

# Developing a Custom Target Template

This page is for developers and partners who want to extend Breeze Interface with a **custom target template**. It uses one worked example – an invoice that references a purchase order – to show how a document is given a target template other than the configured default, and which processing runs when the document is set to **Processed**.

The example: an invoice arrives through Squeeze and is validated as usual. If the document header carries a purchase order number, a purchase invoice should *not* be created as it would be by default. Instead, the purchase order referenced in the header is to be posted.

## Which approach is the right one?

There are two ways to implement your own document creation. They are mutually exclusive – pick the right one before you start:

|                                     | Custom target template (this page)                                                                                   | <code>OnDocumentCreation</code> event         |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| Target template setting             | your own target template                                                                                             | <b>None</b>                                   |
| Validation by Breeze                | yes, with the switches from the target template setup                                                                | no                                            |
| Several document types side by side | yes, one target template per case                                                                                    | a single implementation for all documents     |
| Suited for                          | handling standard purchasing documents differently – such as posting a purchase order instead of creating an invoice | writing documents into entirely custom tables |

If you want to write into your own tables and do not need Breeze's validation, follow the page **Guide to implementing customized document creation** instead. That approach does not fit the example described here: it requires the target template **None** and therefore bypasses exactly the validation an invoice with an order reference needs.

## The standard flow

A document passes these stations on its way from Squeeze to the target document in Business Central:

1. Squeeze hands the document to Core. Core creates a record in the `DXP Document` table.
2. Core uses the **document class** (`DXP Document Class`) to determine the next process step. If that step is **Breeze Interface**, a Breeze document (`DXP BRZ IF Document`) is created.
3. On creation, the Breeze document receives its **target template** from the document class setup – the **Target Template** field on the **Document Class Setup** page.
4. The document is handled in the external workflow and returns with status **Processed**.
5. Breeze Interface then invokes the implementation registered for that target template. It validates the document and hands it to Core, which creates the target document.

Technically, the target template is an enum implementing an interface:

- `DXP Breeze IF Target Template` – the target template enum, `Extensible = true`.
- `DXP BRZIF IDocument Processing` – the interface every target template implements. It has exactly one method:  

```
procedure ProcessDocument(var BREEZEDoc: Record "DXP BRZ IF Document"; JObject: JsonObject)
```

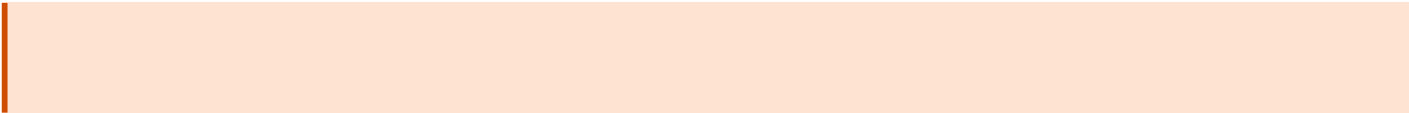
These target templates ship with the app:

| Value | Target template    | Implementation                              |
|-------|--------------------|---------------------------------------------|
| 0     | None               | <code>DXP BRZ Default Processing</code>     |
| 1     | Purchase Document  | <code>DXP BRZ IF PInv/Crdt Memo Proc</code> |
| 2     | Purchase Order     | <code>DXP BRZ IF Purch. Order Proc.</code>  |
| 3     | Order Confirmation | <code>DXP BRZ IF Order Conf. Proc.</code>   |
| 4     | Delivery Note      | <code>DXP BRZ IF Del. Note Proc.</code>     |

{add screenshot of Document Class Setup here}

## Prerequisites

- Your own extension with a dependency on **DEXPRO Breeze Interface** (app id `7c431886-108e-44c3-9328-30e2b93edfa6`) and – carried along through `propagateDependencies` – on **DEXPRO Core**.
- Your own object id range. Do **not** use the range of Breeze Interface (70954950–70954969) or Core (70954575–70954624).
- Your own object name affix as required by your `AppSourceCop.json`.



