

Eigene Zielvorlage entwickeln

Diese Seite richtet sich an Entwickler und Partner, die Breeze Interface um eine **eigene Zielvorlage** (Target Template) erweitern wollen. Sie beschreibt am durchgehenden Beispiel „Rechnung mit Bestellbezug“, wie ein Beleg statt der voreingestellten Zielvorlage eine eigene erhält und welche Verarbeitung beim Setzen des Status **Processed** ausgeführt wird.

Das Beispiel: Eine Rechnung kommt über Squeeze herein und wird wie gewohnt validiert. Ist im Belegkopf eine Bestellnummer eingetragen, soll *nicht* wie im Standard eine Einkaufsrechnung erstellt werden. Stattdessen soll die im Kopf hinterlegte Bestellung gebucht werden.

Welcher Weg ist der richtige?

Für eine eigene Belegerstellung gibt es zwei Wege. Sie schließen einander aus – wählen Sie vor Beginn den passenden:

	Eigene Zielvorlage (diese Seite)	Ereignis OnDocumentCreation
Einrichtung Zielvorlage	Ihre eigene Zielvorlage	Keine
Plausibilitätsprüfung durch Breeze	ja, mit den Schaltern der Zielvorlagen-Einrichtung	nein
Mehrere Belegarten parallel	ja, eine Zielvorlage je Fall	nur eine Implementierung für alle Belege
Geeignet für	Standard-Einkaufsbelege abweichend behandeln – etwa eine Bestellung buchen statt eine Rechnung anzulegen	Belege in vollständig eigene Tabellen schreiben

Wollen Sie in eigene Tabellen schreiben und benötigen die Prüfungen von Breeze nicht, folgen Sie stattdessen der Seite **Leitfaden zur Implementierung einer individuellen Belegerstellung**. Für das hier beschriebene Beispiel ist dieser Weg nicht geeignet: Er setzt die Zielvorlage **Keine** voraus und umgeht damit die Plausibilitätsprüfung, die eine Rechnung mit Bestellbezug gerade braucht.

Der Standardablauf

Ein Beleg durchläuft von Squeeze bis zum Zielbeleg in Business Central diese Stationen:

1. Squeeze übergibt den Beleg an Core. Core legt einen Datensatz in der Tabelle `DXP Document` an.
2. Core ermittelt über die **Dokumentenklasse** (`DXP Document Class`) den nächsten Prozessschritt. Steht dieser auf **Breeze Interface**, wird ein Breeze-Beleg (`DXP BRZ IF Document`) erzeugt.
3. Beim Anlegen erhält der Breeze-Beleg seine **Zielvorlage** aus der Einrichtung der Dokumentenklasse – Feld **Zielvorlage** auf der Seite **Dokumentenklassen-Einrichtung**.
4. Der Beleg wird im externen Workflow bearbeitet und kommt mit dem Status **Processed** zurück.
5. Breeze Interface löst daraufhin die zur Zielvorlage hinterlegte Implementierung aus. Diese prüft den Beleg auf Plausibilität und übergibt ihn an Core, das den Zielbeleg erzeugt.

Technisch ist die Zielvorlage ein Enum, das ein Interface implementiert:

- `DXP Breeze IF Target Template` – das Enum der Zielvorlagen, `Extensible = true`.
- `DXP BRZIF IDocument Processing` – das Interface, das jede Zielvorlage implementiert. Es besitzt genau eine Methode:

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

Im Auslieferungszustand sind diese Zielvorlagen enthalten:

Wert	Zielvorlage	Implementierung
0	Keine (None)	<code>DXP BRZ Default Processing</code>
1	Einkaufsbeleg (Purchase Document)	<code>DXP BRZ IF PInv/Crdt Memo Proc</code>
2	Einkaufsbestellung (Purchase Order)	<code>DXP BRZ IF Purch. Order Proc.</code>
3	Auftragsbestätigung (Order Confirmation)	<code>DXP BRZ IF Order Conf. Proc.</code>
4	Lieferschein (Delivery Note)	<code>DXP BRZ IF Del. Note Proc.</code>

{add screenshot of Dokumentenklassen-Einrichtung here}

Voraussetzungen

- Eine eigene Extension mit Abhängigkeit auf **DEXPRO Breeze Interface** (App-Id `7c431886-108e-44c3-9328-30e2b93edfa6`) und – über `propagateDependencies` mitgeliefert – auf **DEXPRO Core**.
- Ein eigener Objekt-ID-Bereich. Verwenden Sie **nicht** den Bereich von Breeze Interface (70954950–70954969) oder Core (70954575–70954624).
- Ein eigener Objektnamen-Affix, den Ihre `AppSourceCop.json` vorschreibt.

