربط

تقابل قائمة **خيار التحديث** في المصمّم القيمة `mapping.updateMode` :

تسمية المصمّم	القيمة	السلوك
الكيان الحالي	`Local` = 0	يُملأ من المصدر تلقائيًا ويقبل التحرير؛ وتقتصر التعديلات على هذا المُدخل ولا تُكتب إلى المصدر. وهو الخيار الافتراضي في المصمِّم.
عدم التحديث	`Never` = 1	يُملأ من المصدر في المرة الأولى فقط؛ ولا تُنسخ التغييرات اللاحقة على المصدر. ويجب أن يكون الحقل مقفلًا (وإلا رفض الخادم التعريف).
من الكيان الأب	`ParentOnly` = 2	يتبع المصدر: حين تتغيّر قيمة مُدخل المصدر، يُنسخ التغيير إلى هذا الحقل. ويجب أن يكون الحقل مقفلًا.
مزامنة	`TwoWay` = 3	قابل للتحرير في الموضعين؛ ويُكتب التغيير في أيّ جانب إلى الجانب الآخر. ويجب أن يطابق إعداد **مطلوب** لحقل الربط إعدادَ الحقل المصدر.

ويفرض الخادم القواعد التالية عند حفظ الكيان: يجب أن يكون حقل الربط من نمط `Never` أو `ParentOnly` مقفلًا؛ ويجب أن يطابق إعداد الإلزام في حقل الربط من نمط `TwoWay` إعدادَ الحقل المصدر؛ وحين يقع حقل الربط في كتلة قابلة للتكرار بينما يقع حقل المرجع الخاص به خارجها، لا يُسمح إلا بـ `Never` و `ParentOnly` . ولا تتلقّى التغييرات من مُدخل المصدر بعد تحديثه إلا حقول `ParentOnly` و `TwoWay` .

التسلسل: إعدادات التزايد التلقائي

تقابل **اعدادات التزايد التلقائي** في المصمِّم `autoIncrementOptions` : **قيمة البداية** = `startAt` ، و**مقدار التزايد** = `incrementBy` ، و**التسلسل المشروط** = `conditionalIncrement` ، ويمثّل كل زوج **المفتاح / القيمة** (يُضاف باستخدام **إضافة مفتاح**) عنصرًا في `keys` يحمل `keyName` وتعبيرًا في `value` (`calculationMethod` 2). وبعد استخدام العدّاد، تصبح المفاتيح للقراءة فقط.

وإلى جانب قيمة البداية (`startAt`) ومقدار التزايد (`incrementBy`)، تدعم إعدادات التزايد التلقائي (`autoIncrementOptions`) سلوكين متقدّمين:

- **التزايد بالمفاتيح (`keys`):** يخصّص العدّاد لتركيبة من الحقول المفتاحية، فيُحتفظ بتسلسل مستقل لكل تركيبة مفاتيح مختلفة بدلًا من عدّاد عام واحد. فرقم الفاتورة مثلًا يمكن أن يُحتسب بصورة مستقلة لكل فرع أو لكل سنة، بدلًا من أن تشترك المُدخلات كلها في تسلسل واحد.
- **التزايد المشروط (`conditionalIncrement`):** لا يتقدّم التسلسل إلا حين يتحقّق شرط محدّد، فلا تستهلك المُدخلات التي لا تستوفي الشرط رقمًا.

ولمثال عملي متكامل يجمع السلوكين، انظر خطوات المرجع المنسَّق أدناه.

ويعرض المثال العملي للمرجع المنسَّق أدناه طريقة الإعداد الواردة في الحقل المتسلسل [بناء التطبيقات] من منظور مطوّر التكامل.

الخطوة 1 – تسلسل مفتاحه السنة

اضبط إعدادات التزايد التلقائي لحقل التسلسل (`autoIncrementOptions`) مع تفعيل التزايد المشروط ومفتاح واحد يُحَلّ تعبيره إلى السنة:

```
{
  "autoIncrementOptions": {
    "enabled": true,
    "key": "lc-reference",
    "startAt": 1,
    "incrementBy": 1,
    "conditionalIncrement": true,
    "keys": [
      {
        "keyName": "year",
        "calculationMethod": 2,
        "value": "new Date().getFullYear()"
      }
    ]
  }
}
```

ويحدّد `calculationMethod` مصدر قيمة المفتاح:

`calculationMethod`	المصدر
`0`	قيمة ثابتة
`1`	حقل
`2`	تعبير (الافتراضي)
`3`	مؤشِّر

ولا توجد لغة شروط منفصلة: فعند استخدام `calculationMethod: 2`، تحتوي `value` في المفتاح على رماز JavaScript عادي يُقيَّم باستخدام بيانات المُدخل؛ وتكون `value` مرتبطة بقيم المُدخل، وتُتاح الدوال المساعدة مثل `now()` و`today()` في النطاق.

سلوك المفتاح وقت التشغيل:

- تصبح نتيجة المفتاح جزءًا من البحث عن العدّاد؛ إذ يُستخرج الرقم التالي بعملية إدراج أو تحديث (upsert) ذرّية تعتمد في المطابقة على العدّاد وقيمة كل مفتاح، فتحتفظ كل سنة بعدّادها المستقل، الذي يبدأ من `startAt` ويتقدّم بمقدار `incrementBy`. وحين يُنشأ أول مُدخل في سنة جديدة، لا يكون صفّ عدّاد تلك السنة موجودًا بعد، فيبدأ من جديد عند `startAt`، وبذلك لا تحتاج إعادة الضبط السنوية إلى أي خطوة صيانة.
- مع تفعيل `conditionalIncrement`، تفرض المنصة تعطيل التفرّد للحقل (وعند تعطيله، يُفرض على حقل التزايد التلقائي `unique: true`). والتحديث الذرّي للعدّاد هو ما يمنع تكرار الأرقام داخل السنة، لا راية التفرّد.
- عند حفظ تعريف الكيان، تُنسخ `startAt` و`incrementBy` و`keys` إلى مستند العدّاد في المنصة، وهو المرجع المعتمد وقت التشغيل؛ ولا تنتقل التعديلات اللاحقة على `keys` إلى العدّاد إلا ما دام غير مستخدم. لذا حدّد بنية المفاتيح قبل إنشاء مُدخلات فعلية.

الخطوة 2 – الحقل المرافق المقفل

أضف حقلًا ثانيًا مقفلًا – حقل نص مثلًا اسمه النموذجي `ref` – ليحمل القيمة المنسَّقة، واضبط له قيمة محسوبة. ويفترض المثال أنّ الاسم النموذجي لحقل التسلسل هو `seq`:

```
{
  "calculatedValueOptions": {
    "expression": "value.ref ? value.ref : (value.seq ? 'LC-' + new Date().getFullYear() + '-' + String(value.seq).pad
Start(4, '0') : null)",
    "calculationTriggers": [{ "type": 1 }],
    "executionScope": 2
  }
}
```

وفي هذا الإعداد نقطتان أساسيتان:

- **القفل مطلوب كي يُطبَّق الحساب.** لا يكتب الخادم نتيجة محسوبة في المُدخل إلا حين يكون الحقل مقفلًا (أو مخفيًا)؛ أمّا الحقل غير المقفل، فيحتفظ بالقيمة المرسلة. وتعامل المنصة هذا الاقتران على أنه مقصود؛ إذ تُبلَّغ النماذج تلقائيًا بأنّ أي حقل له `calculatedValueOptions` مقفل.
- **يجب ضبط `executionScope`.** عند غيابه، لا يعيد الخادم حساب الحقل عند الحفظ أبدًا، ومن ثمّ لا يُركَّب المرجع. استخدم `1` أو `2` كي تشمل إعادة الحساب المُدخلات الموجودة أصلًا.

`calculationTriggers[].type`	المشغّل
`0`	عند فتح النموذج (في سياق النموذج فقط: يحتفظ الخادم بالقيمة المرسلة في الحقول المقفلة)
`1`	عند الحفظ
`2`	عند تغيّر المُدخل
`3`	عند تغيّر الحقول (يراقب `triggerOptions.watchedFields`)

`executionScope`	يعيد الحساب عند
`0`	الإنشاء فقط (المُدخلات الجديدة)
`1`	التحديث فقط (المُدخلات الموجودة)

executionScope	يعيد الحساب عند
2	الإنشاء والتحديث (كل المُدخلات)
غير مضبوط	لا يُعاد الحساب على الخادم أبدًا

الحارس الذي يضمن ثبات المرجع

الجزء الأول `… : value.ref ? value.ref` هو ما يجعل إعادة تنفيذ التعبير آمنة. فمشغّل الحفظ يعمل عند كل حفظ لاحق للمُدخل؛ ودون الحارس، تُعيد كل عملية حفظ تركيب المرجع، وإذا جرت بعد بداية سنة جديدة، أُعيد بناؤه بالسنة الجديدة، فيتغيّر مرجع صادر مثل `LC-2026-0042` إلى `LC-2027-0042` دون تنبيه. ومع الحارس، تثبت القيمة الأولى المركّبة: فكل إعادة حساب لاحقة تجد `value.ref` مضبوطة فتُعيدها دون تغيير، وبذلك يظلّ المرجع ثابتًا.

ولا يقلّ الحارس الداخلي `value.seq ? … : null` أهمية، بسبب التوقيت الموصوف تاليًا؛ فعند الإنشاء، يُقيَّم التعبير بينما حقل التسلسل لا يزال فارغًا. ودون هذا الفحص، سيركّب التعبير `LC-2026-undefined` ، ثم يثبّت الحارس الخارجي تلك القيمة غير السليمة إلى الأبد. أمّا إرجاع `null` ، فيجعل الحساب يُعدّ فاشلًا، فتترك المنصة الحقل دون ضبط عند الإنشاء (وتُبقي القيمة القديمة عند التحديث)، وهو السلوك الذي تتطلّبه طريقة الإعداد هذه. ويظهر المرجع عند أول حفظ لاحق، متى توفّر رقم التسلسل.

وكلا التعبيرين في طريقة الإعداد هذه – مفتاح العدّاد وحساب الحقل المرافق – رماز JavaScript عادي يُقيَّم باستخدام بيانات المُدخل؛ فإذا وجب أن تطابق السنة المستخدمة في المرجع سنة العدّاد دائمًا، فاشتقّ كليهما من حقل التاريخ نفسه في المُدخل بدلًا من التاريخ الحالي.

التوقيت: يُسند رقم التسلسل بعد التحقّق

هذا الترتيب من سلوك المنصة، وهو ما يحكم طريقة الإعداد بأكملها. فعند إنشاء مُدخل، تتحقّق المنصة منه أولًا – ويجري تقييم الحقول المحسوبة على الخادم ضمن ذلك التحقّق – ثم تسترجع رقم التسلسل وتكتبه في المُدخل، وبعد ذلك فقط تخزّن المُدخل. لذا يجد التعبير المحسوب، عند تقييمه أثناء طلب الإنشاء، حقل التسلسل دائمًا بقيمة `undefined` ؛ فلا تتوقّع المرجع المنسَّق في استجابة الإنشاء. والإنشاء أيضًا هو الوقت الوحيد الذي يُسند فيه رقم تسلسل؛ فالتحديثات لا تُسنده أو تعيد إسناده أبدًا. لذا يكون الرقم موجودًا عند أي حفظ لاحق، ويمكن لإعادة الحساب التي يطلقها الحفظ أن تقرأه.

ويُسجَّل إخفاق العدّاد أثناء الإنشاء، لكنه لا يُلغي الحفظ: فيُنشأ المُدخل بحقل تسلسل فارغ. ويُبقي فرع `null` في الحارس الحقلَ المرافق فارغًا في هذه الحالة أيضًا، بدلًا من تثبيت مرجع غير سليم.

المُدخلات المُنشأة عبر الـ API

لا يحتاج رقم التسلسل نفسه إلى تدخّل العميل؛ فهو يُسند على الخادم عند كل إنشاء، أيًّا كانت القناة التي أنشأت المُدخل. أمّا المرجع المنسَّق، فيختلف سلوكه؛ فلأنّ الرقم يُسترجع بعد تقييم الحقول المحسوبة، لا تستطيع أي قناة – سواء كانت نموذجًا أو API – تركيب المرجع ضمن طلب الإنشاء نفسه، فيبدأ المُدخل المُنشأ عبر الـ API بقيمة `ref` فارغة. وتُملأ هذه القيمة بإحدى ثلاث طرق:

1. **حفظ لاحق** يطابق مشغّله و`executionScope` الخاص به (كما هو مضبوط أعلاه): يعيد الخادم تركيب الحقل بنفسه، دون تدخّل العميل. وهذه هي الطريقة الموصى بها؛ نفّذ عملية حفظ لاحقة بعد إنشاء المُدخل، أو دع التحديث الطبيعي التالي يتولّى ذلك.
2. **يمرّر المستدعي القيمة منسَّقة سلفًا.** لا يُضمن احتفاظ الخادم بالقيمة المرسلة في الحقل المقفل إلا حين يكون مشغّل الحقل عند فتح النموذج (`type: 0`) – وهي الحالة الوحيدة الخاصة بسياق النموذج، حيث لا يُجري الخادم أي حساب ويثق بالقيمة التي أرسلها النموذج (أو مستدعي الـ API) – أو حين يُفتح قفل الحقل عبر الحقول المقفلة القابلة للتجاوز في زرّ نظام. وفي إعدادات المشغّلات الأخرى، تكون الأولوية للحساب على الخادم على القيمة المرسلة.
3. **تعبئة لاحقة بعملية منفصلة** تكتب المرجع مباشرةً.

مقفل وغير مطلوب أبدًا

حقل التسلسل مقفل دائمًا؛ إذ تشترط المنصة أن يكون كل حقل تزايد تلقائي مقفلًا، وترفض تعريف الحقل خلاف ذلك (انظر الحقول المقفلة على الخادم). ومع تعطيل `conditionalIncrement`، يُفرض على حقل التزايد التلقائي `unique: true`.

`Since 1.10` لا يمكن أن يكون حقل التسلسل مطلوبًا أبدًا؛ إذ ترفض المنصة فرض شرط «مطلوب» على حقل تزايد تلقائي من أي مصدر يمكن أن يفرضه: مفتاح «مطلوب» في الحقل نفسه، وقائمة الحقول المطلوبة في المعرض نفسه، وخطوة سير عمل تجعل الكتلة المحيطة بالحقل إلزامية. فالمنصة لا تُسند رقم التسلسل إلا بعد اجتياز المُدخل للتحقّق، والحقل لا يظهر أبدًا في نموذج المُدخل، لذا لا يمكن للمستدعي اتخاذ أي إجراء يستوفي فحص «مطلوب» عليه.

الكتلة: الحقول الفرعية المطلوبة في كتلة مخفية `Since 1.10`

يُعلَّق شرط **مطلوب** للحقل الفرعي المتداخل داخل كتلة كلما كانت الكتلة نفسها مخفية، سواء أُخفيت بشرط الإخفاء الخاص بها أو بخطوة سير عمل تخفيها. ويسري ذلك على أي عمق من التداخل داخل الكتلة، وفي الكتلة **القابلة للتكرار** يُحدَّد تطبيقه لكل تكرار على حدة: فقد يكون تكرار واحد من الكتلة مخفيًا (مثلًا بتعبير لا يخفي إلا الصفّ الأول)، بينما يبقى تكرار آخر ظاهرًا وتظلّ حقوله الفرعية المطلوبة إلزامية كالمعتاد.

الكتلة: فكّ تشفير حقل داخل كتلة قابلة للتكرار `Since 1.10`

للحقل **المشفّر** (انظر الحقول المشفّرة) المتداخل داخل كتلة **قابلة للتكرار** قيمة مخزَّنة لكل تكرار، لذا أصبح كشفه يتطلّب محدِّدًا للصفّ: فإجراء فكّ التشفير يستهدف تكرارًا محدّدًا لا الحقل بأكمله. ويُرفض فكّ تشفير حقل كهذا دون تحديد التكرار، أو عند تحديد تكرار غير موجود. أمّا آلية فكّ التشفير نفسها – نموذج التشفير في حالة السكون، ومنح الوصول، وإجراء فكّ التشفير عمومًا – فموثّقة في الحقول الحساسة [بناء التطبيقات › حماية حقل]؛ وتقتصر هذه الملاحظة على السلوك الخاص بالكتلة القابلة للتكرار.

الكتلة: معرّفات صفوف ثابتة `Since 1.10`

أصبح كل صفّ في الكتلة **القابلة للتكرار** يحمل مفتاحًا محجوزًا، `__id` (uuid)، ليميّز الخادم بين الصفوف عند التحديث. وقبل هذا التغيير، كانت الحالة الخاصة بكل صفّ (القيم المقفلة، والقيم الحسّاسة، والفروق في سجلّ المُدخل) تُنقل عند التحديث **بحسب الموضع**، فلم تكن تتبع الصفّ إذا تغيّر موضعه.

ما يعنيه ذلك عند التصميم أو تطوير التكامل:

- **ليس حقلًا.** لا يظهر `__id` أبدًا في تعريفات حقول الكيان، فلا تشمله عمليات التصدير أو الفهرسة أو عروض الجداول أو منتقيات المرشّحات. والبادئة `__` محجوزة؛ فلا يمكن لأي حقل تعرّفه أن يبدأ بها.
- **أعِد المعرّفات التي تتلقّاها.** الصفّ الذي تطابق قيمة `__id` فيه معرّف صفّ مخزَّن يُعامَل على أنه الصفّ نفسه؛ والصفّ ذو المعرّف غير المعروف يُعامَل على أنه صفّ جديد؛ وتُزال الصفوف المخزَّنة التي لا يقابلها أي صفّ مرسل. ويتولّى عميلا الويب والجوّال ذلك نيابةً عنك.
- **العملاء الذين لا يرسلون أي معرّفات ما زالوا يعملون.** إذا لم يحمل أي صفّ في المصفوفة المرسلة `__id`، يُطابَق كل صفّ بالصفّ المخزَّن في الموضع نفسه، كما كان سابقًا. وإذا حملت بعض الصفوف معرّفات ولم يحملها بعضها الآخر، عوملت الصفوف التي بلا معرّف على أنها صفوف جديدة، ولا ترث أبدًا قيم صفّ مخزَّن. أمّا الحفظ **عبر معرض**، فأشدّ صرامة: انظر القسم التالي.
- **تحصل الصفوف المخزَّنة على معرّفاتها عند الترقية** (`Since 1.10`، ابتداءً من الإصدار 1.10.1). فالترحيل `1.10.0-a52` (`stamp-repeatable-block-row-ids`) يمنح كل صفّ مخزَّن في كل كتلة قابلة للتكرار في المستوى الأعلى، وفي كل مساحة، معرّف `__id`. وتُحفظ المعرّفات الموجودة أصلًا، ولا تُسجَّل الكتابة في السجل التاريخي، ولا تغيّر إعادة تشغيل

الترحيل شيئًا. وقبل ذلك، كان الصفّ المخزَّن بلا معرّف يُمنح معرّفًا جديدًا كلما فُتح في نموذج الويب، فلا يمكن لأي عملية حفظ أن تطابق ذلك الصفّ بالصفّ المخزَّن.

- **الخادم يملك المعرّف.** يُحتفظ بمعرّف uuid السليم الذي ترسله؛ ويولّد الخادم معرّفًا لصفّ بلا معرّف، أو بمعرّف غير سليم، أو بمعرّف يكرّر معرّفًا سابقًا. ويُرفض وجود صفّين بالمعرّف نفسه في مصفوفة مرسلة واحدة، ولا يمكن تغيير `__id` عبر عملية patch.
- **سلوك ذو صلة.** أصبحت تحويلات ترقية المُدخلات بعد تغيير كاسر تُطبَّق على **كل** صفّ في الكتلة القابلة للتكرار، ويقارن السجل التاريخي للمُدخل الصفوف بمعرّفاتها، وتطابق إجراءات التفاعل الخاصة بعناصر الكتلة (إنشاء عنصر كتلة / تحديثه) الصفوف بمعرّفاتها كذلك. وتُقيَّم التعابير في قواعد الحقول داخل الكتلة القابلة للتكرار لكل صفّ على حدة (`scope.index`).