Never use the `DXP` affix for your own objects – it is reserved for the DEXPRO apps. In the code samples on this page, `XYZ` stands for your own affix.

## Step 1 – Register the target template as an enum extension

The `DXP Breeze IF Target Template` enum is extensible. Add a value and point `Implementation` at your own codeunit:

```
enumextension 50100 "XYZ Target Template Ext." extends "DXP Breeze IF Target Template"
{
    value(50100; "XYZ Post Purchase Order")
    {
        Caption = 'Post Purchase Order';
        Implementation = "DXP BRZIF IDocument Processing" = "XYZ Post Purch. Order Proc.";
    }
}
```

The enum value is stored on both the Breeze document and the Core document. Do not change the number once the extension is in production – documents already stored would then point at a different target template.

## Step 2 – Override the default target template

The target template is read from the document class setup when the Breeze document is created. To override it conditionally, subscribe to the `OnAfterFillDocumentInfo` event of codeunit `DXP BRZ IF Document Mgt.`. It is raised while the Breeze document is being populated from the JSON – that is, before it is inserted:

```
codeunit 50100 "XYZ Breeze Doc. Subscribers"
{
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP BRZ IF Document Mgt.",
    'OnAfterFillDocumentInfo', '', false, false)]
    local procedure SetTargetTemplateOnOrderReference(var BrzIfDoc: Record "DXP BRZ IF Document"; JsonData: JsonObject)
```

```

var
    CoreTokenMgt: Codeunit "DXP Core Token Mgt.";
    JsonHelper: Codeunit "DXP Json Helper";
    OrderNo: Code[20];
begin
    // Only redirect documents that would default to a purchase invoice.
    if BrzIfDoc."Target Template" <> BrzIfDoc."Target Template"::"Purchase Document" then
        exit;

    OrderNo := CopyStr(JsonHelper.ValAsTxt(JsonData, CoreTokenMgt.GetOrderNoTok(), false),
1, MaxStrLen(OrderNo));
    if OrderNo = '' then
        exit;

    BrzIfDoc."Target Template" := BrzIfDoc."Target Template"::"XYZ Post Purchase Order";
end;
}

```

The purchase order number sits in the header JSON under the `orderNo` token. Always read it through `DXP Core Token Mgt.` – the token names are maintained centrally there.

The second argument of `ValAsTxt` is `EnsureSuccess`. Pass `false` here: the order number is an optional field, and with `true` a missing token would raise a runtime error.

## Keeping the Core document in sync

Core keeps a copy of the target template in the **Breeze Interface Target Template** field of the `DXP Document` table. When your subscriber changes the target template, update that field as well – otherwise the Core document still shows the old template. The processing from step 3 takes care of it (see `CoreDoc.Validate("DXP Breeze IF Target Template", ...)` in the sample below).

## Step 3 – Implement the processing

When the Breeze document is set to **Processed**, Breeze Interface invokes the implementation registered for its target template:

```

IDocumentProcessing := Rec."Target Template";
IDocumentProcessing.ProcessDocument(Rec, JsonHelper.JObjectFromBlob(Rec.RecordId,
Rec.FieldNo(JSON)));

```

Your codeunit implements `DXP BRZIF IDocument Processing` for this. It is responsible for:

- **Validating** the document – on failure, set the Breeze document to status **Error** and write the messages into the JSON.
- **Handing over** to Core through `DXP Document Mgt.UpdateDocument` when the document is valid.
- **Deleting** the Breeze document after a successful handover. The Core document remains as the record of the transaction.

```
codeunit 50101 "XYZ Post Purch. Order Proc." implements "DXP BRZIF IDocument Processing"
{
    procedure ProcessDocument(var BREEZEDoc: Record "DXP BRZ IF Document"; JObject:
    JsonObject)
    var
        CoreDoc: Record "DXP Document";
        CoreDocMgt: Codeunit "DXP Document Mgt.";
        ValidDoc: Boolean;
    begin
        ValidDoc := CheckDocument(JObject);

        if not ValidDoc then begin
            BREEZEDoc.Validate(Status, "DXP Breeze IF Status"::Error);
            BREEZEDoc.Modify(false);
            exit;
        end;

        CoreDocMgt.UpdateDocument(
            BREEZEDoc."Core Document No.",
            "DXP Document Status"::Transferred,
            "DXP Target Document Process"::"XYZ Post Purchase Order",
            JObject);

        CoreDoc.Get(BREEZEDoc."Core Document No.");
        CoreDoc.Validate("DXP Breeze IF Target Template", BREEZEDoc."Target Template");
        CoreDoc.Modify(true);

        BREEZEDoc.Delete(true);
    end;
}
```

