

# Anleitung zur Nutzung der JSON-basierten Datenübertragung in die Datenbank

---

## 1. Überblick

---

Diese Anleitung beschreibt den vollständigen Datenverarbeitungsprozess, um **Systeminformationen** strukturiert in eine **Microsoft SQL Server Datenbank** zu überführen. Dabei wird ein standardisiertes **JSON-Format** genutzt, welches über die API in die **Inbox** der Datenbank übertragen und anschließend von einem **Event-Handler** verarbeitet wird.

---

## 2. Prozessablauf

---

### 2.1 Definition der zu erfassenden Informationen

Bevor ein System in die Datenbank aufgenommen wird, sind die zu sammelnden Daten zu definieren. Dabei gilt:

- Welche Informationen sollen aus dem System extrahiert werden? (z. B. *Benutzer, Hardware-Details, Software-Lizenzen etc.*)
- Welche dieser Daten sind für Auswertungen notwendig?
- Sind bestimmte Daten sensibel oder erfordern sie besondere Verarbeitungsschritte?

**Ergebnis:** Liste mit allen relevanten Datenfeldern.

---

### 2.2 Evaluation & Erstellung der Datenbankstruktur

Nach der Definition wird geprüft, ob diese Daten in der **bestehenden Datenbankstruktur** bereits abgebildet werden können. Falls nicht, wird eine neue **Tabelle in Microsoft SQL Server angelegt**.

☞ **Beispiel:** Ein System soll Benutzerinformationen speichern. Dazu wird die Tabelle `usr_client_users` erstellt:

**SQL CREATE SCRIPT** für Microsoft SQL Server

```

CREATE TABLE usr_client_users (
    id UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    transaction_id UNIQUEIDENTIFIER NOT NULL DEFAULT NEWID(),
    username NVARCHAR(128) NOT NULL,
    client NVARCHAR(128) NOT NULL,
    usercount VARCHAR(10),
    permissions NVARCHAR(512),
    sid NVARCHAR(128),
    full_name NVARCHAR(256),
    account_status NVARCHAR(32),
    last_logon DATETIME,
    description NVARCHAR(512)
);

```

- Ergebnis:** Eine neue oder erweiterte Datenbankstruktur für die Speicherung der Informationen.
- 

## 2.3 Integration des Erfassungsskripts

Zur **Erfassung der Daten** wird ein **PowerShell-Skript** oder ein entsprechendes Bash-Skript (Linux) geschrieben. Diese Skripte:

- Erfassen die definierten Informationen
- Formatieren diese in das standardisierte JSON-Format
- Speichern die JSON-Datei temporär
- Übertragen die Datei per API an die **Inbox**

### ❖ Ablage der Skripte in `acx_scripts`

- Jedes Skript wird als **Paket** in die global verwaltete Tabelle `acx_scripts` hochgeladen.
- Skripte werden zentral verwaltet und über automatisierte Jobs ausgeführt.

- Ergebnis:** Ein Skript, das die Daten erfasst und standardisiert zur Verarbeitung bereitstellt.
- 

## 2.4 JSON-Format und Speicherung in der Inbox

Jedes Skript sendet die Daten an die **Inbox**-API als standardisiertes JSON.

### ❖ JSON-Beispiel für Benutzerinformationen:

```
{
  "MetaData": {
    "ContentType": "db-import",
    "Name": "Windows User Import",
    "Description": "Collect User Information",
    "Version": "1.0",
    "Creator": "FL",
    "Vendor": "ondeso GmbH"
  },
  "Content": {
    "TableName": "usr_client_users",
    "Consts": [
      {
        "Identifier": "CaptureDate",
        "Value": "2025-01-30T08:45:30Z"
      }
    ],
    "FieldMappings": [
      {
        "TargetField": "id",
        "Expression": "{id}",
        "IsIdentifier": true,
        "ImportField": true
      },
      {
        "TargetField": "transaction_id",
        "Expression": "{transaction_id}",
        "IsIdentifier": true,
        "ImportField": true
      },
      {
        "TargetField": "username",
        "Expression": "{UserName}",
        "IsIdentifier": false,
        "ImportField": true
      }
    ],
    "Data": [
      {
        "id": "d3f2c10e-4f65-4f23-bcd8-9d4a33d3c8a7",
        "transaction_id": "c3a29297-0024-487d-94f6-f6f83e8e8ea1",
        "UserName": "Administrator",
        "ClientName": "MYSERVER",
        "UserCount": "4",
        "Permissions": "Administratoren",
        "SID": "S-1-5-21-123456789-123456789-123456789-500",
        "FullName": "Admin-Konto",
        "AccountStatus": "Active",
        "LastLogon": "2025-01-30 09:30:45",
        "Description": "Administrator-Konto"
      }
    ]
  }
}
```

```
        }
    ]
}
}
```