Verwenden Sie für Ihre Objekte niemals das Affix `DXP`. Es ist für die DEXPRO-Apps reserviert. In den Codebeispielen dieser Seite steht `XYZ` stellvertretend für Ihr eigenes Affix.

Schritt 1 – Zielvorlage als Enum-Erweiterung anlegen

Das Enum `DXP Breeze IF Target Template` ist erweiterbar. Legen Sie einen neuen Wert an und verweisen Sie mit `Implementation` auf Ihre eigene Codeunit:

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

Der Enum-Wert wird auf dem Breeze-Beleg und auf dem Core-Beleg gespeichert. Ändern Sie die Nummer nach dem ersten Produktiveinsatz nicht mehr – bereits gespeicherte Belege würden sonst auf eine andere Zielvorlage zeigen.

Schritt 2 – Die Standardzielvorlage überschreiben

Die Zielvorlage wird beim Anlegen des Breeze-Belegs aus der Dokumentenklassen-Einrichtung gelesen. Um sie fallabhängig zu übersteuern, abonnieren Sie das Ereignis `OnAfterFillDocumentInfo` der Codeunit `DXP BRZ IF Document Mgt.`. Es wird gerufen, während der Breeze-Beleg aus dem JSON gefüllt wird – also bevor er eingefügt ist:

```
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
    // Nur Belege umleiten, die im Standard zur Einkaufsrechnung fuehren.
    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;
}

```

Die Bestellnummer steht im Belegkopf-JSON unter dem Token `orderNo`. Lesen Sie sie ausschließlich über `DXP Core Token Mgt.` aus – die Token-Namen sind dort zentral gepflegt.

Das zweite Argument von `ValAsTxt` ist `EnsureSuccess`. Übergeben Sie hier `false`: Die Bestellnummer ist ein optionales Feld, und bei `true` würde ein fehlendes Token einen Laufzeitfehler auslösen.

Die Zielvorlage im Core-Beleg nachziehen

Core führt im Feld **Breeze Interface Zielvorlage** der Tabelle `DXP Document` eine Kopie der Zielvorlage mit. Setzt Ihr Abonent die Zielvorlage um, aktualisieren Sie dieses Feld mit – sonst zeigt der Core-Beleg weiter die alte Vorlage an. Ihre eigene Verarbeitung aus Schritt 3 erledigt das (siehe `CoreDoc.Validate("DXP Breeze IF Target Template", ...)` im Beispiel unten).

Schritt 3 – Die Verarbeitung implementieren

Wird der Breeze-Beleg auf **Processed** gesetzt, ruft Breeze Interface die zur Zielvorlage hinterlegte Implementierung auf:

```

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

```

Ihre Codeunit implementiert dafür `DXP BRZIF IDocument Processing`. Sie ist verantwortlich für:

- **Prüfen** des Belegs – im Fehlerfall den Breeze-Beleg auf Status **Error** setzen und die Meldungen in das JSON schreiben.
- **Übergeben** an Core über `DXP Document Mgt.UpdateDocument`, wenn der Beleg gültig ist.
- **Löschen** des Breeze-Belegs nach erfolgreicher Übergabe. Der Core-Beleg bleibt als Nachweis bestehen.

```

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;
}

```

}

Als Vorlage für Plausibilitätsprüfungen dient die ausgelieferte Codeunit `DXP BRZ IF PInv/Crdt Memo Proc.`. Nutzen Sie `DXP Plausibility Check Mgt.` für die Prüfungen und `DXP BRZ IF Target Templ. Mgt.`, um die Einrichtung Ihrer Zielvorlage zu lesen – damit greifen die Schalter **Betragsprüfung deaktivieren** und **Buchungsdatum nicht prüfen** auch für Ihre Vorlage.

Schritt 4 – Den Zielbeleg in Core erzeugen

Die Zielvorlage in Breeze bestimmt *welche Prüfung* läuft. *Welcher Zielbeleg* entsteht, entscheidet ein zweites Enum in Core: `DXP Target Document Process`, ebenfalls `Extensible = true`, mit dem Interface `DXP IDocument Processing`.

Die beiden Enums sind getrennt und müssen beide erweitert werden. Die Breeze-Zielvorlage wählt die Prüfung, der Core-Zielbelegprozess die Belegerstellung. Der Wert wird von Ihrer Verarbeitung aus Schritt 3 an `UpdateDocument` übergeben.

