

# Extension Entwicklung

In diesem Kapitel werden wichtige Integration Events und Beispiele aufgeführt, die bei der Entwicklung einer Extension für SQUEEZE for BC helfen. Wichtig: Anpassungen dürfen nur durch entsprechend geschulte Berater/Entwickler durchgeführt werden. Diese befinden sich außerdem immer außerhalb des Standard-Supports.

- [Hinzufügen eines Feldes in der Validierung \(bis Squeeze BC APP Version 1.\\*\)](#)
- [Abweichende Verwendung der SQUEEZE Anhänge](#)
- [Hinzufügen eines Feldes in der Validierung \(ab Version 2.0\)](#)
- [Hinzufügen eines Feldes in der Validierung \(ab Version 2.10\)](#)

# Hinzufügen eines Feldes in der Validierung (bis Squeeze BC APP Version 1.\*)

Im Folgenden wird anhand einer Beispielerextension dargestellt, wie man als Entwickler Felder in der Validierung hinzufügen kann.

## Benötigte Integration Events inkl. beispielhafter Prozeduren:

```
codeunit 50100 EventSubs
{
    //
    //Any header field that you want to transfer to the resulting document, has to be added
to the source JSON Object
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ Document Mgt.",
' OnBeforeAddLineToDocumentJObj', '', false, false)]
    local procedure SQZDocMgtOnBeforeAddLineToDocumentJObj(var DocumentJObj: JsonObject;
DocHeader: Record "DXP SQZ Document Header")
    begin
        AddCustomHeaderFieldsToJson(DocumentJObj, DocHeader);
    end;

    //
    // Any line field that you want to transfer to the resulting document, has to be added to
the source JSON Object
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ Document Mgt.",
' OnBeforeAddLineJObjToLineJArray', '', false, false)]
    local procedure SQZDocMgtOnBeforeAddLineJObjToLineJArray(var LineJObj: JsonObject;
DocLine: Record "DXP SQZ Document Line")
    begin
```

```

        AddCustomLineFieldsToJson(LineJObj, DocLine);
    end;

    //
    // [If you want to perform plausibility checks on the newly added header field, this is
the place]
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ Document Mgt.",
' OnAfterDoHeaderPlausibilityChecks', '', false, false)]

    local procedure SQZDocMgtOnBeforeDoHeaderPlausibilityChecks(DocHeader: Record "DXP SQZ
Document Header"; var PlausibilityCheck: Codeunit "DXP Plausiblity Check Mgt.")
    begin
        CheckCustomer(PlausibilityCheck, DocHeader."Customer No.");
    end;

    //
    // [If you want to perform plausibility checks on the newly added line field, this is the
place]
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ Document Mgt.",
' OnAfterDoLinePlausibilityChecks', '', false, false)]

    local procedure SQZDocMgtOnAfterDoLinePlausibilityChecks(DocHeader: Record "DXP SQZ
Document Header"; DocLine: Record "DXP SQZ Document Line"; var PlausibilityCheck: Codeunit
"DXP Plausiblity Check Mgt.")
    begin
        CheckCustomer(PlausibilityCheck, DocLine."Customer No.");
    end;

    //
    // The value of the previously extended JSON Object now has to saved to the corresponding
field in the SQUEEZE Document Line
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ Document Mgt.",
' OnBeforeModifySQUEEZEDocumentHeader', '', false, false)]

    local procedure SqzDocMgtOnBeforeModifySQUEEZEDocumentHeader(var DocHeader: Record "DXP
SQZ Document Header"; RawJson: JsonObject)
    var
        JSONHelper: Codeunit "DXP Json Helper";
        SQZApiTokenMgt: Codeunit "DXP SQZ API Token Mgt.";
        TokenMgt: Codeunit TokenMgt;