الكتلة: حفظ الصفوف عبر معرض

تسرد عملية التحرير في المعرض مسارات الحقول التي يجوز للمستخدم الكتابة فيها. وتُرسل صفوف الكتلة القابلة للتكرار في مصفوفة واحدة **تحلّ محلّ** المصفوفة المخزَّنة، وكان ذلك سيؤدّي، لولا الإصلاح، إلى اختفاء كل عمود لا يتيح المعرض تحريره من كل صفّ عند الحفظ. ويعالج الخادم ذلك بمطابقة الصفوف المرسلة بالصفوف المخزَّنة في كل كتلة قابلة للتكرار، ثم نسخ القيم **المخزَّنة** لكل مسار لا يتيح المعرض للمستخدم تحريره إلى الصفّ المطابق. وتُستمدّ هذه القيم دائمًا من التخزين، ولا تؤخذ من الحمولة مطلقًا.

- **الإبقاء على الأعمدة غير الممنوحة** متاح `Since 1.4` : وقد أُدرج في إصدارات التصحيح 1.4 و1.6 و1.8 (1.4.119 و1.6.69 و1.8.28) وفي 1.10.1. وفي فروع الإصدارات 1.4 و1.6 و1.8، لا تحمل الصفوف معرّفات، فيُطابَق الصفّ المرسل بصفّ مخزَّن بحسب **قيمه الممنوحة غير المتغيّرة** (في الموضع نفسه، أو في أي موضع حين يزيل الحفظ صفوفًا). وتُبقي تلك الفروع أيضًا الصفوف الفارغة في الكتلة التي لم يغيّرها الحفظ.
- **المطابقة بمعرّف الصفّ** متاحة `Since 1.10` ، ابتداءً من الإصدار 1.10.1 (وليس في 1.10.0): ففي فرع الإصدار 1.10 وما بعده، لا تعتمد المطابقة إلا على `__id` .

ما يحتاج عميل الـ API الذي يكتب عبر معرض إلى معرفته (1.10.1 وما بعده):

1. **أعِد كل صفّ موجود مع `__id` الخاص به.** لا تُستعاد الأعمدة غير الممنوحة إلا للصفّ الذي تطابق قيمة `__id` المرسلة فيه معرّف صفّ مخزَّن. ولا يُستعاد شيء للصفّ الذي لا يُطابَق إلا بالموضع. وإعادة ترتيب الصفوف، أو حذف صفوف تسبق صفًّا ما، آمنة ما دامت المعرّفات ترافق الصفوف.
2. **الصفّ بلا معرّف بجانب صفوف تحمل معرّفات صفّ جديد.** لا يُستعاد له شيء، فتكون أعمدته غير الممنوحة فارغة، ويولّد الخادم معرّفه. لذا لا يمكن لصفّ أُضيف مكان صفّ محذوف أن يرث القيم المقفلة لذلك الصفّ.
3. **المصفوفة التي لا معرّفات فيها إطلاقًا** ما زالت تُطابَق بالموضع لفحص القيم المقفلة، فتظلّ التكاملات الأقدم تعمل، لكنّ الحفظ عبر المعرض **لا** يستعيد **شيئًا** لتلك الصفوف. لذا أرسل المعرّفات.
4. **المعرّف الذي لا يعرفه الخادم** يجعل الصفّ يُعامَل على أنه جديد. وعند الحفظ عبر المعرض، لا يُحتفظ بالمعرّف الذي اخترته؛ إذ ليس `__id` ضمن قائمة حقول أي معرض، فيُسقطه مرشّح المعرض ويولّد الخادم معرّفًا. (وفي الإنشاء أو التحديث الكامل خارج معرض، يُحتفظ بمعرّف uuid السليم والفريد الذي ترسله.)
5. **الصفوف التي تتركها تُزال.** ولا يستطيع المعرض إزالة الصفوف إلا حين يتيح تحرير حقل واحد على الأقل داخل الكتلة؛ أمّا المعرض الذي لا يتيح تحرير أي حقل من الكتلة، فلا يمسّ الكتلة المخزَّنة أبدًا.
6. **تُرفض المعرّفات المكرّرة في مصفوفة واحدة** بـ `400 ENTRY_VALIDATION_ERROR` (والسبب `duplicate __id in <field>: <ids>`)، ولا يجوز لأي عملية patch أن تستهدف `__id` .

ملاحظات الإصدارات

نحتفظ هنا بتوثيق السلوك السابق الذي تغيّر، لتقتصر الأدلّة على وصف السلوك الحالي:

- **دقة النسبة المئوية في القيم المحسوبة (1.8).** كانت القيم المُدخلة وحدها تلتزم بدقة حقل النسبة المئوية، لأنّ التقريب كان يجري عند إدخال المستخدم للرقم. أمّا النتيجة المحسوبة فكانت تُعرض مباشرةً بدقة الفاصلة العائمة الكاملة، لذا كان حقل مضبوط على 0 أو 1 منزلة عشرية قد يعرض قيمة مثل `33.333333333333336%`. ويشمل الإصلاح نمطَي عرض النسبة المئوية كليهما – تسمية النسبة المئوية العادية وتسمية *شريط التقدّم مع النسبة المئوية*؛ ولا يتأثّر ملء شريط التقدّم نفسه.
- **الحقول الفرعية المطلوبة داخل كتلة مخفية (1.10).** قبل هذا الإصلاح، كان الحقل الفرعي المطلوب داخل كتلة مخفية قد يمنع الحفظ مع ظهور الخطأ «Field is required!»، مع أنّ الحقل لم يكن يُعرض للمستخدم قطّ، ولم يكن لديه سبيل إلى ملئه.
- **حقول التسلسل والإلزام (1.10).** قبل هذا الإصلاح، كان حقل التسلسل المحدَّد بوصفه مطلوبًا (عبر مفتاحه، أو قائمة الحقول المطلوبة في معرض، أو خطوة سير عمل تجعل كتلته إلزامية) يمنع **كل** عملية إنشاء مع ظهور الخطأ «Field is required!».
- **حقول الخريطة المفكوك تشفيرها (1.10).** كان حقل الخريطة المكشوف – حقل العنوان مع المحدِّد التفاعلي – يُعرض مضغوطًا إلى نحو نصف عرضه، وفق التخطيط الضيّق المخصّص للنص النائب القصير لقيمة مقنّعة أو مقفلة؛ وأصبح الآن يُعرض بعرضه الكامل المعتاد. أمّا الخريطة المقنّعة أو المقفلة فما زالت تستخدم التخطيط الضيّق.
- **النافذة المنبثقة للقيمة القابلة للتكرار.** كانت النافذة المنبثقة التي تستخدمها عروض تفاصيل المُدخل / الجدول / البطاقة لإظهار القائمة الكاملة لقيم حقل قابل للتكرار ثابتة الارتفاع، أيًّا كان عدد القيم التي تحتويها فعليًّا، فكان الحقل الذي يحتوي على قيمة أو قيمتين يفتح مربعًا فارغًا في معظمه. وأصبحت الآن تتّخذ حجمًا يناسب محتواها، مع بقاء الحدّ الأقصى نفسه. ويقتصر التغيير على العرض؛ ولم تتغيّر بنية القيمة المخزَّنة.
- **حقول الربط والتشفير.** لا يعرض المصمِّم مفتاح التشفير على حقل الربط (وكان توثيق سابق يذكر أنّه يعرضه)، إذ تُنسخ الراية من الحقل المصدر.

أنواع الحقول الداخلية

هذه الصفحة موجّهة إلى **مطوّري التكامل والمطوّرين** الذين يصادفون أنواع الحقول هذه في تعريف كيان يقرؤونه عبر الـ API، أو في كيان توفّره المنصة، أو في حزمة. ولا يظهر أيٌّ منها في قائمة الحقول في مصمِّم الكيان، فلا يستطيع الباني إضافتها؛ إذ تُستخدم في كيانات المنصة نفسها وشاشات الإعداد الخاصة بها.

> **تبني كيانًا من الواجهة؟**
>
> تجد أنواع الحقول التي يستطيع الباني إضافتها في اختر نوع الحقل [بناء التطبيقات]، وتجد شرحًا لطريقة تخزين الحقول في مرجع إعدادات الحقول.

النوع	نوع التخزين	سبب كونه داخليًا
JSON	`ObjectJson`	مسجَّل على أنه مخفي
محدد الاستعلام	`Entity`	عنصر تحكّم تستخدمه شاشات الإعداد، وليس مسجَّلًا بوصفه نوع حقل
أيقونة SVG	`SVGIcon` (`Since 1.8`)	مسجَّل على أنه مخفي
مرجع الصفحة	`Entity`	مسجَّل على أنه مخفي
الخيار الرقمي	`Number`	مسجَّل على أنه مخفي

وتعمل إعدادات هذه الحقول – **قفل الحقل**، و**إخفاء الحقل**، و**مطلوب**، و**إظهار في القائمة** – كما هو موضّح في الإعدادات المشتركة بين الحقول [بناء التطبيقات].

JSON

يتيح حقل JSON تخزين كائن مُهيكل بأي بنية في المُدخل على هيئة JSON خام، ويوفّر محرّر JSON بسيطًا ليتمكّن المستخدمون المتقدمون من حفظ بيانات لا تناسب أنواع الحقول القياسية، مثل كتلة إعدادات أو كائن متداخل يُرجِعه أحد التكاملات.

الاسم	الوصف	نوع البيانات
JSON	حقل يخزّن كائنًا بأي بنية على هيئة JSON خام.	ObjectJson

- حقول JSON **غير قابلة للبحث ولا للاستعلام**، فلا يمكن استخدام محتوياتها في مربع البحث ولا في معايير الاستعلام؛ لذا خزّن البيانات التي تحتاج إلى الترشيح بحسبها في حقول مستقلّة.
- محرّر JSON الخام متاح على **الويب** فقط؛ أما على **الجوّال** فلا تعرض حقول JSON (ObjectJson) أي محتوى في أيٍّ من وضعَي التحرير والعرض.
- لا يقبل حقل JSON قيمة افتراضية متقدمة (تعبيرًا)، ولا يمكن أن يكون فريدًا.
- عند عرضه في قائمة أو في عرض التفاصيل، يمكن فتح محتوى JSON في مربع حوار مخصّص لمعاينته بتنسيق يسهل قراءته.
- إعداداته: **قفل الحقل** (مع تعبير برمجي اختياري يحدّد شرط القفل)، و**إخفاء الحقل** (مع تعبير برمجي)، والإظهار في القائمة.

محدد الاستعلام

يتيح حقل محدد الاستعلام للمُدخل الإشارة إلى استعلام بيانات محفوظ، ويُستخدم عندما يحتاج مُدخلٌ إلى مرجع لتعريف مرشِّح أو معايير مخزَّنة، لتغذية قائمة ديناميكية مثلًا أو مصدر يقبل معاملات في موضع آخر من الإعدادات.

الاسم	الوصف	نوع البيانات
محدد الاستعلام	حقل يتيح اختيار استعلام بيانات محفوظ.	Entity

- يتاح بوصفه عنصر تحكّم تستخدمه المنصة في شاشات إعداد معيّنة، لكنه غير مسجَّل في منتقي الحقول القياسي عند تصميم كيان.
- عند العرض، يُستخرج اسم الاستعلام المختار ليكون عنوان الحقل.
- إعداداته: **قفل الحقل** (مع تعبير برمجي اختياري يحدّد شرط القفل)، و**إخفاء الحقل** (مع تعبير برمجي).

أيقونة SVG `Since 1.8`

يتيح حقل أيقونة SVG تخزين رسم متّجهي قابل للتحجيم (SVG) في المُدخل. وعلى خلاف حقل الأيقونة العادي الذي يشير إلى عنصر من مجموعة الأيقونات المدمجة في المنصة، يخزّن حقل أيقونة SVG ترميز SVG نفسه، فيظهر الرسم بوضوح عند أي حجم، ويتيح الحقل تخزين رسوم مخصّصة.

الاسم	الوصف	نوع البيانات
أيقونة SVG	حقل يخزّن رسمًا متّجهيًا قابلًا للتحجيم (SVG).	SVGIcon

- يمرّ أي محتوى SVG عبر أداة تنقية (sanitizer) قبل تخزينه أو عرضه.
- ليست له قيمة افتراضية.
- إعداداته: **قفل الحقل** (مع تعبير برمجي اختياري)، و**إخفاء الحقل** (مع تعبير برمجي)، والإظهار في القائمة.

مرجع الصفحة

يتيح حقل مرجع الصفحة للمُدخل الإشارة إلى صفحة (معرض عام)، وهو حقل مرجع متخصّص يستمدّ بياناته من صفحات المنصة، ولذلك يُستخدم لربط مُدخل بصفحة معيّنة في النظام.

الاسم	الوصف	نوع البيانات
مرجع الصفحة	حقل يتيح اختيار صفحة بوصفها مرجعًا.	Entity

- يدعم **السماح بتحديدات متعددة** (دون حدّ أدنى أو أقصى)، ويمكن أن يكون فريدًا، ويقبل قيمة افتراضية متقدمة (تعبيرًا).
- إعداداته: **قفل الحقل** (مع تعبير برمجي اختياري)، و**إخفاء الحقل** (مع تعبير برمجي)، والإظهار في القائمة.

الخيار الرقمي

يتيح حقل الخيار الرقمي للمُدخل اختيار قيمة واحدة من مجموعة محدّدة مسبقًا من الخيارات المرقّمة؛ إذ يختار المستخدم تسمية من قائمة، ويخزّن الحقل الرقم المقابل لها. ويفيد ذلك حين يحتاج المُدخل إلى اختيار رقمي مقيّد، مثل مستوى أولوية أو تقييم، بدلًا من رقم يُدخَل بحرّية.

الاسم	الوصف	نوع البيانات
الخيار الرقمي	حقل يتيح اختيار خيار واحد من مجموعة من الخيارات المرقّمة.	Number

- بما أن القيمة المخزَّنة رقم، يظلّ ترشيح الحقل وفرزه ممكنَين، كما في حقول الرقم الأخرى.
- يمكن أن يكون فريدًا، ويدعم **السماح بتحديدات متعددة** (دون حدّ أدنى أو أقصى)، ويقبل قيمة افتراضية متقدمة (تعبيرًا).
- إعداداته: **قفل الحقل** (مع تعبير برمجي اختياري)، و**إخفاء الحقل** (مع تعبير برمجي)، و**مطلوب** (مع تعبير برمجي اختياري يحدّد الشرط)، والإظهار في القائمة.

خادوم MCP لوكلاء الذكاء الاصطناعي

> ⚠ **وصول مبكر — `osos-mcp` 0.2.0**
>
> هذه الوحدة في **طور مبكر**، و**ليست صالحة بعدُ لعمل مستقلّ يُعوَّل عليه**. استعملها مساعدًا تحت الإشراف على قاطن غير إنتاجي.
>
> ولم يُتحقَّق بعد من أي أداة على قاطن حيّ ببيانات اعتماد حقيقيّة، إذ تعمل عُدّة الاختبار على أشكال واجهة مسجّلة، وقد لوحظ سلفًا اختلاف تلك الأشكال عن المنصّة العاملة في مواضع. وأسماء الأدوات عقدٌ عامّ لكنّها **غير مُجمَّدة** حتى الإصدار 1.0.0.
>
> **تصف هذه الصفحة `osos-mcp` 0.2.0.** راجع سجلّ تغييرات الوحدة قبل التعويل على شيء هنا مع بناء أحدث.

يتيح خادوم MCP لوكيل ذكاء اصطناعي أن يعمل مع أسس عبر واجهتها البرمجيّة — فيستكشف نموذج بيانات القاطن، ويستعلم عن المُدخلات ويكتبها، ويقرأ الأذونات، ويقود سير العمل — مستعملًا ميفاق سياق النموذج، وهو ما تتحدّثه معظم أدوات الوكلاء.

ما هو

تخدم بوّابة أسس عدّة مئات من مسارات HTTP عبر خدماتها؛ نحو أربعمئة منها عمليّات موصّفة تصفها مستندات OpenAPI الخاصّة بالخدمات، وهي ما يستطيع هذا الخادوم بلوغه. ولا يُفيد تسليم الوكيل أربعمئة تعريف أداة: فالأوصاف وحدها تزحم المهمّة. لذا يعرض الخادوم فهرسًا صغيرًا مصوغًا على هيئة المهامّ — نحو ثلاثين أداة تغطّي السطح عالي القيمة — مع زوج استكشاف يبلغ ما بقي عند الطلب.

ويعمل بشكلين:

- **stdio** — عمليّة محلّيّة يُطلقها وكيل مكتبي، تُهيَّأ من ملفّ أو من البيئة.
- **HTTP المتدفّق** — خدمة منشورة تحمل فيها كل جلسة قاطنها وبيانات اعتمادها، فتخدم عمليّة واحدة قاطنين كُثرًا.

وهو *عميل* للمنصّة: لا يحتفظ ببيانات، ولا يضيف أذونات، ولا يستطيع شيئًا لم تكن بيانات الاعتماد المُعطاة له تستطيعه أصلًا.

النضج

اقرأ التنبيه أعلاه قبل التخطيط لأي عمل حول هذه الوحدة، وخاصّةً:

الإصدار	0.2.0
الاستقرار	وصول مبكر. وقد تتغيّر أسماء الأدوات قبل 1.0.0.
التحقّق الحيّ	لا شيء بعد. تعمل العُدّة على أشكال مسجّلة.
يصلح لـ	استعمال تفاعلي تحت الإشراف على قاطن غير إنتاجي.
لا يصلح لـ	أتمتة بلا إشراف، أو بيانات إنتاج، أو أي سياق تضرّ فيه إجابة خاطئة صامتة.

وتوثيق هذه الوحدة – هنا وفي برنامج الاعتماد – يذكر الإصدار الذي يصفه. فحين تُحدَّث الوحدة يكون ذلك الإصدار هو ما يُخبرك أنّ الصفحة ما زالت تنطبق.

اشتراط الرخصة

يشترط خادوم MCP رخصة أسس صالحة، ويرفض الطلبات من دونها. انظر رخصتك [الإدارة › رخصتك وحدود خطتك].

ما يستطيعه

مجموعًا بحسب ما يسعى إليه الوكيل:

المجال	أمثلة
الاتّصال	فحص إمكان الوصول وبيانات الاعتماد؛ وسرد لغات القاطن.
استكشاف المخطّط	سرد التطبيقات والكيانات؛ ووصف كيان بحقوله وعلاقاته ومعارضه وأذوناتك الفعليّة في نداء واحد.
قراءة البيانات	عدّ المُدخلات والاستعلام عنها وجلبها، عبر كيان أو عبر معرض.
استكشاف العمليّات	البحث في نحو أربعمئة عمليّة واستدعاء إحداها بالاسم.
الملفّات	رفع مرفق؛ وقراءة بيانات ملفّ.
الأشخاص والأذونات	سرد المستخدمين والمجموعات؛ وقراءة الأذونات الفعليّة؛ وفحص جواز إجراء.
سير العمل	قراءة حالة سير العمل والتصرّف بها؛ والتعليقات؛ والإشعارات؛ وانتظار المهامّ الطويلة.

وعمليّات الكتابة محروسة: فهي تستوجب خطوة تأكيد صريحة بحسب التهيئة، ولا تُسجَّل إطلاقًا في جلسة للقراءة فقط.

انظر مرجع الأدوات.

إلى أين بعد ذلك

- البداية السريعة – تشغيله وتهيئته وجعله يُجيب في دقائق.
- مرجع الأدوات – الفهرس.
- استدعاء واجهة HTTP – الواجهة التي تحته، إن كنت تبني عميلًا خاصًّا بك بدل استعمال هذا.

البداية السريعة مع MCP

> ⚠ **وصول مبكر – `osos-mcp` 0.2.0**
>
> تصف هذه الصفحة `osos-mcp` **0.2.0**، وهو في وصول مبكر وغير صالح لعمل مستقلّ يُعوَّل عليه. استعمل قاطنًا غير إنتاجي.

من الصفر إلى وكيل يُجيب عن سؤال يخصّ قاطنك.

قبل أن تبدأ

- Node 24 وpnpm.
- العنوان **العلني** للقاطن. فاسم مضيفه هو ما تُعرَف به أسس على القاطن، ولذا لن ينجح عنوان IP ولا `localhost` ولا عنوان داخل العنقود – ويرفضها الخادوم بتفسير بدل أن تكتشفها خطأً غامضًا.
- **رمز تطبيق**. يُنشئ مسؤول القاطن **مستخدم تطبيق** (مستخدمًا من النوع *تطبيق* – انظر إضافة مستخدم التطبيق [الإدارة › أنشئ رمز API لعملية تكامل])، فترسل المنصّة الرمز الجديد بالبريد إلى المسؤول الذي أنشأه؛ ولا يُعرض على الشاشة أبدًا ولا يمكن قراءته لاحقًا (فإن ضاع فأعِد توليده). ويحمل الرمز أذونات مستخدم التطبيق ذاك، فأعطِ كل تكامل رمزه الخاصّ.
- قاطن ذو **رخصة صالحة**. انظر رخصتك [الإدارة › رخصتك وحدود خطتك].