#### ❖ Speicherung in der `Inbox` -Tabelle (acx\_inbox)

- JSON wird **unverändert und unprozessiert** als Eintrag abgelegt.
- Der Status steht auf "pending".

**Ergebnis:** JSON ist in der Inbox gespeichert und bereit für die Verarbeitung.

---

## 2.5 Verarbeitung durch den Event-Handler

Ein Event-Handler liest die Inbox **automatisch in Intervallen** aus und verarbeitet die JSON-Daten.

#### ❖ Ablauf:

1. Inbox-Status auf "running" setzen
2. JSON einlesen und Felder validieren
3. Daten in die Ziel-Tabelle übertragen
4. Status in der Inbox auf "success" setzen oder "error" bei Problemen

#### ❖ Beispielhafte Verarbeitung in `process_inbox()`

```

def process_inbox():
    while True:
        with app.app_context():
            pending_entries =
Inbox.query.filter_by(acx_inbox_processing_state="pending").all()

            for entry in pending_entries:
                entry.acx_inbox_processing_state = "running"
                db.session.commit()

            try:
                json_content = json.loads(entry.acx_inbox_content)
                table_name = json_content["Content"]["TableName"]
                data_entries = json_content["Content"]["Data"]

                for record in data_entries:
                    db.session.execute(text(f"INSERT INTO {table_name} (...) VALUES (...)"), record)

                db.session.commit()
                entry.acx_inbox_processing_state = "success"

            except Exception as e:
                entry.acx_inbox_processing_state = "error"
                entry.acx_inbox_processing_log = str(e)
                db.session.rollback()

            db.session.commit()

    time.sleep(10) # Automatische Verarbeitung alle 10 Sekunden

```

**Ergebnis:** Daten werden in die Zieltabelle übertragen.

---

## 2.6 Status-Markierung der Inbox-Einträge

❖ Während der Verarbeitung durch den Event-Handler wird der Status aktualisiert:

Status	Bedeutung
pending	JSON wurde gespeichert, aber noch nicht verarbeitet
running	Verarbeitung läuft aktuell
success	Daten wurden erfolgreich in die Ziel-Tabelle übernommen
error	Fehler bei der Verarbeitung (Log wird gespeichert)

- 
- Ergebnis:** Jede Transaktion ist nachverfolgbar.
- 

## 3. Fazit

Mit diesem System können **beliebige Datenstrukturen aus verschiedenen Quellen** automatisiert erfasst, zwischengespeichert und verarbeitet werden. Die zentrale **Inbox-API** stellt sicher, dass alle Daten standardisiert in der Datenbank landen und **fehlerhafte Einträge nachvollziehbar** sind.

### Vorteile dieser Architektur

- ✓ **Flexibel:** Beliebige Daten können erfasst werden
  - ✓ **Skalierbar:** Erweiterung durch neue Tabellen und Skripte jederzeit möglich
  - ✓ **Sicher:** Inbox trennt Rohdaten von der Verarbeitungslogik
  - ✓ **Nachvollziehbar:** Vollständiges Logging jeder Transaktion
- 

## 4. Nächste Schritte für die Integration

1. Bestimmen, welche Informationen erfasst werden sollen.
2. Prüfen, ob die Zieltabelle existiert, falls nicht: SQL-Create-Skript ausführen.
3. Skript in `acx_scripts` hochladen und testen.
4. JSON an `Inbox` senden und sicherstellen, dass es korrekt abgelegt wird.
5. Monitoring der Inbox-Statusmarkierungen und Fehleranalyse bei Bedarf.