Extension Development

This chapter lists important integration events and examples that help in developing an extension for BREEZE Interface for BC.

Important:
Customizations may only be performed by appropriately trained consultants/developers.
Furthermore, these are always outside of the standard support.

- Guide to implementing customized document creation

Guide to implementing customized document creation

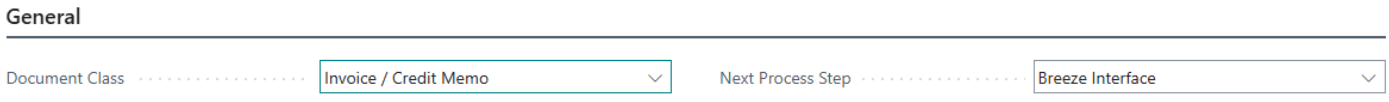
Overview

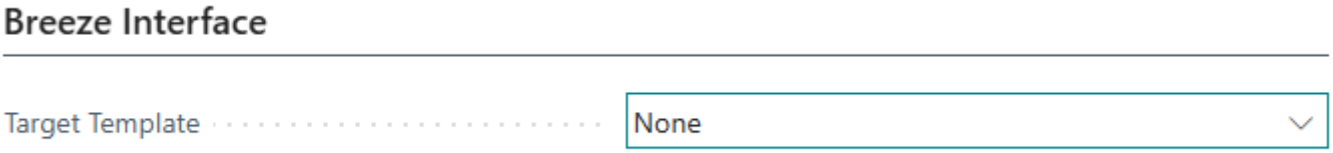
This documentation provides guidance on implementing custom document creation processes using the Integration Event `OnDocumentCreation` in the DEXPRO Core Framework. While the default implementation creates purchase documents (Tables 38, 39), this guide shows you how to implement your own document creation logic using the provided JSON data.

Requirements

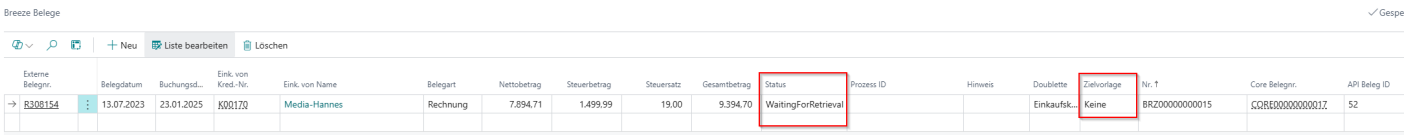
The following setting must be made in the document class setup before validating a document:

- **Next process step:** Breeze Interface
- **Target Template:** None





Have you validated a document with the DEXPRO Squeeze app and transferred it to the next process step, Breeze Interface? If so, you will now see it in Breeze Interface in a waiting status (*WaitingForRetrieval*). At this point, you have the option of transferring the validated document to an approval workflow so that it can be reported back after approval and transferred to the target document creation process. Further information on the Breeze Interface API can be found here.



Once you have changed the status of the Breeze receipt to "*Processed*", it will be reported to the DEXPRO Core for further processing—this is the integration point where you need to start.

Integration point

The central integration point is the `OnDocumentCreation` event in *Codeunit 70954599 "DXP Doc. Creation Def. Impl."* in the DEXPRO Core app. This event is triggered during document processing and enables the implementation of custom document creation logic.

Event Signature

```
[IntegrationEvent(false, false)]
local procedure OnDocumentCreation(JObject: JsonObject; var CreatedDocumentRecordId: RecordId)
```

Parameter

- `JObject`: Contains the complete JSON data with the document information
- `CreatedDocumentRecordId`: Must be set with the RecordId of the created document

Implementation Guide

Step 1: Create Subscriber-Codeunit

First, create a codeunit that subscribes to the OnDocumentCreation event:

```
codeunit 50100 "Custom Document Creation"
{
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP Doc. Creation Def. Impl.",
'OnDocumentCreation', '', false, false)]
    local procedure OnDocumentCreation(JObject: JsonObject; var CreatedDocumentRecordId:
RecordId)
    begin
        // Your implementation here
        CreateCustomDocument(JObject, CreatedDocumentRecordId);
    end;
}
```

