# Portal-Skripte

Um Probleme bei Updates vorzubeugen wird der Großteil der Portal-Skripte wird verschlüsselt ausgeliefert. Nur so kann garantiert werden, dass die Skripte in den Projekten nicht angepasst werden. Durch die verschlüsselten Skripte können jedoch die Objekte und Funktionen nicht eingesehen werden. dies wird mit diesem Kapitel nachgeholt.

# SqlObject() für SQL-Statements

Oft müssen in den Projekten spezifische Datenbank-Abfragen zu Stammdaten erfolgen oder Daten sollen zu einem definierten Zeitpunkt in eine Datenbank-Tabelle geschrieben werden. Dies kann mit relativ viel Code über die Funktionen aus der Portal-Script API umgesetzt werden. Beim Verbindungsaufbau wird das Benutzer-Passwort häufig im Klartext hinterlegt und häufig werden bei Fehlern die Datenbank-Verbindungen nicht geschlossen.

Sowohl die Invoice- als auch die Mailroom-Lösung greifen häufig lesend und schreibend auf die ausgelieferten Datenbanken zu. Um einfache SQL-Befehle auszuführen wurde eine Bibliothek mit dem **SqlObject()** erstellt. Das Objekt wird initial mit 2 Parametern aufgerufen:

| | | | |
|---|---|---|---|
| **@param** | **{string}** | **sqlTable** | **Name der Datenbank-Tabelle** |
| **@param** | **{string}** | **dbConnName** | **Name der Datenbankverbindung aus der dbConn.json** |

Das Objekt enthält diverse property.

| | | | |
|---|---|---|---|
| **@property** | **{string}** | **Error** | **Fehlermeldung in der Kontext-Sprache des aktuellen Benutzers.** |
| **@property** | **{string}** | **Result** | **Wird initial auf boolean 'true' gesetzt und wird bei Fehlern auf 'false' gesetzt. Bei einem SELECT-Befehl enthält das Property das Ergebnis als Array.** |
| **@property** | **{string}** | **Log** | **Enthält die kompletten Log-Informationen.** |
| **@property** | **{string}** | **SQL** | **Speichert den zuletzt ausgeführten SQL-Befehl.** |

| | | | |
|---|---|---|---|
| @property | {string} | SQLWhere | WHERE-Bedingung für SELECT oder UPDATE-Statements. Das "WHERE" selber muss nicht angegeben werden. |
| @property | {boolean} | ExpectHits | Wenn beim SELECT Statement keine Treffer ermittelt wurden, kann dies optional als Fehler interpretiert werden. |
| @property | {boolean} | SelectDistinct | Beim SELECT Statement kann optional ein 'SELECT DISTINCT' ausgeführt werden. |
| @property | {int} | SelectTop | Bei SELECT-Befehlen kann die Trefferanzahl optional auf die gegebene Anzahl eingeschränkt werden. |
| @property | {string} | SQLGroupBy | Hier können SQL-Spalten angegeben werden, worüber ein SELECT Resultset gruppiert ausgegeben werden soll.<br>Das "GROUP BY" muss selber nicht angegeben werden. |
| @property | {string} | SQLOrderBy | Angabe von Spalten über welche die Treffer in einem SELECT sortiert ausgegeben werden. |
| @property | {string} | SQLTable | Speichert den Tabellen-Namen. |
| @property | {string} | FullTable | Aus dem Tabellen-Namen und den Verbindungsinformationen wird der komplette Tabellen-Name mit Datenbank und bei MS SQL auch inklusive "dbo" generiert.<br>Beispiel: "MeineTabelle" -><br>"Datenbankname.dbo.Mein |
| @property | {string} | SQLType | Speichert den Verbindungs-Namen aus der dbConn.json. |

| | | | |
|---|---|---|---|
| **@property** | **{string}** | **ConnType** | **"odbc" oder "mysql" und für Testzwecke "oracle"** |
| **@property** | **{string}** | **ConnStr** | **Verbindungsname aus der "dbConn.json"** |
| **@property** | **{string}** | **SQLUser** | **Benutzer um die DB-Verbindung aufzubauen** |
| **@property** | **{string}** | **DB** | **Datenbankname aus "dbConn.json"** |
| **@property** | **{string}** | **DBO** | **"dbo"-Angabe (nur MS SQL) aus "dbConn.json"** |
| **@property** | **{string}** | **DateFormat** | **Datumsformat für INSERT / UPDATE** |
| **@property** | **{string}** | **ReturnID** | **Beim INSERT kann optional ein Rückgabewert der eingefügten Spalte zurückgegeben werden, wie zum Beispiel eine Auto-Inkrement-Spalte.** |
| **@property** | **{string}** | **ReturnAutoColumn** | **Angabe einer Auto-Inkrement-Spalte, dessen Wert nach einem INSERT zurückgegeben werden soll** |
| **@property** | **{string}** | **ReturnAutoColWhere** | **Bei MS SQL kann bei einem INSERT Befehl die Rückgabe-Spalte direkt im SQL-Statement angegeben werden. Bei MySQL muss nach dem INSERT ein SELECT auf die Datenbank ausgeführt werden. Für die Abfrage wird eine passende WHERE-Bedingung benötigt, welche hier angegeben werden muss.** |
| **@property** | **{*}** | **ColumnArray** | **Array aus Spalten-Objekten für SELECT / INSERT / UPDATE Befehle.** |

Das Objekt unterstützt MySQL und MS SQL. ORACLE ist kaum getestet und ist daher offiziell nicht freigegeben,

```
/** Die Funktion ermittelt zu einem Tabellen-Namen den vollständigen Namen.
 * Aus der Datenbankverbindung wird der Datenbank-Name und bei MS SQL die "dbo"-Angabe hinzugefügt.
 * @param {string} tableName Tabellen-Name
 * @returns {string} Vollständiger Tabellen-Aufruf (Beispiel: "MyTable" -> "DatabaseName.dbo.MyTable")
 **/
SqlObject.getFullTable( tableName )


/** Setzt die Properties auf leere Strings zurück.
 * @returns {boolean} true / false
 **/
SqlObject.resetEntries( )


/** Ändert manuell die Property "SQLTable" und "FullTable".
 * Der Verbindungsaufbau erfolgt weiterhin über die Angabe des Verbindungs-Namen!
 * @param {string} table Neuer Tabellen-Name
 * @returns {boolean} true / false
 **/
SqlObject.setTable( table )


/** Ändert manuell die Property "DB", "DBO" und "FullTable".
 * Der Verbindungsaufbau erfolgt weiterhin über die Angabe des Verbindungs-Namen!
 * @param {string} databaseName Name der neuen Datenbank
 * @param {string} dbo Optionale Angabe für die "dbo"-Angabe (nur MS SQL)
 * @returns {boolean} true / false
 **/
SqlObject.switchDatabase( databaseName, dbo )


/** Ändert die Datenbankverbindung. Es muss ein gültiger Wert aus der "dbConn.json" angegeben werden!
 * @param {string} dbConnectionName Name der Datnbankverbindung aus der "dbConn.json"
 * @returns {boolean} true / false
 **/
SqlObject.switchDbConnection( dbConnectionName )
```

```
/** Die Datenbankspalten für SELECT, INSERT und UPDATE Befehle werden im Property-Array
"ColumnArray" gespeichert.
 * @param {string} sqlColumnName Name der SQL-Tabellen-Spalte
 * @param {string} sqlColumnType Typ zur SQL-Tabellen-Spalte ("varchar", "date", "datetime",
"bit", "int", "decimal")
 * @param {*} optValue Für INSERT und UPDATE Befehle muss ein Wert mitgegeben werden.
 *                     Der Wert muss zum Tabellen-Spalten-Typen passen!
 * @param {int} optDecimalPlaces Bei numerischen Werten muss die Anzahl der Dezimalstellen
angegeben werden!
 **/
SqlObject.addColumn( sqlColumnName, sqlColumnType, optValue, optDecimalPlaces )

/** Über die folgenden Funktionen kann direkt der DB-Spalten-Typ mitgegeben werden **/
SqlObject.addBitColumn( sqlColumnName, optValue )
SqlObject.addDateColumn( sqlColumnName, optValue )
SqlObject.addDatetimeColumn( sqlColumnName, optValue )
SqlObject.addDecimalColumn( sqlColumnName, optValue, optDecimalPlaces )
SqlObject.addIntColumn( sqlColumnName, optValue )
SqlObject.addVarcharColumn( sqlColumnName, optValue )

/** Über die folgende Funktion können unmaskierte Werte (zum Beispiel Funktionsaufrufe)
 * oder eigens maskierte Werte mitgegeben werden.
 * @param {string} sqlColumnName Name der SQL-Tabellen-Spalte
 * @param {string} value Wert
 * @param {string} optType Optionaler Typ für den Rückgabewert ("bit", "date", "datetime",
"decimal", "int", "varchar" /
 *                                Standard:  "varchar")
 * @since 1.0.300
 **/
SqlObject.addColumnValue( sqlColumnName, value, optType )



/** Ein SQL-Befehl kann über diese Funktion selber zusammengesetzt und ausgeführt werden.
 * @param {string} executeStatement Auszuführender SQL-Befehl
 * @returns {boolean} true / false
 **/
SqlObject.executeStatement( executeStatement )



/** Diese Funktion setzt aus den hinzugefügten Datenbank-Spalten und Werten einen INSERT-
Befehl zusammen
 * und führt im Anschluss executeStatement() aus.
 * Über die Funktion kann immer nur eine Datenbank-Spalte hinzugefügt werden.
```

```
 * @returns {boolean} true / false
 **/
SqlObject.insertValues()


/** Diese Funktion ist dafür gedacht, um bei einem INSERT mehrere Spalten als Bulk-Import zu importieren.
 * Die Funktion erstellt den INSERT-Part aus dem "ColumnArray".
 * @returns {string} Erstellt den ersten Teil des INSERT-Statements aus den Spalten-Angaben
 **/
SqlObject.getInsertStatementColumns()


/** Diese Funktion ist dafür gedacht, um bei einem INSERT mehrere Spalten als Bulk-Import zu importieren.
 * Die Funktion erstellt einen Werte-Part aus dem "ColumnArray".
 * Aus den Funktionen getInsertStatementColumns() und dieser Funktion muss der INSERT-Befehl zusammengesetzt werden.
 * Im Anschluss kann der Befehl mit der Funktion executeStatement() ausgeführt werden.
 * @returns {string} VALUES-Angabe eines INSERT-Statements für einen Bulk-Import.
 **/
SqlObject.getInsertStatementValues()


/** Diese Funktion setzt aus den hinzugefügten Datenbank-Spalten und Werten einen UPDATE-Befehl zusammen
 * und führt im Anschluss executeStatement() aus.
 * @returns {boolean} true / false
 **/
SqlObject.updateValues()
```

```
/** Diese Funktion setzt aus den hinzugefügten Datenbank-Spalten und Werten einen UPDATE-Befehl zusammen
 * und führt im Anschluss executeStatement() aus.
 * @param {string} tableType Optionale Angabe.
 *   "usedb": Setzt "USE dbname' vor das SELECT und verwendet nur die dbo-Angabe und den Tabellennamen im SELECT
 *   "justtable": Verwendet im SELECT nur die Property 'SQLTable'.
 *   "usedbjusttable": Kombiniert die beiden Angaben "usedb" und "justtable".
 * Andernfalls wird im SELECT-Statement die Property 'FullTable' verwendet.
 * @returns {*} Gibt ein Array der SQL-Treffer zurück. Bei einem Fehler wird false zurückgegeben.
 **/
SqlObject.selectData(tableType)
```

# SqlObject() Beispiele

**Das folgende Beispiel zeigt eine einfache SQL-Abfrage von 2 Spalten auf die Tabelle "MyTable". Die Angabe "DEX_Workflow" ist die Angabe der Datenbankverbindung aus der "dbConn.json" und nicht der Datenbank-Name!**

```
var sql = new SqlObject("MyTable", "DEX_Workflow");
    sql.addVarcharColumn("ColumnA AS MyTest"); /* Spaltennamen können umbenannt werden */
    sql.addDateColumn("ColumnB");
    sql.SQLWhere = "ColumnA LIKE '%Test%'";
    sql.SelectDistinct = true;
    sql.ExpectHits = false;
    sql.selectData();
util.log(sql.SQL); /* Schreibt den SELECT-Befehl in das Documents-Log */
if( sql.Result===false ){
     return sql.Error;
}
for(var i=0; i<sql.Result.length; i++){
    var strValue = sql.Result[i]["MyTest"];
    var dateValue = sql.Result[i]["ColumnB"];
}
```

**Mit dem SELECT können auch Abfragen via JOIN über mehrere Tabellen ausgeführt werden.**

```
var sql = new SqlObject("Invoice_Posting_Head", "DEX_Workflow");
    sql.addVarcharColumn("head.FileID AS FileID");
    sql.addVarcharColumn("head.InvoiceNumber AS InvoiceNumber");
    sql.addIntColumn("pos.ID AS ID");
    sql.addDecimalColumn("pos.Net AS Net");
    sql.SQLTable = sql.getFullTable("Invoice_Posting_Pos")    + " AS pos LEFT JOIN "
                 + sql.getFullTable("Invoice_Posting_Head") + " AS head ON "
                 + "(head.FileID=pos.FileID)";
    sql.SQLWhere = "(head.PostingStatus=" + sql.parseValueForSql("varchar", "workflow") + ")
AND "
                 + "(head.FileID=" + sql.parseValueForSql("varchar", context.file.getid()) +
")";
    sql.ExpectHits = true;
/* Durch 'justtable' wird der Tabellen-Name nicht automatisch zusammengesetzt,
```

```
 * sondern es wird das Property SQLTable verwendet */
    sql.selectData("justtable");
if( sql.Result===false ){
    util.log(sql.Log); /* Schreibt das Log zum SQl-Objekt in das Documents-Log */
     return sql.Error;
}
/* Iteration über die Ergebnisse  */
for(var x=0; x<sql.Result.length; x++){
    var entry= sql.Result[x];
    var invNo = entry.InvoiceNumber;
    var lineID = entry.ID;
    var lineNet = entry.Net;
 }
```

