Erstellung benutzerdefinierter Dokumente (Individuelle Verarbeitung)

Inhaltsverzeichnis

Überblick

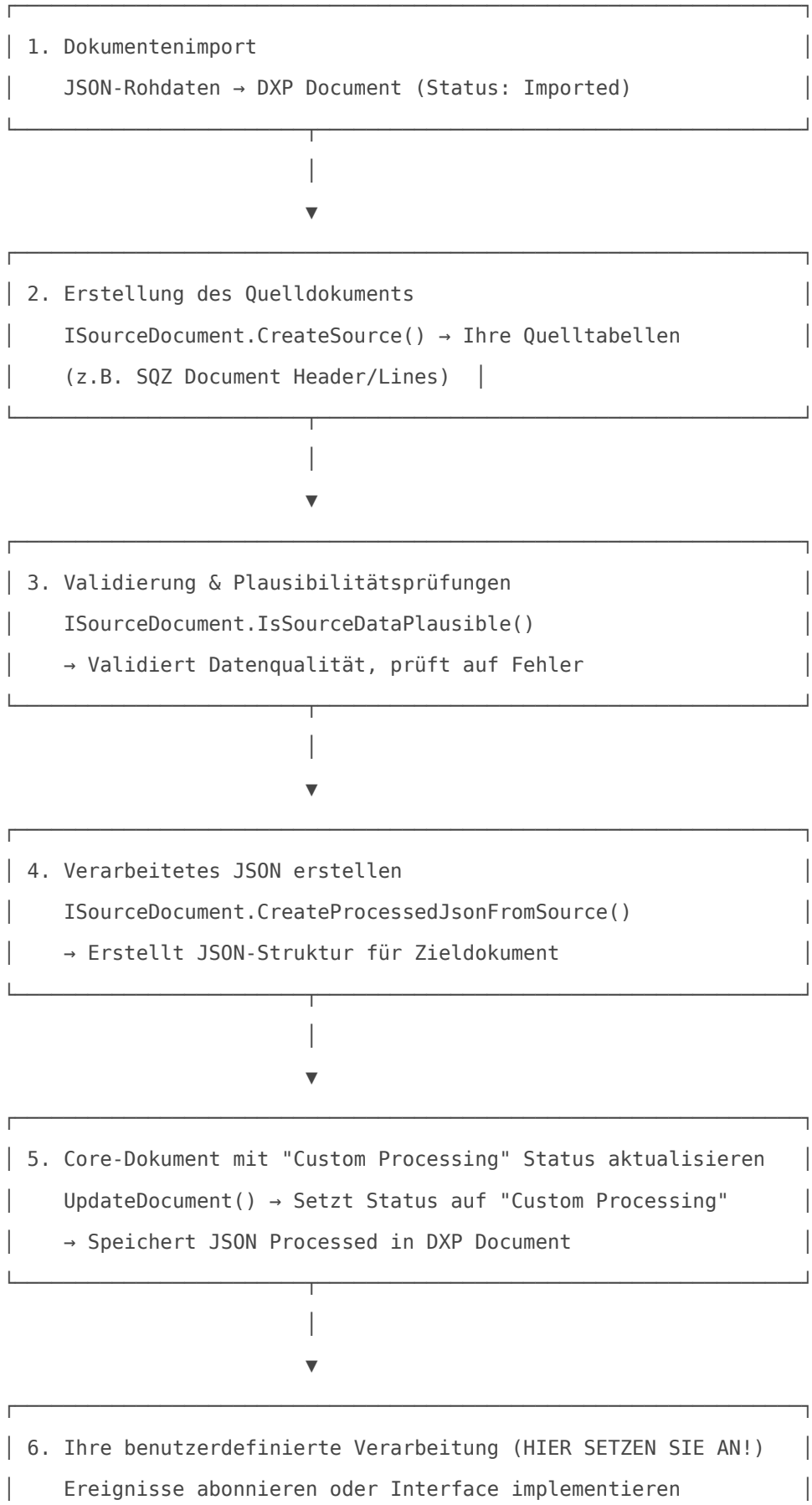
Dieses Handbuch erklärt, wie Sie benutzerdefinierte Business Central-Dokumente aus JSON-Daten erstellen, die von DEXPRO Core bereitgestellt werden. Das System empfängt Dokumente mit dem Status **"Custom Processing"** (Individuelle Verarbeitung), die Drittanbieter-Entwickler in jeden BC-Dokumententyp umwandeln können (Bestellungen, Verkaufsaufträge, Fibu-Buchblätter, Serviceaufträge usw.).

