SQUEEZE Beispiele für UserExits

In diesem Buch sind einige Beispiele für nützliche UserExits hinterlegt.

- Liste aller verfügbaren UserExits
- MasterData
  - Allgemeine Informationen
  - creditors.php
  - Stammdaten per RFC aus SAP holen
- AfterEmailImport
  - Company auf Basis des Empfängers setzen
- BeforeFileImport
  - Attribute einer Importdatei in ein Squeeze Feld übergeben
- Barcodetrennung
  - Barcodetrennung
  - Barcodetrennung mit ausschließen der Trenner-Seite
  - Klassifikation mittels Barcodetrennung
- AfterLocatorExtraction
  - Allgemeine Informationen
- AfterExtraction
  - Eine Position je Bestellnummer
  - Übergabe jeder Bestellnummer Alternative als kommaseparierte Liste
- AfterValidation

    - Versenden einer Email nach der Validierung

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

- ValidateDocument
    - Pflichtfeldprüfung - Entweder Oder

- BeforeExport
    - Text im PDF andrucken

- JS Client UserExits
    - Werteübernahme aus einer DropDown-Liste

- Digitalen Barcode mit Squeeze erzeugen
- Import Dateinamen verwenden

Liste aller verfügbaren UserExits

MasterData

Allgemeine Informationen

Bei jedem Stammdatenimport können die zu importierenden Daten aufbereitet werden.
Für die Aufbereitung der Stammdaten stehen vordefinierte UserExits zur Verfügung, die für diesen Zweck genutzt werden können.

UserExit Verzeichnis

Das UserExit-Verzeichnis für die Nutzung ist immer
`\htdocs\repository\client.server.net\UserExits\MasterData\tabellenname.php`

Ein Beispiel für die Aufbereitung der Lieferantenstammdaten wäre also z.B.:
`\htdocs\repository\localhost\UserExits\MasterData\creditors.php`

UserExit Funktion

In der UserExit Datei der jeweiligen Tabelle muss eine Function mit einem bestimmten Namen vorhanden sein.
Der Name dieser Funktion leitet sich aus dem technischen Tabellennamen ab. Hier der Rumpf für das oben angegebene Beispiel:

```
<?php


function Enhance_Creditors(array $data): ?array
{
    return $data;
}
```

In der Variablen $data ist eine Zeile mit allen Spalten enthalten. Diese Zeile kann nun nach den jeweiligen Anforderungen überprüft und aufbereitet werden.

Sollen z.B. die Leerzeichen einer IBAN entfernt werden und das Land mit Großbuchstaben dargestellt werden, so könnte folgende Funktion genutzt werden:

```
<?php


function Enhance_Creditors(array $data): ?array
{
    // Enhance IBAN
```

```
    if (isset($data['IBAN'])) {
        $data['IBAN'] = strtoupper(str_replace(' ', '', $data['IBAN']));
    }

    return $data;
}
```

Wie in diesem Beispiel zu sehen ist, wird auf die Daten der Spalte IBAN auf das Array `$data['IBAN']` zugegriffen.
Die Leerzeichen dieser IBAN werden mit der Funktion `str_replace(' ', '', $data['IBAN'])` ersetzt und mit der umschließenden Funktion `strtoupper()` werden alle Buchstaben in Großbuchstaben gewandelt.

Aus dem Wert `De12 1234 5678 9012 3456 78` würde mit dieser Funktion der Wert `DE12123456789012345678` erzeugt werden.

Ausnahmen definieren

Soll ein ganzer Datensatz nicht importiert werden, dann muss lediglich der Wert `null` zurückgegeben werden. Soll also zum Beispiel der Datensatz nicht importiert werden, wenn keine IBAN angegeben wurde so wäre folgende Erweiterung möglich:

```
<?php

function Enhance_Creditors(array $data): ?array
{
    // Enhance IBAN
    if (isset($data['IBAN'])) {
        $data['IBAN'] = strtoupper(str_replace(' ', '', $data['IBAN']));

        if ($data['IBAN'] === '') {
            return null;  // verhindert den Import des gesammten Datensatzes
        }
    }

    return $data;
}
```

creditors.php

Die Lieferantenstammdaten müssen i.d.R. noch normalisiert werden.
Dazu gibt es im Invoice Template einen ausgelieferten UserExit, der diese Aufgabe übernimmt.

Zu den Normalisierungsschritten gehört

- das Setzen einer Firmennummer, wenn keine Firmennummer angegeben wurde
- das Entfernen der Leerzeichen in IBANs
- das Entfernen der Leerzeichen in Umsatzsteueridentifikationsnummern
- das Entfernen von Sonderzeichen in Telefon und Faxnummern
- das Setzen der BLZ auf Basis der IBAN, sofern keine BLZ angeben ist
- das Setzen des Kontos auf Basis der IBAN, sofern keine Konto angeben ist
- die Umschlüsselung von Länderkennzeichen

Hier das ausgelieferte Skript, welches beliebig auf die Anforderungen des Kunden angepasst werden kann:

```
<?php

function Enhance_Creditors($data){
    $logger = Logger::getLogger("main");

    // Enhance Company Id
    if(isset($data['CompanyId']) == false) $data['CompanyId'] = "1000";

    // Enhance IBAN
    if(isset($data['IBAN'])){
        $data['IBAN'] = strtoupper(str_replace(" ","",$data['IBAN']));
    }

    // Enhance EU Tax Id
    $data['EUTaxId'] = strtoupper(str_replace(" ","",$data['EUTaxId']));

    // Enhance Phone
    if(isset($data['Phone'])) {
        $data['Phone'] = str_replace(" ", "", $data['Phone']);
        $data['Phone'] = str_replace("-", "", $data['Phone']);
```

```
    $data['Phone'] = str_replace("/", "", $data['Phone']);
    $data['Phone'] = str_replace("(", "", $data['Phone']);
    $data['Phone'] = str_replace(")", "", $data['Phone']);
    $data['Phone'] = str_replace("\\", "", $data['Phone']);
}

// Enhance Fax
if(isset($data['Fax'])) {
    $data['Fax'] = str_replace(" ", "", $data['Fax']);
    $data['Fax'] = str_replace("-", "", $data['Fax']);
    $data['Fax'] = str_replace("/", "", $data['Fax']);
    $data['Fax'] = str_replace("(", "", $data['Fax']);
    $data['Fax'] = str_replace(")", "", $data['Fax']);
    $data['Fax'] = str_replace("\\", "", $data['Fax']);
}

// Enhance Bank Code if IBAN is German
if(isset($data['BankCode']) && $data['IBAN']){
    if($data['BankCode'] == "" && substr($data['IBAN'],0,2) == "DE"){
        $data['BankCode'] = substr($data['IBAN'],4,8);
    }
}

// Enhance Bank Account if IBAN is German
if(isset($data['BankAccount']) && $data['IBAN']){
    if($data['BankAccount'] == "" && substr($data['IBAN'],0,2) == "DE"){
        $data['BankAccount'] = ltrim(substr($data['IBAN'],12,10),"0");
    }
}

// Enhance Bank Account delete leading Zeros
if(isset($data['BankAccount'])){
    $data['BankAccount'] = ltrim($data['BankAccount'],"0");
}

// Enhance Country Code if EU Tax Id is set or IBAN is set
if($data['CountryCode'] == "" && $data['EUTaxId'] != ""){
    $data['CountryCode'] = substr($data['EUTaxId'],0,2);
} elseif($data['CountryCode'] == "" && isset($data['IBAN'])){
    $data['CountryCode'] = substr($data['IBAN'],0,2);
```

```
    } elseif($data['CountryCode'] == "D"){
        $data['CountryCode'] = "DE";
    } elseif($data['CountryCode'] == "A"){
        $data['CountryCode'] = "AT";
    } elseif($data['CountryCode'] == "B"){
        $data['CountryCode'] = "BE";
    } elseif($data['CountryCode'] == "E"){
        $data['CountryCode'] = "ES";
    } elseif($data['CountryCode'] == "F"){
        $data['CountryCode'] = "FR";
    } elseif($data['CountryCode'] == "I"){
        $data['CountryCode'] = "IT";
    } elseif($data['CountryCode'] == "IRL"){
        $data['CountryCode'] = "IE";
    } elseif($data['CountryCode'] == "USA"){
        $data['CountryCode'] = "US";
    } elseif($data['CountryCode'] == "S"){
        $data['CountryCode'] = "SE";
    }
    return $data;
}
```

Stammdaten per RFC aus SAP holen

Voraussetzungen:

1. Es werden folgende Dateien benötigt: php_sapnwrfc.dll, sapnwrfc.dll
2. Es wird ein SAP RFC Baustein je Stammdaten Tabelle benötigt, welcher Stammdaten liefert
3. Es wird ein SAP Benutzer benötigt
4. Es werden die SAP Daten benötigt: Host, SysNr, Client, User, password

Vorbereitung:

1. SAP DLLs im Squeeze/php/ext ablegen
2. In der php.ini *extension=php_sapnwrfc.dll* hinzufügen
3. In der Squeeze/repository/config/clients/<serverFQDN>.json erweitern:

```
"sap": {
        "host": "host",
        "instance": "01",
□□"sysid": "C01",
□□"client": "940",
  □□"user": "xyz",
□□"password": "geheim"
    }
```

1. Unter Squeeze/repository/<ServerFQDN>/UserExits/MasterData eine Datei "SapRfcUpdateSettings.json" anlegen und füllen mit:

```
{
"tables": [
{
"tableid": "1",
"query": "ZZDEX_MASTERDATA",
"variant": " "
}
]
```

```
}
```