**Das folgende Beispiel schreibt einen Datensatz in die Datenbank und liefert den Wert der Auto-Inkrement-Spalte zurück.**

```
var docFile = context.file;
var sql = new SqlObject("Invoice_Posting_Head", "DEX_Workflow");
      sql.addVarcharColumn("FileID", docFile.getid());
/* Optional: Mit Rückgabe der Auto-Inkrement ID */
      sql.ReturnAutoColumn = "ID";
      sql.ReturnAutoColWhere = "(FileID=" + sql.parseValueForSql("varchar", docFile.getid())
+ ")";
      sql.insertValues();

if( sql.Result===false ){
     context.errrorMessage = sql.Error;
     return -1;
}
return sql.ReturnID; /* Rückgabe der Auto-Inkrement ID */
```

**Das folgende Beispiel führt ein Update auf eine Tabellen-Zeile aus.**

```
var docFile = context.file;
var sql = new SqlObject("Invoice_Posting_Head", "DEX_Workflow");
    sql.addVarcharColumn("InvoiceNumber", docFile.InvoiceNumber);
    sql.SQLWhere = "(FileID=" + sql.parseValueForSql("varchar", context.file.getid()) + ")";
    sql.updateValues();

if( sql.Result===false ){
    context.errrorMessage = sql.Error;
```

```
    return -1;
}
```

**Das folgende Beispiel löscht DB-Einträge:**

```
var sql = new SqlObject("Invoice_Posting_Head", "DEX_Workflow");
    sql.executeStatement("DELETE FROM " + sql.FullTable + " WHERE
(UpdateTimestamp<'01.01.2020')");
if( sql.Result===false ){
    context.errrorMessage = sql.Error;
    return -1;
}
```

# DocFile prototype

Die DocFile Klasse wurde um einige spezifische Funktionen erweitert.

```
/** Fügt einen Kommentar zum Historienfeld "Comment" hinzu.
 * Bei einer Split-Mappe wird der Kommentar auch zur Haupt-Mappe hinzugefügt.
 * @param {string} comment Kommentar-Angabe
 * @returns {string} Leerer String oder Fehlermeldung
**/
DocFile.addComment( comment )

/** Prüft ob für den Benutzer eine Wiedervorlage zur Mappe definiert wurde.
 * @param {string} optUserLogin Optional kann ein Benutzer-Login mitgegeben werden.
Andernfalls wird der aktuell sperrende Benutzer geprüft.
 * @returns {boolean} true (Wiedervorlage ist definiert) / false
 * @since 1.1.015
**/
DocFile.checkResubmission( optUserLogin )

/** Prüft ob der aktuelle Workflow-Schritt für den aktuellen Benutzer automatisch im
Bearbeitungs-Modus startet.
 * @param {SystemUser} systemUser SystemUser Objekt
 * @returns {boolean} true (StartEditMode) / false
**/
DocFile.checkStartEditMode( systemUser )

/** Prüft alle Kopf-Pflichtfelder der Mappe, wenn der Parameter
"CheckMandatoryHeadFieldsOnActionEnd" auf 'true' steht.
 * Setzt context.errorMessage
 * @returns {boolean} true (Pflichtfelder gesetzt) / false (Pflichtangaben fehlen)
**/
DocFile.checkMandatoryHeadFields()

/** Vergleicht einen Feldwert mit einem String-Wert.
 * Diese Funktion wird zum Beispiel bei der Auswertung der Workflow-Regeln oder bei der
Parameter-Ermittlung verwendet.
 * @param {string} techFieldName Technischer Feldname
```

```
 * @param {string} stringValue Vergleichs-Angabe
 * @param {string} optCompareType Optionale Angabe eines Vergleichstyps. Als Default-Wert
wird "=" verwendet.
 *                                Erlaubte Typen sind: '=', '<', '<=', '>', '>=', '..', '|~',
'!=', 'expr'
 * @returns {string} true (Vergleichswert passt zum Feldwert) / false (Vergleichswert passt
nicht zum Feldwert)
**/
DocFile.compareFieldValueWithString( techFieldName, stringValue, optCompareType )

/** Löscht alle Sub-Mappen zu einer Haupt-Mappe
 * @returns {string} Leerer String oder Fehlermeldung
**/
DocFile.deleteSubFiles()

/** Liefert die ID eines ausgehenden Kontrollflusses zum Abschluss einer Aktion passend zur
angegebenen Navigation.
 * Bei einer Weiterleitung über eine benutzerdefinierte Aktion muss die anschließende
Navigation
 * über die Skript-Rückgabe gesteuert werden!
 * @param {string} paramNavigation Navigation("keepfile", "next", "overview", "inbox",
"folder").
 * @returns {string} Leerer String oder Kontrollfluss ID
**/
DocFile.getActionEndControlFlowIdByNavigation( paramNavigation)

/** Ermittelt die in den Workflow-Aktionen konfigurierte Navigation nach Abschluss der Aktion.
 * @returns {string} Leerer String oder Navigation("keepfile", "next", "overview", "inbox",
"folder")
**/
DocFile.getActionEndNavigation()

/** Liefert die ID zu einem ausgehenden Workflow-Kontrollfluss-Namen.
 * @param {string} paramCfName Kontrollfluss-Name
 * @param {boolean} paramCfStartsWith Optionale Angabe, wenn der Kontrollfluss-Name nur mit
dem 'paramCfName' beginnt.
 * @returns {string} Leerer String oder Kontrollfluss ID
**/
DocFile.getControlFlowIdByName( paramCfName, paramCfStartsWith )
```

```
/** Wenn zu einem Beleg Split-Mappen erstellt werden, dann wartet die Haupt-Mappe auf die vollständige Abarbeitung
 * aller Split-Mappen.
 * Der Workflow-Verlauf der Split-Mappen ist nur im Monitor der einzelnen Split-Mappen nachzuvollziehen.
 * Die Split-Mappen werden allerdings nach der Verarbeitung gelöscht.
 * Über diese Funktion können Monitor-Einträge zur Hauptmappe hinzugefügt werden.
 * @param {string} comment Optionale Angabe für den Kommentar beim Monitor-Eintrag.
 * @returns {boolean} true / false
**/
DocFile.forwardMainFile( comment )

/** Wenn eine Mappe in einer technischen Aktion durch einen Job verarbeitet werden soll,
 * dann kann der Job die Mappe mit dieser Funktion weiterleiten.
 * Das Skript wechselt hierfür in den Kontext eines sperrenden Benutzers.
 * @param {string} controlFlow OKontrollfluss-Name ('TechActionEnd' oder Default: 'TechActionEndManually').
 * @param {string} comment Optionale Angabe für den Kommentar beim Monitor-Eintrag.
 * @returns {boolean} true / false
**/
DocFile.forwardTechActionJob( controlFlow, comment )

/** Ermittelt die Anzahl der Nachkommastellen aus der Gentable-Feldkonfiguration für die Gentable-Spalte.
 * @param {string} colName Technischer Gentable-Spalten-Name.
 * @param {number} expValue Standardwert für die Anzahl der Nachkommastellen.
 * @returns {number} Anzahl Dezimalstellen
 * @since 1.1.015
**/
DocFile.getGentableColumnDecimal( colName, expValue )

/** Ermittelt das Wiedervorlagedatum für einen Benutzer.
 * @param {string} userLogin SystemUser-Login
 * @returns {string} Wiedervorlagedatum als String
 * @since 1.1.015
**/
DocFile.getResubmissionDate( userLogin )

/** Gibt eine Liste aller sperrenden Benutzer und Gruppen im Workflow aus.
 * Die Funktion iteriert über alle sperrenden Workflow-Schritte.
```

```
 * @param {string} pListSeparator Trennzeichen zwischen den Angaben.
 * @returns {string} true / false
**/
DocFile.listLockingUser( pListSeparator )

/** Setzt die Eigenschaft "ReadOnly" bei allen Feldern der Mappe auf 'true'.
 * @returns {string} true (sync() erfolgreich) / false
**/
DocFile.setAllFieldsReadOnly()

/** Berechnet ein "DeadlineDate"-Datumsfeld anhand eines Ausgangs-Datums und den Angaben vom Parameter "DeadlineDate".
 * Ebenfalls werden die Felder "DeadlineDays" und "DeadlineSign" gesetzt.
 * @param {boolean} syncFile Über den Parameter kann optional direkt ein sync() ausgeführt werden.
 * @param {string} optUseDateField Optional kann ein alternatives Datumsfeld verwendet werden (Default: 'WorkflowStart').
 * @param {boolean} optSetEmptyDate Optional wird ein leeres Datumsfeld auf das aktuelle Tagesdatum gesetzt.
 * @returns {boolean} true / false
**/
DocFile.setDeadlineDate( syncFile, optUseDateField, optSetEmptyDate )

/** Liest die Feldkonfiguration und die Alternativen und setzt die Feldeigenschaften.
 * @returns {boolean} true / false
**/
DocFile.setFieldAttributesByConfig()

/** Wendet die unter "Feldkonfigurationen" -> "Suchfelder" eingestellte Konfiguration an.
 * Die Funktion holt sich die Einträge aus definierten Positions-Spalten und schreibt die Angaben in ein Kopf-Feld.
 * @returns {string} Leerstring oder Fehlermeldung
**/
DocFile.setGentableSearchFields()

/** Wechselt den ausführenden Benutzer an der aktuellen Workflow-Aktion.
 * @param {string} optLogin Optionale Angabe eines gültigen Benutzer-Logins (Default: aktueller Benutzer).
 * @returns {*} true / Fehlermeldung
**/
```

```
DocFile.takeOver( optLogin )

/** Entsperrt eine Mappe.
 * @returns {string} true / false
**/
DocFile.unlockDocFile()
```

Zudem können Parameter-Angaben zum DocFile-Objekt ermittelt werden.

```
/** Ermittelt das Parameter-Objekt zum angegebenen Parameter.
 * @param {string} paramName Technischer Parameter-Name.
 * @param {string} optValue Optionale Angabe, ob direkt der 1., 2. oder 3. Wert zum Parameter zurückgegeben werden soll.
 * @returns {*} Gibt entweder ein Parameter-Objekt zurück oder den spezifischen 1., 2. oder 3. Wert oder null bei Fehlern
**/
DocFile.getParamObject( paramName, optValue )
/* Beispiel */
var docFile = context.file;
var value1String = docFile.getParamObject("MeinParameter", 1);

/** Liefert einen spezifischen Wert zu einem Parameter-Objekt (siehe DocFile.getParamObject()).
 * @param {string} paramObj Optionale Angabe eines gültigen Benutzer-Logins (Default: aktueller Benutzer).
 * @param {string} valueNo Angabe, ob der 1., 2. oder 3. Parameter-Wert zurückgegeben werden soll.
 * @param {*} standardValue Standardwert im passenden Datentypen.
 * @param {string} optParamType Optionale Angabe für den Datentyp des Parameters (Standard: 'string').
 *    Die angaben werden in der Datenbank immer als String gespeichert, werden aber in den Datentypen gewandelt.
 *    Gültige Angaben: 'boolean'/'bool', 'percent'/'numeric'/'decimal', 'select'/'string'
 * @returns {*} In passenden Datentypen umgewandelter Parameter-Wert / Standardwert
**/
DocFile.getParamValue( paramObj, valueNo, standardValue, optParamType)
/* Beispiel */
var docFile = context.file;
var paramObj = docFile.getParamObject("MeinParameter");
var value1Bool = docFile.getParamValue( paramObj, 1, true, "boolean" );
```

```
var value2String = docFile.getParamValue( paramObj, 2, "Test", "string" );
var value3Numeric = docFile.getParamValue( paramObj, 3, 1.5, "numeric" );
```

# Gentable

Das Gentable kann über ein "Gentable"-Objekt realtiv einfach ausgelesen und neu geschrieben werden.

```
var docFile = context.file;
var gentable = new Gentable(docFile);

/* Erstellt ein Array aus Zeilen-Objekten *
 * Zahlen, Datumswerte und Boolsche Werte haben bereits das passende Format */
if( gentable.readFromField()===false || gentable.Result===false ){
  util.log(Gentable.Log); /* Bei Fehlern kann das Log ausgegeben werden */
  context.errorMessage = gentable.Error; /* Fehlermeldung in Client-Sprache */
  return -1;
}

/* Beispiel um Zeilen zu iterieren */
for( var row=0; row<gentable.Rows.length; row++ ){
  var net = gentable.Rows[row]["Net"]; /* Netto aus der Zeile ausgeben */
  gentable.Rows[row]["Gross"] = net; /* Brutto auf den Netto-Wert setzen */
}

/* Summe aus Spaltenwerten bilden. */
var sumNet = gentable.sumLineAmounts("Net");

/* Gentable in die Datenbank-Tabelle 'Invoice_Posting_Pos' schreiben.
 * Die eindeutige Zeilen ID wird zurück in die Gentable-Spalte "ID" geschrieben.
 * Hierüber wird beim nächsten Aufruf ein Update ausgeführt. */
gentable.insertIntoDB();

/* Gentable-String in das Feld 'Gentable' schreiben.
 * Über den Parameter kann optional direkt ein sync() auf das DocFile-Objekt ausgeführt
werden. */
var syncFile = true;
gentable.createGentableStr(syncFile);
```

# Gentable User-Exits

An einigen Skriptstellen wurden User-Exit - Funktionen direkt auf das Gentable-Objekt hinzugefügt. Dadurch muss das Gentable nicht nochmals umständlich ausgelesen werden. Über "this" kann direkt auf das Gentable-Objekt zugegriffen werden.