Was Sie lernen werden

- Wie Sie Dokumente abfangen, die für die individuelle Verarbeitung markiert sind
- Die JSON-Struktur mit Dokumentenkopf, Zeilen, Metadaten und benutzerdefinierten Feldern
- Wie Sie Ihr benutzerdefiniertes Dokument erstellen

Verstehen des Dokumentenablaufs

Kernworkflow



```
│    → JSON Processed aus DXP Document lesen                    │
│    → Ihr BC-Zieldokument erstellen                            │
└──────────────────────────┬────────────────────────────────────┘
                           │
                           ▼
┌──────────────────────────────────────────────────────────────┐
│ 7. Finale Statusaktualisierung                               │
│    Dokumentstatus → Transferred oder Finished                │
│    Linked-to Record Id → Ihr erstelltes BC-Dokument          │
└──────────────────────────────────────────────────────────────┘
```

Wichtige Statuswerte

Status	Beschreibung
Imported	JSON-Rohdaten empfangen, noch nicht verarbeitet
Transferred	Quelldokument erstellt, bereit zur Verarbeitung
Custom Processing	**Ihr Zielstatus** - Dokument bereit für benutzerdefinierte Verarbeitung
Closed	Zieldokument erfolgreich verbucht oder ggfs. bezahlt
Deleted	Verarbeitung abgebrochen, Dokument gelöscht

JSON-Datenstruktur

Überblick

Das `"JSON Processed"` Blob in der `DXP Document`-Tabelle enthält ein strukturiertes JSON-Objekt mit allen Informationen, die zum Erstellen Ihres Zieldokuments benötigt werden.