```

```

begin
    DocHeader."Customer No." := COPYSTR(JsonHelper.ValAsTxt(RawJson,
SQZApiTokenMgt.GetHeaderValueByNameTok(TokenMgt.GetCustomerNoTok()), false), 1,
MaxStrLen(DocHeader."Customer No."));
end;

//
// The value of the previously extended JSON Object now has to saved to the corresponding
field in the SQUEEZE Document Header
//
[EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ Document Mgt.",
'OnBeforeTransferRecognizedValuesOnAfterAssignRecognizedValues', '', false, false)]
local procedure
SQZDocMgtOnBeforeTransferRecognizedValuesOnAfterAssignRecognizedValues(RowJTok: JsonToken;
var DocLine: Record "DXP SQZ Document Line")
var
    JSONHelper: Codeunit "DXP Json Helper";
    SQZApiTokenMgt: Codeunit "DXP SQZ API Token Mgt.";
    TokenMgt: Codeunit TokenMgt;
begin
    DocLine."Customer No." := CopyStr(JsonHelper.ValAsTxt(RowJTok.AsObject(),
SQZApiTokenMgt.GetCellValueTok(TokenMgt.GetCustomerNoTok()), false), 1,
MaxStrLen(DocLine."Customer No."));
end;

//
// The newly added fields now have to be added to the field mapping
// Otherwise the recognized values will be saved as meta data
// This step is necessary to make sure that the fields can be highlighted in the SQUEEZE
Viewer
//
[EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ API Mgt.",
'OnAfterInitMapping', '', false, false)]
local procedure SQZApiMgtOnAfterInitMapping(var HeaderMapping: Dictionary of [Text,
Integer]; var LineMapping: Dictionary of [Text, Integer]; DocClass: Enum "DXP Document Class")
var
    SQZDocHeader: Record "DXP SQZ Document Header";
    SQZDocLine: Record "DXP SQZ Document Line";
    TokenMgt: Codeunit TokenMgt;
begin

```

```

// The standard (without installed extensions) supports one Document Class
case DocClass of
    DocClass::"DXP Invoice / Credit Memo":
        begin
            HeaderMapping.Add( TokenMgt. GetCustomerNoTok(),
SQZDocHeader.FieldNo("Customer No. "));
            LineMapping.Add( TokenMgt. GetCustomerNoTok(), SQZDocLine.FieldNo("Customer
No. "));
        end;
    end;
end;

//
// The value in the JSON Object now has to be transferred to the Purchase Header
//
[EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP Document Transfer Mgt.",
'OnAfterCreatePurchaseHeader', '', false, false)]
    local procedure DocTransferMgtOnAfterCreatePurchaseHeader( var JObject: JsonObject;
PurchaseHeader: Record "Purchase Header")
    begin
        TransferCustomHeaderFieldFromJsonToPurchaseHeader( JObject, PurchaseHeader);
    end;

//
// The value in the JSON Object now has to be transferred to the Purchase Line
//
[EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP Document Transfer Mgt.",
'OnBeforeInsertPurchaseLine', '', false, false)]
    local procedure DocTransferMgtOnAfterInsertPurchaseLineOnBeforeAddDimensions( var
JsonObjectLine: JsonObject; var PurchaseLine: Record "Purchase Line")
    begin
        TransferCustomHeaderFieldFromJsonToPurchaseLine( JsonObjectLine, PurchaseLine);
    end;

    local procedure TransferCustomHeaderFieldFromJsonToPurchaseHeader( var JObject:
JsonObject; PurchaseHeader: Record "Purchase Header")
    var
        JsonHelper: Codeunit "DXP Json Helper";
        TokenMgt: Codeunit TokenMgt;
    begin

```

```

        PurchaseHeader.Validate("DXP Customer No.", JsonHelper.ValAsTxt(JObject,
TokenMgt.GetCustomerNoTok(), false));

        PurchaseHeader.Modify();

    end;

    local procedure TransferCustomHeaderFieldFromJsonToPurchaseLine( var JObjectLine:
JsonObject; var PurchaseLine: Record "Purchase Line")
    var
        JsonHelper: Codeunit "DXP Json Helper";
        TokenMgt: Codeunit TokenMgt;
    begin
        PurchaseLine.Validate("DXP Customer No.", JsonHelper.ValAsTxt(JObjectLine,
TokenMgt.GetCustomerNoTok(), false));
    end;

    local procedure AddCustomHeaderFieldsToJson( var DocumentJObj: JsonObject;
SQZDocumentHeader: Record "DXP SQZ Document Header")
    var
        TokenMgt: Codeunit TokenMgt;
    begin
        DocumentJObj.Add( TokenMgt.GetCustomerNoTok(), SQZDocumentHeader."Customer No.");
    end;

    local procedure AddCustomLineFieldsToJson( var LineJObj: JsonObject; SQZDocumentLine:
Record "DXP SQZ Document Line")
    var
        TokenMgt: Codeunit TokenMgt;
    begin
        LineJObj.Add( TokenMgt.GetCustomerNoTok(), SQZDocumentLine."Customer No.");
    end;

    [ TryFunction]
    local procedure CustomerExists( CustomerNo: Code[ 20] )
    var
        Customer: Record Customer;
    begin
        Customer.Get( CustomerNo );
    end;

    local procedure CheckCustomer( var PlausibilityCheck: Codeunit "DXP Plausiblity Check