التنصيب والبناء

احصل على حزمة `osos-mcp` من مورّد أسس – فهي غير منشورة في سجلّ عام – ونفّذ ما يلي في مجلّدها:

```
pnpm install
pnpm build
```

تهيئة قاطن

تُحلّ التهيئة عبر ثلاث طبقات – قيم في الطلب، ومتغيّرات بيئة، وملفّ تعريف مسمّى – وتُحلّ **حقلًا حقلًا**، فيستطيع النشر تثبيت القاطن مركزيًّا وأخذ بيانات الاعتماد من كل طلب.

وأصغر تهيئة عاملة قيمتان:

```
OSOS_BASE_URL=https://your-tenant.example.com
OSOS_APP_TOKEN=<الرمز الذي نسخته>
```

ولما هو أبعد من تجربة سريعة، فضّل **ملفّ تعريف** مسمّى، فيصير ما تتناقله اسمًا لا سرًّا. ويستطيع ملفّ التعريف أيضًا قراءة سرّه من متغيّر بيئة أو من مدير أسرار بدل تخزينه.

وثمّة قاعدتان تستحقّان المعرفة لأنّهما ليستا «آخر قيمة تفوز» عمدًا:

- بيانات الاعتماد **مقيّدة بالقاطن** الذي هُيّئت له. فتسمية قاطن مغاير في الطلب لن تُعيد استعمال بيانات اعتماد تخصّ غيره في صمت.

- والقيود تتراكم فحسب. فللطلب أن يجعل الجلسة **أشدّ** تقييدًا، ولا يمكنه أن يجعلها أقلّ.

وصله بعميل

تأخذ معظم أدوات الوكلاء كتلةً كهذه:

```
{
  "mcpServers": {
    "osos": {
      "command": "node",
      "args": ["/path/to/osos-mcp/dist/index.js"],
      "env": {
        "OSOS_BASE_URL": "https://your-tenant.example.com",
        "OSOS_APP_TOKEN": "…",
        "OSOS_LOCALE": "ar",
        "OSOS_READ_ONLY": "true"
      }
    }
  }
}
```

والبدء بالقراءة فقط هو الافتراض المعقول لأوّل جلسة: فأدوات الكتابة عندئذٍ غير مُسجَّلة إطلاقًا، فلا يستطيع الوكيل محاولة إحداها.

التحقّق

اطلب من الوكيل تشغيل فحص الاتّصال. فهو يُجيب عن أربعة أسئلة منفصلة، وهذا ما يجعل العطل قابلًا للتشخيص:

١. **هل القاطن قابل للوصول؟** – لا يحتاج بيانات اعتماد، فيعزل العنوان الخاطئ. ٢. **ما لغات القاطن؟** – لا يحتاج بيانات اعتماد أيضًا، ويؤكّد أنّك بلغت القاطن الذي قصدته. ٣. **هل قُبلت بيانات الاعتماد؟** ٤. **هل الرخصة صالحة؟**

فالعنوان الأساسي الخاطئ يُخفق في الأوّل. والرمز الفاسد ينجح في الأوّلين ويُخفق في الثالث. ومشكلة الرخصة تنجح في الثلاثة الأولى وتُخفق في الرابع. ثمّ جرّب «اسرد التطبيقات» و«صِف الكيان».

إن أخفق شيء

العَرَض	السبب الأرجح
رفض قبل تنفيذ أي شيء مع ذكر الرخصة	ليس للقاطن رخصة سارية. تواصل مع المورّد؛ انظر رخصتك [الإدارة › رخصتك وحدود خطتك].
تعذّر التعرّف على القاطن	العنوان الأساسي عنوان IP أو `localhost` أو عنوان داخل العنقود، أو أنّ اسم المضيف ليس المسجَّل.
يُخفق التوثّق مع أنّ القاطن قابل للوصول	الرمز خاطئ أو منتهٍ عمليًّا أو مؤرشف. ولاحظ أنّ المنصّة تَخبَأ نتائج التحقّق دقائق، فالرمز المُنشأ قبل ثوانٍ قد لا يعمل بعد.
قراءة تُعيد قائمة فارغة لم تتوقّعها	يتطلّب سرد مُدخلات الكيان صلاحيّة **التحرير** لا الاطّلاع. والنتيجة الفارغة و«غير مسموح» قد تبدوان متطابقتين.
رفض تغييرات التهيئة	قد لا تشمل الرخصة الإدارة. تواصل مع المورّد.

مرجع أدوات MCP

> ⚠ **وصول مبكر — `osos-mcp` 0.2.0**
>
> الفهرس أدناه هو ما يشحنه `osos-mcp` **0.2.0** — ٢٧ أداة. وأسماء الأدوات عقدٌ عامّ لكنّها **غير مُجمَّدة** حتى 1.0.0، فقد يُضيف بناء أحدث إلى هذه القائمة، وقد يُعيد التسمية داخلها قبل 1.0.0. ويحمل مستودع الوحدة نفسه المرجع المولّد المُحدَّث دائمًا بتفصيل الوسائط كاملًا.

كيف تقرأ هذا

كل أداة إمّا **قراءة** وإمّا **كتابة**. والقراءات آمنة الاستدعاء بحرّيّة. أمّا الكتابات فتُعدّل القاطن، وتستوجب خطوة تأكيد صريحة بحسب التهيئة؛ وفي جلسة للقراءة فقط لا تُسجَّل إطلاقًا، فالفهرس الذي يراه الوكيل بيانٌ صادق عمّا تستطيعه تلك الجلسة.

وكل ما هنا خاضع لحدّين لا تتحكّم بهما الوحدة: أذونات بيانات الاعتماد نفسها، ورخصة [الإدارة › رخصتك وحدود خطتك] القاطن.

الاتّصال

الأداة	النوع	الغرض
فحص الاتّصال	قراءة	إمكان الوصول، ولغات القاطن، وبيانات الاعتماد — مُبلَّغ عنها منفصلةً فيغدو العطل قابلًا للتشخيص.
إصدار المنصّة	قراءة	إصدار الخلفيّة. لا يحتاج بيانات اعتماد؛ وهو أرخص برهان على صحّة العنوان الأساسي.

استكشاف المخطّط

الأداة	النوع	الغرض
سرد التطبيقات	قراءة	التطبيقات (الوحدات) في المساحة.
سرد الكيانات	قراءة	الكيانات، ويمكن حصرها بتطبيق واحد.
وصف الكيان	قراءة	الكيان وحقوله وعلاقاته ومعارضه وأذوناتك الفعليّة — في نداء **واحد**. وهي أكفأ طريقة لفهم كيان.
سرد حقول الكيان	قراءة	أنواع الحقول، ورايات اللزوم والتفرّد، وأهداف المراجع، وخيارات التصنيف.
سرد المعارض	قراءة	معارض كيان أو تطبيق.
وصف المعرض	قراءة	الأعمدة والمرشّح الافتراضي والمرشّحات السريعة والأذونات.

قراءة البيانات

الأداة	النوع	الغرض
عدّ المُدخلات	قراءة	رخيص. ويجدر استدعاؤه قبل استعلام واسع.
الاستعلام عن المُدخلات	قراءة	ترشيح وبحث نصّي حرّ وإسقاط وترقيم.
جلب مُدخل	قراءة	مُدخل واحد بمُعرّفه.
الاستعلام عبر معرض	قراءة	يُطبّق قواعد وصول المعرض ومجموعة أعمدته.
سرد المُدخلات المرتبطة	قراءة	اجتياز علاقة انطلاقًا من مُدخل.

والترقيم بحجم صفحة وإزاحة، ولا مؤشّرات، ويحدّ الخادوم حجم الصفحة كي لا يُغرق استعلام واسع الوكيل.

استكشاف العمليّات

الأداة	النوع	الغرض
البحث في العمليّات	قراءة	البحث في نحو أربعمئة عمليّة بالاسم أو المسار أو الملخّص. ويطوي إصدارات العمليّة الواحدة على أحدثها افتراضيًّا.
استدعاء عمليّة	*متغيّر*	استدعاء إحداها بالاسم. وتُحسب خطورتها لكل عمليّة، فتنطبق القواعد ذاتها كما على أداة مُسمّاة.

وهاتان هما سبب بقاء الفهرس صغيرًا. ويأخذ الاستدعاء **اسم** عمليّة لا عنوانًا، فالطريقة والمسار يأتيان من وصف الواجهة المسجّل، ولا يمكن توجيهه إلى غير ذلك.

الملفّات

الأداة	النوع	الغرض
رفع ملفّ	كتابة	الرفع من مسار أو من محتوى مُضمَّن؛ ويُعيد المُعرّف لإرفاقه بمُدخل.
جلب ملفّ	قراءة	البيانات ورابط تنزيل. ولا يُدفَع المحتوى الثنائي إلى الوكيل قطّ.

الأشخاص والأذونات

الأداة	النوع	الغرض
سرد المستخدمين	قراءة	دليل الأشخاص وحالة كل مستخدم.
سرد المجموعات	قراءة	المجموعات وأعضاؤها ونوع المجموعة.
جلب الأذونات	قراءة	الوصول الفعلي للمستخدم الحالي أو لمستخدم مُسمّى.
فحص الوصول	قراءة	هل يجوز إجراء واحد على مُدخل واحد – وهي أرخص طريقة للإجابة عن «هل يجوز لي؟» قبل الكتابة.

والصلاحيّات سلّم مِنَح لا نطاقات، وأحد مستوياته رفضٌ قاطع يعلو على أي منحة أخرى. ووُجدت **أداة فحص الوصول** كي يشرح الوكيل الرفض بدل أن يُعيد المحاولة.

سير العمل

الأداة	النوع	الغرض
جلب حالة سير العمل	قراءة	الخطوة الراهنة والإجراءات المتاحة والمالكون.
تنفيذ إجراء سير عمل	كتابة	التصرّف بسير عمل مُدخل.
سرد التعليقات	قراءة	تعليقات مُدخل مع تحويل الإشارات إلى أسماء.
إضافة تعليق	كتابة	نشر تعليق.
سرد الإشعارات	قراءة	إشعارات بيانات الاعتماد نفسها.
انتظار مهمّة	قراءة	استقصاء مهمّة طويلة حتى تمامها. ولا نداءات راجعة؛ فالاستقصاء هو النمط.

وإجراء سير العمل المرفوض كثيرًا ما تمنعه *حالة* المُدخل الراهنة لا الأذونات – وتُبلّغ الأدوات بهذا الفرق، فالخلط بينهما يُرسلك للبحث عن وصول تملكه سلفًا.

المُدخلات والقوائم: مرجع السلوك

هذه الصفحة موجّهة إلى مطوّري التكامل والمطوّرين الذين يحتاجون إلى فهم السلوك التقني الدقيق الذي تستند إليه صفحات المستخدم النهائي الخاصة بالمُدخلات والقوائم. وهي تجمع في موضع واحد التفاصيل التقنية التي لا تتناولها تلك الصفحات، وتحيل إلى مرجع الواجهة البرمجية لكل مجال. وفيما يلي صفحات المستخدم النهائي: تصفّح قائمة وغيّر طريقة عرضها [استخدام أسس]، والبحث والترشيح والفرز والتجميع [استخدام أسس › البحث في قائمة وترشيحها وفرزها وتجميعها]، وخصّص المعرض لنفسك [استخدام أسس]، وإنشاء المُدخلات وتعديلها [استخدام أسس]، والقوائم [استخدام أسس › القوائم: اجمع المُدخلات معًا]، وتصدير قائمة [استخدام أسس].

أنواع العرض

تُخزَّن أوجه المعرض («الأوجه») بوصفها أنواع معارض ذات قيم رقمية:

القيمة	النوع	الاسم على الشاشة
`0`	`TableView`	**وجه الجدول**
`1`	`CardView`	**وجه البطاقات**
`2`	`CalenderView`	**وجه التقويم**
`3`	`KanbanBoard`	**وجه لوحة المهام**
`4`	`Timeline`	**عرض الشريط الزمني**
`5`	`TreeView`	**عرض الشكل شجري**
`6`	`MiniCard`	غير موجود في قائمة طرق العرض في بوابة الويب؛ تستخدمه قطع العدّادات، ويعرضه تطبيق الجوّال على هيئة مساحة فارغة.

لا تعرض بوابة الويب وجهًا إلا إذا كان مفعّلًا في المعرض وكان الكيان مناسبًا له: فوجه البطاقات يتطلّب كيانًا من نوع ADT، والتقويم حقل تاريخ، ولوحة المهام حقل لوحة مهام، والشريط الزمني إعدادًا للشريط الزمني، والعرض الشجري حقل مرجع شجري. ولا يُعرض الشريط الزمني ولا العرض الشجري ما دامت قائمة المسوّدات معروضة.

البحث

- لكل مُدخل فهرس واحد للنص الكامل يغطّي جميع الحقول الموسومة **قابل للبحث**؛ وتُسهم فيه حقول التصنيف **بتسميات** خياراتها، وحقول المرجع **بعنوان** المُدخل المرتبط، والمجموعات المرجعية (lookup) بقيمة عرضها، والنص المترجَم بكل لغة مُعدّة.
- يُبنى الفهرس النصّي بالإعداد `default_language: 'none'`، دون تجذيع أو إزالة لكلمات التوقّف، وتقتصر المطابقة على الكلمات الكاملة دون تمييز بين الأحرف الكبيرة والصغيرة.
- تتبع صيغة البحث `$text` في MongoDB: يُجمع بين الكلمات المفصولة بمسافات بـ **OR**، وتُطابق السلسلة المحاطة بعلامات اقتباس بوصفها عبارة مطابقة تمامًا، وتستبعد الشرطة `-` الكلمة التي تَرِد في أولها.
- الفهرسة غير متزامنة: فعند الإنشاء أو التحديث، تُضاف مهمة إعادة فهرسة المُدخل إلى الصفّ، ويؤدي تغيير مُدخل تشير إليه مُدخلات أخرى إلى إعادة فهرستها حين يكون حقل المرجع قابلًا للبحث؛ لذا قد لا يُعثر على المُدخل الجديد أو المعدَّل إلا بعد لحظات.

- لا تُفهرس الحقول المشفّرة أبدًا.
- تطابق منتقيات المراجع والمستخدمين النصَّ المكتوب مع العنوان الرئيسي بكل لغة من لغات القاطن، دون تمييز بين الأحرف الكبيرة والصغيرة، وفي أي موضع من العنوان، مع تهريب المحارف الخاصة في التعابير النمطية (منذ 1.6).

المسار الكامل: المحرك: الفهرسة والبحث.

المرشّحات

العوامل وتسمياتها

عامل المحرّك	التسمية في لوحة الترشيح
`Equal` / `NotEqual`	**= / !=**
`GreaterThan` / `GreaterThanOrEqual`	**> / =>**
`LessThan` / `LessThanOrEqual`	**< / =<**
`In` / `NotIn`	**في / ليس في**
`Contains` / `DoesNotContain`	**يحتوي / لا يحتوي**
`StartsWith` / `DoesNotStartWith`	**يبدأ ب / لا يبدأ ب**
`Like` / `IsNotLike`	**مثل / ليس مثل**
`IsEmpty` / `IsNotEmpty`	**فارغ / مملوء**
`Under`	**اسفل**
`And` / `Or` (المجموعات)	**كل الشروط محققة / بعض الشروط محققة**

عوامل الحقول: `Size` (**عدد القيم**)، و`Sum`، و`Avg` (**المتوسط**)، و`Min`، و`Max`، و`First` (**القيمة الأولى**)، و`Last` (**القيمة الأخيرة**)، و`Any` (**أي قيمة**، وهو الافتراضي). ومصادر القيم: `Fixed`، و`Pointer` (مثل `@me` و`@today`)، و`Dynamic` (**حقل**)، و`Expression`، و`Parameter`.

- لا تميّز العوامل `Contains` و`StartsWith` و`Like` والعوامل النافية لها بين الأحرف الكبيرة والصغيرة.
- تقبل `In` / `NotIn` قيم المراجع وقيم الخيارات مباشرةً.
- تختبر `IsEmpty` / `IsNotEmpty` **الوجود** أيضًا (منذ 1.8): فلا يطابق `IsNotEmpty` إلا حقلًا موجودًا ويحمل قيمة. ويُعدّ الحقل داخل كتلة متكرّرة بلا صفوف غائبًا، ومن ثمّ فارغًا. ويسري الاختبار على الكتلة كلها، لا على كل صفّ.
- يقصر `Under` نتائج الترشيح في كيان شجري على أحفاد عقدة معيّنة (منذ 1.2).

النموذج الكامل: الاستعلامات والمرشّحات [بناء التطبيقات] · صيغ المعايير ومسارات الحقول · المحرك: الاستعلام والتجميع.

كيف يُجمَّع طلب القائمة

1. يُدمج المرشّح المنظَّم مع أي معايير مطبَّقة من استعلام محفوظ باستخدام **And**، ثم تُترجم الشروط المدمجة إلى استعلام قاعدة بيانات.
2. تُضاف مرحلة مطابقة النص الكامل أولًا إذا أُرسل مصطلح بحث وكان المعرض يسمح بالبحث.
3. يُترجم الفرز، ويُلحَق به معرّف المُدخل للفصل بين القيم المتساوية.
4. تُطبَّق قيود المعرض (الشروط المحذوفة، وفرض المُدخلات النشطة فقط، ونحو ذلك) أخيرًا لضمان عدم تجاوزها.

إعدادات المعرض التي تحذف الشروط

حين يعطّل المعرض قدرةً ما، تُحذف الشروط المقابلة لها من الطلب **بصمت** بدلًا من رفضه:

إعداد المعرض	الأثر عند تعطيله
السماح بالبحث	يُتجاهل مصطلح البحث.
السماح بالترشيح على الخطوة الحالية	لا يُطبَّق الترشيح على خطوة سير العمل الحالية.
السماح بالترشيح على الخطوات السابقة	لا يُطبَّق الترشيح على السجل التاريخي لسير العمل.
السماح بترشيح البيانات الوصفية	لا يُطبَّق الترشيح على معلومات الإنشاء أو التحديث ولا على معرّف المُدخل.
السماح بترشيح تفاصيل البيانات الوصفية	لا يُطبَّق الترشيح على من أنشأ المُدخل أو حدّثه.
السماح بمرشّح المستخدم الحالي	لا يُطبَّق الترشيح بمعيار «يخصّني».
السماح بمرشّح مجموعات المستخدم الحالي	لا يُطبَّق الترشيح بمعيار «يخصّ مجموعاتي».
إظهار المتقادمة / غير الصالحة	تُستبعد المُدخلات المتقادمة أو غير الصالحة قسرًا.
إظهار المُدخلات المؤرشفة	لا تُعاد إلا المُدخلات النشطة.

انظر معارض البيانات وأوجهها [بناء التطبيقات].

الحقول الحسّاسة والمشفّرة في المعايير

- أي معيار يتناول حقلًا حسّاسًا لا يملك المستدعي منحة عليه **يُرفض** بخطأ، ولا يُحذف بصمت؛ ويشمل هذا التحقّق أيضًا المسارات الفرعية المترجمة (مثل `salary.en`) ومسارات المراجع الفرعية (مثل `manager.dataEntry`).
- منذ 1.10، يبقى الحقل الحسّاس الذي يملك المستدعي منحة عليه في قائمة حقول المستخدم النهائي حتى قبل فتح جلسة كشف.
- في منطقة الإدارة (تبويب **الاستعلامات** في الكيان، وشروط التفاعلات)، يعرض منتقي الحقول كل حقل حسّاس بصرف النظر عن منحة الباني نفسه، لأن ذكر حقل في معيار لا يكشف شيئًا. ويكون مربع قيمة المرشّح قابلًا للتحرير للحقل الحسّاس على الويب والجوّال.
- لا يمكن لأي مستدعٍ ترشيح حقل مشفّر أو البحث فيه أو الفرز به مطلقًا.