JSON-Struktur (Auszug)

```
{
  // ═══════════════════════════════════════════════════════
  // KOPFBEREICH - Dokumentenebene Informationen
  // ═══════════════════════════════════════════════════════
```

```
"Type": "Invoice",                    // Dokumenttyp: Invoice, Credit Memo, Order, usw.
"VendorNo": "VENDOR001",              // Kreditoren-/Kundennummer
"DocumentDate": "2025-10-15",         // Belegdatum
"PostingDate": "2025-10-16",          // Buchungsdatum
"DocumentReference": "INV-2025-001",  // Externe Belegnummer
"OrderNo": "PO-12345",                // Referenz zur Bestellung (falls vorhanden)
"PostingDescription": "Invoice Q4",   // Buchungsbeschreibung
"NetAmount": 1000.00,                 // Nettobetrag (ohne MwSt.)
"TotalAmount": 1190.00,               // Bruttobetrag (inkl. MwSt.)
"TaxAmount": 190.00,                  // Gesamter Steuerbetrag
"Currency": "EUR",                    // Währungscode

// ──────────────────────────────────────────────────────────────
// DIMENSIONEN - Kopfebene Dimensionen als Schlüssel-Wert-Paare
// ──────────────────────────────────────────────────────────────

"Dimensions": {
  "DEPARTMENT": "SALES",
  "PROJECT": "PROJ001",
  "COSTCENTER": "CC-100"
},

// ──────────────────────────────────────────────────────────────
// BENUTZERDEFINIERTE FELDER - Zusätzliche Kopffelder (siehe unten)
// ──────────────────────────────────────────────────────────────

"CustomFields": [
  {
    "Id": 50100,
    "Name": "CustomerNo",
    "Value": "CUST001",
    "Caption": "Customer Number"
  },
  {
    "Id": 50101,
    "Name": "DeliveryTerms",
    "Value": "EXW",
    "Caption": "Delivery Terms"
  }
```

```
],

// ──────────────────────────────────────────────────────────
// METADATEN - Systemgenerierte Feldzuordnungen
// ──────────────────────────────────────────────────────────

"Metadata": [
  {
    "DocumentClass": "DXP Invoice / Credit Memo",
    "FieldType": "Header",
    "FieldId": 101,
    "Value": "Additional Info"
  }
],

// ══════════════════════════════════════════════════════════
// ZEILEN-BEREICH - Dokumentzeilen
// ══════════════════════════════════════════════════════════

"Lines": [
  {
    // ──────────────────────────────────────────────────────
    // Zeilen-Grundinformationen
    // ──────────────────────────────────────────────────────

    "Type": "Item",                  // Item, G/L Account, Charge (Item), Fixed Asset
    "No": "ITEM001",                 // Artikel-/Kontonummer
    "Description": "Product A",      // Zeilenbeschreibung
    "VendorItemNo": "VEND-SKU-123",  // Kreditorenartikelnummer
    "Quantity": 10.0,                // Menge
    "UnitOfMeasure": "PCS",          // Einheitencode
    "DirectUnitCost": 100.00,        // EK-Preis
    "LineDiscount": 5.0,             // Zeilenrabatt in Prozent

    // ──────────────────────────────────────────────────────
    // Zeilen-Beträge & Steuern
    // ──────────────────────────────────────────────────────

    "NetAmount": 950.00,             // Zeilennettobetrag
```

```
"TotalAmount": 1130.50,          // Zeilenbruttobetrag
"TaxRate": 19.0,                 // Steuersatz in Prozent
"VATBusPostingGroup": "DOMESTIC",
"VATProdPostingGroup": "STANDARD",
"GenBusPostingGroup": "DOMESTIC",
"GenProdPostingGroup": "RETAIL",

// ──────────────────────────────────────────────────────────
// Bestellreferenz (für Zuordnung)
// ──────────────────────────────────────────────────────────

"OrderNo": "PO-12345",           // Referenzierte Bestellnummer
"OrderLineNo": 10000,            // Referenzierte Bestellzeilennummer
"ReceiptNo": "RCP-001",          // Referenzierte Wareneingangsnummer
"ReceiptLineNo": 10000,          // Referenzierte Wareneingangszeile

// ──────────────────────────────────────────────────────────
// Zeilen-Dimensionen
// ──────────────────────────────────────────────────────────

"Dimensions": {
  "DEPARTMENT": "PROD",
  "PROJECT": "PROJ001"
},

// ──────────────────────────────────────────────────────────
// Zeilen-Benutzerdefinierte Felder
// ──────────────────────────────────────────────────────────

"CustomFields": [
  {
    "Id": 50200,
    "Name": "SerialNo",
    "Value": "SN-12345",
    "Caption": "Serial Number"
  }
],

// ──────────────────────────────────────────────────────────
```

```
      // Zeilen-Metadaten
      // ──────────────────────────────────────────────────────────

      "Metadata": [
        {
          "DocumentClass": "DXP Invoice / Credit Memo",
          "FieldType": "Line",
          "FieldId": 201,
          "Value": "Line-specific metadata"
        }
      ]
    }
    // ... weitere Zeilen
  ]
}
```

Benutzerdefinierte Felder vs. Metadaten

Benutzerdefinierte Felder (Custom Fields):

- Benutzerdefinierte Felder aus Ihrer App
- Ggfs. zugeordnet über Custom Field Mapping
- Können jedem Feld in der Squeeze-Validierung zugeordnet werden
- Beispiel: Hinzufügen einer "Kundennr." zur Squeeze-Validierung

Metadaten:

- Systemgenerierte Felder
- Vom Squeeze-System extrahiert, aber keinem Feld in BC zugeordnet

---

Schnellstart: Einfache benutzerdefinierte Verarbeitung

Szenario

Sie möchten einen benutzerdefinierten Dokumenttyp (z.B. einen Warenausgang) aus Dokumenten erstellen, die mit dem Status "Custom Processing" markiert sind.