1. Den folgenden UserExit anlegen:

```
<?php
use Dompdf\Exception;
use Squeeze\xTools;

// Disable xdebug to prevent CLI hanging
if(extension_loaded('xdebug')) {
    xdebug_disable();
}

// Alle Fehler ausser E_NOTICE melden
error_reporting(E_ALL ^ E_NOTICE);
ini_set('display_errors',true);
ini_set("log_errors", true);
ini_set("error_log", dirname(__DIR__) . "/logs/php-error.log");
//ini_set("memory_limit", "-1");
ini_set('max_execution_time', "1200");

require_once __DIR__ . '/../../../../htdocs/bootstrap/app.php';

session_start();

if($argv[2] == '') $argv[2] = '80';

$_SESSION['SERVER_NAME'] = $argv[1];
$_SESSION['SERVER_PORT'] = $argv[2];

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

// ============================================
// start time
// ============================================
$time_start = xTools::microtime_float();

$logger->info("Refresh master data for client ". $argv[1]. " running on port " . $argv[2]);

// ============================================
```

```
// Get Root directory
// ============================================
$dirSep = DIRECTORY_SEPARATOR;
$config = new \Squeeze\SqueezeConfig();
$repoRoot = $config->get('repository.root');

// ============================================
// Get defined Table Settings
// ============================================
$settingsFile = $repoRoot . 'UserExits' . $dirSep . 'MasterData' . $dirSep .
'SapRfcUpdateSettings.json';
$logger->debug($settingsFile);
if(file_exists($settingsFile)){
    $settings = json_decode(file_get_contents($settingsFile));
    $logger->debug($settings);
    foreach($settings->tables as $table){
        $logger->debug($table);
        //$tableResult = refreshMasterDataTableViaRfc($logger, $table, $config,
$_SESSION['SERVER_NAME'], $_SESSION['SERVER_PORT']);
        $tableResult = MasterDataRefreshForSapRfc($table->tableid, $table->query,$table-
>variant);
    }
} else {
    throw(new \Exception('SapRfcUpdateSettings.json does not exist.', 400));
}

// end time
$time_end = xTools::microtime_float();
$time = $time_end - $time_start;
$logger->info("Refresh master data for client ". $argv[1]. " finished");

/**
 * @param Logger $logger
 * @param stdClass $table
 * @param \Squeeze\SqueezeConfig $config
 * @param string $host
 * @param string $port
 * @return \Squeeze\xReturnObject
 */
function refreshMasterDataTableViaRfc(Logger $logger, $table, \Squeeze\SqueezeConfig $config,
```

```
$host, $port){
    try {

        $curl = curl_init();
        $headers = array("Content-Type:multipart/form-data");
        $options = array(
            CURLOPT_URL => 'http://' . $host . ':' . $port .
"/api/MasterDataRefreshViaSapRfc",
            CURLOPT_HEADER => true,
            CURLOPT_POST => 1,
            CURLOPT_HTTPHEADER => $headers,
            CURLOPT_RETURNTRANSFER => true
        ); // cURL options
        curl_setopt_array($curl, $options);
        $args['tableid'] = $table->tableid;
        $args['sapRfcQuery'] = $table->query;
        $args['sapRfcVariant'] = $table->variant;
        curl_setopt($curl, CURLOPT_POSTFIELDS, $args);
        $logger->debug($curl);
        curl_exec($curl);
        if(!curl_errno($curl))
        {
            $info = curl_getinfo($curl);
            if ($info['http_code'] == 200) {
                $logger->debug('MasterData Refresh successful');
            } else {
                $msg = curl_error($curl);
                $logger->error($msg);
            }
        }
        else
        {
            $msg = curl_error($curl);
            $logger->error($msg);
        }
        curl_close($curl);

    } catch (Exception $e) {
        $logger = Logger::getLogger("main");
        $logger->error($e->getMessage());
```

```
        return new \Squeeze\xReturnObject(false, 500, $e->getMessage(), null);
    }
}


function MasterDataRefreshForSapRfc($tableId, $queryName, $queryVariant ) {
    try {
        $logger = Logger::getLogger("main");
        $logger->info("Start");

        ini_set("memory_limit", "-1");

        // ================================================
        // Check if tableid is defined
        // ================================================
        if ($tableId == "") {
            $msg = __FUNCTION__ . ' ' . "No Table ID defined!";
            $logger->error($msg);
            return false;
        }

        // ================================================
        // Check if sapRfcQuery is defined
        // ================================================
        if ($queryName == "") {
            $msg = __FUNCTION__ . ' ' . "No Query defined!";
            $logger->error($msg);
            return false;
        }

        // ================================================
        // Check if sapRfcVariant is defined
        // ================================================
        if ($queryVariant == "") {
            $msg = __FUNCTION__ . ' ' . "No Variant defined!";
            $logger->error($msg);
            return false;
        }

        $uuid = uniqid();
```

```
        $dirSep = DIRECTORY_SEPARATOR;

        // ================================================
        // Get Master Data Table object
        // ================================================
        $xMasterDataTable = new \Squeeze\xMasterDataTable();
        if(strtoupper($queryName) == 'MD_VENDOR') {
            $xMasterDataTable->loadByName('creditors');
        }
        elseif(strtoupper($queryName) == 'ZZDEX_MASTERDATA') {
            $xMasterDataTable->loadByName('creditors');
        }
        elseif(strtoupper($queryName) == 'MD_PO') {
            $xMasterDataTable->loadByName('orders');
        }

        // ================================================
        // Get Configuration value
        // ================================================
        $config = new \Squeeze\SqueezeConfig();
        $sapConfig = [
            'ashost' => $config->get('sap.host'),
            'sysnr' => $config->get('sap.instance'),
'sysid' => $config->get('sap.sysid'),
'lang' => 'DE',
            'client' => $config->get('sap.client'),
            'user' => $config->get('sap.user'),
            'passwd' => $config->get('sap.password'),
'trace' => '1'
        ];

        $logger->debug($sapConfig);

        $c = new \SAPNWRFC\Connection($sapConfig);
$f = $c->getFunction('ZZDEX_MASTERDATA');
        //$result = $f->invoke([
        //    'I_WORKSPACE' => 'G',
        //    'I_USERGROUP' => '/QUERY',
        //    'I_QUERY' => $queryName,
        //    'I_VARIANT' => $queryVariant
```

```
        //]);
$result = $f->invoke();

        if($result['RETURN']['TYPE'] == 'E'){
            $msg = __FUNCTION__ . ' ' . $result['RETURN']['MESSAGE'];
            $code = 400;
            $logger = \Logger::getLogger("main");
            $logger->error(__CLASS__ ." ". __FUNCTION__ ." ". $msg);
            return false;
        }

        // ============================================
        // Check if a userexit script for this table exists
        // ============================================
        //$root = \Squeeze\xTools::getRoot();
        $repo = $config->get('repository.root');
        $userExit = xTools::buildAbsolutePath($repo, 'UserExits' . $dirSep . 'MasterData',
true);
        if (file_exists($userExit . $xMasterDataTable->getName() . '.php')) {
            //$logger->info("File for User Exit of table " . $xMasterDataTable->getName() . "
found.");
            include_once($userExit . $xMasterDataTable->getName() . '.php');
        } else {
            //$logger->info("No UserExit for table " . $xMasterDataTable->getName() . "
defined.");
        }

        $resultTable = array();
        $i = 0;

        if(strtoupper($queryName) == 'MD_VENDOR') {
            foreach ($result['E_DATA'] as $line) {

                $cells = explode(';', $line['LINE']);

                $newLine['CreditorId'] = $cells[0];
                $newLine['CompanyId'] = $cells[1];
                $newLine['Name1'] = $cells[2];
                $newLine['Name2'] = $cells[3];
                $newLine['Name3'] = $cells[4];
```

```
                $newLine['Phone'] = $cells[5];
                $newLine['Street'] = $cells[6];
                $newLine['CountryCode'] = $cells[7];
                $newLine['Zip'] = $cells[8];
                $newLine['City'] = $cells[9];
                $newLine['EUTaxId'] = $cells[10];
                $newLine['NationalTaxId'] = $cells[11];
                $newLine['BankCountry'] = $cells[12];
                $newLine['BankCode'] = $cells[13];
                $newLine['BankAccount'] = $cells[14];
                $newLine['IBAN'] = $cells[15];

                $logger->debug($newLine);
                if (function_exists('Enhance_' . $xMasterDataTable->getName())) {
                    $newLine = call_user_func('Enhance_' . $xMasterDataTable->getName(),
$newLine);
                }
                $resultTable[] = $newLine;
            }
        }
        elseif(strtoupper($queryName) == 'ZZDEX_MASTERDATA') {
            foreach ($result['ET_VND'] as $line) {
//0        1     2    3     4     5   6      7   8        9        10      -11 12     13
14     15        16        17   18      19
//Kreditor BuKr. Land Name1 Name2 ??? Name   Ort Ortsteil Postfach PLZ-Postf. PLZ Region
Suchbegr. Straße USt-Id.Nr SteuerNr. IBAN Adresse ??? ??? ???

$logger->debug($line);
                $cells = explode(';', $line['']);
$logger->debug($cells);

                $newLine['CreditorId'] = ltrim( $cells[0], '0' );
                $newLine['CompanyId'] = $cells[1];
                $newLine['Name1'] = $cells[3];
                $newLine['Name2'] = $cells[4];
                //$newLine['Name3'] = $cells[4];
                //$newLine['Phone'] = $cells[5];
                $newLine['Street'] = $cells[14];
                $newLine['CountryCode'] = $cells[2];
                $newLine['Zip'] = $cells[11];
```

```
                $newLine['City'] = $cells[7];
                $newLine['EUTaxId'] = $cells[15];
                $newLine['NationalTaxId'] = $cells[16];
                //$newLine['BankCountry'] = $cells[12];
                //$newLine['BankCode'] = $cells[13];
                //$newLine['BankAccount'] = $cells[14];
                $newLine['IBAN'] = $cells[17];

                //$logger->debug($newLine);
                if (function_exists('Enhance_' . $xMasterDataTable->getName())) {
                    $newLine = call_user_func('Enhance_' . $xMasterDataTable->getName(),
$newLine);
                }
                $resultTable[] = $newLine;
            }
        }
        elseif(strtoupper($queryName) == 'MD_PO') {
            foreach ($result['E_DATA'] as $line) {

                $cells = explode(';', $line['LINE']);

                $newLine['CompanyId'] = $cells[0];
                $newLine['CreditorId'] = $cells[1];
                $newLine['OrderNumber'] = $cells[2];
                $newLine['OrderItem'] = $cells[3];
                $newLine['Material'] = $cells[4];
                $newLine['Description'] = $cells[5];
                $newLine['Quantity'] = $cells[6];
                //$newLine['Quantity'] = $cells[7];
                $newLine['QuantityUnit'] = $cells[8];
                $newLine['PriceUnit'] = $cells[9];
                //$newLine['Client'] = $cells[10];
                $newLine['NetAmount'] = $cells[11];
                $newLine['Currency'] = $cells[12];

                //$logger->debug($newLine);
                if (function_exists('Enhance_' . $xMasterDataTable->getName())) {
                    $newLine = call_user_func('Enhance_' . $xMasterDataTable->getName(),
$newLine);
                }
```

```
                $resultTable[] = $newLine;
            }
        }

        $result = $xMasterDataTable->createNewTempTable($uuid);
        //$logger->info("xMasterDataTable created");
        if ($result->isSuccess() == false) {
            $logger->error($result->getMessage());
            return false;
        }
        //$logger->info("DB Created");

        $result = $xMasterDataTable->insertData($uuid, $resultTable);
$resultTable = null;
        if ($result->isSuccess() == false) {
            $logger->error($result->getMessage());
            return false;
        }
        //$logger->info("DB Insert done");

        $result = $xMasterDataTable->dropTableAndReplace();
        if ($result->isSuccess() == false) {
            $logger->error($result->getMessage());
            return false;
        }
        //$logger->info("Drop and replace done.");

        // ============================================
        // Check if a UserExit script saving the document exists
        // ============================================
        $repo = $config->get('repository.root');
        $userExit = xTools::buildAbsolutePath($repo, 'UserExits/MasterData/', true);
        if (file_exists($userExit . 'AfterMasterDataRefresh.php')) {
            //$logger->info("File for UserExit AfterMasterDataRefresh found.");
            include_once($userExit . 'AfterMasterDataRefresh.php');
            $userExitResult = call_user_func('AfterMasterDataRefresh', $xMasterDataTable);
        } else {
            //$logger->info("No UserExit for AfterMasterDataRefresh defined.");
        }
```

```
        $logger->info(json_encode($xMasterDataTable));

    } catch (\Exception $e) {
        xTools::handleException($e);
        $code = 500;
        $msg = $e->getMessage();
        $logger = \Logger::getLogger("main");
        $logger->error(__CLASS__ ." ". __FUNCTION__ ." ". $msg);
        return false;
    }
}
```