Die Funktionen selber müssen keinen Rückgabewert zurückgeben. Wenn "this.Result" auf den boolschen Wert "false" gesetzt wird kann über "this.Error" eine Fehlermeldung mitgegeben werden.

```
/** Wird bei der Erstellung der initialen Gentable-Zeile aufgerufen **/
Gentable.prototype.createInitialRow = function (){
}
```

```
/** Wird am Ende des Skripts beim Laden der offenen Bestellpositionen ausgeführt **/
Gentable.prototype.adjustLoadedOrderPositions = function (){
}
```

```
/** User-Exit beim Speichern einer Rechnungs-Akte
 * @since Invoice 1.0.200
 **/
Gentable.prototype.ue_InvoiceOnSave = function (){
	this.Result = false;
	this.Error = "Fehlermeldung";
	var docFile = this.DocFile;
	for( var r=0; r<this.Rows.length; r++){
		var row = this.Rows[r];
		this.Rows[r]["Net"] = 0;
	}
}
```

```
/** User-Exit bei Abschluss einer Workflow-Aktion
 * @since Invoice 1.0.200
 **/
Gentable.prototype.ue_OnActionEnd = function (){
	this.Result = false;
	this.Error = "Fehlermeldung";
	var docFile = this.DocFile;
	for( var r=0; r<this.Rows.length; r++){
		var row = this.Rows[r];
```

```
    this.Rows[r]["Net"] = 0;
  }
}
```

```
/** Wird nach dem Auslesen des Gentable aber noch vor dem Anwenden einer Standardkontierung
ausgeführt.
 * @param {Accounting} standardAccounting Das Standardkontierungs-Objekt
 * @since Invoice 1.1.300
**/
Gentable.prototype.ue_BeforeStandardAccounting = function(standardAccounting){
  /* Beispiel: Kürzt das Gentable auf die gleiche Anzahl an Positionen wie die
Standardkontierung *
    accObj.getAccountingPositions(accObj.AccountingName);
    var accPos = accObj.Accounting[accObj.AccountingName];
    if( this.Rows.length>accPos.length ){
        for( r=this.Rows.length-1; r>=accPos.length ;r-- ){
            if( this.Logg instanceof Log ){
                this.Logg.info("["+r+"] this.Rows.pop();");
            }
            this.Rows.pop();
        }
    }
    */
}

/** Wird nach dem Anwenden einer Standardkontierung ausgeführt.
 * Im Anschluss wird automatisch gentable.createGentableStr(true) ausgeführt.
 * @param {Accounting} standardAccounting Das Standardkontierungs-Objekt wird ab Invoice
1.1.300 mitgegeben
 * @since Invoice 1.0.300
**/
Gentable.prototype.ue_AfterStandardAccounting = function(standardAccounting){
  ;
}
```

## Gentable XML User-Exits

Die Gentable-XML wird automatisch über die WEB-Konfigurations-Tabellen generiert. Um nur mehr Flexibilität zu ermöglichen können die intern generierten Objekte nochmal angepasst werden.Die folgenden Funktionen befinden sich im Skript "**DEXPRO__UserExit_GentableXml**".

Globale XML-Einstellungen können zum Beispiel nur auf einen Wert gesetzt werden. Über die folgende Funktion können Eigenschaften abhängig von Feldwerten an der Mappe manipuliert werden.

```
/** User exit to manipulate gentable Xml settings such as "indexCheckboxes",
"lastRowEditable", "storeFormat", ...
 * Called within "GentableXml.createGentableXml()".
 * @param {any} settingObject Object with Gentable settings
 * @param {string} template Name of the DocFile-template ("Mailroom", "Invoice",
"lcmContract", ...)
 * @param {DocFile} docFile DocFile object
 * @returns {array} (Manipulated) Array with Gentable settings objects.
 **/
function ue_GentableXml_ManipulateGentableXmlSettings(settingObject, template, docFile){
    /* If value is null the standard value will be used:
        xmlSetting.fieldName              // Field name        | must value!!!     ->
technical docFile head field name to store Gentable data
        xmlSetting.database               // Database name     | no standard value ->
database name for database connections
        xmlSetting.alwaysShowToolbar      // "true" or "false" | no standard value -> shows
gentable-buttons even in read-mode
        xmlSetting.url                    // Database url      | no standard value ->
database url (will only be set if datase is not null)
        xmlSetting.driver                 // Database driver   | no standard value ->
database driver (will only be set if datase is not null)
        xmlSetting.user                   // Database user     | no standard value ->
database user (will only be set if datase is not null)
        xmlSetting.password               // Database pw       | no standard value ->
database user password (will only be set if datase is not null)
        xmlSetting.storeFormat            // "xml" or "json"   | Standard: "xml"   ->
gentable store format
        xmlSetting.indexCheckboxes        // "true" or "false" | Standard: "true"  -> sets
checkboxes in front of each row to select rows
        xmlSetting.indexNumbers           // "true" or "false" | Standard: "true"  -> shows
line number in front of each row
        xmlSetting.enableTextSelection    // "true" or "false" | Standard: "false" -> enables
text selection from more than one field (even in read-mode)
```

```
xmlSetting.languagePropertiesFile // technical pf-name | no standard value -> Properties file
name to use pf
                                        //
example: for "gentable_de.properties" set "gentable"
        xmlSetting.lastRowEditable        // "true" or "false" | Standard: "true"  -> set
"true" if you want to edit each column when you add new rows
        xmlSetting.quickFilter            // "true" or "false" | Standard: null    -> adds
search field for each column
        xmlSetting.saveAll                // "true" or "false" | Standard: null    -> saves
all values even if they are not listed in xml definition
        xmlSetting.searchable             // "true" or "false" | Standard: "false" -> shows
search field in head-button-line
        xmlSetting.stickyGroupRows        // "true" or "false" | Standard: null    -> sticks
head line while scrolling down
        xmlSetting.columnsAlwaysVisible   // "true" or "false" | Standard: "true"  -> use
this setting to hide column if each value is invisible
        xmlSetting.resizableRows          // "true" or "false" | Standard: "false" ->
resizable row width
        xmlSetting.resizeColumnsToContent // "true" or "false" | Standard: "false" ->
automatically resizes field size to it's content
                                        //                                        does
not work correct if the content is larger than the technical value (1000 -> 1.000,00)
        xmlSetting.rowHeight              // integer value     | Standard: 20      ->
standard pixel row height
        xmlSetting.moveRows               // "true" or "false" | Standard: "false" -> allows
changing row positions via drag and drop
        xmlSetting.showMoveColumn         // "true" or "false" | Standard: "false" -> shows
extra column to move rows
    */

    return settingObject;
}
```

In der XML können auch Zeilen-Bedingungen definiert werden. Hierüber kann ein Schreibschutz auf ganze Zeilen gelegt werden bzw. können Zeilen abhängig von Feldwerten ausgeblendet werden.

```
/** User exit to build up Gentable Xml row conditions.
 * First type "READONLY" disables  field editing for all row-fields.
 * Second type "INVISIBLE" hides complete row if the rule matches.
```

```
 * Called within "GentableXml.createGentableXml()".
 * @param {String} template Name of the DocFile-template ("Mailroom", "Invoice",
"lcmContract", ...)
 * @param {DocFile} docFile DocFile object
 * @returns {String} Gentable row condition xml tag.
**/
function ue_GentableXml_AddRowConditions(template, docFile){
    var rowCondition = '';
    /* EXAMPLES from Otris Gentable Administration guide: *
    rowCondition = '<rowCondition><rule type="READONLY" field="Field1" value="2222"
/></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY" filefield="Creditor" value="ALFKI"
/></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY" accessprofile="Warehouse"
/></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY" field="Field5" value="Field5"
accessprofile="Directors" /></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY" filefield="Creditor" value="OTRIS"
accessprofile="Administration" /></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY"
autotext="%currentUser.$AllowedAmount%" field="Field4" /></rowCondition>';
    rowCondition = '<rowCondition><rule type="READONLY"
autotext="%currentUser.$AllowedAmount%" filefield="Amount" /></rowCondition>';
    rowCondition = '<rowCondition><rule type="INVISIBLE" invert="true"
accessprofile="Administration" /></rowCondition>';
    */
    return rowCondition;
}
```

Über die Feldkonfigurationen in der WEB-Administration können die Feldeigenschaften bereits abhängig von Kopfwerten geändert werden. Die folgende Funktion wurde zum Beispiel standardmäßig erweitert, um bei einem Klick auf irgendein Feld innnerhalb einer Zeile, diese Zeile als 'aktiv' zu kennzeichnen. Hierüber wird im Squeeze-Viewer zu einer Rechnung die passende Rechnungsposition angezeigt. Über die Funktion können sogar Feldtypen geändert werden.

Einen sehr interessanter Anwendungsbereich bietet die Funktion bei der Auflistung von Auswahl-Listen. Diese können über die WEB-Konfiguration nur starr angegeben werden. Über das Skripting können zum Beispiel sehr einfach Benutzer und Benutzergruppen mit technischem Wert und Anzeige-Wert angegeben werden.

```
/** User exit to manipulate field configuration. This user exit is called for each Gentable
```

```
field.
 * Called within "GentableXml.createGentableXml()".
 * @param {FieldConfig} fieldConfigObject Field configuration object (Standard+Changes)
 * @param {String} template Name of the DocFile-template ("Mailroom", "Invoice",
"lcmContract", ...)
 * @param {DocFile} docFile DocFile object
 * @returns {FieldConfig} Field configuration object.
 **/
function ue_GentableXml_ManipulateFieldSettings(fieldConfigObject, template, docFile){
    var attribute     = "name";
    var techFieldName = fieldConfigObject.TechFieldName;

    /* Beispiel für ein Feld, welches alle Zugriffsprofile listet */
    if( techFieldName==="EnumAccessProfileField" ){
        fieldConfigObject.FieldDataType = "SELECT";
        fieldConfigObject.SelectOption  = "<option value=\"\"></option>"; // Leere Feldwert-
Option
        var iterAP = context.getAccessProfiles(false);
        for( var ap=iterAP.first(); ap; ap=iterAP.next() ){
            var apDisplayName = (ap.getAttribute(attribute)===""){? ap.name :
ap.getAttribute(attribute);
            fieldConfigObject.SelectOption += "<option
value=\""+ap.name+"\">"+apDisplayName+"</option>";
        }
    }

    /* Setzt die angeklickte Zeile als 'aktiv', um im Squeeze-Viewer die entsprechende Zeile
zu markieren */
    if( fieldConfigObject.JsEventType1!=="onFocus" &&
fieldConfigObject.JsEventType2!=="onFocus"  && fieldConfigObject.JsEventType3!=="onFocus"
){
        fieldConfigObject.JsEventType4 = "onFocus";
        fieldConfigObject.JsEvent4     = "setActiveRow";
    }
    return fieldConfigObject;
}
```

# Code-Beispiele

In diesem Abschnitt werden kurze Code-Beispiele vorgestellt.

## Parameter verwenden

Die Abschnitt zeigt, wie man Parameter-Werte im eigenen Code verwendet.

```
/* Zuerst muss über das DocFile-Objekt ein Parameter-Objekt erstellt werden */
var docFile  = context.file;
var paramObj = docFile.getParamObject("MyTechParameterName");
/* Dann kann ebenfalls über das DocFile-Objekt der Parameter-Wert ermittelt werden.
 * 1. Parameter: Das Parameter-Objekt
 * 2. Parameter: Angabe welcher Parameter-Wert ermittelt werden soll (1, 2, 3)
 * 3. Parameter: Gibt den Default-Wert an, falls kein spezifischer Wert zum DocFile definiert
ist
 * Hierdurch funktioniert die Funktion selbst in dem Fall, wenn der Parameter gar nicht
definiert ist.
 * 4. Parameter: Angabe zum parsen des Werts ("bool", "numeric", "string")
 */
var myValue = docFile.getParamValue(paramObj, 1, "DefaultValue", "string");
```

## Aus Base64 String ein Document erstellen

Es kann vorkommen, dass man über einen Request ganze Dokumente zugesendet bekommt. Das folgende Beispiel erstellt aus einem Base64 String ein Dokument.

```
var requ = new myRequestObject();
var ret = requ.getRequest("getDocumentFiles", "docId=1");
var data = JSON.parse(ret.data);
var files = data.result;
for (var i = 0; i < files.length; i++) {
    var file = files[i];
    var base64File = file.base64;
    var filename = file.filename;
```

```
    var byteArr = util.base64Decode(base64File, true);

    var fso = new File("D:\\tmp\\Create\\" + filename, "w+b");
    if (!fso.ok())
       throw fso.error();

    fso.write(byteArr);
    fso.close();
}
```

# DEXPRO Invoice: Archivierung für unterschiedliche Mandanten

```
/* ================================================================================ */
/** Archiving is always custom designed!
 * This function is called for all technical archiving actions ("Archiving",
"ArchivingDirect", ...)
 * and for user defined action "Disqualify". You should call this function with archiving
rights.
 * Use "context.changeScriptUser()" to change script user context and do not forget to switch
back context before "return" statement!
 * @param {string} type Archiving type you can use to address different archives ("Job" /
"Direct" / "Disqualify").
 * @param {DocFile} docFile DocFile object.
 * @returns {boolean} true (archived) / false (error).
 **/
function ue_Archiving(type, docFile){
    /* Switch user context */
    var currentUser = context.currentUser;
    context.changeScriptUser("job");
    var view = '';

    var template = docFile.getAttribute("FromTemplate");
    switch(template){
        case "Mailroom":
        case "Invoice":
            if (docFile.Principal === "Company Name 00") {
                view = "Unit=Default/Instance=Default/View=Company00";
            } else if (docFile.Principal === "Company Name 01") {
                view = "Unit=Default/Instance=Default/View=Company01";
            } else if (docFile.Principal === "Company Name 02") {
                view = "Unit=Default/Instance=Default/View=Company02";
```

```
            } else {
                docFile.insertStatusEntry("ErrArchiving", "Can't find
Principal");
                docFile.sync();
                return false;
            }
            var schema = "Unit=Default/Instance=Default/DocumentSchema=Invoice";

            var ad = new ArchivingDescription();
            ad.targetView = view;
            ad.targetSchema = schema;
            ad.archiveServer = "EEx"; // since Documents 4.0 using multi archive
server
            ad.archiveMonitor = true;
            ad.archiveStatus = true;
            ad.addRegister("Documents");

            if( docFile.archive(ad) ){
                context.changeScriptUser(currentUser);
                var archiveKey = docFile.getArchiveKey();
                if( archiveKey==="" || archiveKey===false ){
                    docFile.insertStatusEntry("ErrArchiving", "Missing archive
key");
                    docFile.sync();
                    return false;
                }
                return true;
            }
            else{
                docFile.insertStatusEntry("ErrArchiving",
docFile.getLastError());
                docFile.sync();
            }
            break;
        default:
            docFile.insertStatusEntry("UndefinedFileTemplate", template);
            docFile.sync();
            break;
    }
    context.changeScriptUser(currentUser);
```

```
    return false;
}
```

# Versteckte User-Exits Mappe

An den Mappentypen sind Skripte für bestimmte Aktionen hinterlegt. Einige der Skripte werden komplett unverschlüsselt als User-Exit-Skripte herausgegeben. Die verschlüsselten Skripte enthalten dafür User-Exits. Die verschlüsselten Skripte importieren auch die spezifischen Modul-Libs sowie die Modul-Costom-Libs.

> Es bietet sich an die UserExits in den spezifischen User-Exit-Libs für das entsprechende Modul hinzuzufügen. Für Invoice wäre dies zum Beispiel die "**Invoice__UserExit_CustomLib**".