```

```

Mgt."; CustomerNo: Code[20]): Boolean
begin
    if CustomerNo = '' then
        exit(false);

    if not CustomerExists(CustomerNo) then begin
        PlausibilityCheck.AddPlausibilityCheckEntry(GetLastErrorText(), Page: "Customer
List");
        exit(false);
    end;

    exit(true);
end;
}

```

## Table Extensions:

### SQUEEZE 4 BC

```

tableextension 50100 "SQZ Doc. Header Ext." extends "DXP SQZ Document Header"
{
    fields
    {
        field(50100; "Customer No."; Code[20])
        {
            TableRelation = Customer;
            ValidateTableRelation = false;
            Caption = 'Customer No.';
        }
    }
}

```

```

tableextension 50101 "SQZ Doc. Line Ext." extends "DXP SQZ Document Line"
{
    fields
    {
        field(50100; "Customer No."; Code[20])
        {
            TableRelation = Customer;
            ValidateTableRelation = false;
        }
    }
}

```

```

        Caption = 'Customer No.';
    }
}
}

```

## Zielbeleg (hier: Einkaufsbeleg)

```

tableextension 50102 "Purchase Header Ext." extends "Purchase Header"
{
    fields
    {
        field(50100; "DXP Customer No."; Code[20])
        {
            DataClassification = CustomerContent;
            TableRelation = Customer;
            Caption = 'DEXPRO Customer No.';
        }
    }
}

```

## Page Extensions:

### SQUEEZE 4 BC

```

pageextension 50100 "SQZ Document Ext." extends "DXP SQZ Document"
{
    layout
    {
        addafter(BuyFromVendorInternal)
        {
            field("Customer No. Internal"; Rec."Customer No.")
            {
                ApplicationArea = All;
                ToolTip = 'Specifies the value of the Customer No. field.';
                trigger OnAssistEdit()
                begin
                    MarkField(Rec.FieldNo("Customer No."), Rec."Customer No.",
Rec);
                end
            }
        }
    }
}

```



```

        end;

        trigger OnValidate()
        begin
            CheckPlausibility();
        end;
    }
}
addafter(BuyFromVendor)
{
    field("Customer No."; Rec."Customer No.")
    {
        ApplicationArea = All;
        ToolTip = 'Specifies the value of the Customer No. field.';
        trigger OnAssistEdit()
        var
            ApiMgt: Codeunit "DXP SQZ API Mgt.";
            ViewerResident: Codeunit "DXP SQUEEZE Viewer Resident";
        begin
            MarkField(Rec.FieldNo("Customer No."), Rec."Customer No.",
Rec);
        end;

        trigger OnValidate()
        begin
            CheckPlausibility();
        end;
    }
}
}

local procedure MarkField(AppFldNo: Integer; FieldVal: Variant; DocHeader: Record "DXP
SQZ Document Header")
var
    ApiMgt: Codeunit "DXP SQZ API Mgt.";
    ViewerResident: Codeunit "DXP SQUEEZE Viewer Resident";
begin
    ViewerResident := GetViewerResident();
    ApiMgt.MarkField(AppFldNo, FieldVal, ViewerResident, DocHeader);
end;

```

```
}
```

```
pageextension 50101 "SQZ Document Sf. Ext." extends "DXP SQZ Document Subform"
{
    layout
    {
        addafter("No.")
        {
            field("Customer No."; Rec."Customer No.")
            {
                ApplicationArea = All;
                ToolTip = 'Specifies the value of the Customer No. field.';
            }
        }
    }
}
```

## Zielbeleg (hier: Einkaufsbeleg)

```
pageextension 50102 "Purchase Invoice Ext." extends "Purchase Invoice"
{
    layout
    {
        addafter("Buy-from Vendor No.")
        {
            field("DXP Customer No."; Rec."DXP Customer No.")
            {
                ToolTip = 'Specifies the value of the DEXPRO Customer No.
field.';
                ApplicationArea = all;
            }
        }
    }
}
```

```
pageextension 50103 "Purchase Inv. Sf Ext." extends "Purch. Invoice Subform"
{
    layout
    {
        addafter("No.")
```

```

        {
            field("DXP Customer No."; Rec."DXP Customer No.")
            {
                ToolTip = 'Specifies the value of the DEXPRO Customer No.
field.';

                ApplicationArea = all;
            }
        }
    }
}