Anpassen des Mappings im UserExit in Zeilen 219ff.

```
Aufruf per "D:\EASY\Squeeze\php\php.exe" RefreshMasterDataViaRfc.php ServerFQDN 80
```

Es wird im UserExit Ordner oder teilweise im Windows/System32 eine *.trc Datei erzeugt, welche die Verbindungsdaten wiedergibt.

> https://github.com/gkralik/php7-sapnwrfc/releases

AfterEmailImport

AfterEmailImport

Company auf Basis des Empfängers setzen

Wenn auf Grund der Empfängeradresse der Mandant gesetzt werden soll, kann der folgende UserExit verwendet werden:

```
<?php

use Squeeze\xDoc;
use Squeeze\Email\EmailFile;
use Squeeze\xReturnObject;

function AfterEmailImport(xDoc $xDoc, EmailFile $emailFile = null)
{
    // Define the valid Email addresses and the assigned company
    $addressMapping = array();
    $addressMapping['mailbox01@company.net'] = '1000';
    $addressMapping['mailbox02@company.net'] = '2000';

    // Loop over all Email addresses
    foreach ($emailFile->getTo() as $address) {

        // Get the current Email address
        $currentAddress = mb_strtolower($address->getEmailAddress());

        // Check if the Email address is defined
        if(isset($addressMapping[$currentAddress])) {

            // Get the assigned company
            $assignedCompany = $addressMapping[$currentAddress];

            // Set the field value for Company
            if($xDoc->getFieldByName("Company") != null) {
                $xDoc->getFieldByName("Company")->getValue()-
>setValueAndText($assignedCompany);
```

```
            }

            // Set the field value for used Email address
            if($xDoc->getFieldByName("EmailToAddress") != null) {
                $xDoc->getFieldByName("EmailToAddress")->getValue()-
>setValueAndText($currentAddress);
            }

            // Optional store the additionalInfo
            $additionalInfo = $xDoc->loadAdditionalInfo();
            $additionalInfo['additionalInfo']['Company'] = $assignedCompany;
            $additionalInfo['additionalInfo']['EmailToAddress'] = $currentAddress;
            $xDoc->storeAdditionalInfo($additionalInfo);

            // break the loop so only the first valid address is used
            break;
        }
    }
    return new xReturnObject(true, 200, 'UserExit processed', null);
}
```

BeforeFileImport

Dieser UserExit wird nach dem Import von Belege über den PollDirectory-Import (Filesystem-Import) ausgelöst.

Attribute einer Importdatei in ein Squeeze Feld übergeben

In diversen Verarbeitungsszenarien kann es notwendig sein, den Dateinamen oder Anteile des Dateinamens im Verarbeitungsprozess zu berücksichtigen.

In dem folgenden Code-Beispiel wird der Dateiname ohne Pfadangabe und ohne Extension in das mehrdimensionale Array "additionalInfo" geschrieben. Dabei wird der Dateiname in den Index "**Filename**" geschrieben.

Zusätzlich wird aus dem Importpfad der Datei, der Mandant abgeleitet und entsprechend vorbelegt.
Der Mandant wird in den Index "**Company**" geschrieben.

Sobald ein identisch benanntes Feld in der Squeeze Dokumentenklasse existiert, wird der Inhalt des Array-Indexes (hier der Dateiname und der Mandant) in das entsprechende Squeeze-Feld geschrieben.

```
<?php

use Squeeze\xDataBaseClass;
use Squeeze\xReturnObject;
use Squeeze\xTools;

/**
 * @param array $params
 * @return xReturnObject
 * @throws Exception
 */
function BeforeFileImport(array $params): xReturnObject
{
    $logger = Logger::getLogger("main");
    $db = xDataBaseClass::getDBConnection();

    return prepareFileImport($db, $params, $logger);
}
```

```
/**
 * @param PDO $db
 * @param array $params
 * @param Logger $logger
 * @return xReturnObject
 * @throws Exception
 */
function prepareFileImport(PDO $db, array $params, Logger $logger): xReturnObject
{
    try {
        $originalFileName = $params['originalFile'];

        if (!isset($originalFileName)) {
            return new xReturnObject(false, 400, 'originalFile not set', null);
        }

        // initialize additionalInfo array
        $params['additionalInfo'] = array();

        // Save fileName without the extension in the additionalInfo
        $params['additionalInfo']['Importfile'] = $originalFileName;
        $params['additionalInfo']['Filename'] =
xTools::getFileNameWithoutExtension($originalFileName);

        // Get the company from the folder structure
        // Example: "C:\tmp\import\Test\1000_Test\0003_63da7bb4b0589621675295.pdf"
        $logger->debug('Original Path and File Name = ' . $originalFileName);
        $directories = explode(DIRECTORY_SEPARATOR, $originalFileName);
        // $directories[0] = 'C:'
        // $directories[1] = 'tmp'
        // $directories[2] = 'import'
        // $directories[3] = 'Test'
        // $directories[4] = '1000_Test'
        // $directories[5] = '0003_63da7bb4b0589621675295.pdf'

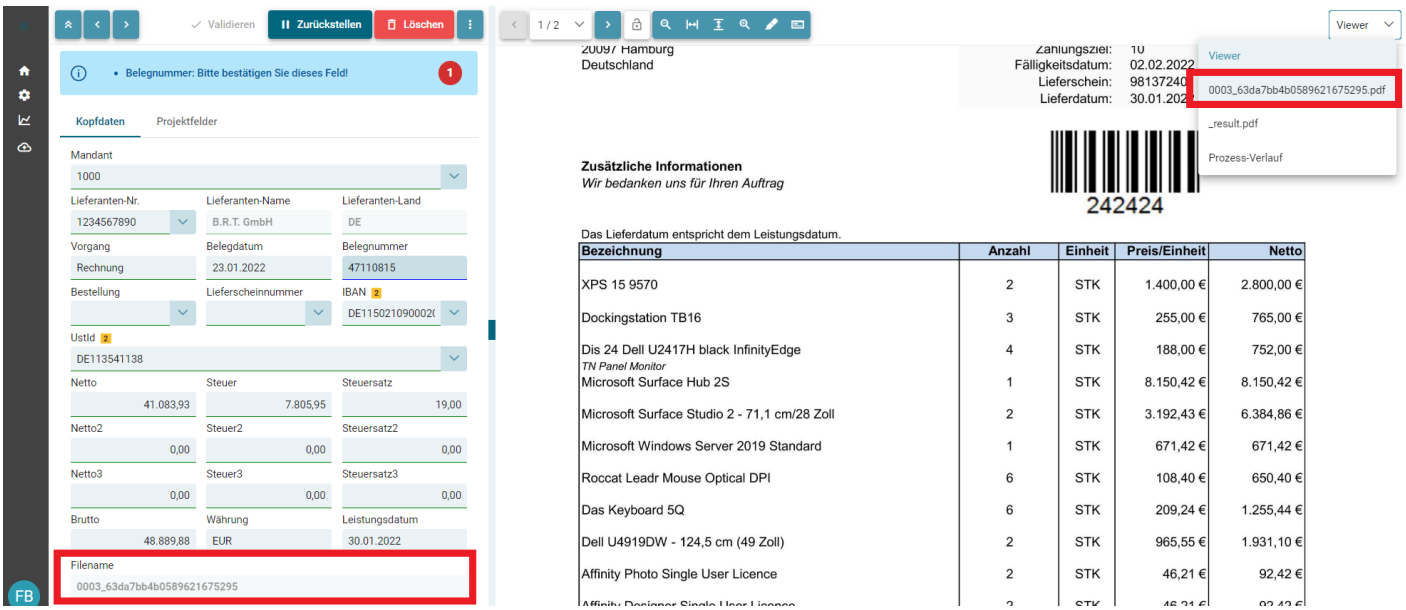
        $companyDir = $directories[4]; // 1000_Test
        $companyId = substr($companyDir, 0, 4);  // 1000
        $params['additionalInfo']['Company'] = $companyId;

    } catch (Throwable $e) {
```

```
            xTools::handleException($e, false, true);
        }

        return new xReturnObject(true, 200, 'File ready for import!', $params);
    }
```

Der Dateiname wird anschließend in dem entsprechenden Feld angezeigt.



