# Squeeze Connector


- 01 – Übersicht
- Installation

# 01 – Übersicht

Der SqueezeConnector-Import holt fertig gelesene Belege per Squeeze REST-API v2 nach DOCUMENTS ab und legt je Beleg eine neue Mappe an. Diese Seite beschreibt Funktionsweise, Ablauf und Grenzen der Variante **ohne Duplikatsprüfung**.

# Was ist der SqueezeConnector-Import

Der Import ist die schlanke Variante des Squeeze Pull-Jobs. DOCUMENTS holt die Belege selbst über die REST-API v2 von Squeeze ab, der SOAP-Export der Klasse entfällt. Je Squeeze-Klasse gibt es ein eigenes Job-Skript, das im Scheduler läuft und `runSqueezeImport` aufruft.

Der Job fragt im Queue-Step „Export“ die Belege der konfigurierten Squeeze-Klasse ab. Je Beleg legt er **immer eine neue Mappe** an, schreibt die Kopffelder (optional auch Positionen und Anhänge) und meldet den Beleg danach über das Export-Interface „Pull“ als FINISHED an Squeeze. Erst dann verlässt der Beleg den Export.

Das Mapping bleibt in Squeeze: Maßgeblich ist der `externalName` jedes Feldes und jeder Tabellenspalte der Squeeze-Klasse. Er muss dem technischen Feldnamen in DOCUMENTS entsprechen. Denselben Wert nutzt der SOAP-Export, beim Umstieg bleibt das Mapping also unverändert. In DOCUMENTS gibt es keine Mapping-Tabelle.

Die Bibliothek ist eigenständig. Sie braucht keine DEXPRO-Solution und keine Parameter und läuft unter DOCUMENTS 5 und 6 (Windows und Linux).

# Abgrenzung zum Pull-Job mit Duplikatsprüfung

| | **Pull-Job (`runSqueezePull`)** | **Import (`runSqueezeImport`)** |
|---|---|---|
| Vorhandene Mappe suchen (SqueezeID/externalId) | ja | nein, immer neue Mappe |
| Feld `SqueezeStatus`, Workflow-Start | ja | nein, liegt beim Aufrufer |
| Pflichtfelder am Mappentyp | `SqueezeID`, `SqueezeStatus`, Register | keine |
| Anhänge | Standard ja | nur mit `register` |
| Positionen | Standard ja | nur mit `positionsField` |
| Sperre gegen parallele Läufe | ja | ja |

> **Ohne Duplikatsprüfung** erkennt der Job nicht, ob es zu einem Beleg schon eine Mappe gibt. Jeder Beleg, der im Export-Step liegt, erzeugt eine neue Mappe. Die Folgen sind im Abschnitt „Grenzen“ und auf der Seite *Betrieb und Fehlersuche* beschrieben.

# Skripte

| Skript | Zweck |
|---|---|
| `SqueezeConnector_Lib` | Die Bibliothek. Sie legt nur das globale Objekt `SqueezeConnector` an (Verbindung, Lesen, Schreiben, FINISHED melden). |
| `SqueezeConnector_Import` | `runSqueezeImport(config, log)`: der komplette Lauf für eine Squeeze-Klasse |
| `SqueezeConnector_JOB_Import` | Job-Vorlage. Je Squeeze-Klasse eine Kopie. |

# Was bei jedem Lauf passiert

Zuerst prüft der Job die Voraussetzungen. Ist eine nicht erfüllt, holt er nichts ab und schreibt den Grund ins Log.

- In der Konfiguration stehen `fileType` und `squeezeClass`.
- Der Benutzer aus `scriptUser` lässt sich setzen. Ohne Benutzer kann DOCUMENTS keine Mappe anlegen.
- Die Zugangsdaten sind vorhanden, in der Job-Konfiguration oder in `SqueezeConfig.js`.
- Das Export-Interface „Pull“ ist an der Squeeze-Klasse vorhanden und aktiv.
- Für den Mappentyp läuft gerade kein anderer Import.

