Export UserExits

- Dynamics365
- EASY Content Server
- Otris SOAP
- Workday
- ELO Indexserver - Before Mapping

Dynamics365

Es ist möglich vor der Übergabe an Dynamics365 weitere Felder hinzuzufügen oder Werte anzupassen.

Hier ein Beispiel um den Netto und Steuerbetrag hinzuzufügen:

```
<?php

use Squeeze\SqueezeAmount;
use Squeeze\xDoc;

function BeforeDynamics365HeaderExport(xDoc $doc, array $fieldList)
{
    // ===========================================
    // fieldList is an array of fields with the
    // following attributes
    //
    // fieldList[]['name']         = Squeeze field name
    // fieldList[]['externalName'] = Squeeze field external name
    // fieldList[]['type']         = Squeeze field type (Text, Amount, Date)
    // fieldList[]['value']        = Squeeze field value
    //
    // ===========================================

    //$logger = Logger::getLogger("main");

    $netAmount = $doc->getFieldByName('NetAmount');
    if($netAmount == null) {
        throw new \Exception('Field NetAmount does not exist.');
    }
    $fieldList['CustomNetAmount']['name'] = 'NetAmount';
    $fieldList['CustomNetAmount']['externalName'] = 'InvoiceAmountExcludingTaxCC';
    $fieldList['CustomNetAmount']['type'] = 'Amount';
    $fieldList['CustomNetAmount']['value'] = SqueezeAmount::formatAndCalc($netAmount-
>getValue()->value);

    $taxAmount = $doc->getFieldByName('TaxAmount');
```

```
    if($taxAmount == null) {
        throw new \Exception('Field TaxAmount does not exist.');
    }
    $fieldList['CustomTaxAmount']['name'] = 'TaxAmount';
    $fieldList['CustomTaxAmount']['externalName'] = 'TaxAmountCC';
    $fieldList['CustomTaxAmount']['type'] = 'Amount';
    $fieldList['CustomTaxAmount']['value'] = SqueezeAmount::formatAndCalc($taxAmount-
>getValue()->value);

    return new \Squeeze\xReturnObject(true, 200, 'UserExit BeforeDynamics365HeaderExport
successful', $fieldList);
}
```

EASY Content Server

Vor der Übergabe eines Dokumentes an den EASY Content Server ist es möglich das Schema sowie Feldwerte und Anlagen zu modifizieren oder zu ergänzen. Hier ein Beispiel für ein UserExit:

```
<?php

use Squeeze\xDoc;
use Squeeze\xReturnObject;

// vor Sqeeze 2.5: BeforeExportEasyContentServer statt BeforeEasyContentServerExport
function BeforeEasyContentServerExport(xDoc $doc, stdClass $ecsRecord): xReturnObject
{
    try {
        // Create a logger instance (optional)
        $logger = Logger::getLogger("main");

        // get the Squeeze field with the name "Company"
□□$company = $doc->getFieldByName('Company');
□□if($company === null) {
            // return an error if the Squeeze field does not exist
□□□return new xReturnObject(false, 500, 'Field Company does not exist', $ecsRecord);
□□}
□□
        // If the value of the field Company is equal 1000
□□if($company->getValue()->value == '1000') {

            // Change the schema to be Test
            $ecsRecord->store = 'Test';

            // Modify the document field value to "TestValue" for the field with the name
"TestField"
            $ecsRecord->fields['TestField'] = 'TestValue';

□□} else {
            // return an error if the field Company is not equal 1000
            return new xReturnObject(false, 500, 'Company ' . $company->getValue()->value . '
is invalid', $ecsRecord);
```

```
        }

        // return a new result with the modified $ecsRecord variable
        return new xReturnObject(true, 200, 'UserExit BeforeExportEasyContentServer
successful', $ecsRecord);

    } catch (Exception $e) {
        // in case of an exception return an error
        return new xReturnObject(false, 500, $e->getMessage(), $ecsRecord);
    }
}
```

Otris SOAP

Vor der Übergabe eines Dokumentes an die Otris SOAP Schnittstelle ist es möglich Modifikationen am Dokument aber auch an den Verbindungsparametern vorzunehmen. Hier ein Beispiel für einen solchen UserExit:

```
<?php

use Squeeze\xDoc;
use Squeeze\xReturnObject;

// vor Squeeze 2.5: BeforeExportOtrisSoap anstatt BeforeOtrisSoapExport
function BeforeExportOtrisSoap(xDoc $doc, array $soapDoc): xReturnObject
{
    try {

        $company = $doc->getFieldByName('Company');
        if ($company === null) {
            // return an error if the Squeeze field does not exist
            return new xReturnObject(false, 500, 'Field Company does not exist', $soapDoc);
        }

        // If the value of the field Company is equal 1000
□□if ($company->getValue()->value == '1000') {
□□    // use a different server
□□□$soapDoc['parameter']['ServerUrl'] = 'http://123.123.123.123:11001';
□□}

        // return a new result with the modified $soapDoc variable
        return new xReturnObject(true, 200, 'UserExit BeforeExportOtrisSoap successful',
$soapDoc);

    } catch (Exception $e) {
        // in case of an exception return an error
        return new xReturnObject(false, 500, $e->getMessage(), $soapDoc);
    }
}
```

Workday

Vor der Übertragung eines Beleges an einen Workday ERP Tenant ist es möglich Werte anzupassen oder zu ergänzen.