Barcodetrennung

Barcodetrennung

In folgendem Ordner der Squeeze Installation:
SQUEEZE\htdocs\repository\<Mandantenname>\UserExits\Global\

muss die PHP-Datei BarcodeSeparation.php liegen.

Unter Barcode Types sind die möglichen Barcode-Typen aufgeführt die verwendet werden können.

In diesem Beispiel: CODE_128

Darüber hinaus kann der zu berücksichtigende Barcode-Wert weiter eingeschränkt werden:

In diesem Beispiel muss der Barcode 12 Stellen haben und die ersten 4 Stellen müssen "DTER" entsprechen.

```
<?php

use Squeeze\xDataBaseClass;
use Squeeze\xDoc;
use Squeeze\xReturnObject;

/**
 * @param xDoc $xDoc
 * @return xReturnObject
 * @throws Exception
 */
function BarcodeSeparation(xDoc $xDoc, string $nextStep, string $newStatus): xReturnObject
{
    $logger = Logger::getLogger("main");
    $db = xDataBaseClass::getDBConnection();

    $logger->debug('BatchClassId    = ' . $xDoc->batchClassId);
    $logger->debug('DocumentClassId = ' . $xDoc->docClassId);

    if ($xDoc->batchClassId == '1') {
        $result = splitByBarcode($db, $xDoc, $logger);
    }
```

```
    return $result;
}

/**
 * @param PDO $db
 * @param xDoc $xDoc
 * @param Logger $logger
 * @return xReturnObject
 * @throws Exception
 */
function splitByBarcode(PDO $db, xDoc $xDoc, Logger $logger): xReturnObject
{
    // ========================================
    // Barcode Types
    // ========================================
    //  AZTEC
    //  CODABAR
    //  CODE_39
    //  CODE_93
    //  CODE_128
    //  COMPOSITE
    //  DATABAR
    //  DATA_MATRIX
    //  DATABAR_EXP
    //  EAN_2
    //  EAN_5
    //  EAN_8
    //  EAN_13
    //  ITF
    //  ISBN_10
    //  ISBN_13
    //  MAXICODE
    //  PDF_417
    //  QR_CODE
    //  RSS_14
    //  RSS_EXPANDED
    //  UPC_A
    //  UPC_E
    //  UPC_EAN_EXTENSION
```

```
    //  UNKNOWN

    $currentSplit = 0;
    $splits = [];

    // Define which field values to keep
    if ($xDoc->getDocumentFieldByName('ScanUser') !== null) {
        $keepFieldValues[] = ['name' => 'ScanUser', 'value' => $xDoc-
>getDocumentFieldByName('ScanUser')->getValue()->value];
    }
    if ($xDoc->getDocumentFieldByName('ScanUser') !== null) {
        $keepFieldValues[] = ['name' => 'DocumentType', 'value' => $xDoc-
>getDocumentFieldByName('DocumentType')->getValue()->value];
    }

    foreach ($xDoc->barcodes as $page => $barcodes) {
        $barcodeType = '';
        $barcodeValue = '';
        $suppressOCR = 0;
        $logger->debug('Checking Barcodes for Page ' . $page);

        if (count($barcodes) == 0) {
            $logger->debug('No Barcode on Page ' . $page);
            if ($page == 1) {
                $currentSplit = $page;
            }
        } else {
            foreach ($barcodes as $key => $barcode) {

                // Suppress OCR if Zero Barcode was found
                //if($barcode['type'] == 'ITF' && $barcode['value'] == '999999'){
                //    $suppressOCR = 1;
                //}

                //$logger->debug($barcode['type']);
                //$logger->debug(substr($barcode['value'], 0, 3));
□□□□
                if ($barcode['type'] == 'CODE_128' && mb_strlen($barcode['value']) == 12 &&
substr($barcode['value'], 0, 4) == 'DTER') {
                    $logger->debug('Start Split here! Barcode (' . $barcode['value'] . ') of
```

```
type (' . $barcode['type'] . ') found on page ' . $page);
                    $currentSplit = $page;
                    $barcodeType = $barcode['type'];
                    $barcodeValue = $barcode['value'];
                }
□□□□//elseif($barcode['type'] == 'ITF' && mb_strlen($barcode['value']) == 8 &&
substr($barcode['value'], 0, 1) == '4'){
                //    $logger->debug('Start Split here! Barcode (' . $barcode['value'] . ') of
type (' . $barcode['type'] . ') found on page '. $page);
                //    $currentSplit = $page;
                //    $barcodeType = $barcode['type'];
                //    $barcodeValue = $barcode['value'];
                //}
            }
        }

        // add page
        $splits[$currentSplit][] = ['page' => $page, 'type' => $barcodeType, 'value' =>
$barcodeValue, 'fields' => $keepFieldValues, 'suppressOcr' => $suppressOCR];
    }

    return new xReturnObject(true, 200, 'Barcode Split', $splits);
}
```

Ob die Trennung funktioniert, sieht man wenn in der Bildaufbereitung aus einem Dokument mehrere werden und im Squeeze.log.

Die oben aufgeführten Barcode Typen sind auch die möglichen Werte für Attachment Barcode Typen an der Dokumentenklasse. Wichtig: Der Wert für den Barcode Typ ist case-sensitiv (Groß-Kleinschreibung beachten!).

Ob die Erkennung des Attachment Barcodes funktioniert hat, sieht man daran das ab diesem Barcode kein Text mehr im Viewer markiert werden kann.

Barcodetrennung mit ausschließen der Trenner-Seite

In folgendem Ordner der Squeeze Installation:
SQUEEZE\htdocs\repository\<Mandantenname>\UserExits\Global\

muss die PHP-Datei BarcodeSeparation.php liegen.

Unter Barcode Types sind die möglichen Barcode-Typen aufgeführt welche verwendet werden können. Diese können dann über die Stapelklasseneigenschaften konfiguriert werden.