Schritt 1: Integration Event abonnieren

Erstellen Sie eine Codeunit, um Dokumente mit dem Status "Custom Processing" abzufangen:

```
codeunit 50100 "My Custom Document Handler"
{
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP Document Mgt.",
        'OnAfterWriteProcessedJsonToBlob', '', true, true)]
    local procedure OnAfterWriteProcessedJsonToBlob(
        var Document: Record "DXP Document";
        var ProcessedJSONObj: JsonObject)
    begin
        // Nur Custom Processing Status behandeln
        if Document.Status <> Document.Status::"Custom Processing" then
            exit;

        // Ihr benutzerdefiniertes Dokument erstellen
        CreateMyCustomDocument(Document, ProcessedJSONObj);
    end;

    local procedure CreateMyCustomDocument(
        var Document: Record "DXP Document";
        ProcessedJSON: JsonObject)
    var
        CoreTokenMgt: Codeunit "DXP Core Token Mgt.";
        JsonHelper: Codeunit "DXP Json Helper";
        MyCustomHeader: Record "My Custom Document Header";
        MyCustomLine: Record "My Custom Document Line";
        LineJArray: JsonArray;
        LineJToken: JsonToken;
        LineJObj: JsonObject;
    begin
        // ═══════════════════════════════════════════════════
        // 1. KOPF ERSTELLEN
        // ═══════════════════════════════════════════════════

        MyCustomHeader.Init();
        MyCustomHeader."No." := '';  // Wird durch Nummernserie zugewiesen
```

```
// Standard-Kopffelder lesen
MyCustomHeader."Vendor No." :=
    JsonHelper.ValAsTxt(ProcessedJSON, CoreTokenMgt.GetVendorNoTok(), false);
MyCustomHeader."Document Date" :=
    JsonHelper.ValAsDate(ProcessedJSON, CoreTokenMgt.GetDocDateTok(), false);
MyCustomHeader."Document Reference" :=
    JsonHelper.ValAsTxt(ProcessedJSON, CoreTokenMgt.GetDocReferenceTok(), false);

MyCustomHeader.Insert(true);

// ═══════════════════════════════════════════════════════
// 2. DIMENSIONEN VERARBEITEN (wenn Ihr Dokument diese unterstützt)
// ═══════════════════════════════════════════════════════

ProcessHeaderDimensions(ProcessedJSON, MyCustomHeader);

// ═══════════════════════════════════════════════════════
// 3. BENUTZERDEFINIERTE FELDER VERARBEITEN
// ═══════════════════════════════════════════════════════

ProcessCustomFields(ProcessedJSON, MyCustomHeader);

// ═══════════════════════════════════════════════════════
// 4. ZEILEN ERSTELLEN
// ═══════════════════════════════════════════════════════

LineJArray := JsonHelper.ReadJArrayFromObj(ProcessedJSON, CoreTokenMgt.GetLinesTok());

foreach LineJToken in LineJArray do begin
    LineJObj := LineJToken.AsObject();

    MyCustomLine.Init();
    MyCustomLine."Document No." := MyCustomHeader."No.";
    MyCustomLine."Line No." := GetNextLineNo(MyCustomHeader."No.");

    // Zeilenfelder lesen
    MyCustomLine."Item No." :=
        JsonHelper.ValAsTxt(LineJObj, CoreTokenMgt.GetNoTok(), false);
    MyCustomLine.Description :=
```

```
                JsonHelper.ValAsTxt(LineJObj, CoreTokenMgt.GetDescriptionTok(), false);
            MyCustomLine.Quantity :=
                JsonHelper.ValAsDec(LineJObj, CoreTokenMgt.GetQtyTok(), false);
            MyCustomLine."Unit of Measure" :=
                JsonHelper.ValAsTxt(LineJObj, CoreTokenMgt.GetUoMTok(), false);

            MyCustomLine.Insert(true);

            // Zeilen-Dimensionen und benutzerdefinierte Felder verarbeiten, falls
erforderlich
            ProcessLineDimensions(LineJObj, MyCustomLine);
