Custom Document Creation (Custom Processing)

Table of Contents

1. Overview
2. Understanding the Document Flow
3. JSON Data Structure
4. Quick Start: Simple Custom Processing

Overview

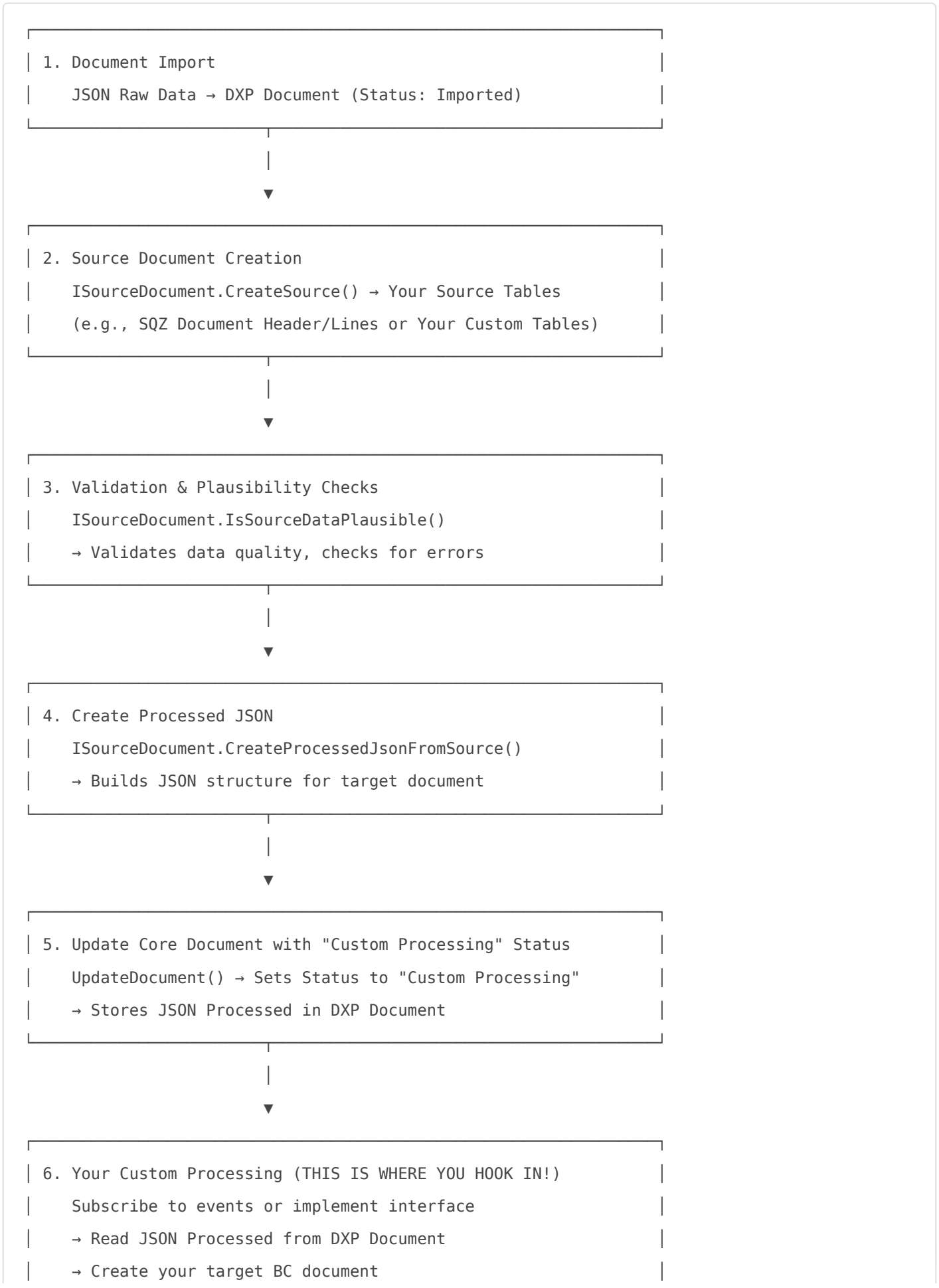
This guide explains how to create custom Business Central documents from JSON data provided by DEXPRO Core. The system receives documents with status **"Custom Processing"**, which third-party developers can transform into any BC document type (Purchase Orders, Sales Orders, G/L Journals, Service Orders, etc.).