Wichtig: Der Wert für den Barcode Typ ist case-sensitiv (Groß-Kleinschreibung beachten!).

```
<?php

use Squeeze\xBatchClass;
use Squeeze\xDoc;
use Squeeze\xQueueEntry;
use Squeeze\xReturnObject;
use Squeeze\xTools;
use App\V2\Base\Config\TenantConfig;
use App\V2\Base\DI\Dependencies;

/**
 * @param xDoc $xDoc
 * @param string $nextStep
 * @param string $newStatus
 * @return xReturnObject
 * @throws Exception
 */
function BarcodeSeparation(xDoc $xDoc, string $nextStep, string $newStatus): xReturnObject
{
    $logger = Logger::getLogger("main");

    // ============================================
```

```
    // Get Barcode Engine from Batch Class Settings
    // ============================================
    $isBarcodeConfigured = true;
    $xQueueEntry = new xQueueEntry();
    $xQueueEntry->getByxDocId($xDoc->id);
    $batchClass = new xBatchClass();
    $batchClass->getById($xQueueEntry->batchclassid);
    $splitBarcodeType = strtolower(trim($batchClass->getSettingValue('SplitBarcodeType')));
    $splitBarcodePattern = $batchClass->getSettingValue('SplitBarcodePattern');
    $splitFixPages = intval(trim($batchClass->getSettingValue('SplitFixPages')));

    if ($splitBarcodeType === null or $splitBarcodeType == '') {
        $isBarcodeConfigured = false;
        //$logger->warn('DocId = '.$xDoc->id.' Barcode Split not possible because
SplitBarcodeType is missing!');
    } else {
        if ($splitBarcodePattern === null or $splitBarcodePattern == '') {
            $isBarcodeConfigured = false;
            //$logger->warn('DocId = '.$xDoc->id.' Barcode Split not possible because
SplitBarcodePattern is missing!');
        }
    }

    if ($isBarcodeConfigured) {
        // get splits by barcode and return the result
        return splitByBarcode($xDoc, $logger, $splitBarcodeType, $splitBarcodePattern);
    }

    if ($splitFixPages > 0) {
        // get splits by fixed pages and return the result
        return splitFixPages($xDoc, $logger, $splitFixPages);
    }

    // return an empty split result
    return new xReturnObject(false, 400, 'No Split configured', []);
}

/**
 * @param xDoc $xDoc
 * @param Logger $logger
```

```
 * @param int $fixedPageSize
 * @return xReturnObject
 * @throws Exception
 */
function splitFixPages(xDoc $xDoc, Logger $logger, int $fixedPageSize): xReturnObject
{
    $config = Dependencies::getContainer()->get(TenantConfig::class);
    $repo = $config->get("repository.work");
    $splits = [];
    $keepFieldValues = [];
    $absolutePath = xTools::buildAbsolutePath($repo, $xDoc->repoPath, false);
    $files = array_diff(scandir($absolutePath . "Viewer"), ['.', '..']);
    $currentPage = 0;
    foreach ($files as $page => $file) {
        $currentPage++;
        $modulo = $currentPage % $fixedPageSize;
        if ($modulo === 0) {
            $logger->debug('DocId = ' . $xDoc->id . ' New split at page ' . $currentPage . '
because of fixed page size');
            $splits[$currentPage][] = ['page' => $currentPage, 'type' => 'FixedPageSplit',
'value' => 'FixedPageSplit-' . $fixedPageSize, 'fields' => $keepFieldValues, 'suppressOcr' =>
false];
        }
    }
    return new xReturnObject(true, 200, 'Fixed Page Split', $splits);
}

/**
 * @param xDoc $xDoc
 * @param Logger $logger
 * @param string $splitBarcodeType
 * @param string $splitBarcodePattern
 * @return xReturnObject
 */
function splitByBarcode(xDoc $xDoc, Logger $logger, string $splitBarcodeType, string
$splitBarcodePattern): xReturnObject
{

    // ========================================
    // Barcode Types
```

```
// ========================================
//  AZTEC
//  CODABAR
//  CODE_39
//  CODE_93
//  CODE_128
//  COMPOSITE
//  DATABAR
//  DATA_MATRIX
//  DATABAR_EXP
//  EAN_2
//  EAN_5
//  EAN_8
//  EAN_13
//  ITF
//  ISBN_10
//  ISBN_13
//  MAXICODE
//  PDF_417
//  QR_CODE
//  RSS_14
//  RSS_EXPANDED
//  UPC_A
//  UPC_E
//  UPC_EAN_EXTENSION
//  UNKNOWN

$currentSplit = 0;
$suppressOCR = 0;
$splits = [];
$keepFieldValues = [];

$splitBarcodeType = str_replace(' ', '', $splitBarcodeType);
$splitBarcodeType = trim($splitBarcodeType, ',;');
$arrSplitBarcodeTypes = explode(';', $splitBarcodeType);

$logger->debug('DocId = ' . $xDoc->id . ' Execute Barcode-Split');

foreach ($xDoc->barcodes as $page => $barcodes) {
```

```
        $logger->debug('DocId = ' . $xDoc->id . ' Checking Barcodes for Page ' . $page);
        $barcodeType = '';
        $barcodeValue = '';
□□$isSplitPage = false;
□□
        if (count($barcodes) == 0) {
            if ($page == 1) {
                $logger->debug('DocId = '.$xDoc->id.' No Barcode on Page '. $page . '. Split
because it is the first page');
                $currentSplit = $page;
            }
        } else {
            foreach ($barcodes as $barcode) {
                foreach ($arrSplitBarcodeTypes as $splitBarcodeType) {
                    if (strtolower($barcode['type']) === strtolower($splitBarcodeType)) {
                        $matches = null;
                        xTools::mb_preg_match_all("/$splitBarcodePattern/i",
$barcode['value'], $matches, PREG_OFFSET_CAPTURE);
                        $hit = $matches[0];
                        foreach ($hit as $match) {
                            if ($match[0] == '') {
                                continue;
                            }
                            $logger->debug('DocId = ' . $xDoc->id . ' Start Split here!
Barcode "' . $matchmatch[0] . '" of type "' . $barcode['type'] . '" found on page ' . $page);
                            //$suppressOCR = 1;
                            $currentSplit = $page - 1;
                            $barcodeType = $barcode['type'];
                            $barcodeValue = $barcode['value'];
□□□□□□□□$isSplitPage = true;

                            // Next Page
                            break 3;
                        }
                    }
                }
            }
        }
        // add page
□□if ($isSplitPage) {
```

```
□□□$logger->debug("IS SPLIT PAGE - DO NOT ADD PAGE $page");
□□} else {
□□□$splits[$currentSplit][] = ['page' => $page, 'type' =>$barcodeType, 'value' => $barcodeValue
□□}
    }

    return new xReturnObject(true, 200, 'Barcode Split', $splits);
}
```

Ob die Trennung funktioniert, sieht man wenn in der Bildaufbereitung aus einem Dokument mehrere werden und im Squeeze.log.

Ob die Erkennung des Attachment Barcodes funktioniert hat, sieht man daran das ab diesem Barcode kein Text mehr im Viewer markiert werden kann.

Klassifikation mittels Barcodetrennung

> Ab SQUEEZE Version 2.5.3 möglich

Um ein Dokument in SQUEEZE mittels Barcodes zu trennen und die Splits direkt anhand des Barcodes zu klassifizieren kann der folgende UserExit verwendet werden.

Voraussetzungen

An der Stapelklasse müssen für den Barcode Split die Stapelklasseneigenschaften

- `SplitBarcodeType`
- `SplitBarcodePattern`

gesetzt werden.

Funktionsweise

Die Splits werden durch den gefundenen Barcode des entsprechenden Typs identifiziert.

Dabei kann die Seite, welche den Split identifiziert (also auf dem der Barcode gefunden wurde) selbst weggelassen werden.
Dafür wird das Flag `isSplitPage` genutzt.

```
// dismiss split pages
if ($isSplitPage) {
$logger->debug(sprintf("Barcode split page %s will be skipped", $page), $logContext);
} else {
  $splits[$currentSplit][] = [
    'page' => $page,
    'type' => $barcodeType,
    'value' => $barcodeValue,
    'fields' => $fieldValuesToKeep,
    'suppressOcr' => $suppressOCR,
    "documentClassId" => $documentClassId
  ];
```

```
}
```

Alternativ kann die Seite, welche den Split identifiziert (also auf dem der Barcode gefunden wurde), dem Split hinzugefügt werden.
Dafür einfach das Array weiter befüllen und das `isSplitPage` Flag ignorieren.

```
// include split pages
$splits[$currentSplit][] = [
    'page' => $page,
    'type' => $barcodeType,
    'value' => $barcodeValue,
    'fields' => $fieldValuesToKeep,
    'suppressOcr' => $suppressOCR,
    "documentClassId" => $documentClassId
];
```

Die Dokumentenklassen ID kann dem Split mit dem Attribut `documentClassId` zugewiesen werden.

> Pro Split wird für das Attribut `documentClassId` der ersten Seite des Splits (erstes Array Element des Splits) ausgewertet.

> Es dürfen nur valide Dokumentenklassen der Stapelklassen gesetzt werden! Invalide Dokumentenklassen werden ignoriert.

Beispielhafter UserExit

In dem folgenden Beispiel werden

- Barcode Pages entfernt
- Die Dokumentenklasse des Splits anhand des Barcodewertes ermittelt, welcher den Namen der Dokumentenklasse enthält.

```
/**
 * Retrieves the document class ID by name.
 *
 * @param string $name The name of the document class.
 * @return int The ID of the document class.
 */
function getDocumentClassIdByName(string $name): int
{
    $documentClass = new xDocumentClass();
```

```
    $documentClass->getByName($name);
    if (is_null($documentClass->getId())) {
        return 0;
    }
    return (int)$documentClass->getId();
}
```

```
if (isMatchingBarcode($splitBarcodeType, $splitBarcodePattern, $barcode["value"],
$barcode["type"])) {
  // ...
  $documentClassId = getDocumentClassIdByName($barcodeValue);
  // ...
}
```