            ProcessLineCustomFields(LineJObj, MyCustomLine);
        end;

        // ═══════════════════════════════════════════════════
        // 5. CORE-DOKUMENTSTATUS AKTUALISIEREN
        // ═══════════════════════════════════════════════════

        UpdateCoreDocument(Document, MyCustomHeader);
    end;

    local procedure ProcessHeaderDimensions(ProcessedJSON: JsonObject; var MyCustomHeader:
Record "My Custom Document Header")
    var
        CoreTokenMgt: Codeunit "DXP Core Token Mgt.";
        DocTransferMgt: Codeunit "DXP Document Transfer Mgt.";
        DimensionsJObj: JsonObject;
        DimSetID: Integer;
    begin
        // Dimensionen aus JSON lesen
        if ProcessedJSON.Contains(CoreTokenMgt.GetDimensionsTok()) then begin
            ProcessedJSON.Get(CoreTokenMgt.GetDimensionsTok(), DimensionsJObj);

            // In Dimensionssatz-ID konvertieren
            if DocTransferMgt.GetDimSetIdFromJsonObj(DimensionsJObj, DimSetID) then begin
                MyCustomHeader."Dimension Set ID" := DimSetID;
                MyCustomHeader.Modify(true);
            end;
        end;
```

```
    end;

    local procedure ProcessCustomFields(ProcessedJSON: JsonObject; var MyCustomHeader: Record
"My Custom Document Header")
    var
        CoreTokenMgt: Codeunit "DXP Core Token Mgt.";
        JsonHelper: Codeunit "DXP Json Helper";
        CustomFieldsJArray: JsonArray;
        CustomFieldJToken: JsonToken;
        CustomFieldJObj: JsonObject;
        FieldName: Text;
        FieldValue: Text;
    begin
        // Benutzerdefinierte Felder-Array lesen
        if not ProcessedJSON.Contains(CoreTokenMgt.GetCustomFieldsTok()) then
            exit;

        CustomFieldsJArray := JsonHelper.ReadJArrayFromObj(ProcessedJSON,
CoreTokenMgt.GetCustomFieldsTok());

        foreach CustomFieldJToken in CustomFieldsJArray do begin
            CustomFieldJObj := CustomFieldJToken.AsObject();
            FieldName := JsonHelper.ValAsTxt(CustomFieldJObj, CoreTokenMgt.GetNameTok(),
false);
            FieldValue := JsonHelper.ValAsTxt(CustomFieldJObj, CoreTokenMgt.GetValueTok(),
false);

            // Zu Ihren benutzerdefinierten Feldern zuordnen
            case FieldName of
                'CustomerNo':
                    MyCustomHeader."Customer No." := CopyStr(FieldValue, 1, 20);
                'DeliveryTerms':
                    MyCustomHeader."Delivery Terms" := CopyStr(FieldValue, 1, 10);
                // Weitere Feldzuordnungen nach Bedarf hinzufügen
            end;
        end;

        MyCustomHeader.Modify(true);
    end;
```

```
    local procedure ProcessLineDimensions(LineJObj: JsonObject; var MyCustomLine: Record "My Custom Document Line")
    var
        CoreTokenMgt: Codeunit "DXP Core Token Mgt.";
        DocTransferMgt: Codeunit "DXP Document Transfer Mgt.";
        DimensionsJObj: JsonObject;
        DimSetID: Integer;
    begin
        if LineJObj.Contains(CoreTokenMgt.GetDimensionsTok()) then begin
            LineJObj.Get(CoreTokenMgt.GetDimensionsTok(), DimensionsJObj);

            if DocTransferMgt.GetDimSetIdFromJsonObj(DimensionsJObj, DimSetID) then begin
                MyCustomLine."Dimension Set ID" := DimSetID;
                MyCustomLine.Modify(true);
            end;
        end;
    end;

    local procedure ProcessLineCustomFields(LineJObj: JsonObject; var MyCustomLine: Record "My Custom Document Line")
    var
