Structure definition

Language: Deutsch · **English**

Structure definition

> **Map CSV columns to BO fields – import modes, Groovy calculation, format and identifiers.**

REFERENZ | APPLICATION DEVELOPER | UPDATED: JUL 2026 | APPLIES TO NUCLOS 4.2026.X

On this page

Why a structure definition?

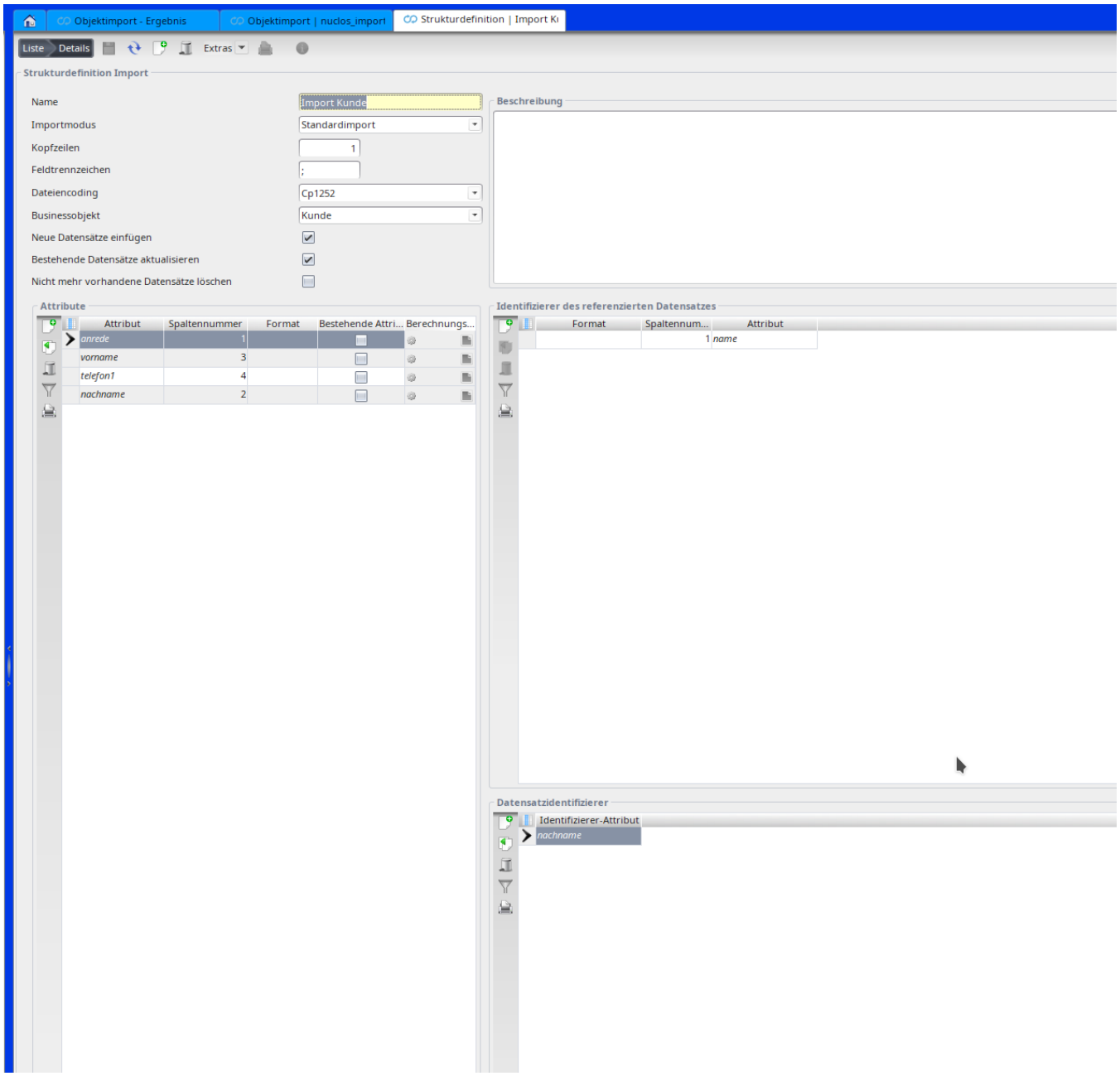
The **structure definition** maps the columns of a source file (CSV) to the fields of a business object. *Menu: Configuration Import & Export Structure definition.*

Basic settings

Field	Meaning
Name	name of the structure definition.
Header rows	number of rows to ignore (e.g. column headings).
Import mode	**standard import** (with Nuclos storage mechanisms and rules) or **direct import** (straight into the DB, without rules).
Field separator	column separator of the CSV (usually `;`).
Business object	target BO of the import.

Insert/update behaviour

Option	Effect
Insert records only	always creates a new record; uniqueness violations cause errors.
Update existing records	existing records (by identifier) are updated.
Delete records no longer present	records missing from the import file are removed.



Structure definition for object import.

Mapping attributes

Link each BO **attribute** to the file's **column number**. *Do not overwrite existing attributes* limits overwriting per field. Boolean values recognise `true/false`, `1/0`, `y/n` etc.

Calculation expression (Groovy)

Instead of a column number an attribute can be computed per row via a **Groovy script**. Available variables: `values` (String[], 0-indexed), `line` (Integer) and `log` (ImportLogger). If a script is set, **no column number** may be given.

```
if (values[2] == null || values[2].isEmpty()) {
    log.info("FALSE "+values[2]);
    return false;
} else {
    log.info("TRUE "+values[2]);
    return true;
}
```

Reference fields may only return `Long`, `null` or an empty string (`null`):

```
if (values[0] == "Frau") {
    return 40000001;
}
else if (values[0] == "Herr") {
    return 40000002;
}
return null;
```

Date value:

```
java.text.SimpleDateFormat dateFormat = new java.text.SimpleDateFormat("dd.MM.yyyy");
java.util.Date date = dateFormat.parse("25.02.2020");

return date;
```

Format (parse string)

Only used when *no* script is set. Meaning per data type:

Type	Format
String / number	`<match>#<replacement>` with regex; example `Nr(\d)+#Nr $1`: `Nr9 Nr 9`.
Date	SimpleDateFormat; default `dd.MM.yyyy`.
Boolean / BigDecimal	format is **not** used.

Identifiers

The **identifier attribute** marks a record as unique (important for updates). For reference fields the **identifier of the referenced record** defines via which field (and column number) the mapping to the referenced BO happens.

> **CSV format**
>
> **Valid CSV** (RFC 4180): fields may be double-quoted; fields with a line break or the field separator *must* be quoted; a `"` inside a value is escaped as `""`.

Related pages

🔌 Object import

Run the import.

Open

📋 XML structure definition

XML import.

Open

📦 Import and export

Overview.

Open