Code

```
<?php

declare(strict_types=1);

namespace Squeeze\UserExits\Common;

use App\V2\Base\Config\ServerConfig;
use App\V2\Base\DI\Dependencies;
use App\V2\Base\Logging\Logging;
use Exception;
use Psr\Log\LoggerInterface;
use Squeeze\xBatchClass;
use Squeeze\xDoc;
use Squeeze\xDocumentClass;
use Squeeze\xQueueEntry;
use Squeeze\xReturnObject;
use Squeeze\xTools;

/**
 * @param xDoc $xDoc
 * @param string $nextStep
 * @param string $newStatus
 * @return xReturnObject
```

```
 * @throws Exception
 */
function BarcodeSeparation(xDoc $xDoc, string $nextStep, string $newStatus): xReturnObject
{
    $logger = Logging::get();

    $queueEntry = new xQueueEntry();
    $queueEntry->getByxDocId($xDoc->getId());
    $batchClass = new xBatchClass();
    $batchClass->getById($queueEntry->getBatchClassId());
    // Get barcode engine from batch class settings
    $splitBarcodeType = $batchClass->getSettingValue('SplitBarcodeType') ?? "";
    $splitBarcodePattern = $batchClass->getSettingValue('SplitBarcodePattern') ?? "";

    if (!empty($splitBarcodePattern) && !empty($splitBarcodeType)) {
        return splitByBarcode($xDoc, $logger, $splitBarcodeType, $splitBarcodePattern);
    }

    $splitFixPages = (int)($batchClass->getSettingValue('SplitFixPages') ?? 0);
    if ($splitFixPages > 0) {
        return splitFixPages($xDoc, $logger, $splitFixPages);
    }

    return new xReturnObject(false, 400, 'No Split configured', []);
}

/**
 * Splits the given xDoc into fixed pages and returns an xReturnObject containing the splits.
 *
 * @param xDoc $xDoc The xDoc to be split.
 * @param LoggerInterface $logger The logger instance.
 * @param int $fixedPageSize The size of each fixed page.
 * @throws Exception If an error occurs during the split process.
 * @return xReturnObject The xReturnObject containing the splits.
 */
function splitFixPages(xDoc $xDoc, LoggerInterface $logger, int $fixedPageSize): xReturnObject
{
    // Define which field values to keep
    $fieldNamesToKeep = [];
    $fieldValuesToKeep = getFieldValuesToKeep($xDoc, $fieldNamesToKeep);
```

```
    // initialize
    $logContext = [Logging::CTX_DOCUMENT_ID => $xDoc->getId()];
    $container = Dependencies::get();
    $config = $container->get(ServerConfig::class);
    $repo = $config->get("repository.work");

    $splits = [];
    $absolutePath = xTools::buildAbsolutePath($repo, $xDoc->getRepoPath(), false);
    $files = array_diff(scandir($absolutePath . "Viewer"), ['.', '..']);
    $currentPage = 0;
    foreach ($files as $page => $file){
        $currentPage++;
        if ($currentPage % $fixedPageSize === 0) {
            $logger->debug(sprintf('Splitting page %s due to fixed page size %s',
$currentPage, $fixedPageSize), $logContext);
            $splits[$currentPage][] = [
                'page' => $currentPage,
                'type' => 'FixedPageSplit',
                'value' => 'FixedPageSplit-' . $fixedPageSize,
                'fields' => $fieldValuesToKeep,
                'suppressOcr' => false
            ];
        }
    }
    return new xReturnObject(true,200, 'Fixed Page Split', $splits);
}

/**
 * Splits the given xDoc into multiple parts based on barcode information.
 *
 * @param xDoc $xDoc The xDoc object to be split.
 * @param LoggerInterface $logger The logger object for logging debug information.
 * @param string $splitBarcodeType The types of barcodes to split on, separated by semicolons.
 * @param string $splitBarcodePattern The pattern to match barcodes against.
 * @return xReturnObject The xReturnObject containing the split parts of the xDoc object.
 */
function splitByBarcode(xDoc $xDoc, LoggerInterface $logger, string $splitBarcodeType, string
$splitBarcodePattern): xReturnObject
{
```

```
    // Define which field values to keep
    $fieldNamesToKeep = [];
    $fieldValuesToKeep = getFieldValuesToKeep($xDoc, $fieldNamesToKeep);

    // initialize
    $logContext = [Logging::CTX_DOCUMENT_ID => $xDoc->getId()];
    $currentSplit = 0;
    $splits = [];
    $documentClassId = 0;
    $barcodeType = '';
    $barcodeValue = '';

    foreach ($xDoc->getBarcodes() as $page => $barcodes) {
        $isSplitPage = false;
        $suppressOCR = 0; // used to suppress OCR at desired page
        $logger->debug(sprintf('Checking barcodes for page %s', $page), $logContext);

        if (count($barcodes) == 0) {
            $logger->debug(sprintf('No barcodes found on page %s', $page), $logContext);
            if ($page == 1) {
                $currentSplit = $page;
                $logger->debug(sprintf('Start split at page %s', $page), $logContext);
            }
        } else {
            foreach ($barcodes as $barcode) {
                if (isMatchingBarcode($splitBarcodeType, $splitBarcodePattern,
$barcode["value"], $barcode["type"])) {
                    $currentSplit = $page - 1;
                    $barcodeType = $barcode['type'] ?? '';
                    $barcodeValue = $barcode['value'] ?? '';
                    $documentClassId = getDocumentClassIdByName($barcodeValue);
                    $isSplitPage = true;
                    $logger->debug(
                        sprintf(
                            'Split at page %s - Barcode "%s" of type "%s" found - Use document
class id %s',
                            $page,
                            $barcode["value"],
                            $barcode["type"],
```

```
                            $documentClassId
                        ),
                        $logContext
                    );
                    break;
                }
            }
        }

        // dismiss split pages
        if ($isSplitPage) {
            $logger->debug(sprintf("Barcode split page %s will be skipped", $page),
$logContext);
        } else {
            $splits[$currentSplit][] = [
                'page' => $page,
                'type' => $barcodeType,
                'value' => $barcodeValue,
                'fields' => $fieldValuesToKeep,
                'suppressOcr' => $suppressOCR,
                "documentClassId" => $documentClassId
            ];
        }
    }

    return new xReturnObject(true, 200, 'Barcode Split', $splits);
}

/**
 * Retrieves the values of the specified fields from the given xDoc object.
 *
 * @param xDoc $xDoc The xDoc from which to retrieve the field values.
 * @param array $fieldNames An array of field names to retrieve the values for.
 * @return array An array of associative arrays containing the field names and values.
 */
function getFieldValuesToKeep(xDoc $xDoc, array $fieldNames): array
{
    $fieldValues = [];
    foreach ($fieldNames as $fieldName) {
        if ($xDoc->getDocumentFieldByName($fieldName) !== null) {
```

```
            $fieldValue = $xDoc->getDocumentFieldByName($fieldName)->getValue();
            if (!is_null($fieldValue)) {
                $fieldValues[] = ['name' => $fieldName, 'value' => $fieldValue->getValue()];
            }
        }
    }

    return $fieldValues;
}

/**
 * Retrieves the document class ID by name.
 *
 * @param string $name The name of the document class.
 * @return int The ID of the document class.
 */
function getDocumentClassIdByName(string $name): int
{
    $documentClass = new xDocumentClass();
    $documentClass->getByName($name);
    if (is_null($documentClass->getId())) {
        return 0;
    }
    return (int)$documentClass->getId();
}

/**
 * Checks if a given barcode matches the specified barcode type and pattern.
 *
 * @param string $splitBarcodeType The type of the barcode to match.
 * @param string $splitBarcodePattern The pattern to match against the barcode value.
 * @param string $barcodeValue The value of the barcode to match.
 * @param string $barcodeType The type of the barcode to match against.
 * @return bool Returns true if the barcode matches the specified type and pattern, false
otherwise.
 */
function isMatchingBarcode(string $splitBarcodeType, string $splitBarcodePattern, string
$barcodeValue, string $barcodeType): bool
{
    if (strtolower($splitBarcodeType) === strtolower($barcodeType)) {
```

```
        $matches = null;
        xTools::mb_preg_match_all("/$splitBarcodePattern/i", $barcodeValue, $matches,
PREG_OFFSET_CAPTURE);
        $hit = $matches[0];
        foreach ($hit as $match) {
            if ($match[0] == '') {
                continue;
            }
            return true;
        }
    }
    return false;
}
```

AfterLocatorExtraction

Nach der Extraktion eines jeden Lokators kann das Ergebnis noch angepasst werden, befor das Dokumentenergebnis erstellt wird. Hier einige Beispiele.

Allgemeine Informationen

Jeder Lokator prüft nach der Erkennung, ob ein UserExit für die Modifikation des Ergebnisses vorhanden ist. Grundsätzlich ist dabei folgender Aufbau der UserExits zu beachten:

Jeder Lokator bietet die Möglichkeit einen UserExit einzubinden. Damit dieser UserExit angesprochen werden kann, müssen folgende Punkte beachtet werden:

UserExit Verzeichnis

Das UserExit Verzeichnis für die Nutzung ist immer `\htdocs\repository\client.server.net\UserExits\Documentenklassenname\AfterLocatorExtraction_Lokatorname.php`

Ein Beispiel für einen Währungslokator wäre also z.B.: `\htdocs\repository\localhost\UserExits\Invoices\AfterLocatorExtraction_Currency.php`

AfterExtraction

Der UserExit AfterExtraction wird immer nach der Extraktion eines Dokumentes ausgeführt und kann genutzt werden um das Ergebnis zu beeinflussen, bevor das Dokument in die Validierung übergeben wird.

AfterExtraction

Eine Position je Bestellnummer

Gerade bei SAP Projekten kann es erforderlich sein, je Bestellnummer eine Position erzeugen zu müssen. In diesem Fall kann der folgende UserExit genutzt werden, um nach der Extraktion des Beleges eine Position je gefundener Bestellnummer zu erzeugen.

Voraussetzungen dieses UserExits:

- Das Kopffeld für Bestellnummern heißt **"OrderNumber"**.
- Es existiert ein Feld in der Positionstabelle mit dem Namen **"PosOrderNumber"**.
- Das Feld für die Positionen heiß **"LineItems"**.

```
<?php

use Squeeze\xDoc;
use Squeeze\xDocumentField;

function AfterExtraction(xDoc $xDoc)
{

    $logger = Logger::getLogger("main");
    createLineItemsByHeadField($xDoc, $logger, "OrderNumber", "PosOrderNumber", "LineItems");

    return true;
}

/**
* @param xDoc $xDoc
* @param Logger $logger
* @param string $originHeadField
* @param string $targetLineField
* @param string $lineItemName
*/
function createLineItemsByHeadField(
    xDoc $xDoc,
    Logger $logger,
    string $originHeadField,
```

```
    string $targetLineField,
    string $lineItemName
): void {
    $head = $xDoc->getDocumentFieldByName($originHeadField);
    $lines = $xDoc->getDocumentFieldByName($lineItemName);

    $head->alternatives = getUniqueAlternatives($head, $logger);
    if (count($head->alternatives) > 1) {
        $head->getValue()->value = '';
        $head->getValue()->text = '';
        foreach ($head->alternatives as $alternative) {
            $line = new Squeeze\xDocValue(0, "", "", "string", 100, 0, 99999, 99999, 0, 0);
            $line->subfields[$targetLineField] = $alternative;
            $line->subfields[$targetLineField]->subFieldName = $targetLineField;
            $lines->alternatives[] = $line;
        }
    }
}

/**
* @param xDocumentField $field
* @param Logger $logger
* @return xDocumentField[]
*/
function getUniqueAlternatives(xDocumentField $field, Logger $logger): array
{
    // ============================================================
    // Only keep unique alternatives
    // ============================================================
    $tmpStore = [];
    $uniqueAlternatives = [];
    $logger->debug('The field ' . $field->name . ' has ' . count($field->alternatives) . '
alternatives');
    foreach ($field->alternatives as $alternative) {
        $logger->debug('Alternative = ' . $alternative->value);
        if (!in_array($alternative->value, $tmpStore)) {
            $logger->debug('Alternative will be kept ' . $alternative->value);
            $tmpStore[] = $alternative->value;
            $uniqueAlternatives[] = $alternative;
        }
```

```
    }

    if (count($uniqueAlternatives) > 0) {
        $field->alternatives = $uniqueAlternatives;
    }

    return $field->alternatives;
}
```

