

NACARSOFT ADABAS SUITE

Your Adabas data, live and where you need it

NSAdaSync replicates every change in your Adabas files to PostgreSQL, Oracle, MySQL, MongoDB, Elasticsearch or Kafka as it happens.

[SEE FEATURES →](#)

9 targets · From the nucleus to your data in seconds



ADABAS

Open Systems



NSAdaSync

broker + workers



PostgreSQL

Relational SQL

ORACLE

Oracle

Relational SQL



MySQL

Relational SQL



MariaDB

Relational SQL



SQL Server

Relational SQL



SQLite

Local SQL



MongoDB

NoSQL · documents



Elasticsearch

Search · indices



Kafka

Streaming · topics

...and one more target whenever you need it

Your Adabas data never reaches where it is queried

01

Batch-only consumption

SQL reports are fed by nightly loads: by the time you query it, the data is already stale.

02

No streaming

Oracle, MongoDB, Elasticsearch or Kafka do not receive nucleus changes as they happen.

03

No control

You do not know what has been replicated, what is queued or what failed until you check it by hand.

- Hours of data lag

- Reports over snapshots

- Zero visibility of the flow

QUÉ ES

Live Adabas replication for SQL, NoSQL and streaming

NSAdaSync is Nacarsoft's Adabas replication service. An exit in the nucleus publishes every change to a persistent broker and the worker of each subscription applies it to the target: SQL, NoSQL, search or streaming.

- ✓ Minimal UEX4 exit: never blocks, never retries, never touches the network
- ✓ Embedded broker, in-process, with no TCP ports
- ✓ File → table subscriptions mapped from the dictionary
- ✓ Web console with RBAC, auditing and DLQ with replay