Step 2: Implement document creation logic

Here is an example of how to implement custom document creation:

```
local procedure CreateCustomDocument(JObject: JsonObject; var CreatedDocumentRecordId:
RecordId)
var
    CustomHeader: Record "Custom Document Header";
    CustomLine: Record "Custom Document Line";
    JsonHelper: Codeunit "DXP Json Helper";
    HeaderJObject: JsonObject;
    LinesJArray: JsonArray;
    LineJToken: JsonToken;
    LineNo: Integer;
begin
    // 1. Create Header
    CustomHeader.Init();
    CustomHeader."Document Type" := GetDocumentType(JObject);
    CustomHeader."Vendor No." := CopyStr(JsonHelper.ValAsTxt(JObject, 'vendorNo', true), 1,
MaxStrLen(CustomHeader."Vendor No."));
    CustomHeader."Document Date" := JsonHelper.ValAsDate(JObject, 'docDate', true);
    CustomHeader."Posting Date" := JsonHelper.ValAsDate(JObject, 'postingDate', true);
    CustomHeader."Document Reference" := CopyStr(JsonHelper.ValAsTxt(JObject, 'docReference',
true), 1, MaxStrLen(CustomHeader."Document Reference"));
    CustomHeader.Insert(true);

    // 2. Process Lines
    LinesJArray := JsonHelper.ReadJArrayFromObj(JObject, 'lines');
    LineNo := 10000;

    foreach LineJToken in LinesJArray do begin
        CustomLine.Init();
        CustomLine."Document No." := CustomHeader."No.";
        CustomLine."Line No." := LineNo;
        CustomLine."Item No." := CopyStr(JsonHelper.ValAsTxt(LineJToken.AsObject(), 'no',
true), 1, MaxStrLen(CustomLine."Item No."));
        CustomLine.Quantity := JsonHelper.ValAsDec(LineJToken.AsObject(), 'qty', true);
        CustomLine."Unit Price" := JsonHelper.ValAsDec(LineJToken.AsObject(), 'unitPrice',
true);
```

```
        CustomLine.Insert(true);

        LineNo += 10000;
    end;

    CreatedDocumentRecordId := CustomHeader.RecordId;
end;
```

JSON-Structure

The JSON input follows this structure:

```
{
   "type":"Invoice",
   "vendorNo":"K00170",
   "iban":"DE58520503530052599766",
   "vatRegNo":"DE540026784",
   "barcode":"123456789",
   "docDate":"2023-07-13",
   "serviceDate":"2023-07-13",
   "postingDate":"2025-01-23",
   "postingDesc":"Die ist eine Buchungsbeschreibung.",
   "docReference":"R308154",
   "orderNo":"B23106029",
   "netAmount":7894.71,
   "netAmount2":0.0,
   "netAmount3":0.0,
   "taxRate":19.0,
   "taxRate2":0.0,
   "taxRate3":0.0,
   "taxAmount":1499.99,
   "taxAmount2":0.0,
   "taxAmount3":0.0,
   "totalAmount":9394.7,
   "currency":"EUR",
   "assignedTo":"BERND.FEDDERSEN",
   "note":"Die ist eine Bemerkung!",
   "dimensions":{
```

```
      "ABTEILUNG":"PROD",
      "EINKÄUFER":"BF",
      "KOSTENTRÄGER":"IT"
   },
   "dimensionSetID":27,
   "customFields":{
      "docClass":"DXP Invoice / Credit Memo",
      "namesAndValues":[
         {
            "name":"Customer_No",
            "value":"01121212"
         },
         {
            "name":"Custom_Date",
            "value":"2025-01-26"
         }
      ]
   },
   "orderMatchDifference":"false",
   "lines":[
      {
         "orderNo":"B23106029",
         "orderLineNo":10000,
         "receiptNo":"ELIEF107004",
         "receiptLineNo":10000,
         "vendorItemNo":"",
         "type":"Item",
         "no":"90002",
         "description":"JogiTek G 1337 Gaming-Headset",
         "qty":17.0,
         "uom":"STÜCK",
         "unitPrice":99.99,
         "lineDisc":0.0,
         "netAmount":1699.83,
         "totalAmount":2022.8,
         "taxRate":19.0,
         "vatBusPostingGroup":"INLAND",
         "vatProdPostingGroup":"MWST.19",
         "dimensions":{
```

```
            "ABTEILUNG":"PROD",
            "BEREICH":"30",
            "EINKÄUFER":"BF",
            "KOSTENTRÄGER":"IT",
            "VERKAUFSKAMPAGNE":"WINTER"
         },
         "dimensionSetID":53,
         "genProdPostingGroup":"HANDEL",
         "genBusPostingGroup":"INLAND",
         "deferralCode":"",
         "quantityDifference":"false",
         "unitPriceDifference":"false",
         "discountDifference":"false",
         "receiptDateDifference":"false",
         "customFields":{
            "docClass":"DXP Invoice / Credit Memo",
            "namesAndValues":[
               {
                  "name":"Custom_Field_1",
                  "value":"Text in Zeile 10000"
               },
               {
                  "name":"Custom_Field_2",
                  "value":"2025-01-31"
               }
            ]
         }
      },
      {
         "orderNo":"B23106029",
         "orderLineNo":30000,
         "receiptNo":"ELIEF107004",
         "receiptLineNo":30000,
         "vendorItemNo":"",
         "type":"Item",
         "no":"90005",
         "description":"Sumsing Galaxy 23",
         "qty":12.0,
         "uom":"STÜCK",
```

```
      "unitPrice":349.99,
      "lineDisc":0.0,
      "netAmount":4199.88,
      "totalAmount":4997.86,
      "taxRate":19.0,
      "vatBusPostingGroup":"INLAND",
      "vatProdPostingGroup":"MWST.19",
      "dimensions":{
         "ABTEILUNG":"PROD",
         "BEREICH":"50",
         "EINKÄUFER":"BF",
         "KOSTENTRÄGER":"IT",
         "VERKAUFSKAMPAGNE":"WINTER"
      },
      "dimensionSetID":55,
      "genProdPostingGroup":"HANDEL",
      "genBusPostingGroup":"INLAND",
      "deferralCode":"",
      "quantityDifference":"false",
      "unitPriceDifference":"false",
      "discountDifference":"false",
      "receiptDateDifference":"false",
      "customFields":{
         "docClass":"DXP Invoice / Credit Memo",
         "namesAndValues":[
            {
               "name":"Custom_Field_1",
               "value":"Text in Zeile 20000"
            },
            {
               "name":"Custom_Field_2",
               "value":"2025-01-30"
            }
         ]
      }
   },
   {
      "orderNo":"B23106029",
      "orderLineNo":20000,
```

```
"receiptNo":"ELIEF107004",
"receiptLineNo":20000,
"vendorItemNo":"",
"type":"Item",
"no":"90000",
"description":"Y-Phone Ultra Pro 23",
"qty":5.0,
"uom":"STÜCK",
"unitPrice":399.0,
"lineDisc":0.0,
"netAmount":1995.0,
"totalAmount":2374.05,
"taxRate":19.0,
"vatBusPostingGroup":"INLAND",
"vatProdPostingGroup":"MWST.19",
"dimensions":{
   "ABTEILUNG":"PROD",
   "BEREICH":"70",
   "EINKÄUFER":"BF",
   "KOSTENTRÄGER":"IT",
   "VERKÄUFER":"JH",
   "VERKAUFSKAMPAGNE":"WINTER"
},
"dimensionSetID":58,
"genProdPostingGroup":"HANDEL",
"genBusPostingGroup":"INLAND",
"deferralCode":"",
"quantityDifference":"false",
"unitPriceDifference":"false",
"discountDifference":"false",
"receiptDateDifference":"false",
"customFields":{
   "docClass":"DXP Invoice / Credit Memo",
   "namesAndValues":[
      {
         "name":"Custom_Field_1",
         "value":"Text in Zeile 30000"
      },
      {
```

```
                "name":"Custom_Field_2",
                "value":"2025-01-29"
            }
        ]
    }
  }
]
}
```