AfterExtraction

Übergabe jeder Bestellnummer Alternative als kommaseparierte Liste

Um jede Bestellnummer Alternative in einer kommaseparierten Liste als Rechnungskopffeld zu übergeben folgende Erweiterungen hinzufügen:

```
function allOrderNumbersInHeadField(xDoc $xDoc, Logger $logger, string $originHeadField): void
{
    $head = $xDoc->getDocumentFieldByName($originHeadField);

    $head->alternatives = getUniqueAlternatives($head, $logger);
    if (count($head->alternatives) > 1) {
        $head->getValue()->value = '';
        $head->getValue()->text = '';
        $strOrdernr = '';
        foreach ($head->alternatives as $alternative) {
            $strOrdernr .= ',' . $alternative->value;
        }
        $strOrdernr = ltrim($strOrdernr, ",");
        $head->getValue()->value = $strOrdernr;
        $head->getValue()->text = $strOrdernr;
    }
}

/**
* @param xDocumentField $field
* @param Logger $logger
* @return xDocumentField[]
*/
function getUniqueAlternatives(xDocumentField $field, Logger e$logger): array
{
    // ============================================================
    // Only keep unique alternatives
```

```
    // ============================================================
    $tmpStore = [];
    $uniqueAlternatives = [];
    $logger->debug('The field ' . $field->name . ' has ' . count($field->alternatives) . '
alternatives');
    foreach ($head->alternatives as $alternative) {
        $logger->debug('Alternative = ' . $alternative->value);
        if (!in_array($alternative->value, $tmpStore)) {
            $logger->debug('Alternative will be kept ' . $alternative->value);
            $tmpStore[] = $alternative->value;
            $uniqueAlternatives[] = $alternative;
        }
    }

    if (count($uniqueAlternatives) > 0) {
        $field->alternatives = $uniqueAlternatives;
    }

    return $field->alternatives;
}
```

Aufruf der Funktion per:

```
allOrderNumbersInHeadField($xDoc, $logger, "OrderNumber");
```

AfterValidation

AfterValidation

Versenden einer Email nach der Validierung

Es ist möglich nach der Validierung eine Email zu versenden.
Hier ein Beispiel wie dieser Versand realisiert werden kann:

```
<?php

// =======================================================
// UserExit after Validation
// =======================================================

use Squeeze\xDoc;
use Squeeze\xTools;
use App\V2\Base\Config\TenantConfig;
use App\V2\Base\DI\Dependencies;

function AfterValidation(xDoc $xDoc, string $nextStep, string $newStatus): ?array
{
    // Optional ein Logger Objekt erzeugen
    $logger = Logger::getLogger("main");

    // Initialisierung der Emailparameter
    $subject = 'Test after Validation'; // Betreff
    $to = ['test(at)test.de'];     // 1 - n Empfänger

    // Definition des Emailinhalts
    // HTML formatierte Emails sind ebenfalls möglich aber komplexer.
    $body = 'Hier der Nachrichtentext.';

    // Absoluten Pfad zum Dokument ermitteln
    // ggf. eine Exception erzeugen, wenn der Pfad nicht existiert
    $config = Dependencies::getContainer()->get(TenantConfig::class);
    $repo = $config->get('repository.root');
    $absolutePath = xTools::buildAbsolutePath($repo, $xDoc->repoPath, false);
```

```
    if ($absolutePath === false) {
        $msg = "AfterValidation Input Folder (" . $repo . $xDoc->repoPath . ") does not
exist.";
        $logger->error($msg);
        throw new Exception($msg);
    }

    // Initalisierung eines Arrays zur Speicherung der Anlagen
    $attachments = [];

    // Laden der PDFs und dem Array der Anlagen hinzufügen
    $files = $xDoc->getBase64Files($absolutePath, 'Input', ['pdf']);
    foreach ($files->getResult() as $file) {
        if ($file->filename != '_result.pdf') {
            $attachments[] = $absolutePath . 'Input' . DIRECTORY_SEPARATOR . $file->filename;
        }
    }

    // Email senden und auf Fehler prüfen
    $result = xTools::sendEmail($subject, $to, $body, 'text', $attachments);
    if (!$result->isSuccess()) {
        throw new Exception($result->getMessage());
    }

    return ['xDoc' => $xDoc, 'nextStep' => $nextStep, 'newStatus' => $newStatus];
}
```

Export UserExits

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);
}
```

ValidateDocument

ValidateDocument

Pflichtfeldprüfung - Entweder Oder

Hin und wieder reicht die einfache Pflichtfeldprüfung nicht aus z.B. wenn eines von zwei Feldern gefüllt sein muss. Für diesen Fall kann unten angegebene Funktion im UserExit ValidateDocument aufgerufen werden.

Die letzten beiden Parameter der Funktion müssen den technischen Feldnamen der zu prüfenden Feldern entsprechen.
Beispiel:

```
checkEitherOrField($db, $fieldList, $logger, 'VatId', 'TaxId');
```

Hier die Funktion, die als Entweder-Order Prüfung genutzt werden kann:

```
function checkEitherOrField(PDO $db, xDocumentFieldCollection $fieldList, Logger $logger,
string $firstFieldName, string $secondFieldName){

$firstField = $fieldList->getDocumentFieldByName($firstFieldName);
$secondField = $fieldList->getDocumentFieldByName($secondFieldName);

if ($firstField == null) {
        $logger->error("Field with name '".$firstFieldName."' does not exist");
return false;
    }

if ($secondField == null) {
        $logger->error("Field with name '".$secondFieldName."' does not exist");
return false;
    }

if ($firstField->getValue()->value == "" and $secondField->getValue()->value == "") {
$firstField->state = "ERROR";
        $firstField->errortext = __('The field %s may not be empty!', $firstField-
>description);
        $fieldList->updateField($firstField);
```

```
□□
□□$secondField->state = "ERROR";
        $secondField->errortext = __('The field %s may not be empty!', $secondField-
>description);
        $fieldList->updateField($secondField);
□} else {
□□$firstField->state = "";
        $firstField->errortext = __("The field is valid!");
        $fieldList->updateField($firstField);
□□
□□$secondField->state = "";
        $secondField->errortext = __("The field is valid!");
        $fieldList->updateField($secondField);
□}
}
```

BeforeExport

BeforeExport

Text im PDF andrucken

Um z.B. den Barcode-Wert auf dem PDF anzudrucken, kann die der UserExit BeforeExport wie folgt angepasst werden:

Voraussetzung:
Squeeze Version: >=1.11.0

```
<?php

use Squeeze\ImageTools;
use Squeeze\xDoc;
use Squeeze\xDocFolders;
use Squeeze\xReturnObject;
use Squeeze\xTools;
use App\V2\Base\Config\TenantConfig;
use App\V2\Base\DI\Dependencies;

/**
 * @param xDoc $doc
 * @return xReturnObject
 */
function BeforeExport(xDoc $doc): xReturnObject
{
    try {
        $dirSep = DIRECTORY_SEPARATOR;
        $config = Dependencies::getContainer()->get(TenantConfig::class);
        $repo = $config->get('repository.root');
        $absolutePath = xTools::buildAbsolutePath($repo, $doc->repoPath, false);

        $inputDir = $absolutePath . xDocFolders::INPUT . $dirSep;
        $workDir = $absolutePath . xDocFolders::WORK . $dirSep;

        $printValue = $doc->getFieldByName('DocumentReference')->getValue()->value;

        $files = scandir($inputDir);
        foreach ($files as $file) {
```

```
            $extension = strtolower(pathinfo($inputDir . $file, PATHINFO_EXTENSION));
            if ($extension == "pdf") {

                //if($file == "_result.pdf") continue;

                $pdfInfo = ImageTools::getPdfInfo($inputDir . $file, 1, 1);
                $pdfWidth = $pdfInfo['pages'][1]['width'] / 300 * 25.4;
                $pdfHeight = $pdfInfo['pages'][1]['height'] / 300 * 25.4;
                $pdf = ImageTools::pdfPrintText($pdfWidth, $pdfHeight, $printValue, 1, 1,
'Courier', 10, 4, true);
                file_put_contents($workDir . 'textlayer.pdf', $pdf);

                $overlayResult = ImageTools::addLayerToPdf($inputDir . $file, $workDir .
'textlayer.pdf');
                if ($overlayResult !== true) {
                    unlink($workDir . 'textlayer.pdf');
                    throw new Exception('Error while adding a PDF Layer');
                }
                unlink($workDir . 'textlayer.pdf');
            }
        }