9

supported targets

1

exit in the nucleus

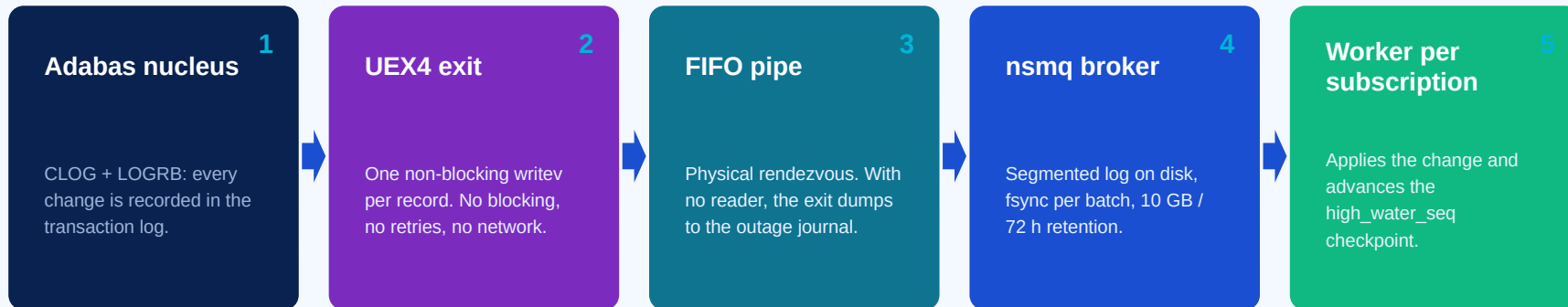
0

changes lost

1

web console

How a change travels, step by step



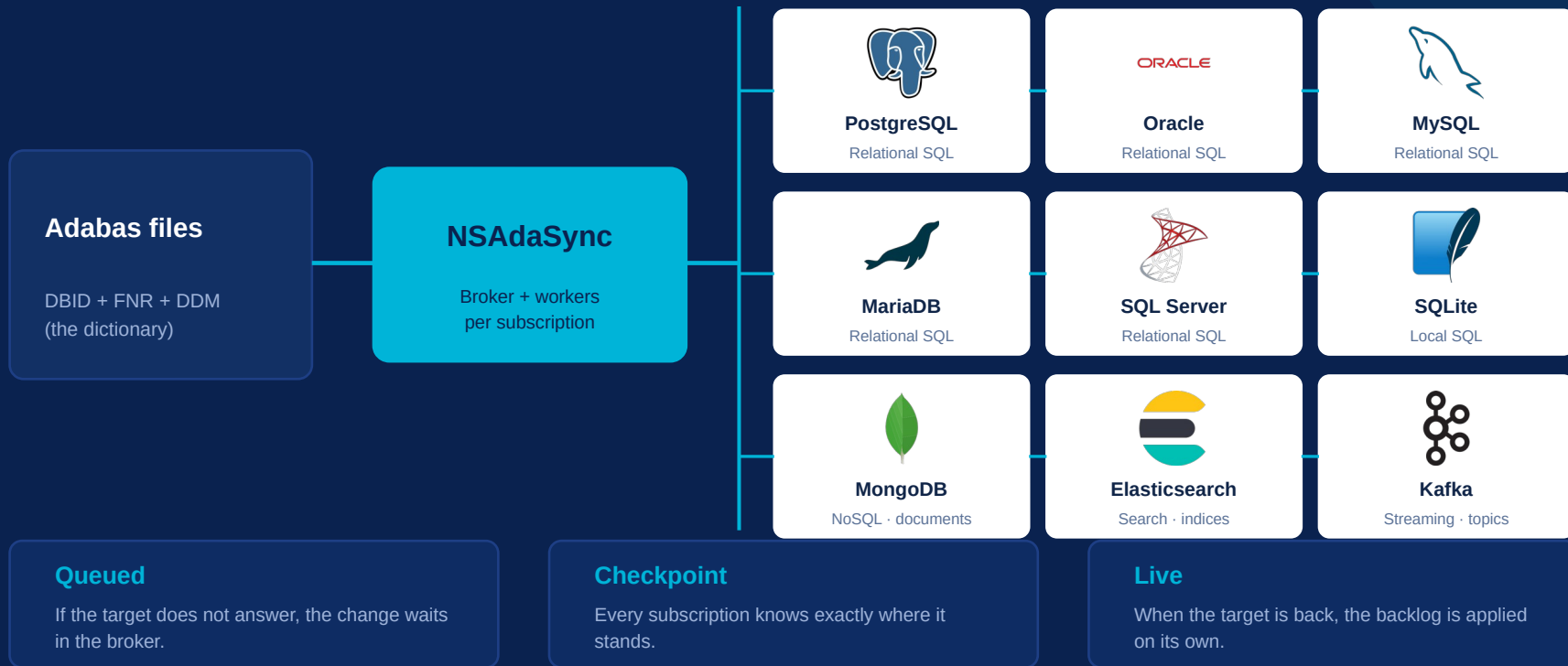
⚠ With no pipe configured the exit stays inert (fail-closed): the nucleus never blocks waiting for anyone.

The nucleus never leaves the network

No clients to install

All in a single server

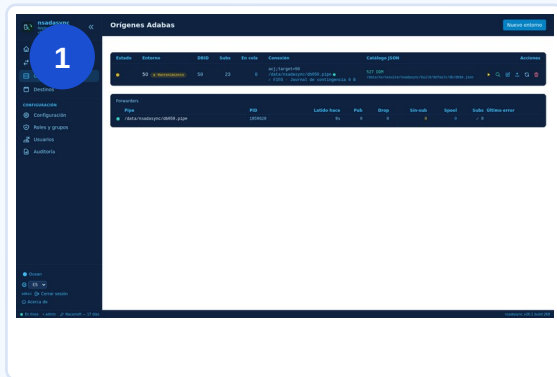
One source, nine targets, a single control point



→ The same subscription is duplicated onto each target: one source, many consumers.

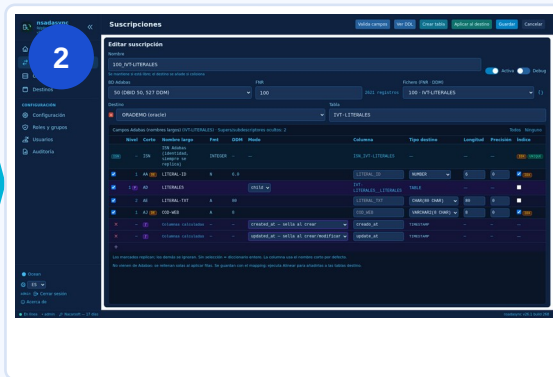
HOW IT WORKS

From the nucleus exit to the target, in three steps



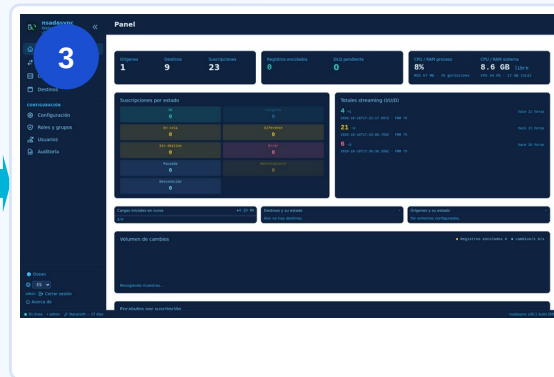
Origin

The Adabas file, its dictionary and the pipe the changes travel through.



