

Load your data into Adabas without writing it by hand.

nsadaloader loads records from JSON, CSV, TSV or XML straight into your Adabas files: STORE, UPDATE and DELETE, with MU fields, PE groups and per-batch control.

[SEE THE FEATURES →](#)

STORE · UPDATE · DELETE · JSON · CSV · XML · Configurable batches



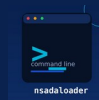
JSON · CSV · XML

auto-detected by extension



Adabas file

STORE, UPDATE or DELETE



THE PROBLEM

Loading data by hand does not scale

Every new file to load means the same work: a program, a transaction, a list of errors and two days of testing.

1

One-off loaders

A development per file, per format and per corner case, with nothing reusable.

2

Errors that only show in production

Truncated fields, MUs that do not fit, transactions left half-way.

3

No traceability

A file loaded half-way and not a single log explaining what happened.

nsadaloader turns loading into a repeatable process, with a dry run and a trace for every record.

WHAT IT IS

Nacarsoft's bulk Adabas loader

nsadaloader reads an input file, validates it against the DDM and writes the records into Adabas in per-batch transactions, with a trace of every operation.

It handles STORE, UPDATE and DELETE, MU fields, PE groups and logical DDM names. With -dry-run you see exactly what it is going to do before it touches the core.

4

input formats

3

operations

1

full trace



nsadaloader

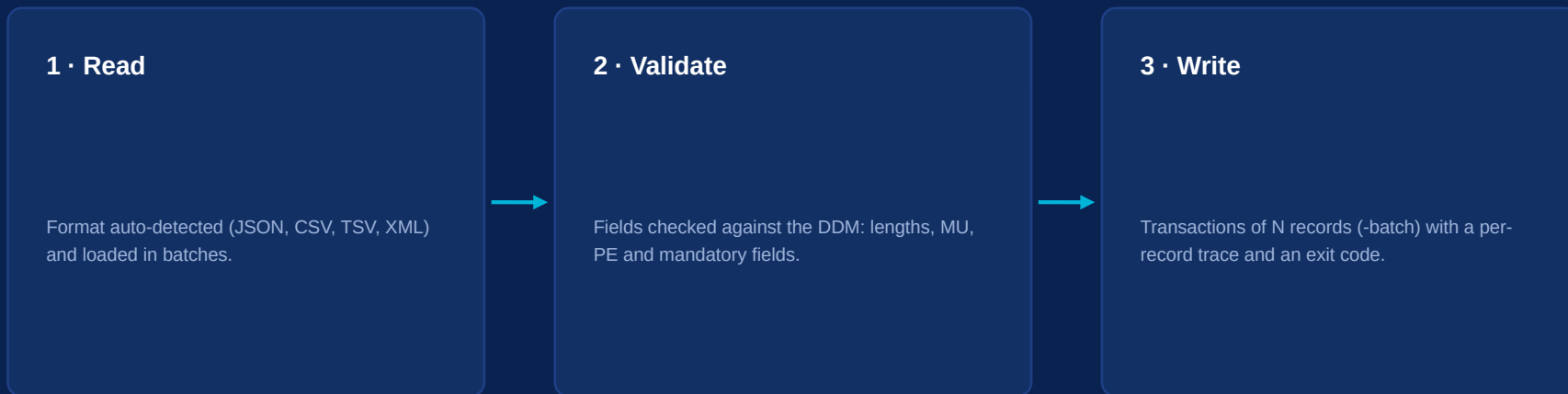
File → Adabas, under control

- JSON, CSV, TSV and XML, auto-detected
- STORE, UPDATE and DELETE in batch
- Native MU fields and PE groups
- -dry-run: simulate before writing

HOW IT WORKS

From the input file to the core, under control

Validation first, transaction per batch and a trace: nothing is written blindly.

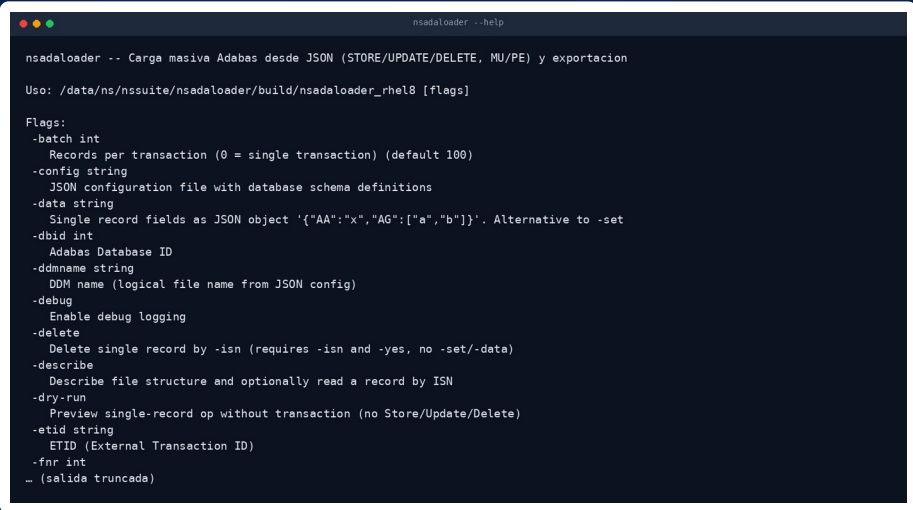


With -dry-run no transaction is opened: you see the plan before running it.