What You'll Learn

- How to intercept documents marked for custom processing
- The JSON structure containing document header, lines, metadata, and custom fields
- How to create your custom document

Understanding the Document Flow

Core Workflow


1. Document Import
JSON Raw Data → DXP Document (Status: Imported)
2. Source Document Creation
ISourceDocument.CreateSource() → Your Source Tables
(e.g., SQZ Document Header/Lines or Your Custom Tables)
3. Validation & Plausibility Checks
ISourceDocument.IsSourceDataPlausible()
→ Validates data quality, checks for errors
4. Create Processed JSON
ISourceDocument.CreateProcessedJsonFromSource()
→ Builds JSON structure for target document
5. Update Core Document with "Custom Processing" Status
UpdateDocument() → Sets Status to "Custom Processing"
→ Stores JSON Processed in DXP Document
6. Your Custom Processing (THIS IS WHERE YOU HOOK IN!)
Subscribe to events or implement interface
→ Read JSON Processed from DXP Document
→ Create your target BC document

```
└──────────────────────────┬──────────────────────────────────────────┘
                           │
                           ▼
┌──────────────────────────────────────────────────────────────────────┐
│ 7. Final Status Update                                               │
│    Document Status → Transferred or Finished                         │
│    Linked-to Record Id → Your created BC document                    │
└──────────────────────────────────────────────────────────────────────┘
```

Key Status Values

Status	Description
Imported	Raw JSON received, not yet processed
Transferred	Source document created, ready for processing
Custom Processing	**Your target status** - Document ready for custom handling
Finished	Target document successfully created
Deleted	Processing complete, document archived

JSON Data Structure

Overview

The `"JSON Processed"` blob in the `DXP Document` table contains a structured JSON object with all information needed to create your target document.

Complete JSON Structure

```
{
  // ═══════════════════════════════════════════════════════
  // HEADER SECTION - Document-level information
  // ═══════════════════════════════════════════════════════

  "Type": "Invoice",                    // Document type: Invoice, Credit Memo, Order, etc.
  "VendorNo": "VENDOR001",              // Vendor/Customer number
```

```
  "DocumentDate": "2025-10-15",         // Document date
  "PostingDate": "2025-10-16",          // Posting date
  "DocumentReference": "INV-2025-001",  // External document number
  "OrderNo": "PO-12345",                // Reference to order (if applicable)
  "PostingDescription": "Invoice Q4",   // Posting description
  "NetAmount": 1000.00,                 // Net amount (excluding tax)
  "TotalAmount": 1190.00,               // Gross amount (including tax)
  "TaxAmount": 190.00,                  // Total tax amount
  "Currency": "EUR",                    // Currency code
  "ResponsibilityCenter": "BERLIN",     // Responsibility center (blank: default from user
setup / vendor)
  "VATBusPostingGroup": "EU",           // VAT business posting group (blank: vendor default)
- the lines' default

  // ──────────────────────────────────────────────────────────
  // DIMENSIONS - Header-level dimensions as key-value pairs
  // ──────────────────────────────────────────────────────────

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

  // ──────────────────────────────────────────────────────────
  // CUSTOM FIELDS - Additional header fields (see below)
  // ──────────────────────────────────────────────────────────

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

```
    "Caption": "Delivery Terms"
  }
],

// ──────────────────────────────────────────────────────────
// METADATA - System-generated field mappings
// ──────────────────────────────────────────────────────────

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

// ══════════════════════════════════════════════════════════
// LINES SECTION - Document line items
// ══════════════════════════════════════════════════════════

"Lines": [
  {
    // ──────────────────────────────────────────────────────
    // Line Basic Information
    // ──────────────────────────────────────────────────────

    "Type": "Item",                  // Item, G/L Account, Charge (Item), Fixed Asset
    "No": "ITEM001",                 // Item/Account number
    "Description": "Product A",      // Line description
    "VendorItemNo": "VEND-SKU-123",  // Vendor's item number
    "Quantity": 10.0,                // Quantity
    "UnitOfMeasure": "PCS",          // Unit of measure
    "DirectUnitCost": 100.00,        // Unit price
    "LineDiscount": 5.0,             // Line discount percentage

    // ──────────────────────────────────────────────────────
    // Line Amounts & Tax
    // ──────────────────────────────────────────────────────
```

```
"NetAmount": 950.00,             // Line net amount
"TotalAmount": 1130.50,          // Line gross amount
"TaxRate": 19.0,                 // Tax percentage
"VATBusPostingGroup": "DOMESTIC",
"VATProdPostingGroup": "STANDARD",
"GenBusPostingGroup": "DOMESTIC",
"GenProdPostingGroup": "RETAIL",