## BeforeEdit

```
/** Das UserExit wird vor der Bearbeitung einer Mappe direkt zu Beginn im Skript aufgerufen.
 * Bei einem Fehler muss "context.errorMessage" in der Funktion gesetzt werden!
 * @return {boolean} true / false.
**/
function ue_BeforeEdit_Start(){
//context.errorMessage = "...";
    return true;
}

/** Das UserExit wird vor der Bearbeitung einer Mappe am Ende des Skripts aufgerufen.
 * Bei einem Fehler muss "context.errorMessage" in der Funktion gesetzt werden!
 * @return {boolean} true / false.
**/
function ue_BeforeEdit_End(){
//context.errorMessage = "...";
    return true;
}
```

## OnSave

```
/** Das UserExit wird beim Speichern einer Mappe direkt zu Beginn im Skript aufgerufen.
 * Bei einem Fehler muss "context.errorMessage" in der Funktion gesetzt werden!
 * @return {boolean} true / false.
**/
```

```
function ue_OnSave_Start(){
	//context.errorMessage = "...";
    return true;
}

/** Das UserExit wird beim Speichern einer Mappe am Ende des Skripts aufgerufen.
 * Bei einem Fehler muss "context.errorMessage" in der Funktion gesetzt werden!
 * @return {boolean} true / false.
**/
function ue_OnSave_End(){
	//context.errorMessage = "...";
    return true;
}

/** Es gibt ein extra UserExit für Gentable-Operationen.
 * Der Vorteil hiervon ist, dass das Gentable nicht noch einmal ausgelesen und geschrieben
werden muss.
 * Das erfolgt automatisch im Anschluss.
 * Bei einem Fehler muss in "this.Error" die Fehlermeldung geschrieben werden und
"this.Error" muss auf false gesetzt werden!
 * @return Kein Rückgabewert erforderlich!
**/
Gentable.prototype.ue_InvoiceOnSave = function(){
	this.Error  = "";
	this.Result = true;
}
Gentable.prototype.ue_ProcurementOnSave = function(){
	this.Error  = "";
	this.Result = true;
}
```

# DecreaseFieldRights

```
/** Im DecreaseFieldRightsOnFileView-Skript wird beim Durchlauf der Felder für jedes Feld
dieses User Exit ausgeführt
 * @param {string} fieldName Aktueller technischer Feldname
 * @param {number} e Nummer für enumval
 * @param {Log} log Optionales Logging-Objekt (nur wenn das Logging aktiv ist)
**/
function ue_ChangeFieldRights(fieldName, e, log){
```

```
    enumval[e] = "-rw";
}
```

# Aufzählungs-Skripte Invoice

Auf einigen Aufzählungs-Feldern liegen Aufzählungs-Skripte und diese enthalten ebenfalls User-Exit-Funktionen zur Manipulation der Werte.

```
/** Bei der Neu-Erstellung des PropCaches für den Mandant können die Daten manipuliert werden (propCache.InvoicePrincipal).
 * Feld: Principal
 * @param {*} principalObject Eingehendes Objekt
 * @param {string} template Modul (Invoice/Mailroom/Procurement)
 * @param {Log} log Logging Objekt
 * @returns {*} Manipuliertes Objekt
**/
function ue_AdjustEnumPrincipals(principalObject, template, log){
    return principalObject;
}
function ue_AdjustEnumPrincipalsPropCache_Invoice(principalObject, log){
    return principalObject;
}
/** Am Ende des Skripts kann das enumval für den Mandanten manipuliert werden.
 * Feld: Principal
 * @param {*} enumval Eingehendes enumval
 * @returns {*} Manipuliertes enumval
**/
function ue_Adjust_Invoice_Enum_Principals(enumval){
    return enumval;
}

/** Bei der Neu-Erstellung des PropCaches für den Buchungskreis können die Daten manipuliert werden (propCache.InvoicePrincipal).
 * Feld: CompanyCode
 * @param {*} companyCodeObject Eingehendes Objekt
 * @param {string} template Modul (Invoice/Mailroom/Procurement)
 * @param {Log} log Logging Objekt
 * @returns {*} Manipuliertes Objekt
**/
function ue_AdjustEnumCompanyCodes(companyCodeObject, template, log){
```

```
    return companyCodeObject;
}
function ue_AdjustEnumPrincipalsPropCache_Invoice(companyCodeObject, log){
    return companyCodeObject;
}
/** Am Ende des Skripts kann das enumval für den Buchungskreis manipuliert werden.
 * Feld: CompanyCode
 * @param {*} enumval Eingehendes enumval
 * @returns {*} Manipuliertes enumval
**/
function ue_Adjust_Invoice_Enum_CompanyCodes(enumval){
    return enumval;
}

/** Über die Funktion können neben Rechnung und Gutschrift weitere Rechnungstypen definiert
werden.
 * Feld: InvoiceCreditVoucher
 * Hierfür kann direkt auf das enumal zugegriffen werden.
 * @returns {*} Kein Rückgabewert!
**/
function ue_AddInvoiceTypes(){
    enumval.push("myinvtype;pf:MyInvoiceType");
}

/** Über die Funktion können zusätzliche Zahlstatus definiert werden.
 * Feld: PaymentStatus
 * Hierfür kann direkt auf das enumal zugegriffen werden.
 * @returns {*} Kein Rückgabewert!
**/
function ue_AddPaymentStatus(){
    enumval.push("mypaytype;pf:MyPaymentType");
}
```

# Beim Aussteuern

```
/** Das User-Exit wird direkt zu Beginn im Skript "DEXPRO_Action_Disqualify" ausgeführt.
 * @param {Log} log Logging-Objekt
 * @param {string} comment Pflicht-Kommentar des Benutzers
 * @returns {boolean} true / false (Ausgabe Fehlermeldung)
**/
```

```
function ue_Disqualify_Start(log, comment){
    // context.errorMessage = "";
    return true;
}

/** Das User-Exit archiviert den Vorgang.
 * @param {string} type Bei ausgesteuerten Belegen immer "Disqualify"
 * @param {DocFile} docFile Auszusteuerndes DocFile-Objekt
 * @param {Log} log Logging-Objekt
 * @returns {boolean} true (Archiviert) / false (Ausgabe Fehlermeldung)
**/
function ue_Archiving(type, docFile, log){
    // context.errorMessage = "";
    return true;
}

/** Das User-Exit wird nach erfolgreicher Archivierung ausgeführt.
 * @param {Log} log Logging-Objekt
 * @param {string} comment Pflicht-Kommentar des Benutzers
 * @returns {boolean} true / false (Ausgabe Fehlermeldung)
**/
function ue_Disqualify_AfterArchiving(log, comment){
    // context.errorMessage = "";
    return true;
}

/** Das User-Exit wird nach erfolgreichem Löschen der Mappe ausgeführt.
 * @param {Log} log Logging-Objekt
 * @param {string} comment Pflicht-Kommentar des Benutzers
 * @returns {boolean} true / false (Ausgabe Fehlermeldung)
**/
function ue_Disqualify_AfterDelete(log, comment){
    // context.errorMessage = "";
    return true;
}
```

# Versteckte User-Exits Workflow

Einige Skripte enthalten User-Exit Funktionen und viele dieser Funktionen sind in den verschiedenen "UserExit"-Bibliotheken enthalten. Wenn neue Funktionen hinzukommen müssten diese Bibliotheken allerdings immer manuell um diese Funktionen erweitert werden. Daher werden neue hinzugefügte UserExit Funktionen nur aufgerufen, wenn Sie als Funktion existieren. Die Funktionen können bei Bedarf in einer UserExit-Bibliothek (zum Beispiel "DEXPRO__UserExit_CustomLib") hinzugefügt werden.

Die Liste bezieht sich jeweils auf die aktuellste Version. Ggf. sind Funktionen in älteren Versionen nicht verfügbar.

## Bei der Initialisierung

```
/** Für initiale projektspezifische Ausführungen.
 * Die Modul-spezifischen Funktionen werden nur beim spezifischen Mappentypen aufgerufen,
wenn die Funktionen existieren.
 * "ue_Initialization()" wird nachfolgend aufgerufen.
 * @portalscript: DEXPRO_WF_Initialization
 * @param {Log} log Logging-Objekt
 * @returns {boolean} true/false
**/
function ue_Initialization(log){
  ;
}
function ue_Initialization_Invoice(log){
  ;
}
function ue_Initialization_Mailroom(log){
  ;
}
function ue_Initialization_Procurement(log){
  ;
}
```

```
/** Für initiale projektspezifische Anpassungen am Gentable (nur Invoice).
 * @portalscript: DEXPRO_WF_Initialization
 * @param {Log} log Logging-Objekt
 * @returns {boolean} true/false
**/
Gentable.prototype.ue_Initialization_Invoice = function(log){
  ;
}
/** Wird bei der Erstellung einer initialen Gentable-Zeile ausgeführt.
 * @portalscript: DEXPRO_WF_Initialization
 * @param {string} optExecType Optionale Ausführungs-Info
("Initialization"/"ReplacePositionsByOrder")
 * @returns Kein Rückgabewert!
**/
Gentable.prototype.createInitialRow = function(optExecType){

}

/** Direkt zu Beginn des Workflows kann über den Parameter "Workflow_InitialDelay"
 * eine Workflow-Verzögerung angesteuert werden.
 * Nur wenn die Verzögerung angesteuert wird, dann wird vorab dieses User-Exit ausgeführt.
 * Die Funktion wird lediglich ausgeführt.
 * @portalscript: DEXPRO_WF_CheckInitDelay
 * @returns Kein Rückgabewert!
**/
function ue_BeforeInitialDelay(){
  ;
}

/** Zu Beginn des Workflows wird das initiale Skript ausgeführt und im Anschluss werden die
Workflow-Regeln ermittelt.
 * In speziellen Fällen wie bei der Erstellung von Split-Mappen kann es gewünscht sein, dass
diese Aktionen übersprungen werden sollen.
 * @portalscript: DEXPRO_WF_InitAbbreviation
 * @param {Log} log Logging-Objekt
 * @returns {boolean} true (Abkürzung nehmen) / false
**/
function ue_InitAbbreviation(log){
    log.info("["+context.file.getid()+"] start function ue_InitAbbreviation()");
 return true;
```

```
}
function ue_InitAbbreviation_Invoice(log){
 return true;
}
function ue_InitAbbreviation_Mailroom(log){
 return true;
}
function ue_InitAbbreviation_Procurement(log){
 return true;
}
```

## Nach der Ermittlung der Workflow-Regeln

```
/** Die User-Exit-Funktion wird ausgeführt, wenn eine Aktion über die Workflow-Regeln
übersprungen wird.
 * @portalscript: DEXPRO_WF_Rules_ActionCheck_SkipAction
 * @returns {boolean} true / false (führt zu einem Workflow-Fehler)
**/
function ue_OnSkipAction(){
return true;
}

/** Nach den Workflow-Regeln wird zunächst auf eine technische Aktion und auf das
Überspringen der Aktion geprüft.
 * Nur wenn die Aktion einer Gruppe oder einem Benutzer zugeordnet wird, wird das Skript
"DEXPRO_WF_Rules_BeforeUserAction" aufgerufen.
 * In dem Skript werden vor allem Zugriffsberechtigungen für die ermittelte Gruppe bzw. für
den ermittelten Benutzer gesetzt.
 * Über die Workflow-Regeln wurde bereits der Benutzer bzw. die Gruppe zugeordnet.
 * Bei einem Split wird das Skript für jede einzelne Split-Mappe ausgeführt!
 * @portalscript: DEXPRO_WF_Rules_BeforeUserAction
 * @returns {boolean} true / false (führt zu einem Workflow-Fehler)
**/
function ue_BeforeUserAction(){
return true;
}
```

## Bei der Benutzer-Aktion

```
/** Das UserExit wird bei Übernahme einer Mappe aus einem Gruppenkorb aufgerufen.
 * Der Aufruf erfolgt über das Skript "DEXPRO_WF_CheckDataForwardUser".
 * In dem Skript wird zuvor lediglich der aktuelle Benutzer in das Feld "ActionUser"
geschrieben.
 * Bei einem Fehler muss die Fehlermeldung zurückgegeben werden - andernfalls ein Leerstring!
 * @portalscript: DEXPRO_WF_CheckDataForwardUser
 * @return {string} ""/ Fehlermeldung.
**/
function ue_CheckDataForwardUser(){
    return "";
}

/** Das UserExit wird beim Zurücklegen in den Gruppenkorb aufgerufen.
 * Der Aufruf erfolgt über das Skript "DEXPRO_WF_CheckDataBackAccessProfile".
 * Bei einem Fehler muss die Fehlermeldung zurückgegeben werden - andernfalls ein Leerstring!
 * @portalscript: DEXPRO_WF_CheckDataBackAccessProfile
 * @return {string} ""/ error-message.
**/
function ue_CheckDataBackAccessProfile(){
    return "";
}
```

## Am Aktions-Ende

```
/** Die Funktionen werden bei Abschluss der Aktion zu Beginn bzw. am Ende ausgeführt.
 * @portalscript: DEXPRO_WF_CheckActionEnd
 * @returns {string} Leerstring oder Fehlermeldung
**/
function ue_OnActionEnd_Start(){
    return "";
}
function ue_OnActionEnd_End(){
    return "";
}
```

```
/** Wird am Ende einer Workflow-Aktion ausgeführt.
 * Hierdurch muss das Gentable nicht mehrfach ausgelesen und geschrieben werden.
 * Das Gentable wird im Anschluss im aufrufenden Skript in das Gentable-Feld zurück geschrieben.
 * @portalscript: DEXPRO_WF_CheckActionEnd
 * @returns Kein Rückgabewert benötigt. Fehler werden über this.Result und this.Error gesteuert.
**/
Gentable.prototype.ue_OnActionEnd = function(optExecType){
this.Error  = "";
    this.Result = true;
}

/** Nach Abschluss einer Aktion kann eine Verzögerung angesteuert werden.
 * Dadurch kann der Anwender ggf. schneller weiter arbeiten.
 * @portalscript: DEXPRO_WF_CheckDelayAfterAction
 * @returns {boolean} true/false
**/
function ue_DelayAtWorkflowActionEnd(){
    return true;
}
```

