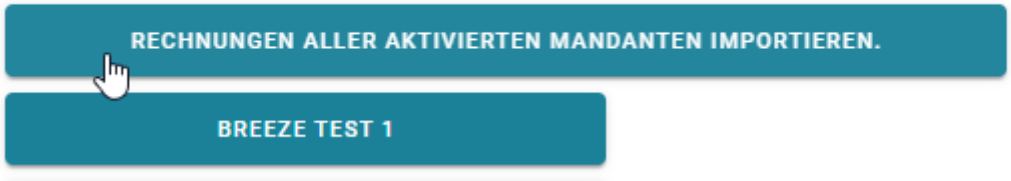
Rechnungsimport

Über den Ordner "**Dynamics 365 BC->Rechnungsimport**" kann ein manueller Rechnungsimport gestartet werden. Im Standard können diese ebenfalls über das Job-Skript **DEXPRO__UserExit_NAV_DynamicsImport** ausgeführt werden.

Auf dem Ordner können entweder die Rechnungen aller Mandanten importiert werden oder nur von einem bestimmten.



Nach dem Import wir ebenfalls angezeigt, wie viele Rechnungen importiert worden sind und ob es ggf. zu Fehlern beim Import kam.

> Fehlerhaft importierte Rechnungen können auf dem öffentlichen Ordner **Fehler beim Import** angesehen und neu gestartet werden,

User-Exits für den Rechnungsimport

Im Standard werden die Felder aus dem Feld-Mapping verwendet, um zu definieren, welche Felder aus BC in welches Feld im Workflow geschrieben werden sollen.

In dem Skript **DEXPRO__UserExit_CustomDynamicsBCLib** befinden sich verschiede User-Exits mit welchem die Daten beim oder nach dem Import modifiziert werden können. Diese User-Exit-Funktionen befinden sich im Standard in der Funktion **userExit_dynamicsFileImport**.

userExit_ParseHeadValue (User-Exit für das Parsing von Kopfwerten beim Import)

Über dieses User-Exit können Kopfwerte so geparst werden, dass diese im Workflow verwendet werden können.
Im Standard sind hier schon UserExits für bestimmte Felder enthalten.

Beispiel am Feld InvoiceCreditVoucher

Für Documents-Feld "**InvoiceCreditVoucher**" wird geprüft, ob in dem BC-Feld typ der Wert " **Rechnung**"/"**Invoice**" bzw. "**Gutschrift**"/"**Creditor Memo**" steht und je nachdem, welcher Werte in dem BC Feld steht, wird entweder der technische Wert "**invoice**" oder "**creditvoucer**" in das Documents-Feld geschrieben.

Über die variable **value** wird immer definiert, welcher Wert in das Feld geschrieben wird.

> Die Variable **value** darf nie gelöscht werden. Ansonsten ist die Wahrscheinlichkeit groß, dass es zu Fehlern im Workflow kommt.

Diese Art von Parsing ist manchmal notwendig, da es Felder gibt, in welchem BC nur die übersetzen Werte liefert, obwohl eigentlich ein technischer Wert für den Workflow notwendig ist.

```
// User Exit welches das Parsen von gemappten Kopfwerten ermöglicht
nav.userExit_ParseHeadValue = function (documentsFieldName, value, dynamicsInvoiceData, docFile) {
    switch (documentsFieldName) {
        case "InvoiceCreditVoucher":
            var invoiceType = dynamicsInvoiceData.type;
            var invoiceTypeParsed = invoiceType;
            if (invoiceType === "Rechnung" || invoiceType === "Invoice") {
                invoiceTypeParsed = "invoice"
            }
            else if (invoiceType === "Gutschrift" || invoiceType === "Credit Memo") {
                invoiceTypeParsed = "creditvoucher"
            }

            value = invoiceTypeParsed;
            break;
        case "NavisionMetadata":
        case "NavisionDimensionData":
            if (typeof value !== "undefined") {
                value = JSON.stringify(value);
            } else {
                value = "";
            }
            break;
        case "Currency":
            // Standardwert bei leerem Wert?
            if (value === "") {
                value = "EUR";
            }
            break;
        default:
            break;
    }

    return value; // Ein Wert muss immer zurückgeben werden!
}
```

userExit_ParsePosValue (User-Exit für das Parsing von Positionswerten beim Import)

Über dieses User-Exit können Positionswerte so geparst werden, dass diese im Workflow verwendet werden können.
Im Standard sind hier schon UserExits für bestimmte Felder enthalten.