Danach holt der Job die Liste aller Belege der Klasse im Queue-Step „Export“ und verarbeitet sie nacheinander, höchstens `maxDocuments` pro Lauf (Standard 100). Je Beleg:

1. **Beleg lesen:** Kopffelder, Positionen und die Liste der Anhänge aus Squeeze. Steht der Beleg inzwischen nicht mehr im Export-Step, wird er übersprungen.
2. **Mappe anlegen:** eine neue Mappe vom Typ `fileType`.
3. **Daten übernehmen:** Kopffelder schreiben, optional Positionen und Anhänge übernehmen, Mappe speichern.
4. **FINISHED melden:** Der Beleg verlässt in Squeeze den Export.

Schlägt Schritt 3 fehl, löscht der Job die Mappe wieder und meldet kein FINISHED. Der Beleg bleibt im Export, der nächste Lauf versucht es erneut.

Ein Fehler bei einem Beleg bricht den Lauf nicht ab, der Job macht mit dem nächsten Beleg weiter. Ausnahme: Fehlt ein Anhang-Register am Mappentyp, bricht der Lauf ab, und die restlichen Belege bleiben im Export.

# Zähler und Rückgabewert

`runSqueezeImport` liefert Zähler zurück. Dieselben Werte stehen in der letzten Logzeile.

| Zähler | Logtext | Bedeutung |
|---|---|---|
| `ok` | angelegt | Mappe angelegt und Beleg als FINISHED gemeldet |
| `error` | fehlerhaft | Lesen, Anlegen, Übernahme oder FINISHED-Meldung gescheitert |
| `aborted` | abgebrochen | Unerwarteter Fehler (Exception) bei einem Beleg |
| `skipped` | noch nicht fertig | Beleg stand beim Lesen nicht mehr im Export-Step |
| `deferred` | auf den nächsten Lauf verschoben | Obergrenze erreicht, Register fehlt oder Sperre verloren |
| `total` | gesamt im Step | Belege der Klasse im Export-Step zu Beginn des Laufs |

Die Job-Vorlage meldet den Lauf als fehlgeschlagen (`context.returnValue = -1`), sobald `error` oder `aborted` größer 0 ist oder ein Fehler über `log.err` gemeldet wurde.

# Grenzen

> **Ohne `register` gehen die Anhänge verloren.** Ist kein Register konfiguriert, entsteht die Mappe ohne Beleg-PDF. Im Log steht nur ein Hinweis, der Beleg wird trotzdem als FINISHED gemeldet und verschwindet aus dem Export. Dasselbe gilt für Positionen ohne `positionsField`. Beide Optionen deshalb immer setzen, wenn Anhänge bzw. Positionen gebraucht werden.

- **FINISHED-Meldung gescheitert:** Die Mappe bleibt stehen, der Beleg bleibt im Export. Der nächste Lauf legt eine zweite Mappe an. Im Log steht dazu ein FEHLER.
- **Unerwarteter Fehler nach dem Anlegen:** Tritt nach dem Anlegen der Mappe eine Exception auf, bleibt die unvollständige Mappe stehen. Der Beleg wird nicht als FINISHED gemeldet, der nächste Lauf legt eine weitere Mappe an.
- **Squeeze nicht erreichbar:** Ist Squeeze nicht erreichbar, der API-Key ungültig oder die Klasse nicht vorhanden, wird die Klasse übersprungen. Das steht nur als Information im Log, der Job meldet **keinen** Fehler. Das Log deshalb regelmäßig prüfen.
- **Kein Rückverweis in Squeeze:** Die FileID der Mappe wird nicht an Squeeze gemeldet (nur über die v1-API möglich). Die Zuordnung Beleg → Mappe gibt es nur über `squeezeIdField` und das Log.

- **Sperre je Mappentyp:** Zwei Jobs für verschiedene Squeeze-Klassen, die in denselben Mappentyp schreiben, blockieren sich gegenseitig. Der zweite Lauf wird übersprungen. Solche Jobs zeitversetzt einplanen.

# Installation

Diese Seite beschreibt Voraussetzungen, Einbau und Einbautest des SqueezeConnector-Imports. Die Optionen der Job-Konfiguration sind auf der Seite *Konfiguration* beschrieben.