انظر الحقول الحسّاسة: الاستعلام والترشيح والاستعلامات: الحقول المشفّرة والحسّاسة [بناء التطبيقات › الاستعلامات والمرشّحات].

الفرز والترقيم

- يُعبَّر عن الفرز بالصيغة `"<field> ASC"` أو `"<field> DESC"`؛ وتُطبَّق بنود الفرز المتعدّدة حسب أولويتها. ويُلحَق معرّف المُدخل دائمًا للفصل بين القيم المتساوية، مما يضمن استقرار الترقيم بالإزاحة. وتُفرز قائمة المُدخلات افتراضيًا ترتيبًا تنازليًا حسب تاريخ التعديل.
- تُقسَّم طلبات القوائم إلى صفحات باستخدام `limit` و`skip` (أو `offset`).
- **القراءات المقيّدة بمعرض محدودة السقف** (منذ 1.8): يُطبَّق السقف على الطلب عبر معرض إذا لم يتضمّن `limit`، وتُخفَّض قيمة `limit` إليه إذا تجاوزته. ويحدّد `MAX_ENTRIES_RESULT_LIMIT` هذا السقف، وقيمته الافتراضية 10,000. انظر قراءة البيانات.
- تُستمدّ معاملات الاستعلام المحفوظ ذي المعاملات، عند تطبيقه مرشّحًا سريعًا (`appliedQuickFilters`)، من الطلب (منذ 1.8).

انظر مرجع التكوين لمعرفة متغيّر السقف.

التجميع والمجاميع

- لا تشمل الحقول القابلة للتجميع الحقولَ متعدّدة القيم، والحقول المتكرّرة، والحقول داخل كتلة متكرّرة، وحقول `_metaData.*`، وحقول القيم المرجعية (lookup) التي لا تكون قيمتها المخزّنة مرجعًا بسيطًا؛ أما المرجع أحادي القيمة فقابل للتجميع.
- يعرض منتقي التجميع لدى المستخدم النهائي الحقول الحسّاسة، لكنه يُخفي المشفّرة. ومستوى التجميع على حقل حسّاس أو مشفّر **يُقبل ويُطوى** في مجموعة واحدة مفتاحها `****` لكل مستدعٍ، بصرف النظر عن المنح؛ ويُطوى كذلك المستوى المخزَّن على حقل مشفّر. ولا يزال تلميح المعلومات في المنتقي يقول إن الحقول المحميّة غير متاحة، وهو ما لم يعد يطابق هذا السلوك.
- يُحفظ التجميع والفرز الخاصان بالمستخدم في مخزن الجلسة في المتصفّح طوال الجلسة؛ ويؤدي تغيير إعداد التجميع في المعرض إلى إسقاط المستويات التي أضافها المستخدم بنفسه.

التفاصيل الكاملة: المجاميع والتجميع [بناء التطبيقات].

الاستعلامات المحفوظة

- يجب أن يكون اسم الاستعلام فريدًا داخل الكيان؛ ويُرفض الاسم المكرّر بالخطأ `ITEM_DUPLICATE`.
- يحمل نموذج الاستعلام قيمة `visibility` هي `Private` أو `Organization` أو `Public`، و`parameters` اختيارية. ولا يطلب مربع الحوار **حفظ** في البوابة (حفظ الاستعلام) إلا اسمًا؛ ويُنشأ الاستعلام بقيمة `relatedTo` تشير إلى المستخدم الحالي والمعرض الحالي، وبذلك يُدرج تحت المعرض في الشريط الجانبي لذلك المستخدم.
- يتطلّب الحفظ عبر معرض تفعيل **السماح بالاستعلامات المحفوظة** في المعرض؛ وإلا فشل الطلب بالخطأ `SaveQueryNotAllowedException`. وتُخفي البوابة أيضًا **حفظ** الاستعلام عن المستخدمين المجهولين وفي قائمة المسوّدات.
- يؤدي تطبيق استعلام محفوظ له معرّف ثم الحفظ مرة أخرى إلى تحديثه؛ أما حفظ مرشّح مشتقّ من استعلام ذي معاملات فينشئ استعلامًا جديدًا.

الواجهة البرمجية: مرجع DataQuery.

الكتابة

- **التحديث الكامل والتعديل الجزئي.** تحفظ البوابة النموذج بتحديث كامل للقيم المتغيّرة، وتحفظ التعديلات المضمّنة بصيغة JSON Patch (عمليات على كل حقل على حدة).
- **الإنشاء الدفعي** (`POST /v1/entities/{entityId}/entries/batch`) ينشئ مُدخلًا رئيسيًا واحدًا مع مُدخلات مرتبطة في استدعاء واحد، ويربط كل مُدخل مرتبط بالرئيسي عبر الحقل الهدف لعلاقته. ويتطلّب تنفيذه عبر معرض إتاحة العملية `AllowCreateNewEntries` في المعرض، ويجب أن تكون قيمة `relationId` لكل مُدخل مرتبط إحدى علاقات المعرض، و**لا** يحتاج إلى `allowSaveInvalidEntries` (إعداد الحفظ السريع)؛ فقد أُزيل هذا الشرط منذ 1.8، إذ كان يمنع خطأً استخدام معارض مُعدّة إعدادًا صحيحًا.
- **تعارضات الحفظ في البوابة.** تحمل المُدخلات قيمة `updateIdentifier`، ولا ترسلها بوابة الويب في التعديل الكامل إلا حين يكون علَم الإعداد وقت التشغيل `OPTIMISTIC_LOCKING_ENABLED` مفعّلًا (وهو معطّل افتراضيًا)؛ أما الواجهة

البرمجية نفسها فتفحص الرمز كلما أرسله العميل. ولا ترسله التعديلات المضمّنة من البوابة. انظر كتابة البيانات: تعارضات التحديث.

- **القيم المقنَّعة عند الكتابة.** تصل القيمة الحسّاسة التي لا يستطيع المستخدم قراءتها على هيئة قناع؛ ويُبقي نموذج التعديل الكامل في البوابة ذلك العنصر معطّلًا حتى يحدّده المستخدم، وعندئذ يُفرَّغ لإرسال ما يكتبه المستخدم (ولو كان فارغًا). أما المحرّر المضمّن فيُبقي العناصر المقنَّعة مفعّلة وفارغة عند البدء. ولا يُكتب القناع المُعاد دون تغيير فوق القيمة المخزّنة أبدًا. انظر حماية حقل: مسار الكتابة [بناء التطبيقات › حماية حقل].
- **سقف المُدخلات.** حين يتجاوز القاطن حدّ باقته بأكثر من 10%، يُرفض الإنشاء (بما في ذلك المسوّدات، وإرسال المسوّدة، والإنشاء الدفعي) برسالة باللغة الإنجليزية فقط تذكر الحدّ، والعدد المخزّن، وعتبة الرفض. وتُحتسب المُدخلات المؤرشفة بربع وزنها. انظر كتابة البيانات: حالات رفض الكتابة المضافة في 1.10.
- **صفوف الكتل المتكرّرة** تحمل معرّفات خاصة بها منذ الإصدار 1.10.1، وتملأ الترقية معرّفات الصفوف المحفوظة سابقًا؛ وإذا كان المعرض لا يمنح إلا بعض أعمدة الكتلة، فتُستعاد بقية الأعمدة من الصفّ المخزّن ذي المعرّف نفسه. أما في سلاسل الإصدارات 1.4 و1.6 و1.8، فتُطابَق الصفوف بالقيم غير المتغيّرة بدلًا من ذلك.
- **الكيانات الوكيلة.** يُوجَّه إنشاء المُدخل المفرد وتعديله وحفظه مسوّدةً عبر معرض وكيل إلى الكيان المضيف حين يسمح بذلك المعرض المرتبط بالوكيل؛ أما الكتابات الجماعية فتُرفض. انظر الكيانات الوهمية: الكتابة عبر التوجيه.

قوائم المستخدمين (المجموعات)

تسمّي البوابة المجموعة **قائمة**.

- **النموذج.** للمجموعة `name` مترجَم (مطلوب)، و`description`، و`icon`، و`dataEntity` واحد، و`items`، ونوع بنية، و`visibility` (`Organization` أو `Private` أو `Public`)، وإعدادات مشاركة؛ وتحتفظ كل مجموعة بمعلومات من أنشأها ومن عدّلها آخر مرة، وبقيمة `updateIdentifier`.
- **أنواع البنية.** يُتحقَّق من `Flat` و`Tree`؛ أما `Reference` و`Multi` فمعرَّفان لكنهما محجوزان. وعناصر البنية المسطّحة مراجع مُدخلات، تُزال المكرّرة منها عند الكتابة. وتحمل عناصر البنية الشجرية القيمة والفهرس والمستوى والأب، ويُتحقَّق منها (لكل عقدة قيمة ومستوى، ولا تعارض في ترتيب الأشقاء، ولا أب مفقود، ولا عقدة تكون أبًا لنفسها).
- **نوعان.** تتبع المجموعات على مستوى الكيان الكيانَ نفسه. أما المجموعات الشخصية فتُنشأ داخل معرض يسمح إعداده بمجموعات المستخدمين؛ وتُوسَم بالمعرض والمالك، وتُسرد ضمن «مجموعاتي» (لكل معرض اختياريًا) و«المشارَكة معي». وتستند حقول التصنيف إلى مجموعات التصنيف (الداخلية)، ويديرها البُناة.
- **الأسماء.** تكون فريدة داخل الكيان، ويكون اسم المجموعة الشخصية فريدًا ضمن مجموعات المالك في ذلك المعرض؛ ويُولَّد `normalizedName` إذا لم يُقدَّم، ويجب أن يكون فريدًا إذا قُدِّم (منذ 1.6).
- **العناصر والمجموعات الضمنية** يصونها النظام: فيُرفض أي تعديل يغيّر عنصرًا ضمنيًا أو يُسقطه، ولا يمكن حذف المجموعة الضمنية ولا تغيير نوع بنيتها.
- **الإضافة والإزالة.** تضيف العملية الواحدة (`set`) أو تزيل (`unset`) مُدخلات محدّدة صراحةً أو مختارة بتحديد الكل بناءً على بحث أو مرشّح؛ وإذا أُرسلتا معًا كانت الأولوية لـ `set`. وتُبلّغ الاستجابة عن `matchedRecords`، و`adjustedRecords`، و`totalRecordsBeforeOperation`، و`totalRecordsAfterOperation`؛ ويقبل إنشاء المجموعة أيضًا تحديد الكل، الذي تُحدَّد نتائجه وقت الإنشاء.
- **المشاركة.** لا تُشارَك إلا المجموعات الشخصية، ولا تُشارَك إلا مع مستخدمين أو مجموعات يستطيعون الوصول إلى معرض المجموعة؛ ولا يمكن مشاركة المعرض غير المصادَق عليه أصلًا. ويدير المالك المشاركة؛ ولا يستطيع المستلم إعادة المشاركة إلا حين تكون المجموعة مجموعة مشارَكة مقيّدة ومصادَقًا عليها، ويكون ضمن مستخدميها أو مجموعاتها المشارَكة.

- **التكرار** يمنح المجموعة معرّفًا واسمًا نموذجيًا جديدين، ويسجّل المجموعة المصدر، ويُسقط كل المشاركات، ويجعل المستدعي مالكًا لها.
- **الحذف.** لا يحذف المجموعات الشخصية إلا مالكها، من داخل معرضها؛ ولا يمكن حذف مجموعة يستخدمها كيان، ولا يؤدي حذف المجموعة إلى حذف المُدخلات أبدًا.
- **تصفّح** مجموعة عبر معرض يطبّق قواعد المعرض (المؤرشفة، وغير الصالحة، والبحث)، ويستبعد المُدخلات الوهمية ما لم يُطلب غير ذلك، ويدعم العدّ، والوسوم، والمرشّحات السريعة، والعرض السريع للعلاقات.
- **التزامن.** يُرفض الحفظ بوصفه تعارضًا إذا لم تعد قيمة `updateIdentifier` فيه مطابقة.

الواجهة البرمجية: مرجع DataCollections · قراءة مجموعة بمعرّفها.

التصدير

- **التصدير السريع** (**تصدير المدخلات**) يتطلّب قائمة مُدخلات المعرض وتفعيل إعداده **السماح بالتصدير**، ويُنشئ ملف XLSX أو CSV يضمّ كل مُدخل يطابق الطلب (البحث، والمرشّح، والمرشّحات السريعة مع قيم معاملاتها، والوسوم، وقيم الترشيح الأولي)، أو المُدخلات المقابلة لما يصل إلى 150 معرّفًا محدّدًا. ويتبع ترتيب الأعمدة ترتيب أعمدة الجدول لدى المستخدم، وإلا فترتيب حقول المعرض. ويُسمّى الملف `<list name>_<yyyy-MM-dd_HH-mm>.<ext>` بالتوقيت المحلي للمستخدم.
- **الحدود.** يُعاد ما يصل إلى `MAX_ENTRIES_RESULT_LIMIT` صفًّا (الافتراضي 10,000) بصورة متزامنة. أما عمليات التصدير الأكبر للمستخدمين المسجّلين فتجري في الخلفية، صفحةً تلو صفحة، بسقف `MAX_BACKGROUND_EXPORT_ROWS` (الافتراضي 100,000)، وتُرسل إشعار «الملف جاهز» ويتضمّن اسم الملف الذي بلغ السقف `first-<n>` . وإذا فشل التصدير في الخلفية، فلا يُرسل ملفًّا ولا إشعارًا. ولا يتلقّى المستدعون المجهولون إلا أول `MAX_ENTRIES_RESULT_LIMIT` صفًّا.
- **CSV.** بترميز UTF-8 مع علامة ترتيب البايتات، ومفصول بفواصل؛ وتُضاف فاصلة عليا قبل النص الذي يبدأ بـ `=` أو `+` أو `-` أو `@` ؛ أما الأرقام والنصوص التي تبدو رقمية فتُترك كما هي. ويُحوَّل حقل الكتلة إلى أعمدة بصيغة `Block › Field` ؛ وفي XLSX تكون له ترويسة مدموجة من صفّين. ولا تُكتب عناوين المجموعات ولا المجاميع.
- **تنسيق القيم** مشترك مع تصدير القوائم إلى XLSX: المراجع بعناوينها، والتواريخ بصيغة `dd/MM/yyyy` ، والتواريخ مع الأوقات بصيغة `dd/MM/yyyy hh:mm` ، وحقل الوقت ذو النطاق بصيغة `from - to` (منذ 1.10)، وكلمات المرور بصيغة `****` ؛ وإذا تعذّر تنسيق قيمة، تُترك خليّتها فارغة بدلًا من إفشال إنشاء الملف (منذ 1.10).
- **تصدير القوائم إلى XLSX** (`GET /v1/entities/{entity}/views/{id}/entries/export/xlsx` ، مع `fileName` و `filter` و `filterFirst` و `include_virtual` و `selectedColumns` و `collectionId`) هو ما يستدعيه **تصدير القائمة**، على الويب وفي تطبيق الجوّال.
- **التقارير** تُصدَّر بصيغة PDF أو XLSX، ولا تُصدَّر بصيغة CSV أبدًا؛ ولا تعرض الصفحة **تصدير PDF** إلا حين يكون إعدادها `allowExportPDF` مفعّلًا. انظر التقارير: نُسخ التصدير.
- **المسؤولون** يتاح لهم التصدير نفسه بصيغة Excel أو CSV في صفحة الكيان ذاتها: إنشاء كيان وتصميمه: التصدير السريع لمُدخلات كيان [بناء التطبيقات › إنشاء كيان وتصميمه].

الواجهة البرمجية: مرجع تصدير مُدخلات الكيان.

استخدم رمز API

هذه الصفحة موجّهة إلى المطوّرين الذين يربطون نصًّا برمجيًا أو خدمة أو أداة أتمتة بـ أسس دون أن يسجّل شخص دخوله. ستتعلّم كيف تحصل على رمز API، وترسله مع طلباتك، وتحصر أذوناته في أضيق نطاق، وتبدّله دون انقطاع في الخدمة.

رمز API هو أبسط وسيلة اعتماد توفّرها المنصّة: لا يتطلّب شاشة تسجيل دخول ولا تبادل رموز. أرسله في الترويسة `x-app-token` مع كل طلب، ليُنفَّذ الطلب باسم **مستخدم التطبيق** الخاص بالرمز، وبأذونات ذلك المستخدم تمامًا.

متى تستخدم رمز API

الجهة المستدعية	استخدم
شخص في متصفّح أو تطبيق جوّال	تسجيل الدخول عبر OAuth (رمز التفويض) – انظر المصادقة OAuthg
نص برمجي موثوق من الطرف الأول يحتفظ بكلمة مرور مستخدم	منحة كلمة المرور في OAuth
خدمة أو نص برمجي أو أتمتة تعمل باسمها هي	**رمز API**. لا بيانات اعتماد العميل في OAuth؛ فذلك الرمز بلا مستخدم، فلا يكاد يصل إلى شيء

وللاطّلاع على معايير الاختيار كاملةً، بما فيها ما لا يمكن التعبير عنه برمز API، انظر اختيار بيانات الاعتماد.

احصل على رمز

ينشئ مسؤول القاطن الرمز بإضافة **مستخدم تطبيق**، ثم ترسل المنصّة الرمز إلى المسؤول بالبريد الإلكتروني مرة واحدة. ولا توجد واجهة برمجية لإنشاء رمز، ولا شاشة تعرضه. اطلب من مسؤولك اتّباع الدليل أنشئ رمز API لعملية تكامل [الإدارة]، ومنح مستخدم التطبيق إذن الوصول إلى البيانات التي تحتاج إليها عملية التكامل فقط.

وستتلقّى منه:

- الرمز – سلسلة سداسية عشرية من 64 محرفًا؛
- مضيف القاطن، مثل `https://acme.example.com`؛
- ما يُسمح لمستخدم التطبيق بقراءته وتغييره.

خزّن الرمز في مخزن أسرار أو متغيّر بيئة، ولا تضعه أبدًا في الرماز المصدري.

أرسل الرمز

ضع الرمز في الترويسة `x-app-token` واستدعِ مضيف القاطن نفسه:

```
export BASE="https://acme.example.com"
export TOKEN="<your-api-token>"

curl -s "$BASE/api/v2/entities/<entity>/entries/count" \
  -H "x-app-token: $TOKEN"

{ "success": true, "result": { "count": 42 } }
```

يُحدَّد القاطن من المضيف الذي تستدعيه؛ فلا ترسل معرّف قاطن أبدًا. وتسري القواعد نفسها — للمسارات، والإصدارات، وغلاف الاستجابة، والأخطاء — مع جميع بيانات الاعتماد: انظر استدعاء واجهة HTTP، والبدء السريع مع الواجهة الخلفية لاستدعاء أول موجَّه خطوة بخطوة.

ما يستطيع الرمز فعله

ليس للرمز **نطاقات أذونات خاصة به**؛ إذ تجري المصادقة به باسم مستخدم التطبيق المرتبط به، ثم تطبّق المنصّة مستوى صلاحيات ذلك المستخدم في النظام، وعضويته في المساحات، وأذوناته، تمامًا كما تفعل مع أي شخص. ولذلك:

- لتقييد عملية تكامل، قيّد أذونات مستخدم التطبيق الخاص بها. فللبيانات التي تقرؤها فقط، اطلب معرضًا تجري مشاركته مع مستخدم التطبيق، واقرأ البيانات عبره (`/api/v1/v/{viewId}/…`، انظر استدعاء واجهة HTTP). ولا يكفي مستوى **عرض** على الكيان نفسه: فهو يُظهر التصميم لا المُدخلات، ويعيد سرد مُدخلات الكيان حينها قائمة فارغة لا خطأ. واطلب **تعديل** للبيانات التي تكتبها فقط.
- امنح كل عملية تكامل مستخدم تطبيق ورمزًا خاصين بها، لتتمكّن من إبطال رمز إحداها دون المساس بالبقية، وليُظهر السجل التاريخي للمُدخل عملية التكامل التي أجرت كل تغيير.
- لا يمنح الرمز أبدًا أذونات تتجاوز أذونات مستخدمه. وتشير استجابة `403` لطلبٍ ما، أو قائمة فارغة حيث تتوقّع مُدخلات، إلى أن مستخدم التطبيق لا يملك إذن الوصول إلى تلك البيانات، لا إلى أن الرمز خاطئ.