Das Prinzip ist dabei identisch zu dem User-Exit der Kopffelder.

```
// User Exit welches das Parsen von gemappten Positionswerten ermöglicht
nav.userExit_ParsePosValue = function (documentsFieldName, value, navisionRow, invoiceData, docFile) {
    switch (documentsFieldName) {
        case "NavisionType":
            var NavisionType = navisionRow["type"];
            var typeParsed = value;
            if (NavisionType === "G_L_Account" || NavisionType === "Sachkonto" || NavisionType === "G/L Account") {
                typeParsed = "G_L_Account";
            }
            else if (NavisionType === "Fixed_Asset" || NavisionType === "WG/Anlage" || NavisionType === "Fixed Asset") {
                typeParsed = "Fixed_Asset";
            }
            else if (NavisionType === "Item" || NavisionType === "Artikel") {
                typeParsed = "Item";
            }
            else if (NavisionType === "Zu-/Abschlag (Artikel)" || NavisionType === "Charge (Item)" || NavisionType === "Charge_Item") {
                typeParsed = "Charge_Item";
            }
            value = typeParsed;
            break;

        case "NavisionMetadata":
            value = JSON.stringify(value);
            break;
        default:
            break;
    }

    return value; // Ein Wert muss immer zurückgeben werden!
}
```

ue_afterInvoiceCreate (User-Exit, welches nach dem Erstellen der Documents-Mappe ausgeführt wird)

Dieses User-Exit kann dafür verwendet werden, um z.B. nachträglich Berechnungen in den Positionen auszuführen oder sonstige Prüfungen an der Rechnungsmappe vorzunehmen.

Im Standard werden folgende Funktionen hier schon ausgeführt:

1. Berechnung von Brutto- und Rabattwerten für den Workflow
2. Wenn es sich um eine Rechnung mit Bestellbezug handelt, werden automatisch die Bestell- und Wareneingangsdaten für diese Bestellung in der BREEZE Stammdatentabelle aktualisiert.
3. Wenn Kontierungswerte in den Positionen bzw. Kreditorenwerte noch nicht in der dazugehörigen Stammdatentabelle vorhanden sind, wird automatisiert dieser Eintrag aus BC gezogen und in die Stammdatentabelle hinzugefügt. Für folgende Daten steht diese Funktion zur Verfügung
    1. Kreditor-Daten
    2. Navision-Typ (Sachkonten, Artikel, W/G Anlage, etc.)
    3. Kostenstellen
    4. Kostenträger
    5. Alle konfigurierten Dimensionen

```
/**
 * Callback, welches nach der Erstellung einer Rechnung aus Dynamics 365 BC aufgerufen kann. Kann genutzt werden, um die Mappe zu
 * @param {*} docFile
 * @param {Object} accountingErrorsPos Object that contains error for every position field
 */
nav.ue_afterInvoiceCreate = function (docFile, accountingErrorsPos) {
    var gentable = new Gentable(docFile);
    gentable.readFromField();
    var sql = new SqlObject(cTableMD_Order, cDbMasterData);

    var sumGross = 0;
    var sumTax = 0;
        (local var) uniqueOrderNumbers: any[]
    var uniqueOrderNumbers = [];

    for (var i = 0; i < gentable.Rows.length; i++) {
        var row = gentable.Rows[i];

        if (typeof row["OrderNumber"] !== "undefined" && row["OrderNumber"] != null && row["OrderNumber"] !== "" && uniqueOrderNu
            uniqueOrderNumbers.push(row["OrderNumber"]);

            // Ziehe die Bestelldaten für diese Bestellnummer
            var call = new ScriptCall(context.currentUser, "DEXPRO__UserExit_NAVImport_OrderDataDynamics", true);
            call.addParameter("paramImportByOrderNumber", "true");
            call.addParameter("paramPrincipal", docFile.Principal);
            call.addParameter("paramOrderNumber", row["OrderNumber"]);
            if (call.launch()) {
                if (call.waitForFinish()) {
                    var retString = call.getReturnValue();
                    this.write(retString);
                }
            }
```

Revision #7
Created 28 October 2022 12:51:23 by Helge Czerwinski
Updated 4 December 2023 14:57:34 by Helge Czerwinski