```

## Optionale Hilfsobjekte:

```

codeunit 50101 TokenMgt
{
    procedure GetCustomerNoTok(): Text
    begin
        exit( CustomerNoTok);
    end;

    var
        CustomerNoTok: Label 'customerNo', Locked = true;
}

```

# Abweichende Verwendung der SQUEEZE Anhänge

Im Folgenden wird kurz erläutert, wie die durch SQUEEZE for BC heruntergeladenen Anhänge für ein Szenario verwendet werden können, das von der vorgesehenen Verwendung abweicht.

Nach der Erstellung des Einkaufsbeleges aus dem validierten SQUEEZE Beleg haben Sie die Möglichkeit, die Anhänge abzugreifen (sofern diese im Vorfeld heruntergeladen wurden).

```
// Set IsHandled to true and implement your code
// Do not forget to deactivate "Transfer attachments to target document" in the Document
Class Setup
[ EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP Document Mgt.",
' OnBeforeTransferCoreAttachmentsToStandardDocument', '', false, false)]
    local procedure DXPDocMgtOnBeforeTransferCoreAttachmentsToStandardDocument(Document:
Record "DXP Document"; var IsHandled: Boolean)
    begin
        IsHandled := true;
        HandleCoreAttachmentsAccordingToYourNeeds(Document, IsHandled);
    end;

    procedure HandleCoreAttachmentsAccordingToYourNeeds(Document: Record "DXP Document"; var
IsHandled: Boolean)
    var
        CoreAttachments: Record "DXP Document Attachment";
        InStr: InStream;
    begin
        CoreAttachments.SetRange("Document No.", Document."No.");
        CoreAttachments.IsEmpty() then
            Exit;
        CoreAttachments.FindSet();
        repeat
            CoreAttachments.CalcFields(Attachment);
            CoreAttachments.Attachment.CreateInStream(InStr);
            // Here you can handle the file stream according to your needs
        [...]
```

```
until CoreAttachments.Next() = 0;  
end;
```

# Hinzufügen eines Feldes in der Validierung (ab Version 2.0)

Im Folgenden wird anhand einer Beispielerweiterung dargestellt, wie man als Entwickler Felder in der Validierung hinzufügen kann (hier: Dokumentenklasse Rechnung/Gutschrift).

## Benötigte Integration Events inkl. beispielhafter Prozeduren:

```
codeunit 50100 EventSubs
{
    //
    //Any header field that you want to transfer to the resulting document, has to be added
    to the source JSON Object
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ P. Inv/Crdt Memo Impl.",
    'OnBeforeAddLineToDocumentJObj', '', false, false)]
    local procedure SQZPInvCrdtMemoImplOnBeforeAddLineToDocumentJObj(var DocumentJObj:
    JsonObject; DocHeader: Record "DXP SQZ Document Header")
    begin
        AddCustomHeaderFieldsToJson(DocumentJObj, DocHeader);
    end;

    //
    // Any line field that you want to transfer to the resulting document, has to be added to
    the source JSON Object
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ P. Inv/Crdt Memo Impl.",
    'OnBeforeAddLineJObjToLineJArray', '', false, false)]
    local procedure SQZPInvCrdtMemoImplOnBeforeAddLineJObjToLineJArray(var LineJObj:
    JsonObject; DocLine: Record "DXP SQZ Document Line")
    begin
```

```

        AddCustomLineFieldsToJson(LineJObj, DocLine);
    end;

    //
    // [If you want to perform plausibility checks on the newly added header field, this is
the place]
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ P. Inv/Crdt Memo Impl.",
' OnAfterDoHeaderPlausibilityChecks', '', false, false)]
    local procedure SQZInvCrdtMemoImplOnBeforeDoHeaderPlausibilityChecks(DocHeader: Record
"DXP SQZ Document Header"; var PlausibilityCheck: Codeunit "DXP Plausiblity Check Mgt.")
    begin
        CheckCustomer(PlausibilityCheck, DocHeader."Customer No.");
    end;

    //
    // [If you want to perform plausibility checks on the newly added line field, this is the
place]
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ P. Inv/Crdt Memo Impl.",
' OnAfterDoLinePlausibilityChecks', '', false, false)]
    local procedure SQZInvCrdtMemoImplOnAfterDoLinePlausibilityChecks(DocHeader: Record "DXP
SQZ Document Header"; DocLine: Record "DXP SQZ Document Line"; var PlausibilityCheck:
Codeunit "DXP Plausiblity Check Mgt.")
    begin
        CheckCustomer(PlausibilityCheck, DocLine."Customer No.");
    end;

    //
    // The value of the previously extended JSON Object now has to saved to the corresponding
field in the SQUEEZE Document Line
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ Document Mgt.",
' OnBeforeModifySQUEEZEDocumentHeader', '', false, false)]
    local procedure SqzDocMgtOnBeforeModifySQUEEZEDocumentHeader(var DocHeader: Record "DXP
SQZ Document Header"; RawJson: JsonObject)
    var
        JSONHelper: Codeunit "DXP Json Helper";
        SQZApiTokenMgt: Codeunit "DXP SQZ API Token Mgt.";
        TokenMgt: Codeunit TokenMgt;