انتهاء الصلاحية

يحدّد المسؤول، لكل مستخدم تطبيق، ما إذا كانت صلاحية الرمز غير محدّدة بمدة أم تنتهي في تاريخ محدّد. ويتوقّف الرمز عن العمل **في نهاية تاريخ الانتهاء**. ويُرفض الرمز المنتهي كأي رمز آخر غير صالح (انظر الأخطاء)؛ والحلّ أن يمدّد المسؤول **تاريخ الانتهاء** لدى مستخدم التطبيق، لا أن يعيد إنشاء الرمز.

`Since 1.10` يستطيع المسؤولون ضبط تاريخ الانتهاء. وقبل 1.10 كان الحقل مخفيًا، وكان كل رمز يُنشأ دون تاريخ انتهاء.

بدّل الرمز دون انقطاع

يستطيع المسؤول الذي يملك مستوى **إدارة** على مستوى النظام **إعادة إنشاء** الرمز. وتبقى هوية مستخدم التطبيق وأذوناته وسجلّه كما هي، ولا يتغيّر إلا السرّ. `Since 1.8`

1. يختار المسؤول **إعادة إنشاء الرمز** لدى مستخدم التطبيق، فيُرسل الرمز الجديد بالبريد الإلكتروني إلى ذلك المسؤول — ولا تُعيده الواجهة البرمجية أبدًا.
2. **يعمل الرمزان كلاهما خلال مهلة سماح** — 24 ساعة افتراضيًا؛ وقد يغيّرها المشغّل لديك.
3. انشر الرمز الجديد في عملية التكامل خلال تلك المهلة. وبعدها يُرفض الرمز القديم.

صمّم عملية التكامل بحيث تقرأ رمزها من الإعدادات، ليقتصر التبديل على تغيير الإعدادات دون إصدار جديد. وإذا ضاعت رسالة البريد التي تحمل الرمز الجديد، فأعد إنشاءه مرة أخرى قبل انتهاء مهلة السماح للرمز القديم.

وتُرفض إعادة إنشاء الرمز لمستخدم تطبيق مؤرشف. أما تبديل الرمز الخاص بوظيفة فيجري على نحو مختلف؛ إذ يُرسَل الرمز الجديد مباشرةً إلى الوظيفة — انظر الوظائف.

الإبطال وتأخّره

يتوقّف الرمز عن العمل حين **يعطّل** المسؤول مستخدم التطبيق الخاص به أو **يؤرشفه**، أو حين تنتهي صلاحيته.

تحتفظ البوّابة بنتيجة التحقّق الناجح مدة قصيرة – **خمس دقائق افتراضيًا** – لذا قد يظل الرمز المُبطَل يعمل إلى أن تنقضي تلك المدة. وراعِ هذا التأخير حين تُبطل رمزًا مسرّبًا: بدّله أو أبطله أولًا، ثم راقب الاستدعاءات التي لا تزال تنجح خلال تلك المدة.

ولمعرفة أثر هذا التأخير في جلسات OAuth وكيفية مراعاته في تصميمك، انظر دورة حياة الرموز.

الأخطاء

الحالة	`error.code`	المعنى	ما تفعله
`400`	`INVALID_APP_TOKEN_API_ERROR`	الرمز غير معروف، أو منتهي الصلاحية، أو معطّل، أو مؤرشف.	لا تُعد المحاولة. تحقّق من قيمة الرمز، واسأل المسؤول عن مستخدم التطبيق.
`403`	يتوقّف على العملية	الرمز صالح، لكن مستخدم التطبيق الخاص به لا يملك إذن الوصول إلى هذه البيانات.	اطلب من المسؤول منح الإذن.
`500`	`APP_TOKEN_VALIDATION_REQUEST_FAILED`	تعذّر على البوّابة الوصول إلى خدمة الهوية للتحقّق من الرمز.	أعد المحاولة مع زيادة الفاصل الزمني تدريجيًا بين المحاولات.

تستخدم الأخطاء بنية جسم الخطأ القياسية في المنصّة. وللاطّلاع على الفهرس الكامل وسياسة إعادة المحاولة، انظر الأخطاء وإعادة المحاولة.

تفاصيل تقنية

- الرمز سرّ سداسي عشري من 64 محرفًا. ولا تخزّن المنصّة إلا بصمة (hash) له، ولهذا لا يستطيع أحد عرضه مرة أخرى.
- الطلب الذي يحمل `x-app-token` يُصادَق عبر تلك الترويسة. أما الطلب الذي لا يحملها فيعود إلى مسار `Authorization: Bearer` المعتاد، فلا يتعارض نوعا بيانات الاعتماد أبدًا.
- في كل مرة تتحقّق فيها المنصّة من الرمز (لا في كل طلب تُستخدم فيه نتيجة مخبوءة)، تسجّل وقت «آخر استخدام».

معلومات مفيدة

- **يُعرض مرة واحدة، وبالبريد الإلكتروني فقط.** إذا ضاع الرمز، يعيد المسؤول إنشاءه؛ فلا سبيل إلى قراءته مرة أخرى.
- **لا نطاقات أذونات لكل رمز.** تُستمدّ أذونات التفويض كاملةً من مستخدم التطبيق.
- **التصرّف نيابةً عن شخص آلية مختلفة.** يتصرّف رمز API دائمًا بصفة مستخدم التطبيق الخاص به. ولتصرّف شخص نيابةً عن آخر أثناء غيابه، انظر التفويض [الإدارة]؛ أما التطبيق من طرف ثالث الذي يتصرّف نيابةً عن مستخدم مسجّل الدخول، فاستخدم معه OAuth.

> **للتعمّق أكثر**
>
> للمسؤولين: أنشئ رمز API لعملية تكامل [الإدارة] · للمشغّلين: عمليات بوابة الواجهة – app-token-auth · لمهندسي المنصّة: الهوية OAuth ورموز الـ API – التفاصيل الداخلية

أسئلة التكامل الشائعة

أسئلة شائعة لدى **المُكاملين والمطوّرين المحترفين** حول استدعاء واجهة HTTP البرمجية، ومصادقة السكربتات والخدمات، والتعامل مع رموز الأخطاء، والاستجابة للأحداث، وكتابة الوظائف، والتحكّم في الحزم عبر الواجهة البرمجية. صُنّفت الأسئلة حسب الموضوع؛ اختر سؤالًا للاطّلاع على إجابته.

تبني في المصمّم؟

تجد الأسئلة المتعلّقة بنمذجة البيانات، والمعارض، وسير العمل، والتفاعلات، والصفحات، ونقل التطبيق عبر شاشات **الحزم** في أسئلة بناء التطبيقات الشائعة [بناء التطبيقات].

التوسعة والتكامل

هل توجد واجهة HTTP API، وكيف تُدار إصداراتها؟

نعم، توفّر المنصة واجهة HTTP API واسعة تُتاح عبر بوابة واحدة، وتندرج جميع مساراتها تحت `/api`. وتُحدَّد إصدارات العمليات **في مسار الرابط** (`/v1/…`، `/v2/…`، `/v3/…`)؛ وتتعايش الإصدارات بدلاً من أن يحلّ أحدها محل الآخر، وغالباً ما تضيف الإصدارات الأحدث قدرات، لذا فضّل أعلى إصدار يوفّره المورد. ويُحدَّد سياق تعدّد القاطنين بالترويسة، فتُعيد العملية نفسها بيانات مختلفة لكل مساحة. وتُغلَّف استجابات JSON الناجحة في `{success, result}`، مع استثناء واحد في 1.10: طلب `GET` تحت `/v3/` يعيد الحمولة المجرّدة؛ راجع `GET /api/v3/…` صار خاماً من جديد.

كيف أصادق سكربتًا أو تكاملًا دون تسجيل دخول مستخدم؟

استخدم رمز API (يُسمّى أيضاً رمز التطبيق). يرسله البرنامج في ترويسة `x-app-token`، وتحدّد البوابة من خلاله المستخدم-التطبيق المالك له. وتُعرَض قيمة الرمز **مرة واحدة فقط** عند الإنشاء؛ فانسخها حينها، إذ لا يمكن استرجاعها لاحقاً. ولإبطال الرمز، أرشفه (مع تأخير طفيف بسبب الخبء). وخصّص لكل تكامل رمزاً مستقلاً لتتمكن من إبطال أحد الرموز دون التأثير في غيره.

هل تحمل رموز الـ API أذوناتها الخاصة؟

لا. ليس لرمز الـ API مجموعة أذونات أو نطاقات منفصلة؛ بل يرث أذونات وصول **المستخدم-التطبيق** المالك له. ولتقييد ما يستطيع التكامل فعله، أنشئ مستخدماً-تطبيقاً مخصّصاً لا يملك إلا الأذونات التي يحتاجها ذلك التكامل، وأصدِر الرمز باسمه.

كيف تُخزَّن الملفات المرفوعة؟

تُرفَع الملفات (في حقول الملف والصورة والمستند، ومرفقات التعليقات) إلى خدمة تخزين تعيد **معرّفاً**؛ ويخزّن المُدخل ذلك المعرّف بدلاً من البايتات. ويضبط المدير أنواع المرفقات المسموحة **لكل معرض** (بحدّ افتراضي للحجم يبلغ نحو 1 ميغابايت)، بدلاً من اعتماد مجموعة ثابتة. وتُفحَص الملفات الواردة من المستخدمين بالتحقّق من قائمة الامتدادات المسموحة، وفحص بايتات المحتوى الفعلي، وفحص جافاسكربت المضمّنة في PDF.

كيف أحصل على مستند OpenAPI لمعرض واحد؟

فعّل **Swagger document** في خيارات عرض ذلك المعرض؛ وعندها فقط ينشر `GET /api/v1/views/{viewId}/openapi.json` مستند OpenAPI 3.0 صغيراً يقتصر على ذلك المعرض. وانتبه إلى اختلاف مقصود واحد: **هذه الاستجابة غير مغلّفة بمغلّف** `{success, result}`؛ لذا حلّل جسمها بوصفه مستند OpenAPI مجرّداً، بينما المستند المُعاد *يصف* ذلك المغلَّف، لأن كل عملية يوثّقها تعيد استجابتها داخله. وثمّة قيدان آخران: يتطلّب استدعاء عملية جلب مخطط الكيان رمزاً دائماً، حتى في معرض مجهول؛ ويُبنى مخطط `value` للمُدخل من الحقول المختارة في المعرض نفسه و**لا يُرشَّح** بحسب الحساسية، فيظهر فيه الحقل الحساس أو المشفَّر الذي يعرضه المعرض باسمه ونوعه وتعدّديته (ولا تظهر قيمته أبداً). راجع مستند OpenAPI للمعرض الواحد. *(Since 1.10)*

لماذا يعيد استدعاء `GET /api/v3/…` جسماً مجرّداً دون `{success, result}`؟

لأن طلب `GET` الذي يبدأ مساره بـ `/v3/` يعيد، في سلسلة إصدارات 1.10، الحمولة الخام مجدداً، لخدمة التكاملات الخارجية التي تقرأ أجسام استجابات `/v3` مباشرة؛ أما عمليات الكتابة عبر `/v3` ومسارات `v2` / `v1` ومسارات JSON غير المرقّمة فتحتفظ بالغلاف `{success, result}`، وتبقى الأخطاء مغلّفة. وتظلّ عمليات النشر من سلسلة 1.8 تغلّف قراءات `/v3` حتى 1.8.21 ضمناً، لذا ينبغي للعميل الذي يتعامل مع السلسلتين أن يستخرج `result` فقط حين يحمل الجسم `success: true`. راجع `GET /api/v3/…` صار خاماً من جديد. *(Since 1.10)*

ماذا يعني `SPACE_ERROR 404` و `SPACE_CONTEXT_MISSING 400`؟

يصدر `SPACE_ERROR 404` عن البوابة حين لا يطابق مضيف الطلب أي مساحة، وكان رمز الحالة `400` قبل 1.10. وتُتجاهل أولاً أي ترويسة `x-space-id` أرسلها العميل، لذا استخدم مضيفاً مسجّلاً لمساحة. أما `SPACE_CONTEXT_MISSING 400` فيعني استدعاء خدمة مباشرة دون ترويسة `x-space-id` صالحة، مع تفعيل فرض نطاق المساحة؛ ولا يحدث هذا عبر البوابة، وعلى المستدعي المباشر أن يرسل ترويسة `x-space-id` واحدة فقط غير فارغة. راجع الرفض المتعلق بالقاطن والمساحة. *(Since 1.10)*

هل يستطيع المتصفح تحميل ملف مخزّن في وسم `<img>` عادي دون رمز حامل؟

نعم: يوقّع `POST /api/storage/v2/files/links` روابط قصيرة الصلاحية لما يصل إلى 100 معرّف ملف، ويبثّ `GET /api/storage/v2/files/:id/content?exp=…&sig=…` الملف اعتماداً على التوقيع وحده بوصفه بيان الاعتماد. وتعامل مع كل رابط بوصفه **قدرة لحامله**؛ إذ يستطيع كل من يحمله جلب الملف إلى أن تنتهي صلاحيته، فلا تسجّل الروابط في السجلات ولا تدرجها في روابط قابلة للمشاركة. وقد أُتيحت هذه الميزة أولاً في سلسلة إصدارات 1.6، ثم أُضيفت إلى سلسلة 1.8 ضمن تحديث. راجع روابط محتوى موقّعة قابلة للتخزين. *(Since 1.6)*

كيف أبدأ تسجيل الدخول مباشرة بمزوّد خارجي محدّد؟

أضف معامل طلب التفويض `idp_hint=<provider name>` إلى `/auth`: فعندما يُفترض عرض صفحة الدخول، تعيد خدمة الهوية التوجيه مباشرة إلى ذلك المزوّد الخارجي **النشط** إذا كان مرتبطاً بالنطاق الذي جاء منه الاستدعاء، وتتجاهل الأسماء المجهولة أو غير النشطة أو غير المرتبطة. والقيمة هي `name` الفريد للمزوّد، لا تسميته المعروضة، ويمرّر الرابط العميق للبوابة `/signin?provider=<name>` هذه القيمة إلى المعامل. وبصورة مستقلة عن ذلك، يتخطّى النطاق الذي يتيح الدخول حصراً عبر مزوّد خارجي واحد صفحةَ الدخول لجميع العملاء افتراضياً. راجع بدء تسجيل الدخول بمزوّد مسمّى. *(Since 1.10)*

كيف أمنح وظيفةً مفتاح API أو سرًّا آخر؟

في خطّ التطوير، أضف صفًّا إلى كتلة **Secrets** (الأسرار) في الوظيفة، يتضمّن `key` (اسم متغيّر البيئة: حروف وأرقام و `_`، ولا يبدأ برقم) و `value`، ثم احفظ مُدخل الوظيفة بالطريقة المعتادة؛ فلا توجد واجهة برمجية منفصلة للأسرار. وتُخزَّن القيمة مشفّرة وتُحجَب عند قراءتها. وتقرؤها الوظيفة بوصفها متغيّر بيئة، مثل `process.env.MY_API_KEY`، عند بدء مراجعة (revision) جديدة؛ إذ يُزامَن التغيير دون إعادة بناء، ويؤدّي إلى إطلاق مراجعة جديدة، أما الجراب الذي يعمل بالفعل فيحتفظ بالقيمة القديمة. راجع أسرار الوظيفة وهويتها الخاصة. *(Since dev)*

بصفة مَن تعمل الوظيفة حين تستدعي واجهة المنصّة البرمجية؟

في خطّ التطوير، تعمل بصفة **مستخدم التطبيق الخاص بها** (المُصدِر `function:<functionId>`)، وتتلقّى رمزه في `OSOS_API_TOKEN`. ويبدأ هذا المستخدم بلا مجموعات ولا أدوار ولا صلاحيات وصول إداري، فلا تستطيع الوظيفة تنفيذ أي إجراء إلا ضمن ما تمنحه لهذا المستخدم صراحةً. ولا تؤدّي إعادة رفع رماز الوظيفة إلى تدوير الرمز، ولا يتلقّى المسؤول الذي يعيد إنشاءه بريداً؛ فالرمز الجديد يُسلَّم إلى الوظيفة مباشرة. راجع هوية الوظيفة الخاصة. *(Since dev)*

لماذا لا يتلقّى تكاملي حدث تحديث بعد عملية كتابة؟

في خطّ التطوير، **لا** تنشر عملية الكتابة حدث إنشاء ولا حدث تحديث إذا حُذف مُدخلها قبل أن تعيد data-service قراءته؛ بل يُبلَّغ عن فشل الكتابة، وللحذف حدثه الخاص `DataEntryDeletedEvent`. أما الحدث الذي يُنشر فيحمل `dataEntry` دائماً، لذا لا تنتظر إلى أجل غير مسمّى حدث تحديث لمُدخل ربما حُذف. راجع نموذج البيانات والأحداث. *(Since dev)*

لماذا فشل حفظ بالرمز `400 FILE_EXPIRED`، وكيف ينبغي لعميلي إرفاق الملفات؟

لأن عملية الحفظ أشارت إلى معرّف ملف لا تجده خدمة التخزين؛ وعادةً ما يكون ملفاً مرفوعاً رفعاً **مؤقتاً** مضى عليه أكثر من يوم، أو معرّفاً لم يوجد قطّ. ومنذ 1.10 تحتفظ خدمة التخزين بسجلّ يبيّن أيّ مُدخل محفوظ يستخدم أيّ ملف، والأسلوب الموثوق هو ما تتّبعه بوابة الويب: ارفع الملف مع `?temp=true`، واحفظ المُدخل باستخدام الواصف المُعاد خلال اليوم، إذ **يرقّي** الحفظُ الملفَ إلى التخزين الدائم بالمعرّف نفسه. ولا تُفحص إلا المعرّفات **الجديدة** المضافة إلى المُدخل المخزّن، فلا تفشل إعادة حفظ مُدخل قديم بسبب ملفات كان يتضمّنها؛ وتسرد `details` كل ملف مفقود مع حقله. والحلّ دائماً هو رفع الملف من جديد وإعادة الحفظ. وإذا تعذّر الوصول إلى خدمة التخزين أثناء إضافة ملف جديد عند الحفظ، تعيد عملية الحفظ الرمز `503 STORAGE_SERVICE_UNAVAILABLE` القابل لإعادة المحاولة. ولا تزال عمليات الرفع الدائم تعمل كما كانت، لكنها غير متتبَّعة. راجع رفع مؤقت ثم حفظ وFILE_EXPIRED وعدم توفّر التخزين. *(Since 1.10)*

هل تحذف المنصة ملفاً لم يعد شيء يشير إليه؟

تحذف الملف **المتتبَّع** فقط، وبعد أن يؤكّد ذلك كل مالك سابق. فكانس الملفات اليتيمة يفحص الملفات التي رُقّيت من رفع مؤقت ثم أُزيلت مراجعها لاحقاً؛ أما الملف المرفوع مباشرة إلى التخزين الدائم، أو الملف المؤقت الذي لم تُستكمل له عملية حفظ، فلا يُكنس أبداً. ويجب أن تنقضي على الملف اليتيم مهلة سماح (7 أيام افتراضياً)، وأيّ استجابة `listed` أو `unknown` من مالك سابق تُبقي الملف. والكانس معطّل افتراضياً في الخدمة؛ ويشغّله مخطط النشر في وضع **التشغيل التجريبي**، فلا يسجّل إلا الملفات التي كان سيحذفها إلى أن يوقف مشغّل المنصة التشغيل التجريبي. ولا يؤدّي حذف إصدار حزمة أو تصدير إلى حذف ملف ZIP الخاص به أبداً. راجع كانس الملفات اليتيمة وسجلّ مراجع الملفات. *(Since 1.10)*

هل يستطيع عميلي ضبط `x-service` أو `x-view-id` على طلب يمرّ عبر البوابة؟

لا. فعند المرور عبر البوابة، تُزال قيمة ترويسة `x-service` التي يرسلها العميل وتُضبط على `API Gateway`، وتُمسح ترويسة `x-view-id` التي يرسلها العميل قبل أي فحص للوصول، ولا تُضبط مجددًا إلا من `{viewId}` في مسار

`/api/v1/v/{viewId}/…` ؛ لذا استخدم صيغة المسار تلك لتنفيذ العمليات عبر معرض. ويُضاف ذلك إلى الإزالة المعمول بها سابقًا لترويسة `x-shared-key` والترويسات الداخلية الأخرى. أما الخدمة التي يُوصَل إليها **مباشرة** فلا تستفيد من أيٍّ من إجراءات الحماية هذه. راجع ما تنزعه البوابة وتُنزَع ترويستا الخدمة والمعرض أيضًا. *(Since 1.10)*

