

Building Event-Driven Workflows with OData Webhooks

Romain Dunand — 1-more-thing



Vienna Calling 2026

Vienna, May, 27-30, 2026

Demo



Romain Dunand — 1-more-thing

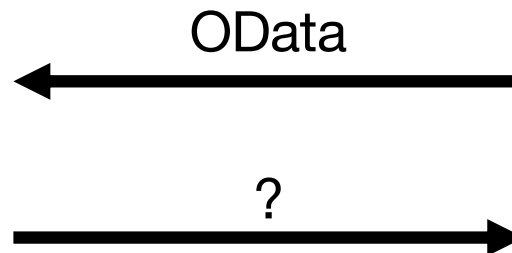
- FileMaker developer since 2004
- Architect — fmcloud.fm
- Expert API, automatisisation & FileMaker Server



Why OData webhooks changes the way to do

Genesis

Thanks Claris Studio !



The gift

A native table trigger inside the Database Engine

- Triggered on any data/structure change (interface, script, API, import...)
- UX agnostic
- OData => Web API

Before vs. then

A paradigm change

BEFORE : triggers on objets, OnRecordCommit, OnWindowTransaction

✗ Depends on the UX, there are unaddressed edge cases

THEN : It doesn't matter where this change originated

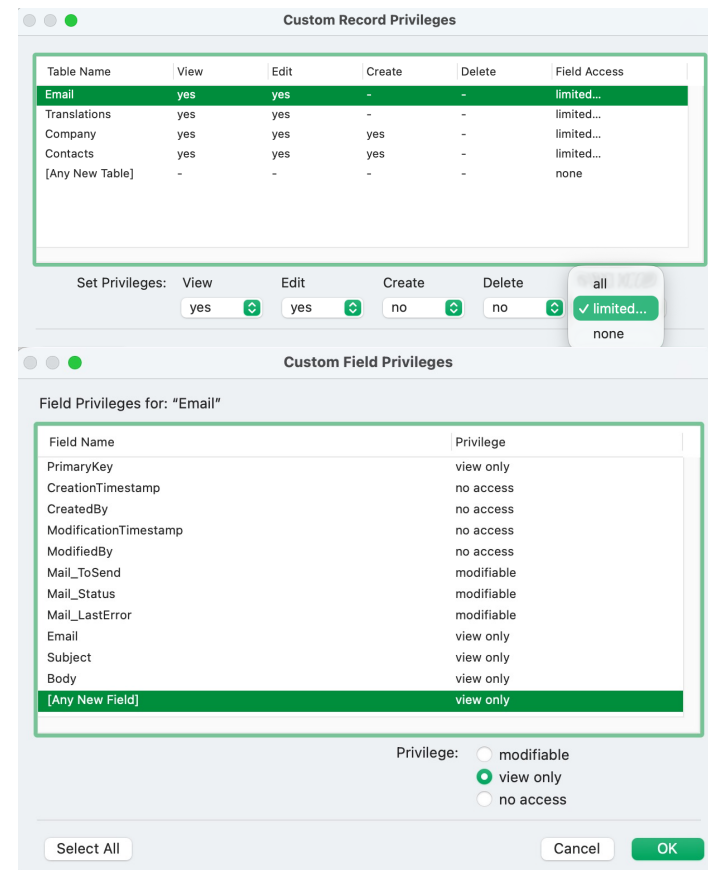
💕 FileMaker natively push

Minimal « clean » setup

Setup FileMaker

One account with oData access

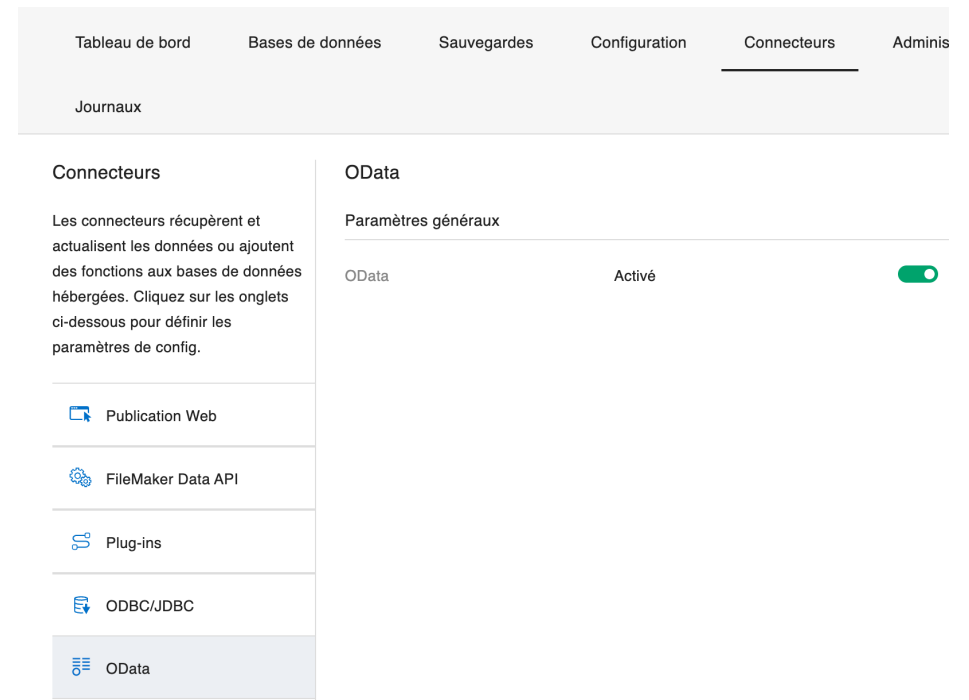
- Dedicated technical account (recommended)
- Restricted privilege set
 - Extended privilege: fmodata
- Avoid unstored calculation fields
- No script access — unnecessary with webhooks



Server Setup

That's all

- Enable OData in FileMaker Server
- Ready to go



Key Considerations

- « [Any new field] = No access » → breaks webhooks (silently) ⚠
- Summary fields are never sent (no context -> no sense)

Create & manage webhooks

What are « OData webhooks »

- These are subscriptions to FileMaker events.
- Triggered by a data or table structure change.
- They call a URL (external API, third-party service...).
- Odata endpoints: used solely to declare/manage subscriptions and apply permissions.

Subscriptions lifecycle

5 actions

- Webhook.Add → Subscribe to a table
- Webhook.GetAll → List subscriptions
- Webhook.Get → Get a subscription detail
- Webhook.Invoke → Trigger webhooks on specified record IDs (great for testing)
- Webhook.Delete → Unsubscribe

Demo

Key Considerations

Avoid the wall

- You can have multiple subscriptions for the same table