# Am Workflow-Ende

```
/** Am Workflow-Ende sollte jeder Vorgang archiviert sein und Rechnungen sollten zum Beispiel den Status "gebucht" haben.
 * Diese User-Exit Funktion wird standardmäßig in der "DEXPRO__USerExit_WorkflowLib" ausgeliefert.
 * Die Prüfungen können beliebig angepasst werden.
 * @portalscript: DEXPRO_WF_CheckDataAtTheEndOfTheWorkflow
 * @return {boolean} true / false.
**/
function ue_CheckFileDataAtTheEndOfTheWorkflow(){
    return true;
}

/** Am Workflow-Ende kann eine Mappe gelöscht ("delete") oder versiegelt ("seal") werden oder
```

```
es passiert nichts mit dem Beleg.
 * Die Information wird in der WEB-Konfiguration zur Workflow ID angegeben und kann über
dieses UserExit pro Mappe manipuliert werden.
 * @portalscript: DEXPRO_WF_End_Seal / DEXPRO_WF_End_Delete
 * @param {string} wfEndType Ermittelte Angabe über die WEB-Konfiguration.
 * @return {string} "delete" / "seal" / "".
**/
function ue_ManipulateWorkflowEndType(wfEndType){
    return wfEndType;
}
```

## Beim Zurücksenden

```
/** Das UserExit wird beim Zurücksenden über das Skript "DEXPRO_WF_CheckSendBack" aufgerufen.
 * In neueren Versionen wurden UserExit-Funktionen pro Modul ergänzt.
 * portalscript: DEXPRO_Action_SendBack / DEXPRO_Action_SendBack_EventReport /
Procurement_Action_SendBackRequester
 * @return {boolean} true / false (Fehler).
**/
function ue_BeforeSendBack(){
// context.errorMessage = "Fehlermeldung";
    return true;
}
/** Nur: DEXPRO_Action_SendBack_EventReport **/
function ue_BeforeSendBack_Invoice(){ return true; }
function ue_BeforeSendBack_Mailroom(){ return true; }
function ue_BeforeSendBack_Procurement(){ return true; }

/** Das UserExit wird beim Zurücksenden über das Skript "DEXPRO_WF_CheckSendBack" aufgerufen.
 * @portalscript: DEXPRO_WF_CheckSendBack
 * @return {string} Leerstring/ Fehlermeldung.
**/
function ue_OnSendBack(){
    return "";
}
```

# Versteckte User-Exits Workflow-Regeln

Bei der Ermittlung der Workflow-Regeln kann an bestimmten Punkten eingegriffen werden.

```
/** Über dieses UserExit kann der Name der Log-Datei angepasst werden.
 * @param {string} logName Aktueller Dateiname
 * @param {string} fileId Aktuelle Mappen-ID
 * @param {string} fileTemplate Mappentyp der aktuellen Mappe
 * @returns {string} Neuer Name für die Log-Datei
**/
function ue_ChangeRulesLogName(logName, fileId, fileTemplate){
	return logName + "_" + fileId + "_" + fileTemplate;
}

/** Über dieses UserExit kann der Zusatz zum Log-Dateinamen angepasst werden.
 * In der Regel wird ein täglicher oder monatlicher Zeitstempel angefügt, damit die einzelnen Logdateien nicht zu groß werden.
 * Der zurückgegebene String wird einfach zum Dateinamen angefügt, falls es nicht einer dieser speziellen Rückgabewerte ist:
 * "year"/"yyyy"/"y" fügt das Aktuelle Jahr hinzu
 * "month"/"mm"/"m" fügt Jahr und Monat hinzu
 * "date"/"day"/"dd"/"d" fügt Jahr, Monat und Tag hinzu
 * "timestamp"/"ts" fügt Jahr, Monat, Tag und Uhrzeit hinzu
 * @param {string} logName Incoming log-name
 * @param {string} fileId Aktuelle Mappen-ID
 * @param {string} fileTemplate Mappentyp der aktuellen Mappe
 * @returns {string} new log add
**/
function ue_ChangeRulesLogAdd(logName, fileId, fileTemplate){
	return "";
}

/** Das User-Exit wird bei den Workflow-Regeln beim Auslesen des Gentable aufgerufen.
 * Über die Funktion können Sonderzeichen in String-Werten ersetzt werden.
```

```
 * Die Funktion wird ausschließlich für String-Werte aufgerufen.
 * @param {string} fieldValue Feldwert
 * @returns {string} Angepasster String-Wert
**/
function ue_FieldConfParseGentableStringValues(fieldValue){
	return fieldValue.replace(/\u001e/g, "").replace(/\u001c/g, "");
}

/** Das UserExit wird direkt nach der Ermittlung der Workflow-Regeln aufgerufen.
 * Der Aufruf erfolgt über das Skript "DEXPRO_WF_Rules_DeterminationUser".
 * Über die Funktion können die ermittelten Regeln manipuliert werden.
 * Der String-Wert 'rulesJSONStr' wird im Anschluss in das Feld 'RulesJSON' geschrieben.
 * @param {string} rulesJSONStr Das Ergebnis der Workflow-Regeln als String.
 * @param {string}workflowid Aktuelle Workflow ID.
 * @param {string} actionid Aktuelle Aktion ID.
 * @param {string} techAction Angabe, ob es sich um eine technische Aktion handelt.
 * @return {string} Angepasste Workflow-Regeln als JSON-string.
**/
function ue_CheckWorkflowRules(rulesJSONStr, workflowid, actionid, techAction){
    var rulesObj = JSON.parse(rulesJSONStr);
    /** An dieser Stelle kann das Objekt manipuliert werden.... **/
    return JSON.stringify(rulesObj);
}

/** Die User-Exit-Funktion wird ausgeführt, wenn eine Aktion über die Workflow-Regeln
übersprungen wird.
 * @returns {boolean} true / false (führt zu einem Workflow-Fehler)
 **/
function ue_OnSkipAction(){
	return true;
}
```

# Versteckte User-Exits: Workflow-Steuerung

Über die Aktions-Konfiguration kann der Anwender definieren ob sich ein Beleg im Ansichtmodus öffnet oder ob der Beleg direkt im Bearbeitungs-Modus geöffnet wird; ob die Mailversendung unterdrückt werden soll und ob die Ablage im Eingangs-Ordner des Benutzers erfolgen soll. Bei Gruppen kann zusätzlich konfiguriert werden ob die Gruppe als Ganzes die Mappe sperrt oder ob die Gruppe aufgelöst wird und die einzelnen Benutzer die Mappe sperren. Zudem kann konfiguriert werden, ob alle Mitglieder der Gruppe oder nur nur ein Mitglied der Gruppe die Aufgabe abschließen muss.

Innerhalb des Prozesses kann es projektspezifisch Teilprozesse geben, bei denen die Konfiguration anders laufen soll. Bei einer über den Workflow gesteuerten Rückfrage (Parameter "**Ask_User_Type**") soll die Mappe zum Beispiel in den Posteingang des Benutzers abgelegt werden - unabhängig von der aktuellen Workflow-Konfiguration. Zudem soll sich der Beleg nicht direkt im Bearbeitungs-Modus öffnen. Dieses Verhalten wird erst ab der Invoice-Version 1.1.000 unterstützt.

Bei anderen projektspezifischen Weiterleitungen kann es ebenfalls gewünscht sein, dass die Aktions-Einstellungen zur Standard-Konfiguration der Workflow-Aktion abweichen. Auch bei der Standard-Zuordnung kann es evtl. gewünscht sein, dass die Konfiguration abhängig von Feldwerten oder abhängig vom Benutzer oder der Gruppe gewählt wird. Um dies zu ermöglichen wurden optionale User-Exit-Funktionen in das Workflow-Routing eingebaut. Technisch werden die Entscheidungen jeweils über 0 und 1 geprüft. Eine Mappe kann nicht im Postingang abgelegt werden (1) oder doch (0) und Sie kann im Bearbeitungs-Modus geönnet werden (1) oder eben nicht (0) und die Mail-Benachrichtigung kann unterdrückt werden (1) oder nicht (0). Hieraus ergeben sich Zahlen-Kombinationen aus 0 und 1.

## Zuordnung Zugriffsprofil

Bei den Zugriffsprofilen wird zuerst geprüft ob die Gruppe aufgelöst werden soll oder nicht. Im Skript ist auch eine Überprüfung der asynchronen Ausführung der Aktion vorgesehen - wird aber bislang nicht weiter berücksichtigt. Bei allen Aktionen ist die Checkbox "Asynchrone Ausführung" deaktiviert.

```
/** Steuerung ob eine Gruppe aufgelöst werden soll oder nicht.
 * Erläuterung zum Code:
 * 1 : asynchrone Ausführung | 0 : keine asynchrone Ausführung (wird derzeit nicht
ausgewertet)
 * 1 : Gruppe auflösen       | 0 : Gruppe sperrt die Mappe
```

```
 * @param {string} code Eingehender Code ("00"/"01"/"10"oder"11")
 * @returns {string} Angepasster Code ("00"/"01"/"10"oder"11")
 **/
function ue_Workflow_ChangeApAsyncDissolveCode(code){
	return code;
}
```

Im Anschluss werden weitere Prüfungen durchgeührt.

```
/** Steuerung Routing für das Zugriffsprofil.
 * Der Workflow unterstützt nur die 4 angegebenen Code-Kombinationen!
 * Bei aufgelösten Gruppen werden nur 2 Kombinationen unterstützt ("01100" und "11100")!
 * Andere Kombinationen können nicht verarbeitet werden!
 * Erläuterung zum Code:
 * 1 : Öffnen im Bearbeitungs-Modus      | 0 : Öffnen im Ansicht-Modus
 * 1 : Aktionsliste anzeigen             | 0 : Aktionsliste ausblenden
 * 1 : Kopierliste anzeigen              | 0 : Kopierliste ausblenden
 * 1 : Mailbenachrichtigung unterdrücken | 0 : Mailbenachrichtigung
 * 1 : Keine Ablage im Posteingang       | 0 : Ablage im Posteingang
 * @param {string} code Eingehender Code ("01100"/"01111"/"11100"oder"11111")
 * @returns {string} Angepasster Code ("01100"/"01111"/"11100"oder"11111")
 **/
function ue_Workflow_ChangeApCode(code){
	return code;
}
```

## Zuordnung Benutzer

Bei der Zuordnung eines Benutzers wird im Workflow zuerst geprüft ob die Mailversendung unterdrückt werden soll oder nicht.

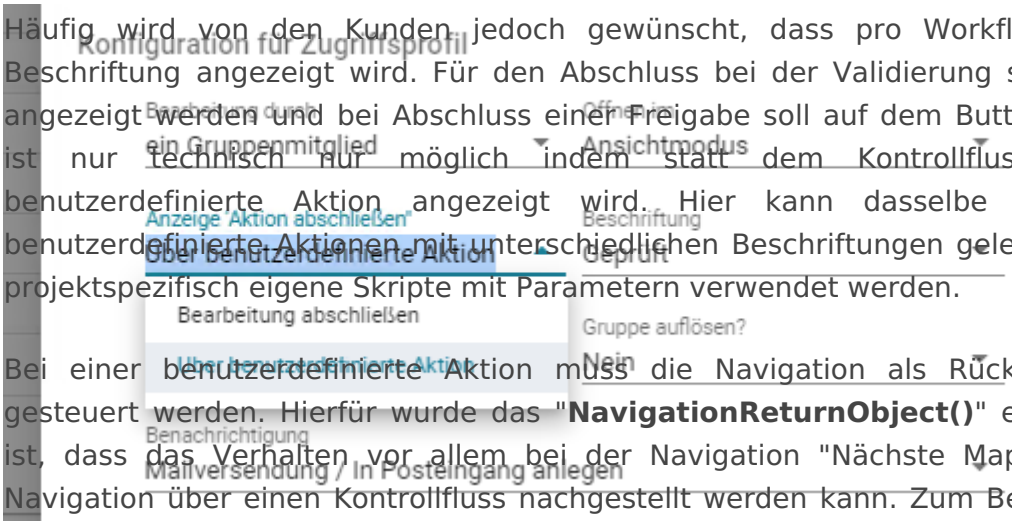
```
/** Steuerung ob die Mailbenachrichtigung bei einem Benutzer unterdrückt werden soll oder
nicht.
 * @param {boolean} Eingehender Wert (true: Mailversendung unterdrücken / false: Mailversendung)
 * @returns {boolean} Ausgehender Wert (true: Mailversendung unterdrücken /
false: Mailversendung)
 **/
function ue_Workflow_UserSuppressMail(suppressMail){
return suppressMail;
}
```

Im zweiten Schritt werden weitere Prüfungen durchgeführt.

```
/** Steuerung Routing für den Benutzer.
 * Der Workflow unterstützt nur die 4 angegebenen Code-Kombinationen!
 * Andere Kombinationen können nicht verarbeitet werden!
 * Erläuterung zum Code:
 * 1 : Öffnen im Bearbeitungs-Modus      | 0 : Öffnen im Ansicht-Modus
 * 1 : Aktionsliste anzeigen             | 0 : Aktionsliste ausblenden
 * 1 : Kopierliste anzeigen              | 0 : Kopierliste ausblenden
 * 1 : Mailbenachrichtigung unterdrücken | 0 : Mailbenachrichtigung
 * 1 : Keine Ablage im Posteingang       | 0 : Ablage im Posteingang
 * @param {string} code Eingehender Code ("0110"/"0111"/"1110"oder"1111")
 * @returns {string} Angepasster Code ("0110"/"0111"/"1110"oder"1111")
 **/
function ue_Workflow_ChangeUserCode(code){
return code;
}
```

# Versteckte User-Exits: Navigation nach Abschluss der Aktion

Bei Abschluss der aktuellen Workflow-Aktion über einen ausgehenden Kontrollfluss gesteuert werden. Der Nachteil dieser Umsetzung ist, dass der Kontrollfluss nur eine Beschriftung möglich ist. Da sich alle Workflow-Aktionen technisch im Hintergrund dieselbe Aktion teilen kann nur eine Beschriftung für den Button (pf:**WfButton_ActionEnd**) gesetzt werden und diese Anzeige gilt dann für alle Workflow-Aktionen wo "Bearbeitung abschließen" ausgewählt wurde. Dafür greift die ausgewählte Navigation ("Mappe beibehalten", "Nächste Mappe", "Zum Eingangsordner") exakt so funktioniert wie man es vom Documents-Workflow kennt.