لماذا صار استدعاءٌ كان يفشل بالرمز `500` يُجاب الآن بـ `4xx` ؟

لأن حالات الرفض المتعلقة بمُدخلات المستدعي أو استحقاقه أصبحت تحمل حالة خاصة بكل منها، مع رمز `error.code` ثابت. ومن الأمثلة: `400 INVALID_POINTER` لمؤشّر مثل `@me` أو `@today('-1d')` لا يمثّل دالة مؤشّر معروفة، و`400 MISSING_REQUIRED_PARAMETER` عند إنشاء مُدخل بطلب أُرسل بلا جسم إطلاقاً، و`403 ACCESS_OBJECT_NO_AUDIENCE` ، و`404 QUERY_NOT_FOUND` ، و`404` لمساحة غير موجودة. وبصورة مستقلة عن ذلك، أصبح الرمز `4xx` الوارد من خدمة أخرى في أسس يُمرَّر بحالة تلك الخدمة بدلاً من تحويله إلى `500` ؛ وقد أُدرج هذا التغيير أيضاً في تحديثات 1.8 (`Since 1.8`). حدّد مسار المعالجة بحسب `error.code` ، وتعامل مع الحالة بوصفها إشارة عامة. راجع حالات رفض مُنمَّطة كانت تُجاب بـ 500 وحالات رفض الكتابة المضافة في 1.10. *(Since 1.10)*

ماذا يعني `503 TENANT_IN_MAINTENANCE` ، وماذا ينبغي لعميلي أن يفعل؟

في النشر متعدّد القاطنين، يعني أن بيانات القاطن **تُستعاد** من نسخة احتياطية، أو أن مشغّل المنصة يحجز القاطن **لنقله** إلى خلية أخرى. ويُجاب كل طلب للقاطن، بما فيه `/auth` و `/token` والتحديث في خدمة الهوية، بالرمز `503` مع `Retry-After`؛ فانتظر المدة المحدّدة وأعد المحاولة بدلاً من تسجيل خروج المستخدم. وفي الأثناء **يركن** مستهلكو الناقل رسائل القاطن ويعيدون تشغيلها لاحقاً، مع استثناء مقصود واحد: الرسالة التي رُكنت أول مرة أثناء استعادة أعادت البيانات إلى حالة سابقة **تُسقَط**، لذا على التكامل الذي يعتمد على حدث لإحداث أثر خارجي أن يتعامل مع الأحداث الصادرة خلال فترة الاستعادة على أنها قد لا تُسلَّم أبداً. ولا توجد حالة صيانة في عمليات التثبيت وحيدة القاطن. راجع التسليم أثناء وضع القاطن تحت الصيانة وحالات رفض الكتابة المضافة في 1.10. *(Since 1.10)*

هل أستطيع أخذ نسخة احتياطية للقاطن أو استعادته أو تصدير بياناته عبر الواجهة البرمجية؟

نعم، بصفتك **مسؤول النظام** لدى القاطن نفسه في النشر متعدّد القاطنين، ولا **شاشة ويب لذلك في 1.10**؛ أما في النشر وحيد القاطن فيُجيب كل مسار بالرمز `501 BACKUP_NOT_SUPPORTED_IN_SINGLE_MODE` . ويُجاب كل طلب بدء بالرمز `202` ، ثم تستطلع الحالة: `POST /api/v1/admin/backups` و `POST /api/v1/admin/exports` ؛ وتتم الاستعادة على خطوتين: معاينة تُرجع `confirmationToken` لاستخدام واحد، ثم استعادة باستخدام ذلك الرمز. ومنذ بدء الاستعادة، يُجيب القاطن بالرمز `503 TENANT_IN_MAINTENANCE` عن جميع الطلبات عدا مسارات `GET` في هذه المجموعة. ويُنزَّل التصدير عبر رابط لاستخدام واحد (خمسة روابط على الأكثر لكل تصدير)، ويتضمّن قواعد بيانات القاطن **دون حسابات المستخدمين**. وتُحتسب طلبات البدء ضمن حصص يومية (`429` مع `Retry-After` عند تجاوزها)، ولا يُتاح الإلغاء. راجع النسخ الاحتياطي للقاطن واستعادته وتصدير بياناته والعمل طويل الأمد. *(Since 1.10)*

لماذا يتلقّى عميلي الإشعارات الفورية من كل المساحات؟

لأن الإشعارات **غير محصورة بالمساحة على المقبس**: فالإشعار المدفوع `notification` يصل إلى غرفة المستخدم أياً كانت المساحة التي أُطلق فيها. ويُسقط عميل الويب المُصدَر الإشعار المدفوع الذي ينتمي إلى مساحة غير المساحة النشطة، لكن هذا الترشيح يجري من جهة العميل وحدها. لذا على العميل المخصّص أن يرشّح الإشعارات بنفسه بحسب `value.space` في المُدخل. راجع الوقت الحقيقي. *(Since 1.10)*

هل تحتاج واجهة المستخدم النهائي V2 إلى عميل OAuth خاص بها؟

لا. فعندما يوفّر النشر واجهة المستخدم النهائي (`END_USER_ENABLED=true` ، على `END_USER_PATH` ، و `/v2` افتراضياً)، يسمح عميل الويب العام القياسي أيضاً بالعنوان `https://<domain><END_USER_PATH>/auth-callback` لكل نطاق لدى كل قاطن، باستخدام المعرّف `client_id` نفسه، ودون تغيير عناوين ما بعد تسجيل الخروج. وعندما تكون واجهة المستخدم النهائي معطّلة، لا يوجد عنوان إعادة التوجيه هذا. ويقتصر ذلك على التوجيه؛ أما تطبيق الويب V2 نفسه فلا يزال `Since dev` . راجع المصادقة و*(Since 1.10)* OAuth.

نمذجة البيانات والمعلومات – الواجهة البرمجية والأخطاء

هل يمكنني نقل كيان إلى وحدة أخرى؟

نعم، بصلاحية مسؤول النظام. فمنذ 1.10 أصبح نقل الكيان (`PATCH /v1/entities/{id}/module`) مقصوراً على مسؤول النظام، ويتيح المسار `POST /v1/entities/move` نقل ما بين كيان واحد و50 كياناً إلى وحدة واحدة في طلب واحد، مع بيان نجاح نقل كل كيان أو فشله؛ ويمكن نقل الكيانات الوهمية أيضاً. وعند نقل الكيان تتبعه معارضه وتجميعاته الفرعية؛ ويُرفض النقل إلى الوحدة الأساسية المحمية أو إلى وحدة يجري حذفها نهائياً (`409 MODULE_BEING_DELETED`). ويظل الاسم المعياري للكيان ونوعه ثابتين أثناء النقل. راجع النقل بصلاحية المسؤول، والنقل الجماعي، وما يصلحه النقل [بناء التطبيقات › إنشاء كيان وتصميمه]. *(Since 1.10)*

هل هناك حدود لحجم البيانات أو عدد المُدخلات التي يمكن تخزينها؟

فقط إذا كانت باقة القاطن تضع حداً، وهو ما أصبح ممكناً منذ 1.10. ولا تزال بعض أنواع الحقول تفرض حدودها الخاصة على الحجم أو العدد، لكن الحدود على مستوى المنصة أصبحت تحدّدها الباقة، ولا يُفرض أي حدّ لم يُضبط:

- **المدخلات** – تُحتسب يومياً بأوزان محدّدة (المُدخل النشط والمسوّدة بوزن 1، والمؤرشف بوزن 0.25؛ ولا تُحتسب إلا الكيانات التي يبنيها المستخدمون). يبدأ التحذير عند 80 % من السقف؛ ويُرفض إنشاء المُدخل – مفرداً أو ضمن دفعة أو بوصفه مسوّدة أو عبر وكيل – بالرمز `403 ENTRY_CAP_EXCEEDED` حين يبلغ المجموع **الحدّ الصارم** (السقف زائد 10 %)، ما لم يُسمح بالتجاوز. ولا تُرفض القراءة ولا التعديل ولا التصدير أبداً.
- **الكيانات** – يُرجِع إنشاء كيان يتجاوز الحدّ `403 ENTITY_CAP_EXCEEDED` ؛ ولا تُحتسب الكيانات الضمنية ولا كيانات الوحدة الأساسية.
- **المساحات** – تُرجِع أي عملية كتابة تضيف مساحة تتجاوز الحدّ `403 SPACE_CAP_EXCEEDED` .
- **التخزين** – تُرجِع خدمة التخزين `403 STORAGE_CAP_EXCEEDED` عند الرفع الذي يبلغ الحدّ أو يتخطّاه.
- **تشغيلات التفاعلات في الشهر** – لا يتسبّب التشغيل الذي يتجاوز الحدّ في فشل أي طلب؛ بل يُسجَّل بحالة `refused` ولا يُنفَّذ أيٌّ من إجراءاته.

ويُرفض تثبيت الحزمة مسبقاً إذا كان سيتجاوز حدّ المُدخلات. راجع حدود الخطة: الاستهلاك والرفض وسقف الكيانات، وللاطّلاع على ما يظهر للمسؤول، راجع حدود الخطّة [الإدارة › رخصتك وحدود خطتك]. *(Since 1.10)*

هل يمكنني حذف كيان أو وحدة كاملة حذفاً نهائياً؟

نعم، منذ 1.10، بصلاحية مسؤول النظام. فالمسار `DELETE /v1/entities/{id}/permanent` يزيل نهائياً ودون رجعة كياناً من نوع Custom أو Request أو كياناً وهمياً وكل ما يملكه، ويزيل المسار `DELETE /v1/data-module/{id}/permanent` وحدةً وكل ما فيها ضمن مهمة في الخلفية؛ أما مسارات الحذف العادية فلم تتغيّر. ويخضع كلاهما لقاعدة «فُكّ الارتباط أولاً»:

فيُرفض الطلب بالرمز `409 ENTITY_HAS_DEPENDENTS` / `409 MODULE_HAS_DEPENDENTS` ما دام مورد خارجي يشير إلى الكيان أو الوحدة، وتُرفض الموارد المثبّتة من حزمة بالرمز `403` . راجع الحذف النهائي (purge) [بناء التطبيقات › إنشاء كيان وتصميمه والوحدات – الحذف النهائي [بناء التطبيقات › الوحدات]. *(Since 1.10)*

هل يمكنني إضافة فهارس قاعدة بيانات خاصة بي إلى كيان؟

نعم. يتيح تبويب **الفهارس** في باني الكيان للمدير تعريف ما يصل إلى **10** فهارس MongoDB إضافية لكل كيان، يضمّ كلٌّ منها من حقل واحد إلى 8 حقول، مع خيار **فريد** ومرشِّح جزئي اختياري، لتسريع عمليات الترشيح أو الفرز المتكررة أو لفرض التفرّد على مجموعة حقول. ويجب أن يتضمّن الفهرس الفريد مرشِّحاً جزئياً يستبعد القيم الفارغة، لأن MongoDB يفهرس القيمة المفقودة بوصفها `null` ؛ ولا يُعرض التبويب لكيانات الوحدة الأساسية ولا لكيانات `Request` ولا للكيانات الوهمية. راجع فهارس الكيان (تبويب الفهارس) [بناء التطبيقات › إنشاء كيان وتصميمه]. *(Since 1.10)*

الحقول وأنواع البيانات – الواجهة البرمجية والأخطاء

هل يمكن لحقل التصنيف أن يحمل عدة قيم، وماذا يحدث حين أفعّل ذلك؟

نعم، منذ 1.10: فعّل **Multiple** في إعدادات قيمة الحقل، مع حدّين اختياريين، أدنى وأقصى، لعدد الاختيارات، فيخزّن الحقل مصفوفة `[{id}, …]` بدلاً من كائن `{id}` واحد. ويُعدّ تفعيله **تغيير كاسر**، إذ يُرفع إصدار الكيان وتُوضَع المُدخلات القائمة في مصفوفات من عنصر واحد عند ترقيتها؛ ويُرفض التبديل بالرمز `400 CATEGORY_FIELD_MULTI_VALUE_IN_USE` ما دامت لوحة Kanban أو تجميع أو فرز أو لوحة معلومات أو شريط زمني يعتمد على الحقل. ولا يمكن أبداً استخدام حقل التصنيف متعدد القيم بوصفه التصنيف الرئيسي أو حقل أعمدة Kanban أو فرزاً افتراضياً أو تجميعاً للوحة معلومات (`400 MULTI_VALUE_CATEGORY_FIELD_NOT_ALLOWED`). راجع القيم المتعددة [بناء التطبيقات › حقل التصنيف]. *(Since 1.10)*

هل يستطيع المستخدمون النهائيون إضافة خيارات حقل التصنيف أو تعديلها بأنفسهم؟

نعم، من بطاقة إنشاء المُدخل أو تعديله، عند تفعيل إعدادين معاً: **السماح للمستخدمين بإدارة الخيارات** على حقل التصنيف (ولا يُسمح به على حقل تُربط خياراته من حقل آخر)، و**إضافة خيارات التصنيف** / **تعديل خيارات التصنيف** على عملية الإنشاء أو التعديل في المعرض. وتُرسَل التغييرات مع عملية كتابة المُدخل بوصفها `categoryOptionChanges` وتُحفظ معه؛ ويُرفض التغيير المتعارض مع تعديل متزامن (`CATEGORY_OPTION_CONFLICT` أو `CATEGORY_OPTIONS_CHANGED`)، وعندئذ يعيد المستخدم التحميل. راجع السماح للمستخدمين بإدارة الخيارات [بناء التطبيقات › حقل التصنيف]. *(Since 1.10)*

العمل مع البيانات – المعارض والأوجه والمُدخلات – الواجهة البرمجية والأخطاء

كيف أتيح للمستخدمين النهائيين إعادة ترتيب القيم المتكررة؟

فعّل `allowRepeatableReorder` ، وهو خيار في إعداد المعرض يُضبط **لكل عملية** على كل مُدخل في قائمة العمليات المدعومة ضمن إعداد التعديل، وليس على تعريف الحقل. لذا يمكن أن يكون الحقل نفسه قابلاً لإعادة الترتيب في «إنشاء» وثابتاً في «تعديل» ضمن المعرض الواحد، أو قابلاً لإعادة الترتيب في معرض دون آخر. ولا يتاح تحديد الخيار إلا إذا احتوت قائمة حقول العملية على هدف واحد قابل للتكرار على الأقل (حقل قابل للتكرار أو كتلة قابلة للتكرار)، ويعود تلقائياً إلى حالة الإيقاف إذا أزلت آخر حقل مؤهَّل. وعند تفعيله، يظهر لكل تكرار مقبض سحب في نموذج الإنشاء أو التعديل؛ ولا يظهر المقبض إلا عند وجود أكثر من تكرار واحد، ويجعل السحبُ النموذجَ في حالة تعديلات غير محفوظة، فيُحفَظ الترتيب الجديد مع المُدخل.

و**يحترم العميلان كلاهما هذا الإعداد**: إذ يقرأ تطبيق الجوال العلامة نفسها الخاصة بكل عملية من إعداد المعرض، ويعرض مقبض السحب الخاص به على كل تكرار بعد الأول، فيعمل المعرض المُعدّ لإعادة الترتيب بالطريقة نفسها في بوابة الويب وعلى الهاتف. (وكانت إعادة ترتيب التكرارات خاصةً بالويب عند إطلاقها أول مرة؛ ولم تعد مقتصرة عليه.) راجع معارض البيانات [بناء التطبيقات › معارض البيانات وأوجهها]. *(Since 1.8)*

لماذا أجاب حفظ معرض أو صفحة بالرمز `409 UPDATE_CONFLICT` ؟

لأن شخصاً آخر حفظ المعرض أو الصفحة بعد أن فتحتهما. فكلاهما يُحفظ بوصفه مستنداً كاملاً تحميه قيمة `updateIdentifier` . وقبل 1.10، كانت مقارنة الرمزين تقتصر على قراءة مسبقة، فكان من الممكن أن تجتاز عمليتا حفظ هذه المقارنة معاً، ثم تكتب إحداهما فوق الأخرى؛ أما الآن فالكتابة نفسها مشروطة، وتُرفض الثانية من عمليتي حفظ متزامنتين بالرمز `409` . أعد القراءة ثم احفظ من جديد. وأصبحت حالتا رفض أخريان لحفظ المعرض تُرجِعان رمزين محدّدين بدلاً من `500` : فتفعيل تعديل المُدخلات دون أي عملية مدعومة يُرجِع `400 DATA_VIEW_INVALID_ALLOWED_OPERATIONS` ، والتحقّق من قدرة المستدعي على فتح معرض أو صفحة أو تقرير لا يفعّل **أياً** من الجمهورين يُرجِع `403 ACCESS_OBJECT_NO_AUDIENCE`. راجع حفظ المعرض: التعارضات وحالات الرفض [بناء التطبيقات › معارض البيانات وأوجهها]. *(Since 1.10)*

هل أستطيع إعادة تسمية معرض أو صفحة مثبّتة من حزمة، أو تغيير أيقونتها؟

نعم. فالمعرض أو الصفحة التي أنشأتها عملية تثبيت حزمة تكون مقفلة، لكن يمكن تغيير **اسمها ووصفها وأيقونتها** في المحرّر (عدا اسم المعرض الضمني)، ويعرض المحرّر قيمة الحزمة بجانب قيمتك. وتُحفَظ تسميتك عند تحديث الحزمة؛ وتُسجَّل قيمة الحزمة الأصلية في `packageLabels` ، التي لا يستطيع العميل الكتابة فيها. وعبر الواجهة البرمجية، أرسل التسميات إلى `PATCH /api/v1/entities/{entity}/views/{id}` أو `PATCH /api/v1/generic-views/{id}` مع قيمة `updateIdentifier` الحالية (وإلا أُرجِع `409 UPDATE_CONFLICT`)؛ وتمسح `description: null` الوصف، أما `icon: null` فلا تمسح الأيقونة **إلا** مع `?clearIcon=true` . وتبقى حالة النشر وترتيب العرض والجمهور وحقوق الوصول تحت تحكّمك أيضاً. وما سوى ذلك لا يزال يُرجِع `403 PACKAGE_RESOURCE_READ_ONLY` . راجع المعارض المثبَّتة [بناء التطبيقات › معارض البيانات وأوجهها] وتدفّق التحديث: التزامن والصفحات المثبَّتة [بناء التطبيقات › الصفحات والقطع]. *(Since 1.10)*

الاستعلامات والبحث والفرز — الواجهة البرمجية والأخطاء

كيف يحصل الاستعلام السريع ذو المعاملات على قيمه؟

منذ 1.10، يحمل كل استعلام سريع مطبَّق قيمه الخاصة: فيظهر للمستخدم النهائي مربع حوار **معاملات الاستعلام**، وتتلقّى الواجهة البرمجية `quickFilterParameters` مفهرسةً بمعرّف الاستعلام (`{"<queryId>": {"<parameterName>": {value}}`)، مع الرجوع إلى الكائن المسطّح `parameters` لضمان استمرار عمل لوحات Grafana القائمة. ويؤدّي غياب قيمة إلى فشل الطلب بالرمز `422 QUICK_FILTER_PARAMETER_VALUE_MISSING` ، ولا يمكن أبداً تطبيق استعلام سريع ذي معاملات افتراضياً. راجع الاستعلامات السريعة ذات المعاملات [بناء التطبيقات › الاستعلامات والمرشّحات]. *(Since 1.10)*

الخدمات وسير العمل — الواجهة البرمجية والأخطاء

كيف أسلّم خطوة سير عمل إلى نظام خارجي؟

استخدم **خطوة خارجية** (`StepType.ExternalStep`). فإعدادها `ExternalStepConfig` إلزامي، ويتضمّن **الرابط**، و**الطريقة** (`GET` أو `POST`)، و**فتح كـ**، أي كيفية عرض الوجهة للمالك. ويمكن لخطوة `POST` أن تحمل كذلك **حمولة مخصصة**؛ ويرسل

الطلب افتراضياً تفاصيل المُدخل مع إجراءات الخطوة. وضمّن رابط العودة باستخدام الرمز `{{@returnUrl}}` في الرابط أو الحمولة، واستخدم `{{@returnUrl:actionId}}` حين يكون للخطوة أكثر من إجراء واحد، إذ لا يُستبدل الرمز المجرّد تلقائياً إلا عند وجود إجراء واحد بالضبط. ويجب أن يُفضي تحديد مالك الخطوة إلى مستخدم واحد محدّد بالضبط (وتُرفض ملكيات المجموعات والملكيات الديناميكية والتعبيرية عند الحفظ)، ولا تكتمل الخطوة إلا حين يبلّغ النظام الخارجي. راجع مرجع نموذج سير العمل وواجهة API. *(Since 1.8)*.