```

```

begin
    DocHeader."Customer No." := COPYSTR( JsonHelper.ValAsTxt(RawJson,
SQZApiTokenMgt.GetHeaderValueByNameTok( TokenMgt.GetCustomerNoTok()), false), 1,
MaxStrLen(DocHeader."Customer No."));
end;

//
// The value of the previously extended JSON Object now has to saved to the corresponding
field in the SQUEEZE Document Header
//
[ EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ Document Mgt.",
' OnBeforeTransferRecognizedValuesOnAfterAssignRecognizedValues', '', false, false)]
    local procedure
SQZDocMgtOnBeforeTransferRecognizedValuesOnAfterAssignRecognizedValues(RowJTok: JsonToken;
var DocLine: Record "DXP SQZ Document Line")
var
    JSONHelper: Codeunit "DXP Json Helper";
    SQZApiTokenMgt: Codeunit "DXP SQZ API Token Mgt.";
    TokenMgt: Codeunit TokenMgt;
begin
    DocLine."Customer No." := CopyStr( JsonHelper.ValAsTxt(RowJTok.AsObject(),
SQZApiTokenMgt.GetCellValueTok( TokenMgt.GetCustomerNoTok()), false), 1,
MaxStrLen(DocLine."Customer No."));
end;

//
// The newly added fields now have to be added to the field mapping
// Otherwise the recognized values will be saved as meta data
// This step is also necessary to make sure that the fields can be highlighted in the
SQUEEZE Viewer
//
[ EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ API Mgt.",
' OnAfterInitMapping', '', false, false)]
    local procedure SQZApiMgtOnAfterInitMapping(var HeaderMapping: Dictionary of [Text,
Integer]; var LineMapping: Dictionary of [Text, Integer]; DocClass: Enum "DXP Document Class")
var
    SQZDocHeader: Record "DXP SQZ Document Header";
    SQZDocLine: Record "DXP SQZ Document Line";
    TokenMgt: Codeunit TokenMgt;
begin

```



```

        case DocClass of
            DocClass::"DXP Invoice / Credit Memo":
                begin
                    HeaderMapping.Add( TokenMgt. GetCustomerNoTok(),
SQZDocHeader.FieldNo("Customer No. "));
                    LineMapping.Add( TokenMgt. GetCustomerNoTok(), SQZDocLine.FieldNo("Customer
No. "));
                end;
            end;
        end;

//
// The value in the JSON Object now has to be transferred to the Purchase Header
//
[EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP Document Transfer Mgt.",
'OnAfterCreatePurchaseHeader', '', false, false)]
    local procedure DocTransferMgtOnAfterCreatePurchaseHeader( var JObject: JsonObject;
PurchaseHeader: Record "Purchase Header")
    begin
        TransferCustomHeaderFieldFromJsonToPurchaseHeader( JObject, PurchaseHeader);
    end;

//
// The value in the JSON Object now has to be transferred to the Purchase Line
//
[EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP Document Transfer Mgt.",
'OnBeforeInsertPurchaseLine', '', false, false)]
    local procedure DocTransferMgtOnAfterInsertPurchaseLineOnBeforeAddDimensions( var
JObjectLine: JsonObject; var PurchaseLine: Record "Purchase Line")
    begin
        TransferCustomHeaderFieldFromJsonToPurchaseLine( JObjectLine, PurchaseLine);
    end;

    local procedure TransferCustomHeaderFieldFromJsonToPurchaseHeader( var JObject:
JsonObject; PurchaseHeader: Record "Purchase Header")
    var
        JsonHelper: Codeunit "DXP Json Helper";
        TokenMgt: Codeunit TokenMgt;
    begin
        PurchaseHeader.Validate("DXP Customer No.", JsonHelper.ValAsTxt( JObject,