Häufig wird von den Kunden jedoch gewünscht, dass pro Workflow-Aktion eine spezifische Beschriftung angezeigt wird. Für den Abschluss bei der Validierung soll zum Beispiel "Validiert" angezeigt werden und bei Abschluss einer Freigabe soll auf dem Button "Freigeben" stehen. Das ist nur technisch nur möglich indem statt dem Kontrollfluss-Button ein Button als benutzerdefinierte Aktion angezeigt wird. Hier kann dasselbe Skript auf beliebig viele benutzerdefinierte Aktionen mit unterschiedlichen Beschriftungen gelegt werden. Es können auch projektspezifisch eigene Skripte mit Parametern verwendet werden.

Bei einer benutzerdefinierte Aktion muss die Navigation als Rückgabewert über das Skript gesteuert werden. Hierfür wurde das "**NavigationReturnObject()**" erstellt. Ein großer Nachteil ist, dass das Verhalten vor allem bei der Navigation "Nächste Mappe" nicht 1:1 wie bei der Navigation über einen Kontrollfluss nachgestellt werden kann. Zum Beispiel kann bei selektierten Belegen in einem Ordner nicht automatisch einer der nächsten selektierten Belege angezeigt werden, da "context.selectedFiles" auch bei selektierten Belegen ein leeres FileResultset zurückliefert. Es kann auch nicht irgendein Beleg aus dem aktuellen Ordner angezeigt werden, da "context.folder", "context.folderName" und "context.folderFiles" keine Werte liefern.

Für die Anzeige des nächsten Belegs wird der Aufgaben-Ordner des aktuellen Benutzers ermittelt und es wird aus diesem Ordner ein möglichst passender Beleg ermittelt. Diese Ermittlung kann unter Umständen lange dauern, wenn der Anwender sehr viele Belege in seinen Aufgaben hat. Projektspezifisch könnte der neu anzuzeigende Beleg zum Beispiel über einen spezifischen Filterordner ermittelt werden und der Filterordner kann anhand einiger Daten wie die Aktion-ID und ggf. weiteren Informationen wie dem Mandanten abgeleitet werden.

Für solche Umsetzungen wird das NavigationReturnObject() ab der Invoice Version 1.1. um optionale UserExit-Funktion pro Navigation erweitert. Sobald eine der Funktionen definiert ist und

diese Funktion den Wert true zurückliefert ersetzt diese das Navigations-Standardverhalten! Folgende UserExit-Funktionen wurden ergänzt:

- **ue_Keep()**
- **ue_Next()**
- **ue_Overview()**
- **ue_Folder()**
- **ue_Inbox()**

Alle Funktionen haben denselben Aufbau - daher wird nur eine der Funktionen ein Beispiel-Skript angezeigt.

> Wenn eine UserExit-Funktion verwendet wird, dann wird der Standard-Code nicht ausgeführt! Die Rückgabe-Werte müssen in jedem Fall durch die UserExit-Funktion gesetzt werden!

```
/** Rückgabewert für die Navigation 'Mappe beibehalten'.
 * Diese Funktion zeigt ein Beispiel wie man nach der Validierung den nächsten Validierungs-Beleg anzeigen kann.
 * @returns {boolean} true (verwende UserExit Navigation) / false (verwerfe Rückgabewerte und ermittle nach Standard)
 **/
NavigationReturnObject.prototype.ue_Next(){
  if( this.ActionID==="ValidationInvoice" ){
    var folderIter = context.getFoldersByName("ValidationInvoice", "dynamicpublic");
      if( folderIter && folderIter.size()===1 ){
          var folder = folderIter.first();
          var files  = folder.getFiles();
            /* ... nächste Mappe ermitteln. */
          var showFile = ...
    this.ReturnType = "multipleAction";
this.ReturnVal  = JSON.stringify([
            { returnType : 'showFolder', returnValue : folder.id },
                { returnType : 'showFile', returnValue : showFile.getid() }
            ]);
            return true;
        }
    }
  return false;
}
```

Das NavigationReturnObject() hat Folgende Attribute:

| Attribut | Beschreibung | Voreinstellung |
|---|---|---|
| **ReturnType** | Rückgabetyp (context.returnType) | "stay" |
| **ReturnVal** | Rückgabewert | |
| **ForwardAction** | Weiterleitungs-Art | "FinishAction"/"Forward"/"AnswerQuesti |
| **Navigation** | Navigation | "keepfile"/"next"/"overview"/"inbox"/"fol |
| **DocFile** | Aktuelle Mappe | context.file |
| **DocFileID** | Documents ID der aktuellen Mappe | context.file.getid() |
| **Filetype** | Mappentyp der aktuellen Mappe | context.file.getAttribute("FromTemplate |
| **ActionID** | Aktion ID der aktuellen Mappe | context.file.ActionID |
| **WorkflowID** | Workflow ID der aktuellen Mappe | context.file.WorkflowID |
| **SystemUser** | Aktueller Benutzer als SystemUser Objekt | context.getSystemUser() |
| **MonoView** | Angabe, ob beim aktuellen Benutzer der MonoView aktiv ist (ab Invoice 1.1.015). | ""/"true" |

Am Ende des Aufrufs gibt es ein zusätzliches User-Exit, über welches generell jegliche Navigation nochmals manipuliert werden kann. Das gilt auch für die oben genannten UserExits. Dieses UserExit bietet sich zum Beispiel an, wenn bestimmte Benutzer immer dieselbe Navigation haben möchten.

```
/** Manipulation des Objekts am Funktions-Ende.
 * @returns Kein Rückgabewert!
**/
NavigationReturnObject.prototype.ue_NavigationManipulation = function(){
  this.Log += "[INFO] function ue_NavigationManipulation()\n";
  if( this.SystemUser instanceof SystemUser ){
    this.Log += "[INFO] valid system user\n";
    var lastUsedFolder = this.SystemUser.getPrivateFolder("lastused");
    if( lastUsedFolder ){
            /* Dieses Beispiel öffnet den privaten Ordner 'Zuletzt benutzt' und zeigt die
aktuelle Mappe an. */
      this.ReturnType = "multipleAction";
```

```
        this.ReturnVal  = JSON.stringify( [
            { returnType : 'showFolder', returnValue :  lastUsedFolder.id },
            { returnType : 'showFile', returnValue :  this.DocFile.getid() }
                ] );
            }
        else{
            this.Log += "[ERROR] invalid private folder(lastused)\n";
        }
    }
    else{
        this.Log += "[ERROR] invalid system user\n";
    }
    util.log(this.Log);
}
```

# Versteckte User-Exits Invoice-Jobs

## Invoice_JOB_StartWorkflow

Bei der Übergabe von Squeeze an den Documents-Workflow kann es zu Problemen mit der Anzahl der erlaubten SOAP-Sessions kommen. Bei einer Überschreitung der erlaubten Sessions (im Standard sind es 3) werden alle Sessions abgebrochen. Wenn ein Workflow bereits gestartet wurde, dann fehlen im Workflow-Verlauf abrupt die Zugriffsberechtigungen. Die Workflow-Skripte lassen den Vorgang in einen Fehler-Status laufen. durch die negative Rückmeldung werden Belege teils mehrfach übergeben.
Bei der Anlage einer neuen Mappe wurde bislang immer direkt der Workflow gestartet. Das führt allerdings dazu, dass über die SOAP-Session nicht nur der Beleg erzeugt wird, sondern auch der Workflow gestartet wird. Je mehr Zeit eine Session benötigt, desto größer ist das Risiko, dass die Sessions abbrechen und es zu Fehlern kommt. Als Lösung wurde bereits eine initiale Verzögerung in den Workflow eingebaut.
Um noch mehr Zeit zu sparen kann der Workflow auch nachgelagert über diesen Job gestartet werden. Das Skript setzt context.superMode(true) und sucht via FileResultset nach allen Rechnungen mit leerer WorkflowID. In einer Schleife wird zunächst nochmal das Attribut "InCirculation" an der Mappe überprüft.

```
/** Diese optionale Funktion wird pro Mappe ausgeführt.
 * Wenn der Rückgabewert ungleich true ist wird der Workflow nicht gestartet!
 * Das UserExit wird auch über die Skripte "Invoice_Action_StartWorkflow" und
"Invoice_Action_Folder_UDA_StartWorkflow" ausgeführt.
 * Diese werden im Kontext eines Benutzers ausgeführt und in den Fällen muss bei Fehlern
context.errorMessage gesetzt werden,
 * damit der Anwender eine Fehlermeldung erhält.
 * @param {Log} log Logging-Objekt * @returns {boolean} true / false (Workflow wird nicht
gestartet)
 * @since Invoice 1.1.015
**/
function ue_beforeStartWorkflow(docFile, log){
log.info("["+docFile.getid()+"] function ue_beforeStartWorkflow(docFile, log)");
return true;
}
```

## Invoice_JOB_CheckPostingStatus

Der Job überprüft den Buchungs-Status (Spalte "**PostingStatus**") in der Tabelle "**Invoice_PostingHead**" für alle Rechnungs-Mappen mit der Aktions-ID "**Posting**" und dem Aktions-Status "**TechActionJob**". Aus der Tabelle werden folgende Feldwerte in die Mappe übernommen:

- PostingStatus
- PostingNumber
- PostingPeriod
- PostingDate
- PostingUser
- PostingError

Bei einem Fehler-Status ("**error**") wird der Beleg zur im Feld "**ActionAccessProfile**" hinterlegten Gruppe zur Nachbearbeitung gesendet. Technisch kann in das Feld "ActionAccessProfile" auch ein Benutzer-Login geschrieben werden. Bei erfolgter Buchung (Status "**posted**") wird der Beleg zur nächsten Workflow-Aktion gesendet.

Im Skript wird zunächst das FileResultset ermittelt. Erst wenn mindestens 1 Beleg gefunden wurde wird ein Log erzeugt und erst dann werden auch die folgenden User-Exit-Funktion ausgeführt. Vor dem FileResultset wird "**context.setSuperMode(true)**" gesetzt. Alle User-Exits werden im Super-Mode ausgeführt! Der Job importiert die "**Invoice__ImportLib**". Die UserExits können demnach zur "**Invoice__UserExit_CustomLib**" hinzugefügt werden.

```
/** Diese optionale Funktion wird noch vor der FileResultset-Schleife ausgeführt und kann
verwendet werden,
 * um zum Beispiel den Buchungs-Status aus einem externen System zu übertragen.
 * @param {Log} log Logging-Objekt
 * @returns {} Die Rückgabe wird nicht ausgewertet
 * @since Invoice 1.1.015
**/
function ue_checkPostingStatus_BeforeLoop(log){
  log.info("["+docFile.getid()+"] function ue_checkPostingStatus_BeforeLoop(log)");
}

/** Diese optionale Funktion wird direkt zu Beginn in der Schleife ausgeführt.
 * @param {DocFile} docFile Aktuelle Rechnungs-Mappe im FileResultset
 * @param {Log} log Logging-Objekt
 * @returns {boolean} true / false (durch false wird die Überprüfung der Mappe abgebrochen)
 * @since Invoice 1.1.015
```

```
**/
function ue_checkPostingStatus_DocFileStart(docFile, log){
  log.info("["+docFile.getid()+"] function ue_checkPostingStatus_DocFileStart(docFile, log)");
  return true;
}

/** Diese optionale Funktion wird nur ausgeführt, wenn der Buchungsstatus in der Tabelle "error" zurückliefert.
 * Die Feldwerte werden gesetzt aber die Mappe wurde noch nicht weitergeleitet.
 * @param {DocFile} docFile Aktuelle Rechnungs-Mappe im FileResultset
 * @param {Log} log Logging-Objekt
 * @returns {boolean} true / false (durch false wird die Mappe nicht an die Gruppe zur Überprüfung weitergeleitet)
 * @since Invoice 1.1.015
**/
function ue_checkPostingStatus_OnPostingError(docFile, log){
  log.info("["+docFile.getid()+"] function ue_checkPostingStatus_OnPostingError(docFile, log)");
  return true;
}

/** Diese optionale Funktion wird nur ausgeführt, wenn der Buchungsstatus in der Tabelle "posted" zurückliefert.
 * Die Feldwerte werden gesetzt aber die Mappe wurde noch nicht weitergeleitet.
 * @param {DocFile} docFile Aktuelle Rechnungs-Mappe im FileResultset
 * @param {Log} log Logging-Objekt
 * @returns {boolean} true / false (durch false wird die Mappe nicht zur nächsten Workflow-Aktion weitergeleitet)
 * @since Invoice 1.1.015
**/
function ue_checkPostingStatus_Posted(docFile, log){
  log.info("["+docFile.getid()+"] function ue_checkPostingStatus_Posted(docFile, log)");
  return true;
}

/** Diese optionale Funktion wird nur ausgeführt, wenn der Buchungsstatus in der Tabelle "posted" zurückliefert.
 * Die Feldwerte werden gesetzt und die Mappe erfolgreich weitergeleitet.
 * @param {DocFile} docFile Aktuelle Rechnungs-Mappe im FileResultset
```

```
 * @param {Log} log Logging-Objekt
 * @returns {} Die Rückgabe wird nicht ausgewertet
 * @since Invoice 1.1.015
**/
function ue_checkPostingStatus_PostedAndForwarded(docFile, log){
  log.info("["+docFile.getid()+"] function ue_checkPostingStatus_PostedAndForwarded(docFile, log)");
  return true;
}
```

## Invoice_JOB_Archiving

Der Job archiviert die Belege, die eine der folgenden Aktions ID haben:

- Archiving, Archiving1, Archiving2, Archiving3, ... Archiving14, Archiving15

```
/** Diese Funktion archiviert die Rechnungen.
 * Die Funktion wird bei allen Archivierungsvorgängen - auch beim Aussteuern - aufgerufen.
 * Die Funktion wird in der DEXPRO__USerExit_TechActionLib ausgeliefert.
 * @param {string} type Über den Aufruf aus diesem Job wird immer "Job" mitgegeben.
 * @param {DocFile} docFile Aktuelle Rechnungs-Mappe im FileResultset
 * @param {Log} log Optionales Logging-Objekt
 * @returns {boolean} true (Archiviert) / false (Fehler)
**/
function ue_Archiving(type, docFile, log){
  log.info("["+docFile.getid()+"] function ue_Archiving("+type+", docFile, log)");
  if( docFile.archive() ){
    log.info("["+docFile.getid()+"] archived("+docFile.getArchiveKey()+")");
return true;
  }
  log.err("["+docFile.getid()+"] " + docFile.getLastError();
  return false;
}
```