لماذا عاد إجراء سير العمل بالرمز `409 UPDATE_CONFLICT`؟

لأن شخصاً آخر غيّر سير عمل المُدخل أولاً. فمنذ 1.10، لا تُحفظ أي عملية كتابة على سير العمل – تنفيذ إجراء، أو بدء تدفق فرعي أو إكماله، أو إدراج خطوة مخصّصة، أو تغيير المالك أو الخطوة – إلا إذا ظلّ سير العمل في الحالة التي استند إليها التحقّق، وتُرفض العملية التي تخسر سباق التزامن بالرمز `409 UPDATE_CONFLICT` دون تسجيل أي شيء أو نشره. أعد تحميل المُدخل وقرّر من جديد؛ ويُعاد إثبات التحقّق الإضافي للمدة المتبقية من صلاحيته كي تستخدمه إعادة المحاولة. راجع حارس التزامن على عمليات كتابة سير العمل. *(Since 1.10)*

لماذا صار تنفيذ إجراء مخفيّ وأخذ ملكية خطوة مملوكة يُجابان بالرمز `422`؟

لأن كليهما يمثّل حالة رفض للطلب، لا عطلاً في الخادم. فتنفيذ إجراء مخفيّ على المُدخل يُرجِع `422 WORKFLOW_ACTION_EXECUTION_NOT_ALLOWED`، وأخذ ملكية خطوة لها مالك بالفعل يُرجِع `422 CURRENT_STEP_ALREADY_OWNED`؛ وكان كلاهما يُرجِع `500` . وقد اختير `422` بدلاً من `403` عمداً، لأن عميل الويب يوجّه كل استجابة `403` إلى صفحة رفض الوصول. ويجب تعديل رماز العميل الذي كان يطابق `500` ليطابق رمز الخطأ بدلاً منه. راجع حالات رفض مُنمَّطة كانت تُجاب بـ 500. *(Since 1.8)*

التفاعلات والأتمتة – الواجهة البرمجية والأخطاء

هل يستطيع التفاعل استدعاء نظام خارجي أو تشغيل رمازي الخاص؟

نعم. يرسل إجراء التفاعل **استدعاء API** (`CallApi` ، 90) طلب HTTP صادرًا (GET/POST/PATCH/PUT/DELETE) ويلتقط الاستجابة، ويشغّل إجراء **تنفيذ وظيفة** وظيفة بلا خادم (serverless) نشرتَها. استخدمهما عندما تحتاج إلى استدعاءات شبكية خارجية أو منطق مخصّص يتجاوز كتالوج الإجراءات المدمج عديم الرماز.

ما الذي حلّ محلّ مفتاح «UI Event» على مشغّل الإجراء المخصّص؟

إعدادان. فالمشغّل يُسمّى الآن **إجراء مخصص** في الباني (وما يزال `CustomEvent` في الرماز وواجهة الـ API)، ويتضمّن:

- **مستوى الإجراء المخصص** (`customActionType` ، `CustomActionTypeEnum`) – قائمة منسدلة بثلاث قيم. تشير **UI** *(إجراء واجهة)* إلى أن الإجراء يُستدعى من واجهة مستخدم، أي من زرّ نظام، وتكافئ تفعيل مفتاح *UI Event* القديم. ويمكن استدعاء **External** *(إجراء خارجي)* عبر تنفيذ إجراء مخصّص [بناء التطبيقات › التفاعلات والأتمتة] من تفاعل آخر في أي وحدة، واستدعاء **Local** *(إجراء محلي)* من داخل وحدته فقط؛ وترفض reaction-service أي «تنفيذ إجراء مخصّص» يشير إلى إجراء UI. وقد طُرحت القيمتان أول مرة باسمَي `Module` و `Internal` ، ثم أُعيدت تسميتهما بترحيل أحادي الاتجاه في دورة 1.8 نفسها (`Module` ← `Local` ، و `Internal` ← `External`).
- **سياقي** (`contextual`) – مفتاح تبديل مفعَّل افتراضيًا، يُخزَّن على المشغّل ويُعرض في المحرِّر. ولا يرتبط به بعدُ أيّ سلوك وقت التشغيل؛ فهو علامة لاستخدام مستقبلي، وليس مفتاحًا يغيّر طريقة عمل التفاعل حاليًا.

ويترتّب على النوع حكمان: لا تنطبق خيارات **إظهار للمستخدم** (مؤشّر التنفيذ ورسالتا بدء التنفيذ ونجاحه) إلّا حين يكون النوع **UI** – إذ ترفضها الخدمة على مشغّل External أو Local – ولم يعد طلبُ مُدخل من المستخدم مرتبطًا بالنوع. ولا تتأثّر الأزرار على مستوى البيانات المتبادلة: فلا يزال DTO زرّ النظام يعرض قيمة منطقية بسيطة `uiEvent`، لكنها صارت **مُشتقّة** من كون النوع UI بدل أن تُقرأ من علَم مُخزَّن.

والتفاعلات القائمة رُحِّلت جميعها إلى النوع **UI** مع تفعيل «سياقي»، وأُزيل العلَم الملغى. والترحيل أحادي الاتجاه، لذا راجِع أيّ مشغّل ينبغي أن يصبح الآن External أو Local وغيّر نوعه يدويًا. راجع نوع الإجراء المخصّص وخيار Contextual [بناء التطبيقات › التفاعلات والأتمتة]. *(Since 1.8)*

هل يمكن أن يُشغَّل تفاعل على مُدخل من كيان مختلف؟

لا. يُشغَّل التفاعل **بوصفه كيانه الخاص** – فإجراءاته تحلّ مسارات الحقول وتكتب عبر تعريف ذلك الكيان – وأصبح المحرّك يفرض ذلك صراحةً. فالمُدخل الذي يختلف كيانه عن كيان التفاعل **يُتخطّى** مع تسجيل تحذير في سجلّ الخدمة، بدلًا من تشغيل التفاعل عليه باستخدام تعريفات حقول كيان غير صحيح. وفي تفاعلات الكيانات المرتبطة، تُرشَّح المُدخلات المُستقدَمة عبر علاقة لكلّ مشغّل على حدة، بحيث تقتصر على كيان ذلك المشغّل نفسه *قبل* جلبها. وتبرز أهمية ذلك في حالة **حقل مرجع يقبل أكثر من كيان**: إذ كان كلّ تفاعل يشترك في تلك العلاقة يتلقّى سابقًا كلّ مُدخل مُشار إليه أيًّا كان كيانه. والمشغّل الذي يتعذّر حلّ الكيان المعرَّف له – كأن يكون الكيان محذوفًا – يُتخطّى مع تحذير بدل إفشال العملية بأكملها.

ولا يُطلَق خطأ جديد لأيٍّ من ذلك. فنُسجَّل حالة المُدخل التابع لكيان آخر أو الكيان الذي يتعذّر حلّه كتخطٍّ لا كإخفاق، ومن ثمّ لا يظهر شيء في سجلّ تنفيذ التفاعل. فإن بدا أن تفاعلًا لم يُطلَق على بعض المُدخلات، فابحث عن تحذيرات التخطّي في سجلّ reaction-service، لا عن نتيجة إجراء فاشلة. راجع عزل الكيانات [بناء التطبيقات › التفاعلات والأتمتة]. *(Since 1.8)*

هل يستطيع التفاعل منح أذونات الوصول أو إزالتها؟

نعم، منذ 1.10، باستخدام الإجراءين **منح الإذن** و**إزالة الإذن**، لمنح مستخدمين أو مجموعات إذنًا على مورد من نوع System أو Module أو كيان أو معرض أو صفحة أو تقرير، أو إزالته؛ ولا يُمنح المستويان `Restricted` و `None` أبدًا، ولا يخفض إجراء المنح مستوى منحة قائمة أعلى. ولا يستطيع التفاعل منح مستوى يتجاوز ما يملكه من يحفظه؛ إذ يُفحص كل إجراء إذن مقابل مستوى وصول من يحفظ التفاعل عند الحفظ، ويحمل الرفض الرمز `PERMISSION_ACTION_EXCEEDS_SAVER_ACCESS`. راجع إجراءا منح الإذن وإزالته [بناء التطبيقات › التفاعلات والأتمتة]. *(Since 1.10)*

لماذا رُفض حفظ تفاعل بالرمز `REACTION_AUTHORING_FORBIDDEN`؟

لأن إنشاء التفاعل أو تحديثه أو حذفه يتطلّب وصولًا فعليًا بمستوى **Developer أو أعلى** على كيانه، والمستدعي لا يملكه. ويُعاد الرفض بالحالة `403`. اطلب مستوى Developer على الكيان (أو وحدته)، أو اطلب ممّن يملكه حفظ التفاعل. راجع حالات رفض مُنمَّطة كانت تُجاب بـ 500 والتفاعلات والأتمتة [بناء التطبيقات]. *(Since 1.8)*

لماذا ظهر تشغيلٌ لتفاعل بحالة *مرفوض* دون أن يُنفَّذ أيٌّ من إجراءاته؟

لأن القاطن بلغ **حدّ تشغيلات التفاعلات الشهري** في باقته (`maxReactionRunsPerMonth`). ويُفرض هذا الحدّ داخل خدمة الأتمتة ولا يتسبّب في فشل أي طلب: يُسجَّل التشغيل بالحالة `refused` (بالرمز `REACTION_RUN_CAP_EXCEEDED`، مع الحدّ والعدد المستهلك والفترة)، ولا يُنفَّذ أيٌّ من إجراءاته. ويمكن العثور على هذه التشغيلات باستخدام المعامل `outcome=refused` في سجلّ التشغيلات، و `status=refused` في سجلّات التفاعلات، والعدّاد `refusedCount` الذي تُرجعه

قائمة التفاعلات إلى جانب `executedCount` . ولا تُحتسب تشغيلات التفاعلات في النشر وحيد القاطن. راجع حدود الخطة: الاستهلاك والرفض. *(Since 1.10)*

هل يستطيع التفاعل إرسال رسالة WhatsApp؟

نعم، باستخدام إجراء **Send WhatsApp**، الذي يرسل أحد **قوالب رسائل WhatsApp المعتمدة** للمساحة. وتُجلب قائمة القوالب عبر `GET /api/reaction/v1/whatsapp/templates` (وصول بمستوى View على النظام أو أعلى)، الذي يُرجِع `409` `whatsapp_not_configured` إلى أن يُعَدّ حساب WhatsApp التجاري للمساحة، ويستبعد قوالب فئة المصادقة. ويُهيَّأ الحساب في كيان إعدادات النظام – ولا توجد شاشة إدارة مخصّصة له في 1.10 – وتُسجَّل حالة تسليم كل رسالة في سجلّ رسائل WhatsApp. راجع WhatsApp، وللإجراء نفسه التفاعلات والأتمتة [بناء التطبيقات]. *(Since 1.10)*

لماذا لم يعمل تفاعل `ReactionFailed` حين أوقف حدّ التشغيل الشهري تشغيلًا؟

لأن التشغيل المرفوض ليس تشغيلًا فاشلًا. فحين يستنفد القاطن حصّته الشهرية من التشغيلات، يُسجَّل التشغيل بالحالة `refused` ، ولا يُنفَّذ أيٌّ من إجراءاته، ولا يُنشر حدث «انتهى هذا التفاعل»، فلا يُشغَّل بسببه تفاعل `ReactionFailed` ولا تفاعل `ReactionSucceeded` . لذا راقب التشغيلات المرفوضة باستخدام المعامل `outcome=refused` في سجلّ التشغيلات أو عدّاد التفاعل `refusedCount` . راجع حدّ التشغيل الشهري والتشغيلات المرفوضة. *(Since 1.10)*

لماذا لا يصل إجراء Call API إلى مضيف داخلي؟

لأن إجراء Call API يرفض الوجهات الخاصة والداخلية: عناوين النطاقات الخاصة وloopback وlink-local (ومنها عنوان بيانات السحابة الوصفية) وCGNAT وmulticast والمحجوزة، في IPv4 أو IPv6. وتُفحص أسماء المضيفات عند حلّها، فيُرفض كذلك الاسم العام الذي يشير إلى عنوان خاص، ويُعاد الفحص عند كل إعادة توجيه. ولا يستطيع السماح بالوصول إلى وجهة خاصة بعينها إلا مشغّل المنصة المسؤول عن نشرك، عبر قائمة السماح `CALL_API_ALLOWED_PRIVATE_DESTINATIONS` ، فاطلب ذلك منه. وتنطبق حدود أخرى على كل استدعاء: HTTP(S) فقط، و30 ثانية للتبادل كلّه، وحدّ أقصى قدره 10 MiB للجسم. راجع Call API: ما تسمح به المنصّة [بناء التطبيقات › التفاعلات والأتمتة]. *(Since 1.8)*

الأزرار – الواجهة البرمجية والأخطاء

لماذا أُزيل زر نظام بصمت عند حفظ المعرض؟

خمسة من معرّفات أزرار النظام **ضمنية** – `VIEW` و`EDIT` و`ARCHIVE` و`CREATE` و`DUPLICATE` . وهي ليست أزرارًا مخزَّنة، بل تُشتقّ من إعداد المعرض نفسه. فإن كانت حمولة الحفظ لديك ما زالت تتضمّن زرًا لم يعد الإعداد الحالي يسمح به – كأن يبقى زرّ `EDIT` أو `DUPLICATE` بعد تعطيل العملية المقابلة – فإن الحفظ الآن **ينجح ويقتطع الزر الضمني غير المسموح** بدل أن يفشل بخطأ «زر النظام غير موجود». ولا ينطبق هذا الاقتطاع إلا على تلك المعرّفات الخمسة: فأزرار النظام المخصّصة وأزرار القوالب لا تُقتطع أبدًا، وتظلّ الإشارة إلى زر غير موجود خطأً. راجع معارض البيانات [بناء التطبيقات › معارض البيانات وأوجهها]. *(Since 1.8)*

الأذونات والمشاركة والبيانات الحسّاسة – الواجهة البرمجية والأخطاء

هل توجد واجهة API لإدارة منح الحقول الحساسة؟

نعم – عبر نقاط النهاية `/v1/sensitive-field-grants*` ، التي تعتمد عليها صفحة **الأذونات الحساسة** لدى المدير وتبويبات كل مستخدم وكل مجموعة. يُعيد `GET` جميع المنح السارية على مستوى المساحة، أو المنح المباشرة لمستخدم

واحد والمنح التي يرثها من مجموعاته، أو المنح الخاصة بمجموعة، أو المُخوَّلين الذين تشملهم منحة شاملة تغطّي حقلًا بعينه. ويُنشئ `POST` المنح (مع الإبلاغ عن الصفوف المُنشأة والمكرّرات المتخطّاة)، ويحدّث `PATCH` المدّة فقط، وثمة مساران منفصلان للحذف النهائي وللأرشفة. وهي لا تُضيف **أي تخزين جديد** – إذ يبقى `AccessRight` مصدر الحقيقة الوحيد، وهذه طرق جديدة لقراءة السجلّات نفسها التي يكتبها محرّر الحقول والكتابة فيها. وبخلاف محرّر الحقول، تستطيع هذه النقاط إنشاء نطاقات شاملة ونوافذ صلاحية. راجع الحقول الحساسة [بناء التطبيقات ‹ حماية حقل]. *(Since 1.8)*

لماذا يُرفض منح إذن بالرمز `PERMISSION_GRANT_EXCEEDS_CALLER_ACCESS` ؟

لأنك إذا كنت دون مستوى **Admin** على النظام، فلا تستطيع منح مستوى أعلى من مستواك الفعلي على الهدف، ولا تغيير منحة تتجاوز ذلك المستوى أو إزالتها. ويُعاد الرفض بالحالة `403`، وتتضمّن تفاصيله `callerLevel` و`requestedLevel` و`currentLevel`. ويُحتسب مستواك الفعلي استنادًا إلى منح النظام لديك، ومنحك المباشرة ومنح مجموعاتك على الهدف، وما ترثه من وحدته أو كيانه؛ وعند العمل بموجب تفويض، تُحتسب أذونات التفويض. ويسري السقف نفسه أصلًا على التفاعلات التي تضبط الأذونات. راجع لا تتجاوز المنحة مستواك أنت [الإدارة ‹ الأذونات والوصول]. *(Since 1.8)*

الصفحات والواجهات – الواجهة البرمجية والأخطاء

ما الخاصية `layouts` على `GET /v2/views/end-user/{id}` ، وهل أستطيع استخدامها؟

بالنسبة إلى المعرض المبني على قالب العرض **Layout builder**، يحلّ ذلك المسار مُدخلات SHELL التي يسمّيها القالب (`layout_list` و`layout_view` و`layout_edit`) إلى `layouts` – `{listEntry, viewEntry, editEntry}`، وكلٌّ منها `{id, name, definition}` أو `null` حين لا يُربط مُدخل صالح. ولا يخزّن القالب إلا المراجع، لذا ينعكس تعديل التخطيط على كل معرض مرتبط دون إعادة حفظه، ويتلقّى الزوار المجهولون التعريف أيضًا؛ ولا يتضمّن مسار v1 الخاصية `layouts`. ويتوفّر الجزء الخاص بالخادم في 1.10، لكن **لا يستخدمه أي عميل مُصدَر بعد**: فقالب Layout builder وأداة بناء الواجهة اللذان يقرآن `layouts` ما يزالان `Since dev`. راجع مراجع التخطيط على مسار المستخدم النهائي [بناء التطبيقات ‹ معارض البيانات وأوجهها]. *(Since 1.10)*

صيغ التحزيم وواجهاتها البرمجية

كيف أنقل تطبيقاً بين البيئات (تطوير ← تجهيز ← إنتاج)؟

استخدم التحزيم. فهو يجمع الموارد المختارة – الوحدات والكيانات والمعارض والاستعلامات والعلاقات والتفاعلات والتقارير والصفحات والأزرار والتجميعات والدوال وغيرها – مع جميع اعتمادياتها في ملف ZIP واحد تصدّره من بيئة وتستورده في أخرى. ويجري الاستيراد عبر تدفّق موجّه قائم على جلسات (تحقّق ← تهيئة الوصلات ← تنفيذ)، مع إعادة ربط معرّفات المصدر بمعرّفات الهدف أثناء التنفيذ. وشغّل دائماً **تحليل** التصدير أولاً وعالج تحذيرات الاعتماديات الخارجية قبل نقل الحزمة.

ما الذي يُترَك عند تصدير حزمة؟

تُستبعد بعض الموارد حتى لو اخترت: الكيانات **المؤرشفة** (وشجرتها المتفرّعة)، والموارد **المرتبطة بمستخدم**، والكيانات/التفاعلات **الضمنية**. ولا تُكتَب كيانات الهوية الأساسية (المستخدمون، المجموعات، الوحدات التنظيمية، ساعات العمل) داخل الحزمة أبداً؛ فهي موجودة بصورة مستقلة في كل بيئة وتُعامَل بوصفها مراجع، وهنا يأتي دور ربط الأشخاص وملفات الربط (mapping profiles). ولا يُضمن انتقال سوى الاعتماديات الصلبة مع الحزمة؛ أما المراجع الرخوة (الاختيارية) فقد تُفرَّغ عند الاستيراد.

لماذا أدّى اختيار هدف استيراد لمستخدم أو مجموعة مشتركة إلى إعادة توجيه مرجع مورد آخر أيضاً؟

لأن الموردين كانا مرتبطين بقرار **الصفّ** نفسه لذلك الرئيس. فحين يُشار إلى المستخدم أو المجموعة أو الوحدة التنظيمية نفسها من أكثر من **مورد** في الاستيراد الواحد – كمالك خطوة سير عمل على كيان، والمستخدم الذي يقتصر عليه معرض – أصبحت نسخة كل مورد من ذلك المرجع قابلة لإعادة التوجيه على حدة؛ إذ يتعرّف المعالج على ظهور الرئيس ضمن أكثر من صفّ مورد، ويجعل مجموعة كل مورد تقرأ مواضع ظهوره وتكتبها وحدها. أمّا الرئيس المحصور في مورد واحد فلا يتأثر ويظلّ خاضعاً لقرار الصفّ، وهو أيضاً القرار الذي يحتفظ به ملفّ تعريف الربط المخزَّن.