# Voraussetzungen

**Squeeze**

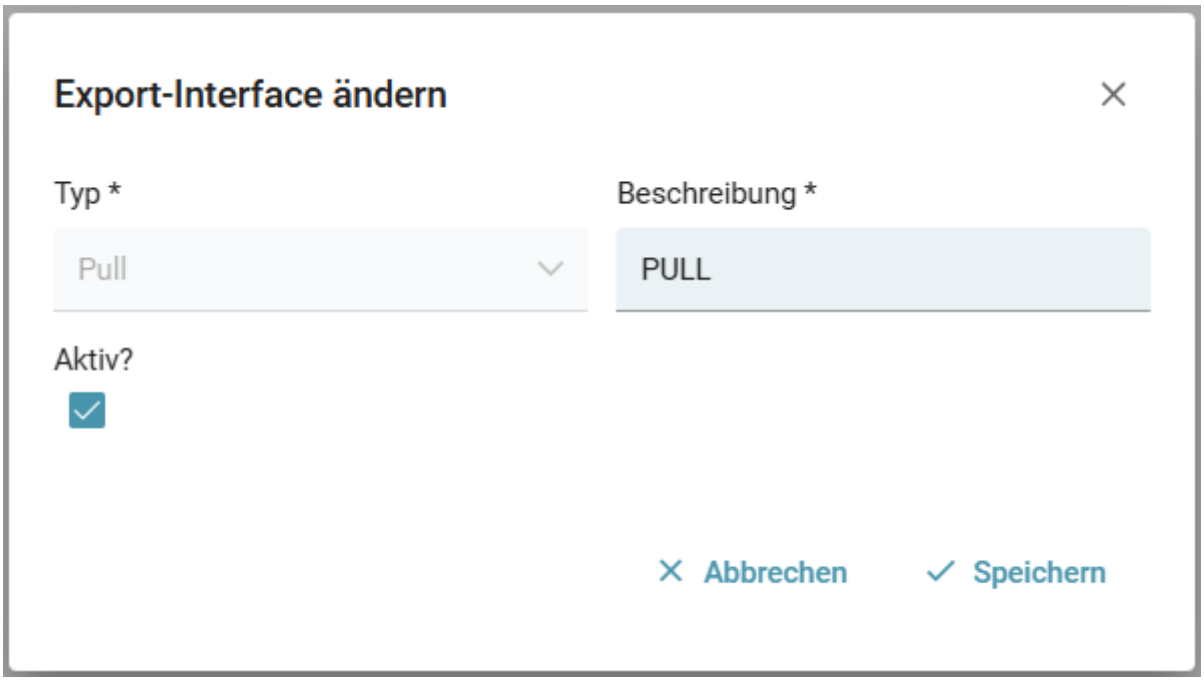
- REST-API v2 ist erreichbar. Der DOCUMENTS-Server kommt auf die Squeeze-URL.
- Es gibt einen API-Key mit Zugriff auf die Dokumentklasse, die Queue und die Export-Interfaces.
- Die Belege der Klasse durchlaufen den Queue-Step „Export“.

**DOCUMENTS**

- DOCUMENTS 5 oder 6.
- Der Mappentyp ist vorhanden. Die technischen Feldnamen entsprechen den `externalName`-Werten in Squeeze.
- Es gibt einen Benutzer mit Rechten auf den Mappentyp, der im Mandanten existiert ( `scriptUser`).
- Für Anhänge: ein Dokumentenregister am Mappentyp.
- Für Positionen: ein Feld am Mappentyp, das die Positionen als JSON aufnimmt (z. B. `Gentable`).
- Optional: ein Textfeld für die Squeeze-ID des Belegs (z. B. `SqueezeID`). Empfohlen, weil es der einzige Rückverweis von der Mappe auf den Beleg ist.