        CoreTokenMgt: Codeunit "DXP Core Token Mgt.";
        JsonHelper: Codeunit "DXP Json Helper";
        CustomFieldsJArray: JsonArray;
        CustomFieldJToken: JsonToken;
        CustomFieldJObj: JsonObject;
        FieldName: Text;
        FieldValue: Text;
    begin
        if not LineJObj.Contains(CoreTokenMgt.GetCustomFieldsTok()) then
            exit;

        CustomFieldsJArray := JsonHelper.ReadJArrayFromObj(LineJObj, CoreTokenMgt.GetCustomFieldsTok());

        foreach CustomFieldJToken in CustomFieldsJArray do begin
            CustomFieldJObj := CustomFieldJToken.AsObject();
            FieldName := JsonHelper.ValAsTxt(CustomFieldJObj, CoreTokenMgt.GetNameTok(),
```

```
false);
            FieldValue := JsonHelper.ValAsTxt(CustomFieldJObj, CoreTokenMgt.GetValueTok(),
false);

            // Zu Ihren benutzerdefinierten Zeilenfeldern zuordnen
            case FieldName of
                'SerialNo':
                    MyCustomLine."Serial No." := CopyStr(FieldValue, 1, 50);
                // Weitere Feldzuordnungen hinzufügen
            end;
        end;

        MyCustomLine.Modify(true);
    end;

    local procedure UpdateCoreDocument(var Document: Record "DXP Document"; MyCustomHeader:
Record "My Custom Document Header")
    var
        DocumentMgt: Codeunit "DXP Document Mgt.";
    begin
        // Core-Dokument aktualisieren, um es mit Ihrem erstellten Dokument zu verknüpfen
        Document.Status := Document.Status::Transferred;
        Document."Linked-to Record Id" := MyCustomHeader.RecordId;
        Document.Modify(true);

        // Optional: Anhänge von Core zu Ihrem Dokument übertragen
        TransferAttachments(Document, MyCustomHeader);
    end;

    local procedure TransferAttachments(Document: Record "DXP Document"; MyCustomHeader:
Record "My Custom Document Header")
    var
        DocAttachment: Record "DXP Document Attachment";
        MyDocAttachment: Record "Document Attachment";
        InStr: InStream;
    begin
        DocAttachment.SetRange("Document No.", Document."No.");
        if DocAttachment.FindSet() then
            repeat
```

```
                MyDocAttachment.Init();
                MyDocAttachment.ID := 0;

                DocAttachment."File Content".CreateInStream(InStr);
                MyDocAttachment.SaveAttachmentFromStream(
                    InStr,
                    MyCustomHeader.RecordId,
                    DocAttachment."File Name");
            until DocAttachment.Next() = 0;
    end;

    local procedure GetNextLineNo(DocumentNo: Code[20]): Integer
    var
        MyCustomLine: Record "My Custom Document Line";
    begin
        MyCustomLine.SetRange("Document No.", DocumentNo);
        if MyCustomLine.FindLast() then
            exit(MyCustomLine."Line No." + 10000);
        exit(10000);
    end;
}
```

Wichtige Hilfs-Codeunits

Codeunit	Zweck
DXP Core Token Mgt.	Stellt Token-Namen für JSON-Felder bereit (GetVendorNoTok(), GetDocDateTok(), usw.)
DXP Json Helper	JSON-Parsing-Hilfsfunktionen (ValAsTxt(), ValAsDate(), ReadJArrayFromObj(), usw.)
DXP Document Transfer Mgt.	Dimensionsverarbeitung, Metadatenübertragung, Behandlung benutzerdefinierter Felder
DXP Document Mgt.	Kern-Dokumentenverwaltung (UpdateDocument(), UpdateDocumentStatus(), usw.)

Bei Fragen oder Unklarheiten wenden Sie sich bitte an die DEXPRO Solutions GmbH.

Revision #12
Created 2025-10-16 09:50:53 UTC by Bernd Feddersen
Updated 2026-02-11 12:32:52 UTC by Bernd Feddersen