// ──────────────────────────────────────────────────────────
// Order Reference (for matching)
// ──────────────────────────────────────────────────────────

"OrderNo": "PO-12345",           // Referenced order number
"OrderLineNo": 10000,            // Referenced order line number
"ReceiptNo": "RCP-001",          // Referenced receipt number
"ReceiptLineNo": 10000,          // Referenced receipt line number

// ──────────────────────────────────────────────────────────
// Line Dimensions
// ──────────────────────────────────────────────────────────

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

// ──────────────────────────────────────────────────────────
// Line Custom Fields
// ──────────────────────────────────────────────────────────

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

```
      // ──────────────────────────────────────────────────────────
      // Line Metadata
      // ──────────────────────────────────────────────────────────

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

Custom Fields vs. Metadata

Custom Fields:

- User-defined fields from your app
- Mapped via Custom Field Mapping
- Can be mapped to any field in the Squeeze Validation
- Example: Adding a "Customer No." to the Squeeze Validation

Metadata:

- System-generated fields
- Extracted by the Squeeze System but not mapped to any Field in BC

Quick Start: Simple Custom Processing

Scenario

You want to create a custom document type (e.g., a Warehouse Shipment) from documents marked with "Custom Processing" status.

Step 1: Subscribe to the Integration Event

Create a codeunit to intercept documents with "Custom Processing" status:

```
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
        // Only handle Custom Processing status
        if Document.Status <> Document.Status::"Custom Processing" then
            exit;

        // Create your custom document
        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. CREATE HEADER
        // ═══════════════════════════════════════════════════

        MyCustomHeader.Init();
        MyCustomHeader."No." := '';  // Will be assigned by number series
```

```
// Read standard header fields
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. PROCESS DIMENSIONS (if your document supports them)
// ═══════════════════════════════════════════════════════

ProcessHeaderDimensions(ProcessedJSON, MyCustomHeader);

// ═══════════════════════════════════════════════════════
// 3. PROCESS CUSTOM FIELDS
// ═══════════════════════════════════════════════════════

ProcessCustomFields(ProcessedJSON, MyCustomHeader);

// ═══════════════════════════════════════════════════════
// 4. CREATE LINES
// ═══════════════════════════════════════════════════════

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.");

    // Read line fields
    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);

            // Process line dimensions and custom fields if needed
            ProcessLineDimensions(LineJObj, MyCustomLine);
            ProcessLineCustomFields(LineJObj, MyCustomLine);
        end;

        // ═══════════════════════════════════════════════════
        // 5. UPDATE CORE DOCUMENT STATUS
        // ═══════════════════════════════════════════════════

        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
        // Read dimensions from JSON
        if ProcessedJSON.Contains(CoreTokenMgt.GetDimensionsTok()) then begin
            ProcessedJSON.Get(CoreTokenMgt.GetDimensionsTok(), DimensionsJObj);

            // Convert to Dimension Set ID
            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
        // Read custom fields array
        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);

            // Map to your custom fields
            case FieldName of
                'CustomerNo':
                    MyCustomHeader."Customer No." := CopyStr(FieldValue, 1, 20);
                'DeliveryTerms':
                    MyCustomHeader."Delivery Terms" := CopyStr(FieldValue, 1, 10);
                // Add more field mappings as needed
            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);

            // Map to your line custom fields
            case FieldName of
                'SerialNo':
                    MyCustomLine."Serial No." := CopyStr(FieldValue, 1, 50);
                // Add more field mappings
            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
        // Update the core document to link it to your created document
        Document.Status := Document.Status::Transferred;
        Document."Linked-to Record Id" := MyCustomHeader.RecordId;
        Document.Modify(true);

        // Optionally, transfer attachments from Core to your document
        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;
}
```

Key Helper Codeunits

Codeunit	Purpose
DXP Core Token Mgt.	Provides token names for JSON fields (GetVendorNoTok(), GetDocDateTok(), etc.)
DXP Json Helper	JSON parsing utilities (ValAsTxt(), ValAsDate(), ReadJArrayFromObj(), etc.)
DXP Document Transfer Mgt.	Dimension processing, metadata transfer, custom field handling
DXP Document Mgt.	Core document management (UpdateDocument(), UpdateDocumentStatus(), etc.)

For questions or clarifications, please contact DEXPRO Solutions GmbH.

Revision #3
Created 2025-10-16 09:53:41 UTC by Bernd Feddersen
Updated 2026-10-21 11:50:29 UTC by Bernd Feddersen