- Webhook config in (don't play with it !):
`Data/Preferences/ClarisWebhooks.json`
- ⚠ Only works with local tables
- ⚠ No de-duplication — Check if your subscription already exists first !

What and how is it sent ?

What is sent ?

A batched and delayed notification

- Not a transaction log
- Final state of records when data are pushed
- Batch every ~10 secondes
 - 👉 If a record is modified 3 time within the 10s delay → 1 push with the final state
 - 👉 Not suitable for a sync or auditlog mechanism

4 types of events

- Record created (POST)
- Record modified (PATCH)
- Record deleted (DELETE) => only send the recordID!
- Table modified (POST)

Demo

Key Considerations & limits

This is not « real time »

But it is reliable

- Batch every ~10 s – 200 records max/call
- Ordered by RecordId (not the commit/change order)
- Sequential sending: one subscription at a time.
- Queue preserved
 - if webhook unavailable (timeout)
 - Webhook response code = 401
- ⚠ Any other response code threatened as sent !!! 😬

Limits

- Final State — Not an auditlog
- No insight on what was actually modified
- DELETE only send the RecordId
- Does not work on External data sources

Pitfalls

- Filter (re)evaluated before send : your record can disappear from the queue
- Duplicated subscriptions : Same event received multiple times
- « Any new field = No access » → webhooks broken

Conclusion

The real shift.

- Table triggers not depending on the UX
- Works regardless of their origin (API, ODBC, import, etc.)
- Centralized the logic at the data level
- Not a single line of script in FileMaker!

It's up to you to imagine what comes next...

But here's a few ideas

- Trigger a business workflow pipeline in an external system (Connect, n8n, industrial robot, etc.)
- Replicate your data into external databases (FM, SQL, etc.)
- Update a website
- “Real-time” alerts (Slack, email, notifications)
- Expose data to other developers

Q & A

OData Webhooks



Vienna Calling 2026

Vienna, May, 27-30, 2026