Best practice

1. **Error handling**: Implement proper error handling for JSON parsing and database operations:

```
local procedure CreateCustomDocument(JObject: JsonObject; var CreatedDocumentRecordId:
RecordId)
var
    CustomHeader: Record "Custom Document Header";
begin
    if not DoesJsonHaveRequiredFields(JObject) then
        Error('Required fields are missing in the JSON payload');

    if not TryCreateCustomHeader(JObject, CustomHeader) then
        Error('Failed to create document header');

    // ... rest of the implementation
end;
```

2. **Validation**: Implement proper validation before document creation:

```
local procedure ValidateDocument(JObject: JsonObject): Boolean
var
    Vendor: Record Vendor;
    VendorNo: Code[20];
begin
    VendorNo := CopyStr(JsonHelper.ValAsTxt(JObject, 'vendorNo', true), 1,
MaxStrLen(VendorNo));
    if not Vendor.Get(VendorNo) then
```

```
        Error('Vendor %1 does not exist', VendorNo);
    // Add more validation as needed
    exit(true);
end;
```

Common scenarios

Scenario 1: Create different document types

```
local procedure GetDocumentType(JObject: JsonObject): Enum "Custom Document Type"
var
    DocType: Text;
begin
    DocType := JsonHelper.ValAsTxt(JObject, 'type', true);
    case DocType of
        'Invoice':
            exit("Custom Document Type"::Invoice);
        'CreditMemo':
            exit("Custom Document Type"::"Credit Memo");
        else
            Error('Unknown document type: %1', DocType);
    end;
end;
```

Scenario 2: Processing custom fields

```
local procedure ProcessCustomFields(JObject: JsonObject; var CustomDoc: Record "Custom
Document Header")
var
    CustomFieldsObj: JsonObject;
    CustomFieldsArr: JsonArray;
    FieldToken: JsonToken;
begin
    if not JsonHelper.TokenExists(JObject, 'customFields') then
        exit;

    CustomFieldsObj := JsonHelper.ReadJObjectFromObj(JObject, 'customFields');
```

```
    CustomFieldsArr := JsonHelper.ReadJArrayFromObj(CustomFieldsObj, 'namesAndValues');

    foreach FieldToken in CustomFieldsArr do
        ProcessCustomField(FieldToken.AsObject(), CustomDoc);
end;
```

Troubleshooting

1. **RecordId is empty**: Ensure that you set the CreatedDocumentRecordId parameter:

```
CreatedDocumentRecordId := CustomHeader.RecordId;
```

2. **JSON-Parsing-Error**: Use the JsonHelper functions for secure parsing:

```
if not JsonHelper.TokenExists(JObject, 'vendorNo') then
    Error('Vendor number is missing in JSON');
```

3. **Data validation**: Implement proper data validation:

```
if JsonHelper.ValAsDec(LineJToken.AsObject(), 'qty', true) <= 0 then
    Error('Quantity must be greater than 0');
```