# Versteckte User-Exits: otrUser / otrAccessProfile

Anzeige der benutzerdefinierten Aktionen:

```
/** Anzeige der projektspezifischen benutzerdefinierten Aktionen am Mappentypen otrUser.
 * Die Standard-Aktionen können über diese User-Exit-Funktion nicht verändert werden!
 * Die Funktion wird für jede projektspezifische benutzerdefinierte Aktion separat
aufgerufen.
 * @param {int} e Aktueller Aufzählungswert von enumval
 * @param {string} enumValue Technischer Name der benutzerdefinierten Aktion
 * @param {SystemUser} su SystemUser-Objekt zur aktuellen Akte
 * @since Invoice 1.0.300
 **/
function ue_SystemUser_UDA( e, enumValue, su ){
  /* Beispiel */
  if( enumValue==="MeineBenutzerdefinierteAktion" ){
    if( su instanceof SystemUser ){
      if( su.hasAccessProfile("Administration") ){
      return;
      }
    }
  }
  enumval[e] = "";
}
```

```
/** Anzeige der projektspezifischen benutzerdefinierten Aktionen am Mappentypen
otrAccessProfile.
 * Die Standard-Aktionen können über diese User-Exit-Funktion nicht verändert werden!
 * Die Funktion wird für jede projektspezifische benutzerdefinierte Aktion separat
aufgerufen.
 * @param {int} e Aktueller Aufzählungswert von enumval
 * @param {string} enumValue Technischer Name der benutzerdefinierten Aktion
 * @param {AccessProfile} ap AccessProfile-Objekt zur aktuellen Akte
 * @since Invoice 1.0.300
```

```
 **/
function ue_AccessProfile_UDA( e, enumValue, ap ){
  /* Beispiel */
  if( enumValue==="MeineBenutzerdefinierteAktion" ){
    if( ap instanceof AccessProfile ){
      if( ap.name==="Administration" ){
      return;
      }
    }
  }
  enumval[e] = "";
}
```

Erweiterung der Eigenschaften-Liste:

```
/** Erweiterung der Eigenschafts-Liste bei der benutzerdefinierten Aktion "UDA_SetProperty"
am Mappentypen otrUser.
 * @param {Array} enumval Aufzählungswerte
 * @since Invoice 1.0.300
 **/
function ue_ExtendSystemUserPropertyList( enumval ){
  /* Beispiel */
  enumval.push("MyProjectSpecificProperty;MyProjectSpecificProperty
(ValueOption1|ValueOption2)");
  enumval.sort();
}
```

# Verstecke User-Exits Logging

Einige Skripte schreiben Loginformationen in Textdateien unter "**..\DEXPRO\Logs\..**". Über die folgenden User-Exits kann der Pfad bzw. der Dateiname manipuliert werden werden.

```
/** Über dieses UserExit kann der Log-Pfad generell für alle internen Logs umgestellt werden!
 * @param {string} path Aktueller Pfad.
 * @return {string} Angepasster Pfad.
**/
function ue_ChangeLogPath(path){
    if( context.scriptName==="MeinSkriptName" ){
        return "E:\\Logs";
    }
    return path;
}

/** Über dieses UserExit kann der Name der Log-Datei angepasst werden.
 * @param {string} logName Aktueller Dateiname
 * @param {string} fileId Aktuelle Mappen-ID
 * @param {string} fileTemplate Mappentyp der aktuellen Mappe
 * @returns {string} Neuer Name für die Log-Datei
**/
function ue_ChangeRulesLogName(logName, fileId, fileTemplate){
return logName + "_" + fileId + "_" + fileTemplate;
}

/** Über dieses UserExit kann der Zusatz zum Log-Dateinamen angepasst werden.
 * In der Regel wird ein täglicher oder monatlicher Zeitstempel angefügt, damit die einzelnen Logdateien nicht zu groß werden.
 * Der zurückgegebene String wird einfach zum Dateinamen angefügt, falls es nicht einer dieser speziellen Rückgabewerte ist:
 * "year"/"yyyy"/"y" fügt das Aktuelle Jahr hinzu
 * "month"/"mm"/"m" fügt Jahr und Monat hinzu
 * "date"/"day"/"dd"/"d" fügt Jahr, Monat und Tag hinzu
 * "timestamp"/"ts" fügt Jahr, Monat, Tag und Uhrzeit hinzu
 * @param {string} logName Incoming log-name
 * @param {string} fileId Aktuelle Mappen-ID
```

```
 * @param {string} fileTemplate Mappentyp der aktuellen Mappe
 * @returns {string} new log add
**/
function ue_ChangeRulesLogAdd(logName, fileId, fileTemplate){
return "";
}
```

# Versteckte User-Exits: 3-Way-Match

Bei der Prüfung der Rechnungsmenge gegen die Bestellmenge und den Wareneingang wird pro Rechnungszeile ein **QuantityObject** erstellt und die Zeile wird gegen die Bestellung und gegen den Wareneingang geprüft. Die Prüfung bietet einige Eingriffspunkt zur Manipulation. Die Prüfung wird pro Gentable-Zeile ausgeführt. Das Objekt enthält das DocFile-Objekt ("**DocFile**"), die aktuelle Zeilennummer ("**RowNo**") und die Informationen zur Gentable-Zeile ("**OrderNumber**", "**OrderPosition**", "**GoodsReceiptID**", "**GoodsReceiptPos**", "**GoodsReceiptCheck**", "**ExtraLine**"). Zudem enthält das Objekt die Prüfergebnisse. Bei einer Abweichung zur Bestellung gibtt es die Bool-Werte "**OrderResult**", "**OpenQuantityResult**" und die Fehlermeldung wird in "**OrderError**" ausgegeben.
Bei der Prüfung gegen den Wareneingang sind es entsprechend "**GoodsReceiptResult**" und "**GoodsReceiptError**".

**QuantityObject.prototype.ue_3WayMatch_ParseQuantity = function(){ ... }**

Vor dem Vergleich der Mengen können die Mengen umgerechnet werden. das **QuantityObject** enthält die Menge aus der Rechnungsposition ("**Quantity**") , die Bestellmenge ("**QuantityOrdered**") sowie die gelieferte Menge ("**QuantityDelivered**"). Zudem gibt es die verbrauchten Mengen ("**UsedOrderedQuantity**" / "**UsedDeliveredQuantity**") und die offene Bestellmenge ("**OpenOrderedQuantity**") sowie die offene Liefermenge ("**OpenDeliveredQuantity**"). Darüber hinaus gibt es auch die Mengeneinheiten ("**QuantityUnit**" / "**QuantityUnitOrdered**" / "**QuantityUnitDelivered**").

Nach dem Aufruf der UserExit-Funktion werden "**QuantityUnit**" und "**QuantityUnitOrdered**" verglichen. Bei einer Abweichung wird eine Fehlermeldung ausgegeben. Zudem gibt es eine Fehlermeldung, wenn die "**Quantity**" größer als die "**QuantityOrdered**" ist. Die Prüfung gegen den Wareneingang erfolgt nur, wenn "**GoodsReceiptCheck**" den Wert "**true**" hat. Hier werden ebenfalls die Mengeneinheiten abgeglichen und die Menge darf die gelieferte Menge nicht überschreiten. Eine Fehlermeldung wird ausgegeben, wenn die Preisabweichung größer **0** ist.

**QuantityObject.prototype.ue_3WayMatch_CheckPriceDiff = function( priceCalculated, priceOrdered ){ ... }**

Über die Funktion können bei Preisabweichungen sehr spezifische Toleranz-Regeln umgesetzt werden. Bei den Preisen gibt es nur die Werte aus der Rechnungsposition ("**Price**", "**PriceUnit**" )

und aus der Bestellung ("**PriceOrdered**", "**PriceUnitOrdered**"). Aus Preis und Preiseinheit ergibt sich der kalkulierte Preis ("**PriceCalc**" / "**PriceCalcOrdered**") und die Preisdifferenz ("**PriceDiff**").

**QuantityObject.prototype.ue_3WayMatch_CheckEnd = function(){ ... }**

Wenn die Funktion aufgrund eines Fehlers abbricht, dann wird diese Funktion am Ende nicht mehr ausgeführt! Über die Funktion können die Ergebnisse manipuliert werden und es können weitere Prüfungen ausgeführt werden.

```
/** User exit after loading order master data to manipulate or to add order information.
 * Called: docFile.replacePositionsByHeadOrderValue()
 * Script: Invoice_Action_GetOpenOrderPosition
 * @param {string} optExecType Optional parameter to set execution type information
**/
Gentable.prototype.adjustLoadedOrderPositions = function(optExecType){
    /*
    var id    = (this.DocFile)? this.DocFile.getid() : "Unknown";
    this.Log += "[INFO]["+id+"] function adjustLoadedOrderPositions()"+this.LineBreak;
    for( var r=0; r<this.Rows.length; r++ ){

    }
    */
}
```

# Aufzählungs-Skripte (enumval)

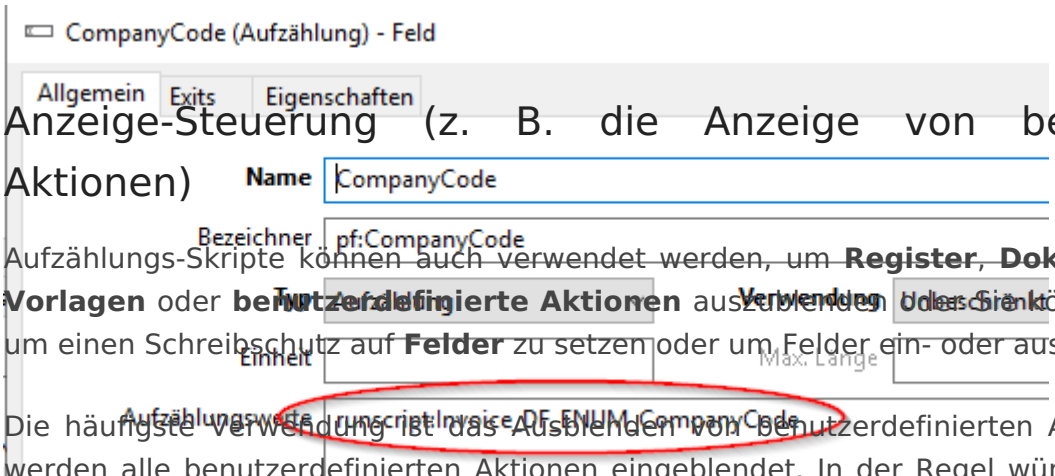
In Documents können Skripte fürAufzählungen erstellt werden. Das Ergebnis kann zum Beispiel als Aufzählungsliste in einem Feld verwendet werden. In Aufzählungs-Skripten gibt es immer das Array "**enumval**". Aufzählungs-Skripte sind Documents-Standard. Weitere Informationen finden Sie entsprechend beim Hersteller.

## Verwendung in Auswahllisten-Feldern

Bei Aufzählungswerten für ein Auswahllisten-Feld kann zwischen einem technischen Wert und einem Anzeigewert unterschieden werden. Die Angaben werden durch ein "**;**" getrennt. Das folgende Beispiel speichert im Auswahllisten-Feld den Wert "0" oder "1". Dem Benutzer wird je nach Anmeldesprache ein Text angezeigt.

```
// Einfaches Beispiel für ein Aufzählungsskript mit technischem Wert und Anzeige-Wert
enumval.push("0;de:Aus;en:Off");
enumval.push("1;de:An;en:On")
```

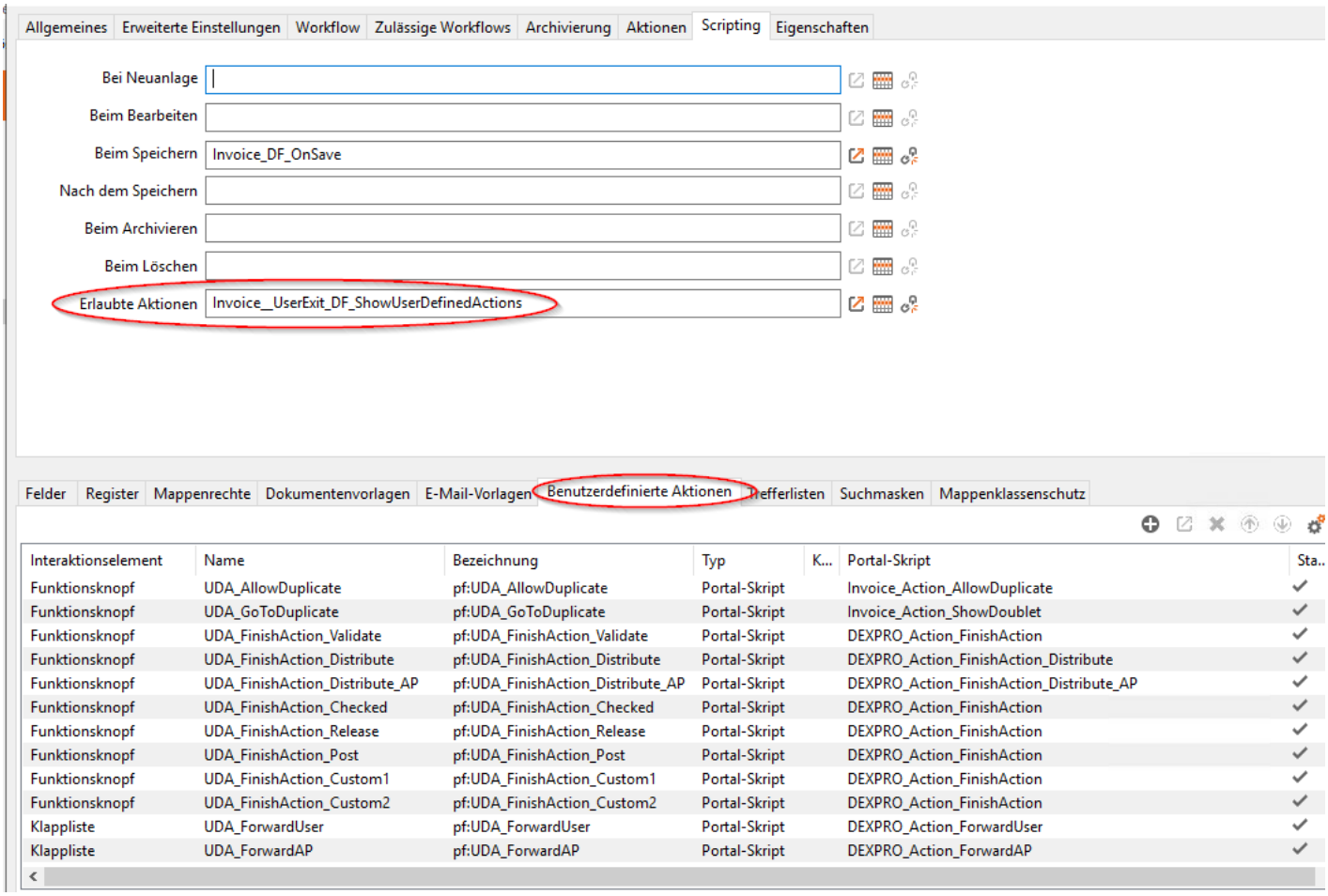
Die Skripte werden so eingebunden:




## Anzeige-Steuerung (z. B. die Anzeige von benutzerdefinierten Aktionen)

Aufzählungs-Skripte können auch verwendet werden, um **Register**, **Dokumenten**- oder **Email-Vorlagen** oder **benutzerdefinierte Aktionen** auszublenden. Sie können verwendet werden um einen Schreibschutz auf **Felder** zu setzen oder um Felder ein- oder auszublenden.