Use the shipped codeunit `DXP BRZ IF PInv/Crdt Memo Proc` as the model for your validation. Use `DXP Plausibility Check Mgt.` for the checks themselves, and `DXP BRZ IF Target Templ. Mgt.` to read your template's setup – that way the **Disable Amounts Check** and **Disable Posting Date Check** switches apply to your template too.

## Step 4 – Create the target document in Core

The target template in Breeze decides *which validation* runs. *Which target document* is created is decided by a second enum in Core: `DXP Target Document Process`, also `Extensible = true`, with the interface `DXP IDocument Processing`.

The two enums are separate and both need extending. The Breeze target template selects the validation; the Core target document process selects the document creation. The value is passed to `UpdateDocument` by your processing from step 3.

```
enumextension 50101 "XYZ Target Doc. Process Ext." extends "DXP Target Document Process"
{
    value(50100; "XYZ Post Purchase Order")
    {
        Caption = 'Post Purchase Order';
        Implementation = "DXP IDocument Processing" = "XYZ Post P. Order Creation";
    }
}
```

The `DXP IDocument Processing` interface has a single method returning the `RecordId` of the created document. Core stores it in the **Linked-to Record Id** field of the Core document:

```
procedure ProcessStandardDocument(JObject: JsonObject): RecordId
```

Your implementation reads the order number from the header, retrieves the purchase order and posts it:

```
codeunit 50102 "XYZ Post P. Order Creation" implements "DXP IDocument Processing"
{
    procedure ProcessStandardDocument(JObject: JsonObject): RecordId
    var
        PurchaseHeader: Record "Purchase Header";
        PurchInvHeader: Record "Purch. Inv. Header";
```

```

CoreTokenMgt: Codeunit "DXP Core Token Mgt.";
JsonHelper: Codeunit "DXP Json Helper";
TargetDocumentMgt: Codeunit "DXP Target Document Mgt.";
PurchPost: Codeunit "Purch.-Post";
DocRecRef: RecordRef;
OrderNo: Code[20];
VendorNo: Code[20];

begin
    OrderNo := CopyStr(JsonHelper.ValAsTxt(JObject, CoreTokenMgt.GetOrderNoTok(), true),
1, MaxStrLen(OrderNo));
    VendorNo := CopyStr(JsonHelper.ValAsTxt(JObject, CoreTokenMgt.GetVendorNoTok(), true),
1, MaxStrLen(VendorNo));

    PurchaseHeader.SetRange("Document Type", PurchaseHeader."Document Type"::Order);
    PurchaseHeader.SetRange("No.", OrderNo);
    PurchaseHeader.SetRange("Buy-from Vendor No.", VendorNo);
    PurchaseHeader.FindFirst();

    // Transfer the invoice data from the JSON onto the purchase order
    // (document date, posting date, vendor invoice no., quantities to invoice)
    UpdatePurchaseHeaderFromJson(JObject, PurchaseHeader);

    PurchPost.SetSuppressCommit(true);
    PurchPost.Run(PurchaseHeader);

    DocRecRef.Get(PurchaseHeader.RecordId());
    TargetDocumentMgt.SetProcessedCompletely(DocRecRef);

    PurchInvHeader.SetRange("Order No.", OrderNo);
    if PurchInvHeader.FindLast() then
        exit(PurchInvHeader.RecordId());

    exit(PurchaseHeader.RecordId());
end;
}

```

Never return an empty `RecordId`. Core checks the return value; if it is empty, processing stops with an error.