```

```

TokenMgt.GetCustomerNoTok(), false));
    PurchaseHeader.Modify();
end;

local procedure TransferCustomHeaderFieldFromJsonToPurchaseLine( var JObjectLine:
JsonObject; var PurchaseLine: Record "Purchase Line")
var
    JsonHelper: Codeunit "DXP Json Helper";
    TokenMgt: Codeunit TokenMgt;
begin
    PurchaseLine.Validate("DXP Customer No.", JsonHelper.ValAsTxt(JObjectLine,
TokenMgt.GetCustomerNoTok(), false));
end;

local procedure AddCustomHeaderFieldsToJson( var DocumentJObj: JsonObject;
SQZDocumentHeader: Record "DXP SQZ Document Header")
var
    TokenMgt: Codeunit TokenMgt;
begin
    DocumentJObj.Add( TokenMgt.GetCustomerNoTok(), SQZDocumentHeader."Customer No.");
end;

local procedure AddCustomLineFieldsToJson( var LineJObj: JsonObject; SQZDocumentLine:
Record "DXP SQZ Document Line")
var
    TokenMgt: Codeunit TokenMgt;
begin
    LineJObj.Add( TokenMgt.GetCustomerNoTok(), SQZDocumentLine."Customer No.");
end;

[ TryFunction]
local procedure CustomerExists( CustomerNo: Code[ 20] )
var
    Customer: Record Customer;
begin
    Customer.Get( CustomerNo );
end;

local procedure CheckCustomer( var PlausibilityCheck: Codeunit "DXP Plausiblity Check
Mgt."; CustomerNo: Code[ 20] ): Boolean

```

```

begin
    if CustomerNo = '' then
        exit(false);

    if not CustomerExists(CustomerNo) then begin
        PlausibilityCheck.AddPlausibilityCheckEntry(GetLastErrorText(), Page: "Customer
List");
        exit(false);
    end;

    exit(true);
end;
}

```

## Table Extensions:

### SQUEEZE 4 BC

```

tableextension 50100 "SQZ Doc. Header Ext." extends "DXP SQZ Document Header"
{
    fields
    {
        field(50100; "Customer No."; Code[20])
        {
            TableRelation = Customer;
            ValidateTableRelation = false;
            Caption = 'Customer No.';
        }
    }
}

```

```

tableextension 50101 "SQZ Doc. Line Ext." extends "DXP SQZ Document Line"
{
    fields
    {
        field(50100; "Customer No."; Code[20])
        {
            TableRelation = Customer;
            ValidateTableRelation = false;
            Caption = 'Customer No.';
        }
    }
}

```

```

    }
}
}

```

## Zielbeleg (hier: Einkaufsbeleg)

```

tableextension 50102 "Purchase Header Ext." extends "Purchase Header"
{
    fields
    {
        field(50100; "DXP Customer No."; Code[20])
        {
            DataClassification = CustomerContent;
            TableRelation = Customer;
            Caption = 'DEXPRO Customer No.';
        }
    }
}

```

## Page Extensions:

### SQUEEZE 4 BC

```

pageextension 50100 "SQZ Document Ext." extends "DXP SQZ Document v2"
{
    layout
    {
        addafter(BuyFromVendorInternal)
        {
            field("Customer No. Internal"; Rec."Customer No.")
            {
                ApplicationArea = All;
                ToolTip = 'Specifies the value of the Customer No. field.';
                trigger OnAssistEdit()
                begin
                    MarkField(Rec.FieldNo("Customer No."), Rec."Customer No.",
Rec);
                end;
            }
        }
    }
}