ويُحتفَظ بافتراضي الصفّ حين تنحرف عنه إحدى المجموعات. فالاختيار على مستوى موضع الظهور ليس إلا تجاوزاً *جزئياً* يُطبَّق فوق قرار الصفّ، ويرفض الخادم أي إعادة توجيه على مستوى العنصر لا تحمل هدفاً خاصاً بها. ولذلك فإن أول هدف تختاره لرئيس مشترك يحدّد كذلك الخيار الافتراضي للصفّ، ثم تحيد مجموعة لاحقة *عن* ذلك الخيار بدلاً من أن تحلّ محلّه؛ وبذلك يبقى الاختيار المتَّخذ في «الإعداد» سارياً حتى «التنفيذ». أمّا **الإنشاء التلقائي** لرئيس يظهر في عدّة موارد فيُطبَّق دائماً على مستوى الصفّ، لأن مفتاح المُدخلات المُنشأة تلقائياً هو الرئيس نفسه، لا موضع ظهوره. راجع التحزيم. *(Since 1.8)*

هل أستطيع استيراد حزمة بمخطط 1.2 إلى بيئة لم تُرقَّ بعد؟

فقط إذا كانت البيئة الهدف تعمل بالإصدار **1.10 أو أحدث**. أما عمليات النشر السابقة للإصدار 1.10 فما تزال تقبل `1.0` و`1.1` وحدهما، وتُرفض فيها أي حزمة تعلن `1.2` بخطأ حاجب `INCOMPATIBLE_SCHEMA` قبل أي بحث في المساحة الهدف. وتحدّد قائمة السماح الإصدارات المقبولة صراحةً، ولا تتحقّق من وقوعها ضمن نطاق؛ فالإصدار «الأحدث» ليس بالضرورة «مسموحاً به». ولهذا يحدّد التصدير **أدنى** إصدار يصف الحزمة – `1.2` فقط حين تتضمّن فعلاً الملفّ الجانبي `field-mapping-hints.json`، و`1.1` فيما عدا ذلك – فيبقى التصدير الخالي من الحقول المربوطة قابلاً للاستيراد في بيئة لم تُرقَّ. وتترتّب على ذلك القاعدة التشغيلية: **رقِّ البيئة الهدف قبل المصدر**. وينطبق الأمر نفسه على `1.3` و`1.4` – اللذين يعلنهما، على الترتيب، الإصدار المنشور من حزمة وأي حزمة تحمل README – إذ ترفضهما البيئة الهدف السابقة للإصدار 1.10 أيضاً بالرمز `INCOMPATIBLE_SCHEMA`؛ راجع ما الذي يضيفه 1.3 و1.4. راجع إصدارات المخطط. *(Since 1.10)*

هل يمكنني نشر قالب في المتجر وتثبيت التطبيقات من متجر داخل التطبيق؟

نعم، في 1.10: تملك **الحزمة** وحدات، وتمرّ **إصداراتها المنشورة** (و**صفحة متجرها**) بصندوق مراجعة داخلي قبل أن تصل إلى المتجر، ويثبّت **المتجر داخل التطبيق** من الفهرس. والمتجر **موقوف إلى أن يفعّله مشغّل** (ويُجيب كل مسار للمتجر في الأثناء بالرمز `503 MARKETPLACE_DISABLED`). وحين يُفعَّل:

- **التصفّح لا يحتاج إلى ربط**: أما التثبيت فيحتاج إليه، ويربط التثبيت الأول البيئة بعد تأكيدك.
- **الربط والنشر منفصلان.** فالربط يتيح للبيئة التثبيت فقط؛ ويحتاج النشر أيضاً إلى **ناشر** يُعَدّ بعد ذلك، وإلى استحقاق الباقة `marketplacePublish`.
- **تأتي الأذونات من ثلاث مجموعات نظام** – الناشرون والمراجعون والمثبِّتون. ويحتاج النشر والمراجعة إلى العضوية في المجموعة **حتى لمسؤول النظام**.
- **تُقبَل شروط الناشر عند الاعتماد**، ويقبلها المراجِع حينئذ، لا عند إرسال الطلب.
- **يُحدَّث التطبيق المثبّت في مكانه** إلى إصدار أحدث مدرَج؛ فلا حاجة إلى إزالته ثم إعادة تثبيته.

راجع الحزم والمتجر والمتجر داخل التطبيق. *(Since 1.10)*

لماذا لا أستطيع تعديل مورد مثبّت من حزمة أو تصديره؟

لأن التثبيت يقفل الموارد التي أنشأها؛ إذ يرفض data-service وreaction-service تحديثها أو حذفها بالرمز `403 PACKAGE_RESOURCE_READ_ONLY` لكل مستدعٍ، بمن فيهم مسؤولو النظام، ويُمنع الاستيراد العادي فوقها بالرمز `403 EXPORT_INSTALLED_RESOURCE`. وتبقى بعض خصائص المعرض أو الصفحة المقفلة تحت تحكّمك – حالة النشر وترتيب العرض والجمهور وحقوق الوصول، والاسم والوصف والأيقونة – ويحتفظ بها تحديث الحزمة. ولتغيير ما سوى ذلك **اشتقّ** التثبيت (`POST /v1/installations/{id}/fork`): فتصبح موارده قابلة للتعديل، لكنها لا تعود تتبع الحزمة ولا تتلقّى تحديثات ولا يمكن تصديرها أو نشرها أبداً. راجع وضع التثبيت والاشتقاق والإزالة. *(Since 1.10)*

من يستطيع تصدير حزمة، ومن يستطيع استيرادها؟

منذ 1.10 أصبح التصدير إجراءً **للمطوّر**: فمن يملك منحاً على مستوى النظام بدرجة Developer أو أعلى يستطيع تصدير أي جذور، ومن كان منحه على مستوى النظام بدرجة None يُرفض طلبه، وفيما عدا ذلك يتطلّب تصدير كل جذر درجة Developer أو أعلى على الوحدة أو الكيان المالك له. أما الاستيراد، فالتحقق والإعداد والتنفيذ وإعادة المحاولة والإلغاء كلها تتطلّب منحاً إدارياً على مستوى النظام، لأن عمليات الكتابة أثناء الاستيراد تتجاوز المنح على مستوى المورد. راجع إذن التصدير وحواجز الوصول والتنفيذ في الاستيراد. *(Since 1.10)*

هل أحتاج إلى ربط بالمتجر لتصفّح المتجر داخل التطبيق؟

لا. فحين لا تكون البيئة مربوطة – لم تُربط قطّ، أو أُلغي ربطها – يُقرأ الفهرس والناشرون وصفحات التطبيقات دون هوية من الفهرس العام للمتجر. ولا يحتاج إلى ربط حيّ إلا **التثبيت**: فعند الضغط على **تثبيت** في بيئة غير مربوطة بعد، يُطلب منك التأكيد في حوار *الربط بالمتجر*، ثم تُربط البيئة ويجري التثبيت. والربط بهذه الطريقة لا ينشئ ناشراً. ويبقى المتجر محتاجاً إلى أن يفعّل المشغّل المتجر، وإلى المساحة الأساسية، وإلى إذن `install`. راجع مسارات المتجر والتثبيت من المتجر يربط أولاً. *(Since 1.10)*

نحن مربوطون بالمتجر – فلماذا يُرفض إرسال إصدار للمراجعة؟

لأن الربط لا يجعلك ناشراً. فالاتصال الجديد **للتثبيت فقط**؛ ويُرفض تحضير طلب النشر بالرمز `409 MARKETPLACE_PUBLISHER_REQUIRED` إلى أن **يُعِدّ** أحدٌ **النشر** (`POST /v1/marketplace/publisher`)، فيختار اسم الناشر وعنوان المتجر اختيارياً. ويحتاج الإعداد إلى مسؤول نظام أو حامل `publish`، وإلى استحقاق الباقة `marketplacePublish`؛ وبعده لا يستطيع إعادة تسمية الناشر أو تغيير عنوان المتجر إلا مسؤول النظام. وثمة رفضان آخران شائعان في هذه المرحلة: `403 MARKETPLACE_PUBLISH_NOT_IN_PLAN` حين لا تشمل الباقة النشر، و`409 PACKAGE_OWNED` حين يملك التطبيقَ ناشرٌ آخر في المتجر. راجع الربط وإعداد النشر. *(Since 1.10)*

لماذا عُطّل *إرسال للمراجعة* مع تحذير «النشر موقوف»؟

لأن المتجر أوقف ناشرك عن النشر دون إنهاء الاتصال. فتعرض صفحة الحزمة تاريخ الإيقاف وسبب المتجر كما كُتب، وتعرض بطاقة الاتصال وسم *النشر موقوف*؛ ولا يمكن إرسال إصدارات جديدة ولا تغييرات صفحة المتجر حتى يعيد المتجر النشر. ويبقى تثبيت التطبيقات وتحديثها متاحًا. وليست بيئتك هي التي تقرّر ذلك: فالمتجر يرفض الإرسال (`502 MARKETPLACE_REFUSED`)، فالحلّ عند المتجر. متاح من الإصدار 1.10.2. راجع إيقاف المتجر للنشر. *(Since 1.10)*

أنا مسؤول نظام – فلماذا لا أستطيع إرسال حزمة للمراجعة أو اعتمادها؟

لأن النشر والمراجعة يأتيان من **العضوية في المجموعة وحدها**. فمسؤول النظام لا يملك دون الانتماء إلى مجموعة إلا `install` و`update`؛ ويحتاج `publish` إلى العضوية في `MARKETPLACE_PUBLISHERS`، و`review` إلى العضوية في `MARKETPLACE_REVIEWERS`، وإلا كان الجواب `403 MARKETPLACE_PERMISSION_REQUIRED`. أضف نفسك إلى المجموعة؛ تُخبَّأ العضوية مدة 60 ثانية، لكن مسؤول النظام الذي يُرفض يعيد قراءة مجموعاته مرة واحدة متجاوزًا الخبء، فلا انتظار. وتُطلب **شروط الناشر** عند الاعتماد – إذ يقبلها المراجِع في حوار الاعتماد حين يكون الجواب `409 MARKETPLACE_TERMS_REQUIRED` – لا عند إرسال الطلب. راجع مجموعات المتجر الثلاث وطلبات النشر وصندوق المراجعة. *(Since 1.10)*

كيف أنقل تطبيقاً مثبّتاً إلى إصدار أحدث؟

حدّثه **في مكانه** – دون إزالته ثم إعادة تثبيته. فتطبيق المتجر المثبّت الذي يتوافر له إصدار أحدث مدرَج يعرض **تحديث** في صفحة تثبيته وفي صفحة تطبيقه في المتجر؛ وكلاهما يفتح معاينة للتحديث تتضمّن ملاحظات الإصدارات بين الإصدارين. وعبر الواجهة البرمجية: أنشئ جلسة تحديث (`POST /v1/installations/{id}/update-sessions`)، ثم شغّلها (`POST /v1/installations/{id}/update`)، مع توافر مسار لإعادة محاولة التحديث الفاشل. وتتطلّب مسارات التحديث أن تكون مسؤول البيئة في المساحة الأساسية **مع** إذن `update` (مجموعة المثبّتين). ولا يمكن تحديث التثبيت المشتق (`409 INSTALLATION_FORKED`)، ولا التحديث إلى إصدار غير أحدث (`409 UPDATE_NOT_NEWER`). راجع تحديث التطبيق المثبّت في مكانه. *(Since 1.10)*

فشل تثبيت – فكيف أستعيده دون أن أترك نصف تطبيق خلفي؟

تبقى الموارد التي أنشأها التثبيت الفاشل مقفلة، فلا يُترك شيء دون حماية. وتُظهر صفحة التثبيت خيارات الاستعادة: **أعد المحاولة** من جلسة استيراده ما دامت موجودة، أو **ارفع ملف الحزمة نفسه مرة أخرى** لاستكمال التثبيت الفاشل. وحين يتوفّر **إصدار أحدث** متوافق، يُعرض **تحديث** بدلاً من ذلك: فيكتب فوق ما أنشأه التشغيل الفاشل ويثبّت الباقي، فلا تعود معالجة الفشل الناجم عن الحزمة نفسها تتطلّب الإزالة أولاً. وتبقى الإزالة ممكنة من الحالة `failed`، وتعرض الإزالة غير النشطة خيار *إعادة المحاولة*. وقبل بدء التثبيت، يبيّن فحص التثبيت في المتجر (`POST …/install-check`) ما إذا كان يمكن تثبيت الإصدار الآن، والأسماء المتعارضة. راجع قائمة المثبتة وصفحة التثبيت وفحص التثبيت من المتجر. *(Since 1.10)*

كيف أُبلغ ناشرًا بأن تطبيقه فشل في التثبيت؟

من صفحة التثبيت: فالتثبيت الفاشل من المتجر، أو التحديث غير النشط من المتجر، يعرض **إبلاغ *الناشر***. ويعاين أولًا ما سيُرسَل بالضبط (الحزمة وإصدارها، والخطأ، والموارد التي فشلت في التشغيل)، ويمكنك إضافة ملاحظة واختيار مشاركة اسمك وبريدك الإلكتروني أو عدمها. ويلزمك أن تكون مسؤول بيئة في مساحة ROOT مع اتصال حيّ بالمتجر؛ ولا يلزم استحقاق النشر. وإن تغيّر الخطأ منذ المعاينة يُرفض الإرسال بـ `409 REPORT_STALE`، فعاين مجدّدًا. متاح من الإصدار 1.10.1. راجع الإبلاغ عن تثبيت أو تحديث فاشل إلى ناشره. *(Since 1.10)*

هل أستطيع إزالة تثبيت اشتققته؟

نعم. فالإزالة مسموحة من `active` أو `failed` أو `forked` أو `uninstalling`، وتتطلّب أن تكون مسؤول البيئة. وعند إزالة التثبيت المشتق يحذّر الحوار من أن **تغييراتك على الموارد المشتقة تُحذف معها**. وفي الحالتين يسرد الحوار ما سيُحذف، والمُدخلات التي أنشأها المستخدمون وستُحذف معه، والروابط من خارج الحزمة؛ ويبقى زرّ **الإزالة** معطّلاً حتى تُزال تلك الروابط الخارجية، ويرفض الخادم الاستدعاء الذي يتجاوز الحوار بالرمز `409 UNINSTALL_BLOCKED_BY_OUTSIDE_LINKS`. وتُستبقى الوحدة التي ما تزال تضمّ كيانات أنشأها مستخدموك. راجع الاشتقاق والإزالة. *(Since 1.10)*

كيف أجد التثبيتات الفاشلة أو المُزالة؟

في صفحة **المثبتة** (`/admin/packages/installed`)، التي يمكن ترشيح محتواها من شريط الأدوات: قائمة اختيار **الحالة** (*المثبتة* و*الفاشلة* و*المُزالة*) وقائمة اختيار **المصدر** بجانب البحث والترتيب. وتبدأ القائمتان فارغتين، فتظهر جميع الحالات والمصادر حتى تختار قيمة في إحداهما؛ ولا يوجد خيار *الكل*. ويُظهر كل صفّ من ثبّت الحزمة، وناشرها في حال التثبيت من المتجر. والمكافئ في الواجهة البرمجية هو `GET /v1/imports` مع `state` (`installed` أو `failed` أو `uninstalled`) و `source` . (كانت هذه الصفحة تذكر أن 1.10 يعرض تبويبات للحالة؛ وقائمتا شريط الأدوات موجودتان في الإصدار `1.10.0` .) راجع قائمة المثبتة وصفحة التثبيت وتحسينات المتجر و«المثبتة». *(Since 1.10)*

ماذا يحفظ السجل التاريخي للحزمة وسجلّات التصدير، وهل يحرّر حذف إصدار أو تصدير ملفّ ZIP الخاص به؟

يسجّل الشريط الجانبي **للسجل التاريخي** لصفحة الحزمة أحداث دورة حياتها: الإنشاء، والتعديل (مع بيان ما تغيّر)، وتعديل README، وإنشاء الإصدارات وإهمالها والتراجع عن إهمالها وحذفها، وأحداث المراجعة. ويُنشئ كل تصدير **سجلّ تصدير** (`GET /v1/exports`)، ويُدرَج في تبويب *التصديرات* بصفحة البناء)، يتيح إعادة فتح التصدير نفسه وتشغيله من جديد.

وحذف أيٍّ منهما **يزيل السجلّ وحده – ويبقى ملف ZIP في التخزين**، لأن النسخ الاحتياطية للقاطن تسرد الملفات المخزّنة دون نسخها. ولا يُحذف الإصدار إلا إذا لم يتلقّه المتجر قطّ: فالإصدار المُرسَل للمراجعة أو المثبّت في مكان ما يُرفض حذفه (`VERSION_SENT` ، `VERSION_INSTALLED`)، ورقم الإصدار المحذوف لا يُعاد استخدامه أبداً. راجع سجلّات التصدير وصفحة البناء وصفحة الحزمة. *(Since 1.10)*

لماذا يتوقّف الاستيراد عند *اختر حقلاً* لحقل ربط؟

لأن الاستيراد لم يستطع تحديد الحقل الذي ينسخ منه حقل الربط على الكيان الأب الهدف، و**المثبِّت هو من يقرّر الآن**. فالاستيراد يطابق حقل المصدر بالاسم أولاً؛ وحين يتعذّر ذلك تُبلّغ مرحلة التحقق عن الحقل بالحالة `unmatched` أو `unnamed` وتُصدر الخطأ الحاجب `MAPPED_FIELD_SOURCE_UNDECIDED` ، فيبقى `canProceed` بالقيمة `false` . اختر **النسخ من** حقل على الأب الهدف – ولا تُعرض إلا الحقول من النوع نفسه، ذات أهداف علاقة متوافقة، وبإعداد الإلزام نفسه في حالات الربط ثنائية الاتجاه – أو **عدم النسخ**. ويُعاد الفحص عند التنفيذ وإعادة المحاولة (`409 MAPPED_FIELD_SOURCE_UNDECIDED`). ويُدرج التثبيت أو التحديث من المتجر الذي يجري دون المعالج الحقلَ ضمن القرارات التي عليك اتخاذها في صفحة الاستيراد. أما التحذير غير الحاجب السابق `MAPPED_FIELD_SOURCE_UNRESOLVED` فلم يعد موجوداً في 1.10. راجع حقول الربط: المثبِّت يقرّر وواجهة الاستيراد البرمجية. *(Since 1.10)*

هل يمكن أن تحمل الحزمة صفحات لا تنتمي إلى أي وحدة؟

نعم. فالصفحة **العامة** – التي لا تقع ضمن نطاق وحدة أو كيان – يمكن أن تكون جذراً في الحزمة، وحدها أو إلى جانب الوحدات، وقد تقتصر الحزمة على صفحات. ويُرجِع حفظ حزمة تُحدّد صفحةً تنتمي إلى وحدة أو كيان الخطأ `422` `PACKAGE_PAGE_NOT_GLOBAL`، ويرفض النشر الصفحةَ التي انتقلت لاحقاً إلى أحدهما (`PUBLISH_PAGE_NOT_GLOBAL`) أو التي ثُبّتت هي نفسها أو اشتُقّت من حزمة أخرى. وتُثبَّت القطعة ذات الاستعلام الإحصائي التي تشير بياناتها إلى خارج الحزمة **غير مهيّأة** بدلاً من أن يفشل التثبيت. راجع الصفحات العامة في الحزم [بناء التطبيقات › الصفحات والقطع]. *(Since 1.10)*

هل تنتقل أسرار الوظيفة داخل الحزمة؟

لا. في فرع التطوير تحمل الوظيفة المحزومة **أسماء** أسرارها فقط: فتُصدَّر كل قيمة على هيئة `null` ، ويُستبعد `OSOS_API_TOKEN` الخاص بالوظيفة وهويتها، لأن القيم المشفّرة مرتبطة بمفاتيح البيئة المصدر. ويُزيلها التثبيت مرة أخرى،

فلا تستطيع حتى حزمة أقدم إدراجها. اضبط القيم في كل بيئة هدف بعد التثبيت. راجع أسرار الوظيفة تبقى في بيئتها. *(Since dev)*