Subscription

The file → table mapping, with your own fields and computed columns.



Panel

State, queued records, latency and errors of the whole flow, live.

No nightly scripts and no extractors: one nucleus exit, one persistent broker, and the console does the rest.

SUBSCRIPTIONS

From an Adabas file to a target table



DDM field	Target column	Type
AA001	name	VARCHAR(120)
AA002	surname	VARCHAR(120)
MESES (MU)	MESES_1 ... MESES_8	TEXT

- State per subscription: ok, queued, no target, error
- Last initial load and how long it took
- Insert, update and delete counters
- Control SEQ: where each flow stands

The screenshot shows the "nsadasync" web interface. The main section is titled "Suscripciones" (Subscriptions). At the top, there's a red banner indicating a replication pause for maintenance. Below this, there's a table listing various subscriptions. The table has columns for "Nombre" (Name), "Destino / Tabla" (Destination / Table), "Carga inicial" (Initial Load), "CRUD" (Create, Read, Update, Delete), and "SRO high-water" (Source Record Offset high-water). The subscriptions listed include "100_NF_LITERALES", "120_N2-CIS-2", "50 (120)", "120_N2-CIS-2-2", "50 (120)", "120_N2-CIS-2-MARIA", "50 (120)", "120_N2-CIS-2-MYSQL", and "50 (120)". Each row shows details about the subscription's status, destination table, and performance metrics.

The load first, the live stream after

Prepare + DDL preview

1

Fields are validated against the dictionary and the DDL is previewed without running it.

Ensure table

2

Creates the parent table, child tables and indexes on the target.

Initial load

3

merge, empty or drop modes, with per-target capacity limits.

Live streaming

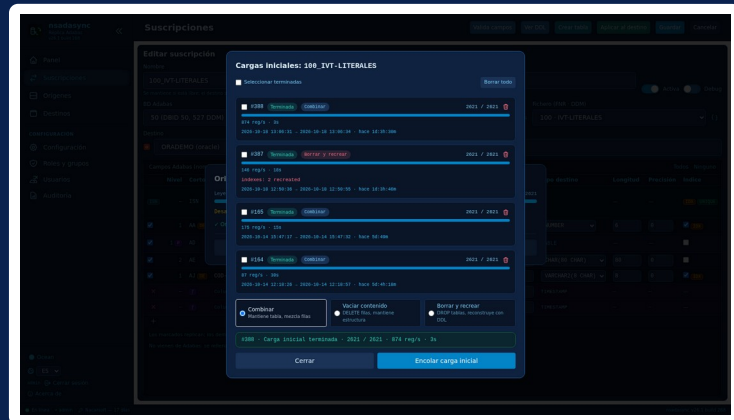
4

Every later change is applied instantly and recorded.

high_water_seq checkpoint

If the process is interrupted, the load resumes from the checkpoint: it never starts over.

✓ Before queuing a load, the target is pinged and the source is checked: if either is down the load is rejected instead of dying halfway.



SCHEMA CHANGES

The schema can change: no reload needed

A · Adabas changes

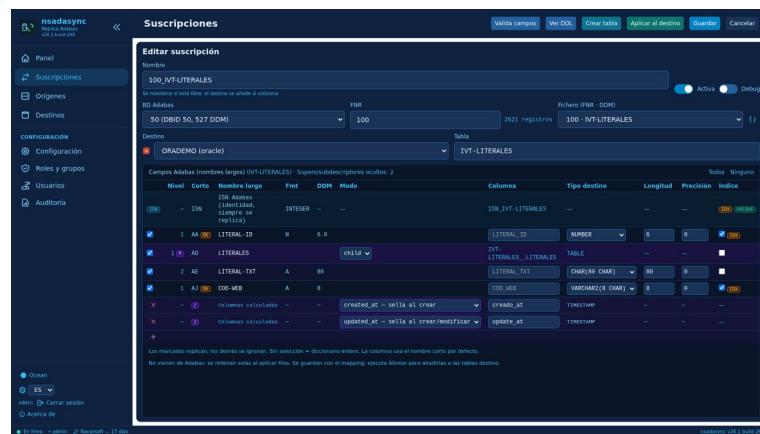
A new field in the file or an updated dictionary: reload the dictionary, add the field to the mapping and «Apply to destination» adds the column to the existing table without touching data, counters or checkpoint. Streaming continues.