```

```

        trigger OnValidate()
        begin
            CheckPlausibility();
        end;
    }
}
addafter(BuyFromVendorExternal)
{
    field("Customer No."; Rec."Customer No.")
    {
        ApplicationArea = All;
        ToolTip = 'Specifies the value of the Customer No. field.';
        trigger OnAssistEdit()
        var
            ApiMgt: Codeunit "DXP SQZ API Mgt.";
            ViewerResident: Codeunit "DXP SQUEEZE Viewer Resident";
        begin
            MarkField(Rec.FieldNo("Customer No."), Rec."Customer No.",
Rec);

        end;

        trigger OnValidate()
        begin
            CheckPlausibility();
        end;
    }
}
}

```

```

local procedure MarkField(AppFldNo: Integer; FieldVal: Variant; DocHeader: Record "DXP
SQZ Document Header")
var
    ApiMgt: Codeunit "DXP SQZ API Mgt.";
    ViewerResident: Codeunit "DXP SQUEEZE Viewer Resident";
begin
    ViewerResident := GetViewerResident();
    ApiMgt.MarkField(AppFldNo, FieldVal, ViewerResident, DocHeader);
end;

```

```
}
```

```
pageextension 50101 "SQZ Document Sf. Ext." extends "DXP SQZ Document Subform"
{
    layout
    {
        addafter("No.")
        {
            field("Customer No."; Rec."Customer No.")
            {
                ApplicationArea = All;
                ToolTip = 'Specifies the value of the Customer No. field.';
            }
        }
    }
}
```

## Zielbeleg (hier: Einkaufsbeleg)

```
pageextension 50102 "Purchase Invoice Ext." extends "Purchase Invoice"
{
    layout
    {
        addafter("Buy-from Vendor No.")
        {
            field("DXP Customer No."; Rec."DXP Customer No.")
            {
                ToolTip = 'Specifies the value of the DEXPRO Customer No.
field.';
                ApplicationArea = all;
            }
        }
    }
}
```

```
pageextension 50103 "Purchase Inv. Sf Ext." extends "Purch. Invoice Subform"
{
    layout
    {
        addafter("No.")
```

```

        {
            field("DXP Customer No."; Rec."DXP Customer No.")
            {
                ToolTip = 'Specifies the value of the DEXPRO Customer No.
field.';

                ApplicationArea = all;
            }
        }
    }
}

```

## Optionale Hilfsobjekte:

```

codeunit 50101 TokenMgt
{
    procedure GetCustomerNoTok(): Text
    begin
        exit( CustomerNoTok);
    end;

    var
        CustomerNoTok: Label 'customerNo', Locked = true;
}

```

# Hinzufügen eines Feldes in der Validierung (ab Version 2.10)

Im Folgenden wird anhand einer Beispielextension dargestellt, wie man als Entwickler Felder in der Validierung hinzufügen kann (hier: Dokumentenklasse Rechnung/Gutschrift).

## Optionale Integration Events inkl. beispielhafter Prozeduren:

```
codeunit 50100 EventSubs
{
    //
    // [If you want to perform plausibility checks on the newly added header field, this is
    the place]
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ P. Inv/Crdt Memo Impl.",
'OnAfterDoHeaderPlausibilityChecks', '', false, false)]
    local procedure SQZPInvCrdtMemoImplOnBeforeDoHeaderPlausibilityChecks(DocHeader: Record
"DXP SQZ Document Header"; var PlausibilityCheck: Codeunit "DXP Plausiblity Check Mgt.")
    begin
        CheckCustomer(PlausibilityCheck, DocHeader."Customer No.");
    end;

    //
    // [If you want to perform plausibility checks on the newly added line field, this is the
    place]
    //
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ P. Inv/Crdt Memo Impl.",
'OnAfterDoLinePlausibilityChecks', '', false, false)]
    local procedure SQZPInvCrdtMemoImplOnAfterDoLinePlausibilityChecks(DocHeader: Record "DXP
SQZ Document Header"; DocLine: Record "DXP SQZ Document Line"; var PlausibilityCheck:
Codeunit "DXP Plausiblity Check Mgt.")
```



```

begin
    CheckCustomer(PlausibilityCheck, DocLine."Customer No.");
end;

[ TryFunction]
local procedure CustomerExists(CustomerNo: Code[20])
var
    Customer: Record Customer;
begin
    Customer.Get(CustomerNo);
end;

local procedure CheckCustomer(var PlausibilityCheck: Codeunit "DXP Plausiblity Check
Mgt."; CustomerNo: Code[20]): Boolean
begin
    if CustomerNo = '' then
        exit(false);

    if not CustomerExists(CustomerNo) then begin
        PlausibilityCheck.AddPlausibilityCheckEntry(GetLastErrorText(), Page: "Customer
List");
        exit(false);
    end;

    exit(true);
end;
}