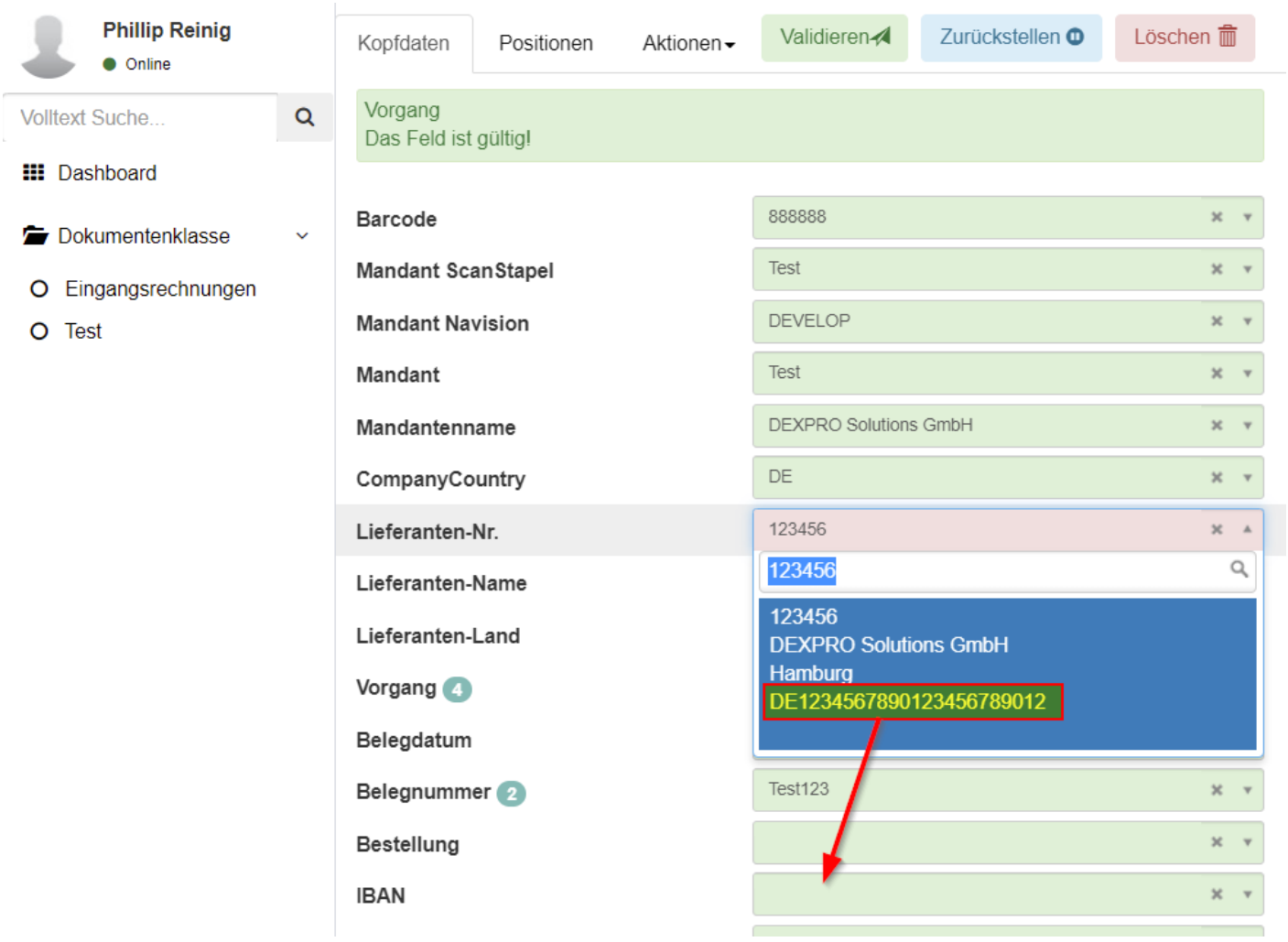
        return new xReturnObject(true, 200, 'UserExit BeforeExport successful', null);
    } catch (Exception $e) {
        $logger = Logger::getLogger("main");
        $logger->error($e->getMessage());
        return new xReturnObject(false, 500, $e->getMessage(), null);
    }
}
```

JS Client UserExits

Werteübernahme aus einer DropDown-Liste

Bei Feldern mit Eingabehilfe besteht die Möglichkeit die Info-Werte aus der Datenbank in andere Felder zu übertragen.
Das bedeutet, dass sobald der Benutzter einen Eintrag aus der DropDown Liste auswählt, kann ein beliebiger Wert aus dieser Liste in andere Kopffelder übertragen werden.

Hier ein Beispiel:



Um diese Funktion nutzen zu können muss ein JavaScript UserExit eingerichtet werden.
Die JavaScript UserExit Datei muss mit dem Namen `customValidation.js` unter folgendem Pfad gespeichert werden:
`.../repository/mandant.server.net/UserExits/DocClassName/js/customValidation.js`

In dieser Datei kann die folgende Funktion hinterlegt werden die in diesem Fall die IBAN aus dem DropDown in das IBAN Feld überträgt:

```
function customAfterSelectHeaderFieldOption(field, e){
    if(field.name === 'Creditor') {
□□var ibanData = {};
□□ibanData.id = "0";
□□ibanData.text = e.object.row.IBAN;
□□ibanData.value = e.object.row.IBAN;
□□$("#IBAN").select2('data', ibanData);
□□updateFieldValue("IBAN", ibanData);□□
    }
}
```

Aus dem Skriptbespiel wird ersichtlich, dass das Feld, welches aufgeklappt wurde, abgefragt werden kann (`Creditor`).
Die letzten beiden Zeilen sind für die Aktualisierung des Feldes und der Daten des Dokumentes verantwortlich.

Digitalen Barcode mit Squeeze erzeugen

Digitalen Barcode mit SQUEEZE erzeugen

Datenbank Counter anlegen

SQUEEZE Datenbank Tabelle "counters" nutzen.

Counter mit Namen und Initialwert in die Datenbank einfügen.

Counter hochzählen und auslesen

Mittels der xTools Funktion `incrementCounter("<COUNTER_NAME>", <VALUE_TO_ADD>);`

Barcode erstellen und Count einfügen

Der Barcode soll meist aus einer alphanumerischen Zeichenkette bestehen (wie etwa: BC2100001).

Dazu kann mittels der PHP Funktion `sprintf(<FORMAT>, <COUNTER_VALUE>);` der Barcode entsprechend zusammengesetzt werden.

In der Formatangabe `%05d` bedeutet die `5` dass bis zu 5 Nullen aufgefüllt werden und das `d`, dass eine Ganzzahl verwendet werden soll (PHP Referenz).

Beispiel

```
use Squeeze\xTools;


$counter = xTools::incrementCounter('DEXPRO_INVOICE', 1);
$barcode = sprintf('BC' . date('y') . '%05d', $counter);


// BARCODE: BC2100001 [BC + 21 + bis zu fünf Nullen aufgefüllt + Aktueller Counter]
```

Import Dateinamen verwenden

In diesem Beispiel enthalten die Dateinamen der importierten Datei bereits Indexinformationen, die wir in Kopffelder schreiben wollen.

Was passiert mit importierten Dokumenten?

Beim Import von Dokumenten können einzelne Dateien in mehrere resultieren.
Ein Beispiel dafür sind E-Mails die mehreren Anhängen, die alle als einzelne Eingangsdokumente behandelt werden sollen. Des weiteren werden auch XML-Anlagen innerhalb von PDFs oder direkt als Anhang für diese Funktionalität berücksichtigt.

Damit die Import Dateinamen sowie deren Zuordnung zur verarbeiteten Datei in SQUEEZE also nicht verloren gehen, werden diese Informationen gespeichert.
Zu jedem SQUEEZE Dateinamen gibt es dann also mindestens einen oder im Zweifel mehrere Import Dateinamen.

AdditionalInfo

Zu jedem SQUEEZE Vorgang (xDoc) gibt es `AdditionalInfo`, welche zusätzliche Informationen zu den Vorgängen und Dateien abspeichert.
Aktuell ist dies eine JSON Datei die am Vorgang im Repository innerhalb des "Work" Ordners abgespeichert ist.

Darin enthalten sind neben Pipeline Keys auch der/die Dateiname/n der importierten Datei/en.

FileImportMap

Die AdditionalInfo eines SQUEEZE Vorgangs (xDoc) enthält also auch eine `FileImportMap`.

Diese stellt ein Mapping zwischen den Dateinamen der importierten Dateien zu den jeweiligen SQUEEZE Dateinamen, die zur Bearbeitung / Extraktion verwendet werden.

Attachments behalten ihre originalen Dateinamen, der Zugriff zu ihnen ist über das Attachment Handling des SQUEEZE Vorgangs (xDoc) möglich und wird in diesem Beispiel nicht näher beschrieben.

Datenstruktur

In der `_additionalInfo.json` sieht das Mapping so aus:

Mit dem Zugriff auf ein Mapping (also einen SQUEEZE Dateinamen) erhält man ein Array mit den Import Dateinamen, aus welcher dann die SQUEEZE Datei geworden ist.

```
{
    "additionalInfo": {
        "IMPORT_FILENAMES": {
            "0002_64999a0020378394230462.xml": [
                "import.pdf"
            ],
            "64999a0020378394230462.pdf": [
                "another.pdf",
                "import.pdf"
            ]
        }
    }
}
```

In diesem Beispiel JSON gibt es also eine Datei namens "import.pdf" welche zusätzlich XML Content enthält und durch die Stapelklasseneigenschaft wurde "another.pdf" mit "import.pdf" zusammengefügt. Zu den SQUEEZE Dateinamen (hier die Schlüssel "0002_64999a0020378394230462.xml" und "64999a0020378394230462.pdf") werden die ursprünglichen Dateinamen gespeichert.

Die FileImportMap konzentriert sich nur auf den JSON Schlüssel "IMPORT_FILENAMES".

Das `->get()` der `FileImportMap` gibt also das Array mit den originalen Dateinamen der angegebenen SQUEEZE Datei zurück.

Beispiel

Zugriff im UserExit

Um auf die originalen Dateinamen beim Import zuzugreifen werden die `AdditionalInfo` benötigt, die wiederum die Zugriffsklasse `FileImportMap` enthält.

Mit dieser ist es einfach, zu allen SQUEEZE Dateinamen die jeweiligen Import Dateinamen zu erhalten.

Außerdem kann man zusätzlich Dateinamen einem SQUEEZE Dateinamen zuordnen.
Das `->set()` fügt den Import Dateinamen zum Mapping des übergebenen SQUEEZE Dateinamen hinzu oder legt dieses Mapping an, sollte es noch nicht existieren.

```
// get AdditionalInfo
$additionalInfo = $xDoc->loadAdditionalInfoTyped();
// get FileImportMap
$fileImportMap = $additionalInfo->getFileImportMap();

// get all mapped import file names
$allSqueezeDocumentsToImportDocuments = $fileImportMap->getAll();
foreach($allSqueezeDocumentsToImportDocuments as $squeezeFileName => $importDocuments) {
  foreach($importDocuments as $importFileName) {
    // do something
  }
}
// get import file names for squeeze file name
$importFileNamesOf = $fileImportMap->get($squeezeFileName);
foreach($importFileNamesOf as $importFileName) {
  // do something
}
// add import file name for squeeze file name
$fileImportMap->set($importFileName, $squeezeFileName);
```