# 1. Squeeze vorbereiten

1. **Mapping prüfen:** In der Dokumentklasse hat jedes Feld, das übernommen werden soll, als `externalName` (Name im Zielsystem) den technischen Feldnamen aus DOCUMENTS. Dasselbe gilt für die Tabellenspalten. Felder und Spalten ohne `externalName` werden nicht übernommen.
2. **Export-Interface „Pull“ anlegen und aktivieren:** An der Dokumentklasse ein Export-Interface mit dem Namen `Pull` anlegen und aktiv schalten. Der Name muss exakt stimmen, auch in der Groß- und Kleinschreibung. Ein abweichender Name wird über die Option `exportInterface` angegeben.
3. **SOAP-Export der Klasse deaktivieren.**




| ID | Name | Beschreibung | Typ | Standard Wert | Lokator | Lokator-Unterfeld | Name (Zielsystem) |
|---|---|---|---|---|---|---|---|
| 2 | Company | Mandant | Text | 1000 | Mandant | | Principal |
| 195 | CompanyName | Mandantenname | Text | | Mandanten-Name | | |
| 196 | CompanyCountry | Mandantenland | Text | | Mandanten-Land | | |
| 4 | Creditor | Lieferant | Text | | Lieferant | | CreditorID |
| 5 | CreditorName | Lieferantenname | Text | | Lieferantenname | | CreditorName |
| 6 | CreditorCountry | Lieferantenland | Text | | Lieferantenland | | Creditor_Country |
| 11 | IBAN | IBAN | Text | | IBAN | | IBAN |
| 12 | VatId | Ust-ID | Text | | UstId | | Creditor_TaxID |
| 3 | DocumentType | Vorgang | Text | | Vorgang | | InvoiceCreditVoucher |
| 8 | DocumentReference | Belegnummer | Text | | Belegnummer | | InvoiceNumber |
| 7 | DocumentDate | Belegdatum | Datum | | Belegdatum | | InvoiceDate |

> **Pro Squeeze-Klasse darf nur ein Export aktiv sein:** SOAP-Export oder Pull. Laufen beide, entstehen doppelte Mappen.

> Ein `externalName` darf je Klasse nur einmal vergeben sein. Ist ein DOCUMENTS-Feld doppelt gemappt, steht ein Hinweis im Log, und es ist nicht festgelegt, welcher Wert in der Mappe landet.

# 2. Zugangsdaten hinterlegen

Ohne `server` und `apiKey` in der Job-Konfiguration liest der Import die Zugangsdaten aus `SqueezeConfig.js`, derselben Datei wie der Squeeze-Viewer.

**Speicherort der Datei** (die letzte zutreffende Zeile gilt):

| Mandanteneigenschaft | Pfad |
|---|---|
| keine | `<DOCUMENTS-Installationsverzeichnis>/DEXPRO/Squeeze/SqueezeConfig.js` |
| `$SqueezeFolderPath` | `<Wert>/www/SqueezeViewer/SqueezeConfig.js` |
| `$SqueezeServerConfigPath` | `<Wert>/SqueezeConfig.js` |

Mit `$SqueezeConfigFileName` lässt sich ein anderer Dateiname angeben. `$SqueezeUseProxy = false` schaltet die Proxy-Ermittlung von DOCUMENTS ab.

**Inhalt der Datei:** Ausgewertet wird das JSON ab der ersten geschweiften Klammer. Gibt es einen Block `SqueezeConfig_<Mandantenkürzel>`, gilt dieser, sonst die oberste Ebene.

```
var SqueezeConfig = {
    "SqueezeConfig_meinmandant": {
        "squeezeServer": "https://kunde.squeeze.one",
        "squeezeApiKey": "...",
        "squeezeCaCert": ""
    }
}
```

`squeezeCaCert` ist optional und nennt eine PEM-Datei für die Zertifikatsprüfung.

> Der API-Key gehört nicht in das Job-Skript. `SqueezeConfig.js` ist der Standardweg. `server` und `apiKey` in der Job-Konfiguration sind nur für Ausnahmen gedacht.