```

## Table Extensions:

### SQUEEZE 4 BC

```

tableextension 50100 "SQZ Doc. Header Ext." extends "DXP SQZ Document Header"
{
    fields
    {
        field(50100; "Customer No."; Code[20])
        {
            TableRelation = Customer;
            ValidateTableRelation = false;
        }
    }
}

```

```

        Caption = 'Customer No.';
    }
}
}

```

```

tableextension 50101 "SQZ Doc. Line Ext." extends "DXP SQZ Document Line"
{
    fields
    {
        field(50100; "Customer No."; Code[20])
        {
            TableRelation = Customer;
            ValidateTableRelation = false;
            Caption = 'Customer No.';
        }
    }
}

```

## Zielbeleg (hier: Einkaufsbeleg)

```

tableextension 50102 "Purchase Header Ext." extends "Purchase Header"
{
    fields
    {
        field(50100; "DXP Customer No."; Code[20])
        {
            DataClassification = CustomerContent;
            TableRelation = Customer;
            Caption = 'DEXPRO Customer No.';
        }
    }
}

```

## Page Extensions:

### SQUEEZE 4 BC

```

pageextension 50100 "SQZ Document Ext." extends "DXP SQZ Document v2"

```

```

{
    layout
    {
        addafter(BuyFromVendorInternal)
        {
            field("Customer No. Internal"; Rec."Customer No.")
            {
                ApplicationArea = All;
                ToolTip = 'Specifies the value of the Customer No. field.';
                //The following code is optional. It is used to highlight a recognized value in
the Squeeze Viewer
                trigger OnAssistEdit()
                begin
                    MarkField(Rec.FieldNo("Customer No."), Rec."Customer No.",
Rec);

                end;

                trigger OnValidate()
                begin
                    CheckPlausibility();
                end;
            }
        }
        addafter(BuyFromVendorExternal)
        {
            field("Customer No."; Rec."Customer No.")
            {
                ApplicationArea = All;
                ToolTip = 'Specifies the value of the Customer No. field.';
                //The following code is optional. It is used to highlight a recognized value
in the Squeeze Viewer
                trigger OnAssistEdit()
                var
                    ApiMgt: Codeunit "DXP SQZ API Mgt.";
                    ViewerResident: Codeunit "DXP SQUEEZE Viewer Resident";
                begin
                    MarkField(Rec.FieldNo("Customer No."), Rec."Customer No.",
Rec);

                end;
            }
        }
    }
}

```

```

        trigger OnValidate()
        begin
            CheckPlausibility();
        end;
    }
}

local procedure MarkField(AppFldNo: Integer; FieldVal: Variant; DocHeader: Record "DXP
SQZ Document Header")
var
    ApiMgt: Codeunit "DXP SQZ API Mgt.";
    ViewerResident: Codeunit "DXP SQUEEZE Viewer Resident";
begin
    ViewerResident := GetViewerResident();
    ApiMgt.MarkField(AppFldNo, FieldVal, ViewerResident, DocHeader);
end;
}

```

```

pageextension 50101 "SQZ Document Sf. Ext." extends "DXP SQZ Document Subform"
{
    layout
    {
        addafter("No. ")
        {
            field("Customer No."; Rec."Customer No.")
            {
                ApplicationArea = All;
                ToolTip = 'Specifies the value of the Customer No. field.';
            }
        }
    }
}

```

## Zielbeleg (hier: Einkaufsbeleg)

```

pageextension 50102 "Purchase Invoice Ext." extends "Purchase Invoice"
{
    layout
    {

```

```

    addafter("Buy- from Vendor No.")
    {
        field("DXP Customer No."; Rec."DXP Customer No.")
        {
            Tooltip = 'Specifies the value of the DEXPRO Customer No.
field. ';
            ApplicationArea = all;
        }
    }
}

```

```

pageextension 50103 "Purchase Inv. Sf Ext." extends "Purch. Invoice Subform"
{
    layout
    {
        addafter("No. ")
        {
            field("DXP Customer No."; Rec."DXP Customer No.")
            {
                Tooltip = 'Specifies the value of the DEXPRO Customer No.
field. ';
                ApplicationArea = all;
            }
        }
    }
}

```