# Marking the document as completely processed

Core extends the **Purchase Header** table with a **Completely Processed** field. Set it through `DXP Target Document Mgt.SetProcessedCompletely` once the document has been fully handled. The shipped implementations do the same; the field drives the display and downstream processing.

## What happens when the order is posted

Posting creates a new document – the posted purchase invoice – while the purchase order is deleted or kept depending on the remaining quantities. Return the *posted* document, as shown above. If **Linked-to Record Id** points at a deleted purchase order, the drill-down from the Core document leads nowhere.

## Target template setup

Every target template has a record in the `DXP BRZ IF Target Templ. Setup` table, shown on the **Target Template Setup** page. It is created automatically on first access. These switches are available:

- **Disable Amounts Check** – suppresses the reconciliation of header and line amounts.
- **Disable Posting Date Check** – suppresses the posting date validation.
- **Ignore Vendor Blocked (Payment)** – allows documents for blocked vendors.
- **Do not release** – suppresses the automatic release of the target document.

If your target template needs defaults other than the empty standard values, set them on creation. Codeunit `DXP BRZ IF Target Templ. Mgt.` does this for the shipped templates; for your own, subscribe to the table's `OnInsert` trigger.

{add screenshot of Target Template Setup here}

## Checklist

| Step | Object                                                                       | Purpose                                            |
|------|------------------------------------------------------------------------------|----------------------------------------------------|
| 1    | <code>enumextension</code> on <code>DXP Breeze IF Target Template</code>     | Register your target template                      |
| 2    | <code>Subscriber</code> on <code>OnAfterFillDocumentInfo</code>              | Override the default target template conditionally |
| 3    | <code>codeunit</code> implements <code>DXP BRZIF IDocument Processing</code> | Validate and hand over to Core                     |

| Step | Object                                                                 | Purpose                                                         |
|------|------------------------------------------------------------------------|-----------------------------------------------------------------|
| 4    | <code>enumextension</code> on <code>DXP Target Document Process</code> | Register your target document process                           |
| 5    | <code>codeunit</code> implements <code>DXP IDocument Processing</code> | Create the target document – here: post the purchase order      |
| 6    | <b>Document Class Setup</b>                                            | Confirm the next process step is set to <b>Breeze Interface</b> |

# Troubleshooting

| Symptom                                                               | Cause                                                                                                                                                                                                                 |
|-----------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The document keeps the old target template.                           | The subscriber from step 2 is not taking effect. Check that the document class really points at the <b>Purchase Document</b> target template and that the <code>orderNo</code> token is populated in the header JSON. |
| Setting the document to <b>Processed</b> runs the default processing. | The <code>Implementation</code> reference is missing from the enum value. Without it the enum's <code>DefaultImplementation</code> applies, which is <code>DXP BRZ Default Processing</code> .                        |
| Error <i>Document creation failed. RecordId is empty.</i>             | Your <code>ProcessStandardDocument</code> implementation returned no <code>RecordId</code> .                                                                                                                          |
| The drill-down from the Core document opens nothing.                  | <b>Linked-to Record Id</b> points at a deleted document – return the posted document, not the purchase order.                                                                                                         |
| The document sits in <b>Error</b> with no message shown.              | Your processing sets the status but writes no validation messages into the JSON. Use <code>DXP Document Transfer Mgt.AddErrorsToJson</code> .                                                                         |

# Related topics

- **Target Template Setup** – the switches per target template
- **Document Class Setup** – mapping document class, next process step and target template
- **Breeze Interface Documents** – the document list and its statuses
- **Guide to implementing customized document creation** – the alternative approach through `OnDocumentCreation` for custom document tables
- **Breeze Interface API** – returning the document from the external workflow

Revision #2

Created 2026-09-08 11:56:24 UTC by Bernd Feddersen

Updated 2026-09-08 12:07:13 UTC by Bernd Feddersen