B · The target changes

If someone adds a column, the upsert respects it. If the table is dropped, the subscription goes «no table» and the worker parks in the broker: recreate it and the backlog resumes. No reload.

C · An unmapped field arrives

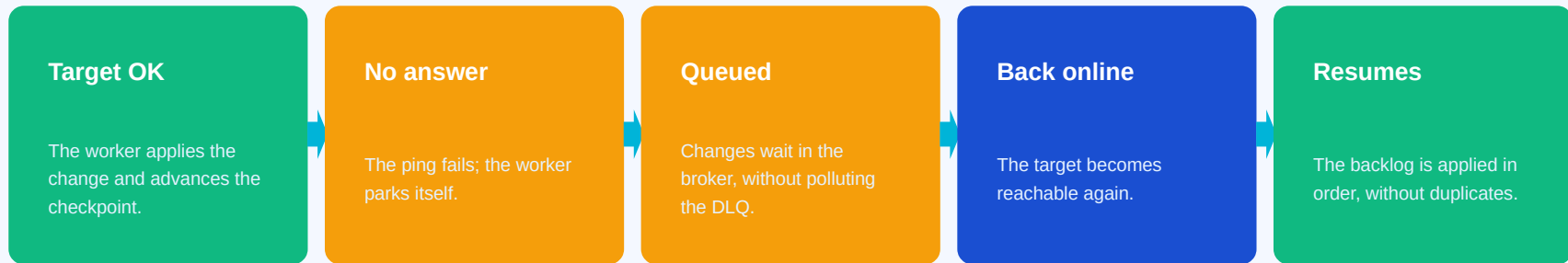
The apply is rejected with «unmapped fields... check catalog» and the entry goes to the DLQ. Fix the mapping and replay it: no initial load either.



The subscription editor: «Validate fields», «View DDL», «Create table» and «Apply to destination».

The initial load is only the backfill. From then on, the schema evolves without loading again.

If a target goes down, the data waits. Nothing is lost



DLQ with replay

If the apply really fails, the entry goes to the error queue with its reason. It is inspected and retried. No event is ever lost.

The console says it itself

«4 not reachable – the changes are queued in the broker:
ELADEMO, MARIADEMO, MSDEMO, kafdemo».

Built for production

1 worker per subscription

Isolation: a slow target never blocks the others.

Batch 200 with fsync

Persisted on disk per batch: nothing is lost on restart.

10 GB / 72 h retention

The broker bounds the disk; purge by topic and by age.

max_loads per target

Loads are serialised or parallelised per target.

max_records

Optional record limit per load, for tests and migrations.