Hier ein Beispiel:

```
<?php

use Squeeze\xDoc;
use Squeeze\xReturnObject;

// vor Squeeze 2.5: BeforeExportWorkday statt BeforeWorkdayExport
function BeforeWorkdayExport(xDoc $doc, array $workdayDoc): xReturnObject
{
    // This UserExit allows to modify the
    // Workday data structure right before the document is submitted.

    try {
        $logger = Logger::getLogger("main");

        if (isset($workdayDoc['Supplier_Invoice_Data'])) {
            $isInvoice = true;
            $rootTag = 'Supplier_Invoice_Data';
        } else {
            $isInvoice = false;
            $rootTag = 'Supplier_Invoice_Adjustment_Data';
        }

        if ($isInvoice) {
            $workdayDoc[$rootTag]['Adjustment_Reason_Reference']['ID']['_'] = 'Other';
        }

        // Remove Accounting_Date_Override (PostingDate)
        unset($workdayDoc[$rootTag]['Accounting_Date_Override']);

        // ================================================
        // iterate over every line item of the document
```

```
        // ================================================
        for($i = 0; $i < count($workdayDoc[$rootTag]['Invoice_Line_Replacement_Data']); $i++)
{

            // ================================================
            // Check if Tax Option is set
            // this means it's a reverse charge line item
            // ================================================
            if
(isset($workdayDoc[$rootTag]['Invoice_Line_Replacement_Data'][$i]['Tax_Rate_Options_Data']['Ta
x_Option_1_Reference']['ID'])) {

                // If the Tax Option is set also set the Tax Recoverability

$workdayDoc[$rootTag]['Invoice_Line_Replacement_Data'][$i]['Tax_Rate_Options_Data']['Tax_Recov
erability_1_Reference']['ID'] = array('_' => '100%_Fully Recoverable', 'type' =>
'Tax_Recoverability_Object_ID');
                $logger->debug('Tax Option is set. Set Tax Recoverability to 100%_Fully
Recoverable');
            }
        }

        return new xReturnObject(true, 200, 'UserExit BeforeExportWorkday successful',
$workdayDoc);

    } catch (Exception $e) {
        $logger = Logger::getLogger("main");
        $logger->error($e->getMessage());
        return new xReturnObject(false, 500, $e->getMessage(), $workdayDoc);
    }
}
```

ELO Indexserver - Before Mapping

Vor der Übertragung bzw. dem Export an ELO via Indexserver REST API kann das Mapping der Felder mit diesem UserExit manipuliert oder gar gänzlich geändert werden.

Als Parameter erhält der UserExit das xDoc, mit dem ein feldabhängiges Mapping erstellt werden kann.

Außerdem das Default Mapping, in dem alle Felder mit externen Namen bereits aufgelistet sind.

Hier können auch Felder gemappt werden, die nicht im sogenannten "Sord" (Metadaten Objekt) enthalten sind.
Bspw. das Datumsfeld `xDateIso` oder der Name des ELO Vorgangs `name` (nicht zu verwechseln mit dem Dateinamen selbst).

Hier ein Beispiel:

```
<?php

declare(strict_types=1);

namespace Squeeze\UserExits\Invoices;

use Squeeze\xDoc;
use Squeeze\xReturnObject;

function BeforeEloIndexServerMapping(xDoc $doc, array $mapping): xReturnObject
{
    $company = $doc->getActualFieldValue('Company');

    // Use a completely different mapping for each company.
    $mapping = match ($company) {
        '<COMPANY_1>' => [
            //"ELO_FIELD_NAME" => "SQUEEZE_FIELD_NAME",
            "XDateIso" => "DocumentDate", // XDateIso is a reserved field name which refers to
the ELO process field "Date".
            "name" => "DocumentName" // Name is a reserved field name which refers to the ELO
process field "name".
```

```
        ],
        '<COMPANY_2>' => [
            //"ELO_FIELD_NAME" => "SQUEEZE_FIELD_NAME",
            "XDateIso" => "DocumentDate", // XDateIso is a reserved field name and is not mapped like the other fields, but is included here.
            "name" => "DocumentName" // Name is a reserved field name and is not mapped like the other fields, but is included here.
        ],
        default => [], // empty mapping will result in an error, please provide the original mapping if no change is needed.
    };

    return new xReturnObject(true, 200, "Mapping ok", $mapping);
}
```