THE TOOL

One loader, every input

The same binary for JSON, CSV, TSV and XML, with extension auto-detection and validation against the DDM.



```
nsadaloader --help
nsadaloader -- Carga masiva Adabas desde JSON (STORE/UPDATE/DELETE, MU/PE) y exportacion
Usa: /data/ns/nssuite/nsadaloader/build/nsadaloader_rhel8 [flags]

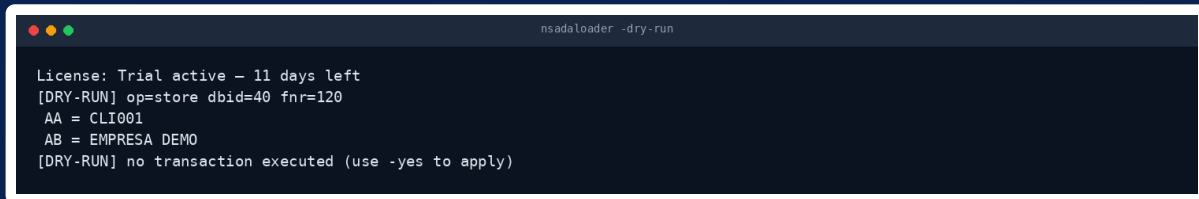
Flags:
  -batch int
    Records per transaction (0 = single transaction) (default 100)
  -config string
    JSON configuration file with database schema definitions
  -data string
    Single record fields as JSON object '{"AA":"x","AG":["a","b"]}'. Alternative to -set
  -dbid int
    Adabas Database ID
  -ddmname string
    DDM name (logical file name from JSON config)
  -debug
    Enable debug logging
  -delete
    Delete single record by -isn (requires -isn and -yes, no -set/-data)
  -describe
    Describe file structure and optionally read a record by ISN
  -dry-run
    Preview single-record op without transaction (no Store/Update/Delete)
  -etid string
    ETID (External Transaction ID)
  -fnr int
    ... (salida truncada)
```

Real output of nsadaloader --help

- No custom development
- DDM validation
- Per-batch transactions

Simulate first, load later

-dry-run shows the operation, the values and the target file without opening a single transaction against the core.

A terminal window with a dark background and a white border. The title bar at the top says "nsadaloader -dry-run". The output text is as follows:

```
License: Trial active - 11 days left
[DRY-RUN] op=store dbid=40 fnr=120
AA = CLI001
AB = EMPRESA DEMO
[DRY-RUN] no transaction executed (use -yes to apply)
```

- Plan before running
- No transactions
- Per-record trace

Real output of nsadaloader with -dry-run

FEATURES

What it does, point by point

Built for migrations and tests, with the same control as in production.



Four input formats

JSON, CSV, TSV and XML with extension auto-detection, no configuration.



MU fields and PE groups

Native support for repeated elements and periodic groups, no tricks.



Per-batch transactions

-batch controls the size of each transaction (100 records by default).



Templates with -describe

Describes the file structure and reads a record by ISN, read-only.



STORE, UPDATE and DELETE

Insert, update and delete by ISN within the same load.



Simulation (-dry-run)

See the operation and its values without running any transaction.



Logical DDM names

With the nsddm configuration, fields are named as in the DDM.



JSON export

-output dumps the file to JSON: also useful to extract and to test.

KEY OPTIONS

The parameters used every day

They combine with each other; every other flag is in --help.

-db id <id>

Target Adabas database. Required.

-fnr <n> | -ddm name

File by number or by logical DDM name (requires -config).

-input <file>

Input file: JSON, CSV, TSV or XML.

-set | -data

A single record as pairs or as a JSON object.

-dry-run

Simulates the operation without opening a transaction.

-batch <n>

Records per transaction (0 = a single transaction).

-describe

Describes the file structure and reads a record by ISN.

-output <file>

Exports the file to JSON instead of writing to Adabas.

Where it is used every day

The same use cases we solve with customers in banking, healthcare and the public sector.

01

Migrating a file

Initial load of an Adabas file from the export of another system.

02

Fixing data

Mass UPDATE of specific fields over thousands of records, with a dry run first.

03

Environment testing

Load a subset of data into the test environment and delete it afterwards.

04

Loading reference data

Load catalogues and reference tables validated against the DDM.

Install and load in five minutes

Requirements

- Linux x86-64 (RHEL 8 or RHEL 9)
- Native Adabas client (acj) in the PATH
- Write access to the database
- Product licence or the 30-day trial
- DDM in JSON (nsddm) for logical names

Install and first run

```
# Install the suite
./installsh --prefix /opt/ns

# Simulate a load
nsadaload -dbid 40 -input clients.json \
-fnr 120 -dry-run

# Load for real (in batches)
nsadaload -dbid 40 -input clients.json \
-fnr 120 -batch 500 -yes
```

The licence is resolved from \$NS_LICENSE_FILE, next to the product data or in /opt/ns/licenses/. Without one: 30-day trial.

FREQUENTLY ASKED QUESTIONS

What customers always ask

Does it write straight into the core?

Yes, in per-batch transactions, and with -dry-run you can see beforehand exactly what it will write.

What if a record fails?

It is recorded in the trace with the ISN and the reason, and the process returns an exit code other than zero.

Does it support MU and PE fields?

Yes, natively: repeated elements and periodic groups with the same semantics as the DDM.

Do I need the DDM?

Not to work by file number. With the JSON from nsddm you can use logical field names and validate the load.

NEXT STEP

Load your data without writing a single line of your own code.

Ask for a demo with your DDM and your input file: we show the simulation and the real load against your database.

REQUEST A DEMO →

nacarsoft.com · support@nacarsoft.com · +34 910 038 502



NacarSoft Soluciones Informáticas S.L. · nacarsoft.com · nsadaloader

NSADALoader at a glance

- JSON, CSV, TSV or XML input
- STORE, UPDATE and DELETE in batch
- Native MU fields and PE groups
- -dry-run before touching the core