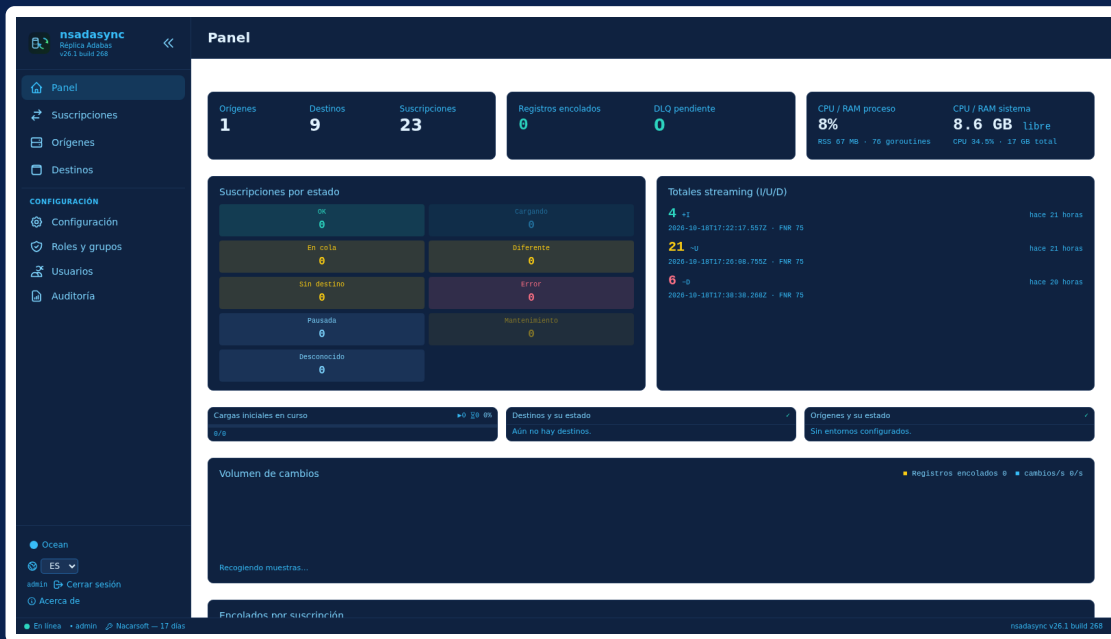
Checkpoints per subscription

Each one knows where it stands and resumes without repeats.

El exit no bloquea · El broker persiste por lote · Cada suscripción se recupera sola

OPERATIONS

The whole state on a single screen



- Sources and targets with their state and latency
- Initial loads running and queued
- State of every subscription: ok, no target, queued, error
- Streaming totals of insert, update and delete
- Change volume and queued records

Queued per subscription

Change volume

Chain health

Latency per target

Who sees what, and everything is recorded

Access control

- RBAC with system roles
- Admin, SyncOperators, Viewers
- Groups and users
- Per-area permissions

Automation

- Machine tokens nstm_...
- Scope per subscription
- Configurable expiry
- CIDR allowlist

Traceability

- Audit of every action
- User and client IP
- Per-subscription log
- Configurable retention

config.yaml 0600 · database 0700 · bcrypt · JWT HS256 · CSP headers · reverse proxy with HTTPS

USE CASES

Where it pays off from day one

Analytics and BI

Oracle, PostgreSQL or MySQL feed reporting live, with no nightly loads and no intermediate files.

Event streaming

Kafka receives one topic per table, keyed by ISN, with tombstones on delete: it converges with compaction.

Search and text

Elasticsearch keeps one index per table, created during the load and fed live.

Modern applications

MongoDB and SQL Server receive documents and rows that feed APIs and applications without touching the nucleus.

→ A single source can feed several targets at once: duplicate the subscription onto the target you need.

From zero to data in the target, in three steps

1

Enable the exit

Build libnsadasync.so and set ADAUEX_4, USEREXITS=4, LOGGING=(CB) and LOGRB in the nucleus.

2

Register the source

nsddm2file JSON dictionary and the pipe_path of the source row.

3

Targets and subscriptions

Create the target with ping and test, ensure-table, initial load and streaming.

REQUIREMENTS

- Linux (RHEL 8 / 9)
- Adabas Open Systems with CLOG and LOGRB
- JSON dictionary per source (≤ 8 MiB)
- Reverse proxy with HTTPS in production
- Web console on port 11035

```
nsadasync init ./data → nsadasync start ./data → web console :11035
```

FREQUENTLY ASKED

The questions we hear most

Do we have to stop the nucleus?

No. The exit writes one frame per record with a single non-blocking writev. With no reader it dumps to the journal and carries on.

What if the target goes down?

The changes queue in the broker and are applied when it is back. The DLQ only records real apply failures.

And deletes?

They travel without after-image: they are applied by ISN, without decoding the record image.

How are MU and PE handled?

As JSON (default) or as normalised child tables, resolved the same way in the DDL, the load and the apply.

Does it handle large loads?

Yes: initial load in batches with resumable checkpoints and per-target capacity limits (max_loads).

What can I monitor?

Source, target, states, queued records, latency, DLQ and auditing; the broker exposes offsets and checkpoints.

Your Adabas data, live wherever you need it

Try NSAdaSync with your own files: we set it up and watch it run on your data.

[Guided demo](#)

[Try it in your environment](#)

[Let's talk about your case](#)

nacarsoft.com · support@nacarsoft.com