# 3. Skripte einspielen

Alle drei Skripte in den DOCUMENTS-Manager einspielen:

| Skript | Hinweis |
|---|---|
| `SqueezeConnector_Lib` | Gibt es je Server nur einmal. Ist auch der Pull-Job mit Duplikatsprüfung installiert, muss die neuere Fassung der Lib eingespielt sein. |
| `SqueezeConnector_Import` | Importiert `SqueezeConnector_Lib` |
| `SqueezeConnector_JOB_Import` | Vorlage, wird nicht direkt eingeplant |

# 4. Job-Skript je Squeeze-Klasse anlegen

1. `SqueezeConnector_JOB_Import` kopieren und sprechend benennen, z. B. `Invoice_SqueezeConnector_JOB_Import`.
2. In der Kopie `squeezeConfig` anpassen:

```
var squeezeConfig = /** @type {Object.<string,any>} */ ({
    fileType:       "Invoice",      // Mappentyp in DOCUMENTS
    squeezeClass:   "Invoices",     // Squeeze-Klasse, Name oder ID
    scriptUser:     "dexadmin",     // Benutzer mit Rechten auf den Mappentyp
    register:       "Documents",    // Register für die Anhänge
    positionsField: "Gentable",     // Feld für die Positionen
    squeezeIdField: "SqueezeID"     // Feld für die Squeeze-ID
});
```

3. Das Logging der Vorlage schreibt mit `util.out` ins Server-Log (Präfix `[SqueezeImport]`). Bei Bedarf durch das eigene Logging ersetzen: ein Objekt mit den Funktionen `log.info(text)` und `log.err(text)`.

> Die ausgelieferte Vorlage enthält nur `fileType`, `squeezeClass` und `scriptUser`. `register` und `positionsField` müssen ergänzt werden, sonst werden Anhänge und Positionen nicht übernommen und der Beleg trotzdem als FINISHED gemeldet.

# 5. Verbindung testen

Vor dem ersten Lauf die Verbindung mit einem kurzen Testskript prüfen. Es schreibt nichts.

```
// #import "SqueezeConnector_Lib"
var fileCfg = SqueezeConnector.readConfigFile();
if (fileCfg.error !== "") {
    util.out(fileCfg.error);
} else {
    var sqz = new SqueezeConnector(fileCfg.connection);
    var chk = sqz.checkConnection();
    util.out(chk.status + " " + chk.errorMessage + " " + JSON.stringify(chk.data));
    var pull = sqz.getExportInterface("Invoices");
    util.out(pull.status + " " + pull.errorMessage + " " + JSON.stringify(pull.data));
```

```
}
```

Erwartet wird zweimal `OK`. Die erste Zeile zeigt die Squeeze-Version und alle Dokumentklassen mit ID und Name, die zweite das Export-Interface mit `active: true`.

# 6. Einbautest

1. Einen Testbeleg in Squeeze bis in den Queue-Step „Export“ bringen.
2. Im Job-Skript vorübergehend `maxDocuments: 1` und `testKeepInExport: true` setzen.
3. Das Job-Skript **einmal** manuell ausführen.
4. Im Log prüfen: `Dokument <ID> - Mappe <FileID> angelegt, gesetzte Felder: …`. Hinweise zu fehlenden Feldern und Typabweichungen beachten.
5. Die Mappe prüfen: Kopffelder, Positionen, Anhänge im richtigen Register, Umlaute.
6. Die Testmappe löschen.
7. `testKeepInExport` und `maxDocuments` wieder entfernen.
8. Das Job-Skript erneut ausführen. Der Beleg muss danach in Squeeze den Export-Step verlassen haben.

> **Testmodus nie eingeplant laufen lassen.** Mit `testKeepInExport: true` wird kein FINISHED gemeldet. Jeder Lauf legt für dieselben Belege erneut Mappen an.

# 7. Job einplanen

Das Job-Skript im DOCUMENTS-Manager als Job einplanen, z. B. viertelstündlich. Je Squeeze-Klasse ein Job. Schreiben mehrere Klassen in denselben Mappentyp, die Jobs zeitversetzt einplanen, weil die Sperre je Mappentyp gilt.

[BILD 03 – DOCUMENTS-Manager: eingeplanter Job]

> **Pro Squeeze-Instanz darf nur ein DOCUMENTS-System abholen.** Test- und Produktivsystem dürfen nicht gleichzeitig dieselbe Klasse abholen, sonst fehlen die Belege im jeweils anderen System.