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

Das Interface `DXP IDocument Processing` hat eine Methode, die die `RecordId` des erzeugten Belegs zurückgibt. Core speichert sie im Feld **Verknüpfte Datensatz-ID** des Core-Belegs:

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

Ihre Implementierung liest die Bestellnummer aus dem Kopf, holt die Bestellung und bucht sie:

```
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();

    // Rechnungsdaten aus dem JSON in die Bestellung uebernehmen
    // (Belegdatum, Buchungsdatum, Externe Belegnummer, zu berechnende Mengen)
    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;
}

```

Geben Sie niemals eine leere `RecordId` zurück. Core prüft den Rückgabewert; bleibt er leer, bricht die Verarbeitung mit einer Fehlermeldung ab.

Vollständig verarbeitet kennzeichnen

Core erweitert die Tabelle **Einkaufskopf** um das Feld **Vollständig verarbeitet**. Setzen Sie es über `DXP Target Document Mgt.SetProcessedCompletely`, sobald der Beleg fertig verarbeitet ist. Die ausgelieferten Implementierungen tun das ebenfalls; das Feld steuert die Anzeige und die nachgelagerte Verarbeitung.

Wenn die Bestellung gebucht wird

Beim Buchen entsteht ein neuer Beleg – die gebuchte Einkaufsrechnung –, während die Bestellung je nach Restmenge gelöscht wird. Verweisen Sie deshalb im Rückgabewert auf den *gebuchten* Beleg, wie im Beispiel oben. Zeigt **Verknüpfte Datensatz-ID** auf eine gelöschte Bestellung, läuft der Absprung vom Core-Beleg ins Leere.

Die Einrichtung der Zielvorlage

Zu jeder Zielvorlage gehört ein Datensatz in der Tabelle `DXP BRZ IF Target Templ. Setup`, sichtbar auf der Seite **Zielvorlagen-Einrichtung**. Er wird beim ersten Zugriff automatisch angelegt. Diese Schalter stehen zur Verfügung:

- **Betragsprüfung deaktivieren** – unterdrückt den Abgleich von Kopf- und Zeilenbeträgen.
- **Buchungsdatum nicht prüfen** – unterdrückt die Prüfung des Buchungsdatums.
- **Gesperzten Kreditoren (Zahlung) ignorieren** – lässt Belege gesperrter Kreditoren zu.
- **Nicht freigeben** – unterdrückt die automatische Freigabe des Zielbelegs.

Sollen für Ihre Zielvorlage andere Vorgabewerte gelten als die leeren Standardwerte, setzen Sie sie beim Anlegen. Die Codeunit `DXP BRZ IF Target Templ. Mgt.` tut das für die ausgelieferten Vorlagen; für eigene Vorlagen abonnieren Sie den `OnInsert`-Trigger der Tabelle.

{add screenshot of Zielvorlagen-Einrichtung here}

Checkliste

Schritt	Objekt	Zweck
---------	--------	-------

1	<code>enumextension</code> auf <code>DXP Breeze IF Target Template</code>	Die eigene Zielvorlage bekannt machen
2	Abonnent auf <code>OnAfterFillDocumentInfo</code>	Die Standardzielvorlage fallabhängig überschreiben
3	<code>codeunit</code> implements <code>DXP BRZIF IDocument Processing</code>	Prüfen und an Core übergeben
4	<code>enumextension</code> auf <code>DXP Target Document Process</code>	Den eigenen Zielbelegprozess bekannt machen
5	<code>codeunit</code> implements <code>DXP IDocument Processing</code>	Den Zielbeleg erzeugen – hier: die Bestellung buchen
6	Dokumentenklassen-Einrichtung	Prüfen, dass der nächste Prozessschritt auf Breeze Interface steht

Fehlersuche

Beobachtung	Ursache
Der Beleg behält die alte Zielvorlage.	Der Abonnent aus Schritt 2 greift nicht. Prüfen Sie, ob die Dokumentenklasse tatsächlich auf die Zielvorlage Einkaufsbeleg zeigt und ob das Token <code>orderNo</code> im Kopf-JSON gefüllt ist.
Beim Setzen auf Processed läuft die Standardverarbeitung.	Der <code>Implementation</code> -Verweis im Enum fehlt. Ohne ihn greift die <code>DefaultImplementation</code> des Enums, also <code>DXP BRZ Default Processing</code> .
Meldung <i>Document creation failed. RecordId is empty.</i>	Ihre <code>ProcessStandardDocument</code> -Implementierung hat keine <code>RecordId</code> zurückgegeben.
Der Absprung vom Core-Beleg öffnet nichts.	Verknüpfte Datensatz-ID zeigt auf einen gelöschten Beleg – geben Sie den gebuchten Beleg zurück, nicht die Bestellung.
Der Beleg steht auf Error , ohne dass eine Meldung erscheint.	Ihre Verarbeitung setzt den Status, schreibt aber keine Prüfmeldungen in das JSON. Nutzen Sie <code>DXP Document Transfer Mgt.AddErrorsToJson</code> .