Strukturdefinition

Sprache: **Deutsch** · English

Strukturdefinition

> **CSV-Spalten auf BO-Felder mappen – Importmodi, Groovy-Berechnung, Format und Identifizierer.**

REFERENZ | ANWENDUNGSENTWICKLER | STAND: JUL 2026 | GILT FUER NUCLOS 4.2026.X

Auf dieser Seite

Wozu eine Strukturdefinition?

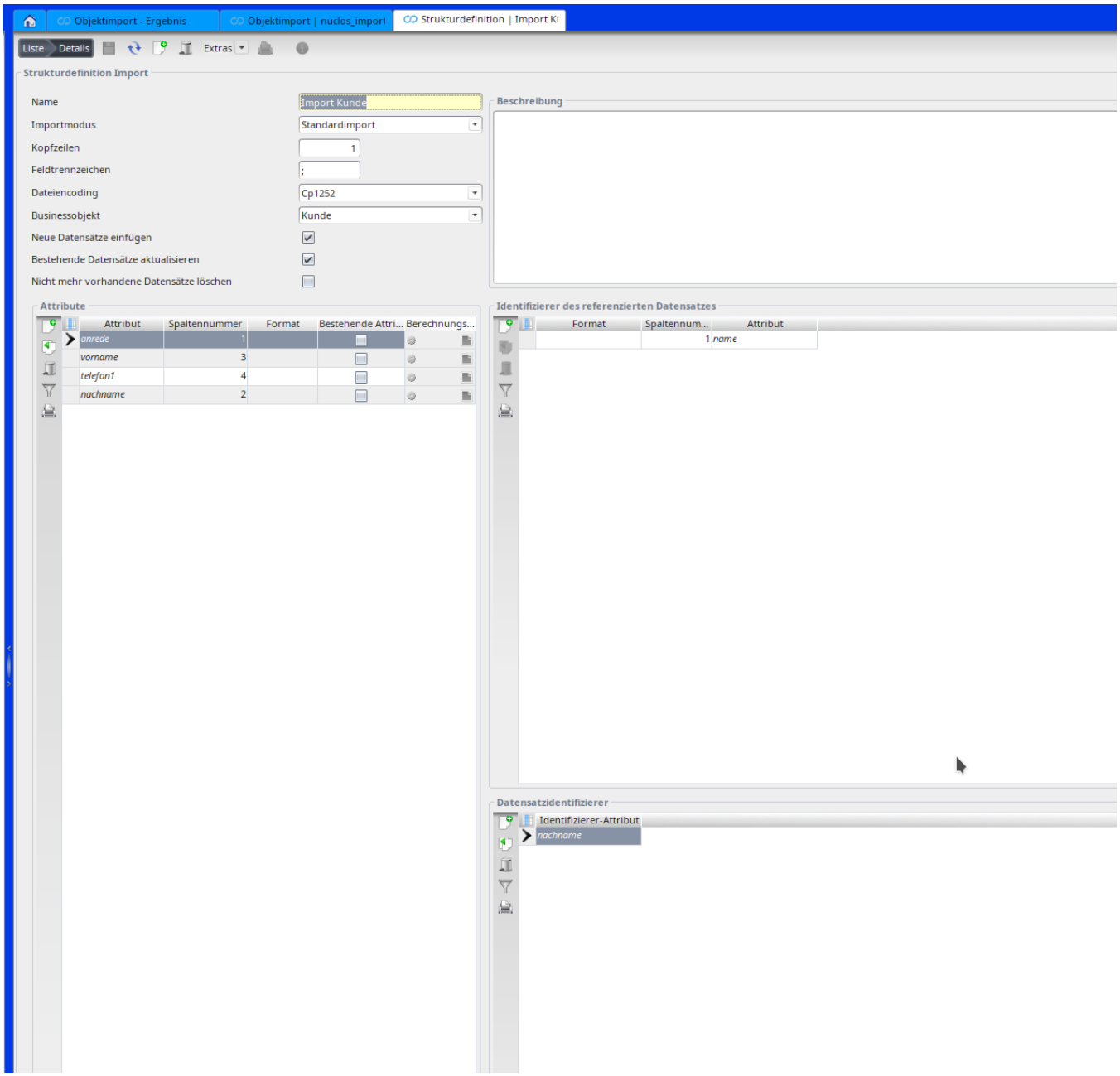
Die **Strukturdefinition** ordnet die Spalten einer Quelldatei (CSV) den Feldern eines Businessobjekts zu. *Menü: Konfiguration Import & Export Strukturdefinition.*

Grundeinstellungen

Feld	Bedeutung
Name	Bezeichnung der Strukturdefinition.
Kopfzeilen	Anzahl der zu ignorierenden Zeilen (z.B. Spaltenüberschriften).
Importmodus	**Standardimport** (mit Nuclos-Speichermechanismen und Regeln) oder **Direktimport** (direkt in die DB, ohne Regeln).
Feldtrennzeichen	Spaltentrenner der CSV (meist `;`).
Businessobjekt	Ziel-BO des Imports.

Anlage-/Update-Verhalten

Option	Wirkung
Nur Datensätze einfügen	es wird stets ein neuer Datensatz angelegt; Eindeutigkeitsverletzungen führen zu Fehlern.
Bestehende Datensätze aktualisieren	vorhandene Datensätze (per Identifizierer) werden aktualisiert.
Nicht mehr vorhandene Datensätze löschen	Datensätze, die in der Importdatei fehlen, werden entfernt.



Strukturdefinition für Objektimport.

Attribute zuordnen

Je Zeile ein **Attribut** des BOs mit der **Spaltennummer** der Datei verknüpfen. Mit *Bestehende Attribute nicht überschreiben* lässt sich das Überschreiben auf Feldebene eingrenzen. Boolean-Werte erkennen `true/false`, `1/0`, `j/n` u.a.

Berechnungsausdruck (Groovy)

Statt einer Spaltennummer kann ein Attribut per **Groovy-Script** je Zeile berechnet werden. Verfügbare Variablen: `values` (String[], 0-indiziert), `line` (Integer) und `log` (ImportLogger). Ist ein Script hinterlegt, darf **keine Spaltennummer** angegeben werden.

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

Referenzfelder dürfen nur `Long`, `null` oder Leerstring (`null`) liefern:

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

Datumswert:

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

return date;
```

Format (Parsestring)

Wird nur genutzt, wenn *kein* Script hinterlegt ist. Bedeutung je Datentyp:

Typ	Format
String / Number	`<match>#<replacement>` mit Regex; Beispiel `Nr(\d)+#Nr $1`: `Nr9 Nr 9`.
Date	SimpleDateFormat; Standard `dd.MM.yyyy`.
Boolean / BigDecimal	Format wird **nicht** verwendet.

Identifizierer

Das **Identifizierer-Attribut** kennzeichnet einen Datensatz eindeutig (wichtig für Updates). Für Referenzfelder legt der **Identifizierer des referenzierten Datensatzes** fest, über welches Feld (und welche Spaltennummer) das Mapping auf das Referenz-BO erfolgt.

CSV-Format

Valides CSV (RFC 4180): Felder dürfen in doppelte Anführungszeichen; Felder mit Zeilenumbruch oder Feldtrennzeichen *müssen* gequotet werden; ein `"` im Wert wird durch `""` maskiert.

Verwandte Seiten

🔌 Objektimport

Import ausführen.

Öffnen

📋 XML Strukturdefinition

XML-Import.

Öffnen

📦 Import und Export

Übersicht.

Öffnen