Die häufigste Verwendung ist das Ausblenden von benutzerdefinierten Aktionen. Standardmäßig werden alle benutzerdefinierten Aktionen eingeblendet. In der Regel wünscht man sich eher den umgekehrten Fall, dass die Aktionen nur zum passenden Zeitpunkt für die passenden Benutzer angezeigt werden. Hierfür kann am Mappentypen auf dem Register "Scripting" ein Skript hinterlegt werden.




| Interaktionselement | Name | Bezeichnung | Typ | K... | Portal-Skript | Sta.. |
|---|---|---|---|---|---|---|
| Funktionsknopf | UDA_AllowDuplicate | pf:UDA_AllowDuplicate | Portal-Skript | | Invoice_Action_AllowDuplicate | ✓ |
| Funktionsknopf | UDA_GoToDuplicate | pf:UDA_GoToDuplicate | Portal-Skript | | Invoice_Action_ShowDoublet | ✓ |
| Funktionsknopf | UDA_FinishAction_Validate | pf:UDA_FinishAction_Validate | Portal-Skript | | DEXPRO_Action_FinishAction | ✓ |
| Funktionsknopf | UDA_FinishAction_Distribute | pf:UDA_FinishAction_Distribute | Portal-Skript | | DEXPRO_Action_FinishAction_Distribute | ✓ |
| Funktionsknopf | UDA_FinishAction_Distribute_AP | pf:UDA_FinishAction_Distribute_AP | Portal-Skript | | DEXPRO_Action_FinishAction_Distribute_AP | ✓ |
| Funktionsknopf | UDA_FinishAction_Checked | pf:UDA_FinishAction_Checked | Portal-Skript | | DEXPRO_Action_FinishAction | ✓ |
| Funktionsknopf | UDA_FinishAction_Release | pf:UDA_FinishAction_Release | Portal-Skript | | DEXPRO_Action_FinishAction | ✓ |
| Funktionsknopf | UDA_FinishAction_Post | pf:UDA_FinishAction_Post | Portal-Skript | | DEXPRO_Action_FinishAction | ✓ |
| Funktionsknopf | UDA_FinishAction_Custom1 | pf:UDA_FinishAction_Custom1 | Portal-Skript | | DEXPRO_Action_FinishAction | ✓ |
| Funktionsknopf | UDA_FinishAction_Custom2 | pf:UDA_FinishAction_Custom2 | Portal-Skript | | DEXPRO_Action_FinishAction | ✓ |
| Klappliste | UDA_ForwardUser | pf:UDA_ForwardUser | Portal-Skript | | DEXPRO_Action_ForwardUser | ✓ |
| Klappliste | UDA_ForwardAP | pf:UDA_ForwardAP | Portal-Skript | | DEXPRO_Action_ForwardAP | ✓ |

Das Array "**enumval**" listet beim Skript-Aufruf bereits alle benutzerdefinierten Aktionen mit dem technischen Namen, denn wie erwähnt werden alle benutzerdefinierten Aktionen standardmäßig immer angezeigt.

"**enumval[0]**" enthält zum Beispiel den ersten Eintrag. Wenn man den Screenshot als Beispiel nimmt würde der String "**UDA_AllowDuplicate**" der zugehörige Wert sein. Das Array wird in der Regel mit einer einfachen `for`-Schleife durchlaufen und innerhalb der Schleife kann via `if()`-Abfragen oder via `switch() .. case` geprüft werden um welche Aktion es sich handelt. Um ein Aktion auszublenden muss der Wert auf einen Leerstring "" gesetzt werden.

```
/* ================================================================ */
/* Iterate user defined actions                                     */
/* ================================================================ */
for( var uda=0; uda<enumval.length; uda++ ){
    switch( enumval[uda] ){

        /* Start release workflow */
        case "UDA_FinishAction_StartRelease":
            if( inCirculation===true && isArchiveFile===false && cu.login!==docFile.Requester || actionID!=="CreateProcurement" ){
                enumval[uda]="";
            }
            break;
        case "UDA_SaveAsStandardAddress":
            if( inCirculation===true && isArchiveFile===false && cu.login!==docFile.Requester || actionID!=="CreateProcurement" ){
                enumval[uda]="";
            }
            break;

        /* Zurück an den Anforderer */
        case "UDA_ReturnToRequester":
            if( inCirculation===true && isArchiveFile===false ){
```

In den ausgelieferten Modulen wird das Skript für die Anzeige der benutzerdefinierten Aktionen

generell als UserExit-Skript ausgeliefert, denn in diesem Skript müssen in den Projekten fast immer Anpassungen vorgenommen werden. Für gewöhnlich werden Feldwerte oder Eigenschaften der aktuellen Mappe und/oder Gruppenzugehörigkeiten des aktuell angemeldeten Benutzers abgefragt und ausgewertet.

Diese Skripte werden relativ häufig aufgerufen. Damit das Kundensystem performant bleibt sollten aufwändige Prozeduren wie Abfragen auf externe Systeme, aufwändige Berechnungen oder `sync()`-Befehle möglichst vermieden werden.

> Aufzählungs-Skripte sind Documents-Standard.
> Weitere Informationen können in der Hersteller Dokumentation nachgelesen werden.

# Versteckte User-Exits: Zugriffsberechtigungen manipulieren (GACL)

Die Mappentypen "**Invoice**", "**Procurement**" und "**Mailroom**" verwenden den Mappenklassenschutz und im Standard ist hier das Text-Feld "**Rights**" hinterlegt. Der Mappenklassenschutz berücksichtigt Zugriffsprofile und Benutzer-Logins. Standardmäßig setzen sich die Berechtigungen aus den Feldern "**RightsInitial**" und "**RightsWorkflow**" zusammen. Die Berechtigungen werden bei einer Weiterleitung automatisch erweitert. Jede Gruppe bzw. jeder Benutzer welcher am Workflow teilnimmt erhält automatisch Zugriffsrechte.

Im Feld "**RightsInitial**" werden die initialen Berechtigungen festgehalten, welche über die entsprechende WEB-Konfiguration gesetzt werden können. Diese Rechte werden bei einer weiterleitung neu ermittelt! Wenn initiale Berechtigungen abhängig von Feldwerten definiert wurden (zum Beispiel vom Mandanten-Feld) und sich der Feldwert in einer Mappe ändert, dann können neue Zugriffsberechtigungen hinzukommen und es kann auch vorkommen, dass Zugriffsberechtigungen wieder entzogen werden!

Das Feld "**RightsWorkflow**" speichert hingegen die Gruppen und Benutzer, die am Workflow zum Vorgang teilgenommen haben. Diese Liste wird immer weiter fortgeführt.

Die Zugriffsberechtigungen werden technisch über das Skript-Objekt "**RightsGACL**" gesteuert. Dieses Objekt wird mit dem DocFile-Objekt als Parameter aufgerufen und enthält folgende Methoden:

- **addWorkflowEntry**(String login/tech ap name)
- **addAccessProfile**(AccessProfile)
- **addSystemUser**(SystemUser, Boolean add agent in case of absence)
- **checkRights**(SystemUser)
- **createGACLString**()
- **readInitialRights**()
- **readRightsFromField**(String fieldName)

Das Objekt hat folgende Eigenschaften:

| Eigenschaft | Typ | Beschreibung |
|---|---|---|
| Error | String | Fehlermeldung |

| | | |
|---|---|---|
| Result | Boolean | Ergebnis |
| DocFile | DocFile-Objekt | Aktuelle Mappe |
| DocFileID | DocFile-ID | Aktuelle Mappen-ID |
| Log | String | Logging-Informationen |
| RightsInitial | Objekt | Objekt mit den initialen Berechtigungen |
| RightsInitialStr | String | GACL-String mit den initialen Berechtigungen |
| RightsWF | Objekt | Objekt mit den Workflow-Berechtigungen |
| RightsWFStr | String | GACL-String mit den Workflow-Berechtigungen |

Die Methode "**createGACLString**()" liefert den fertigen GACL-String (ohne diesen direkt in das Feld "**Rights**" zu schreiben). Ab Invoice ***V.1.1.200*** enthält diese Funktion 2 optionale UserExit-Funktionen, um das GACL-Ergebnis zu manipulieren. Die Funktionen können bei Bedarf in eine UserExit-Bibliothek hinzugefügt werden.

```
/** Die Funktion wird nach dem Erstellen des GACL-Strings ausgeführt.
**/
RightsGACL.prototype.ue_BeforeCreateGACLString = function(){

}

/** Die Funktion wird nach dem Erstellen des GACL-Strings ausgeführt.
 * In dem Beispiel wird ein zusätzliches Feld 'AdditionalRights' mit zusätzlichen
Zugriffsberechtigungen ausgewertet.
**/
RightsGACL.prototype.ue_AfterCreateGACLString = function(){
    if( this.DocFile.hasField("AdditionalRights") ){
        this.Log += "[INFO]["+this.DocFileID+"] has field 'AdditionalRights'" +
this.LineBreak;

        var userObj = {};
        var gaclArr = this.GACL.split("\r\n");
        for( var i=0; i<gaclArr.length; i++ ){
            var su = context.findSystemUser(gaclArr[i]);
            if( su instanceof SystemUser ){
                userObj[su.login] = su.login;
            }
```

```
        }

        var userArr = this.DocFile.AdditionalRights.split("\r\n");
        for( var j=0; j<userArr.length; j++ ){
            var su2 = context.findSystemUser(userArr[j]);
            if( su2 instanceof SystemUser ){
                if( userObj[su2.login] ){
                    this.Log += "[INFO]["+this.DocFileID+"]["+j+"] system user("+su2.login+")
already has rights" + this.LineBreak;
                }
                else{
                    this.Log  += "[INFO]["+this.DocFileID+"]["+j+"] system
user("+su2.login+") add GACL rights" + this.LineBreak;
                    this.GACL += su2.login+"\r\n";
                }
            }
        }
    }
    //util.out(this.Log);
}
```

# Versteckte User-Exits: 3-Way-Match (Warten auf Wareneingang)

Bei Rechnungen mit Bestellbezug erfolgt die Versendung der Rechnung häufig zeitgleich oder noch vor der Warenversendung. Selbst wenn die Rechnung zeitgleich mit der Ware ankommt, erfolgt die Wareneingangserfassung häufig erst nachdem die Rechnung im Workflow angelegt wurde. In der technischen Workflow-Aktion "**Wait4IncomingGoods**" ("Warten auf Wareneingang") können Rechnungen mit Bestellbezug eine definierte Anzahl an Tagen auf den Wareneingang warten.

Das Jobskript "**Invoice_JOB_CheckGoodsReceipt**" iteriert die auf Wareneingang wartenden Rechnungen mit Bestellbezug. Das Skript überprüft zunächst, ob jeder Rechnungsposition eine Bestell-Position zugeordnet wurde, denn ohne die Zuordnung zu einer Bestellposition kann der Position kein Wareneingang zugeordnet werden (ausgenommen sind als Zu-Abschlagszeilen markierte Positionen). Die Prüfung übernimmt die Funktion "Gentable.checkOrder()". Vor dieser Prüfung wird jetzt die UserExit-Funktion **Gentable.ue_BeforeGoodsReceivedCheck()** ausgeführt.

```
/** Diese UserExit-Funktion wird bei allen Rechnungen ausgeführt,
 * die im Workflow-Schritt "Warten auf Wareneingang" liegen.
 * @param {Log} log Logging-Objekt
 * @since Invoice 1.1.300
**/
Gentable.prototype.ue_BeforeGoodsReceivedCheck = function(log){
  ;
}
```

Wenn allen Positionen eine Bestell-Position zugeordnet wurde, dann werden über Gentable.checkIncomingGoods() die Wareneingänge geprüft und im Anschluss wird die neue UserExit-Funktion **Gentable_ue_AfterGoodsReceivedCheck()** ausgeführt.

```
/** Diese UserExit-Funktion wird bei den Rechnungen ausgeführt,
 * die im Workflow-Schritt "Warten auf Wareneingang" liegen
 * und wo allen Positionen eine Bestellung zugeordnet wurde.
 * Die Prüfung auf einen Wareneingang (Gentable.checkIncomingGoods()) wurde bereits
ausgeführt.
```

```
 * @param {Log} log Logging-Objekt
 * @since Invoice 1.1.300
**/
Gentable.prototype.ue_AfterGoodsReceivedCheck = function(log){
  ;
}
```

Wenn einer Rechnungs-Position keine Bestell-Position zugeordnet wurde und die Position auch nicht als Zu-Abschlagszeile markiert wurde, dann kann keine automatische Zuordnung zu einem Wareneingang erfolgen und die Rechnung muss zur manuellen Bearbeitung an einen Benutzer oder an eine Gruppe weitergeleitet werden. Vor der Weiterleitung wird jetzt die Funktion **DocFile.ue_GoodsReceivedCheck_MissingOrder()** ausgeführt.

```
/** Diese UserExit-Funktion wird bei den Rechnungen ausgeführt,
 * die im Workflow-Schritt "Warten auf Wareneingang" liegen
 * und wo mindestens einer Positionen keine Bestellung zugeordnet wurde.
 * @param {Log} log Logging-Objekt
 * @since Invoice 1.1.300
**/
DocFile.prototype.ue_GoodsReceivedCheck_MissingOrder = function(log){
  ;
}
```

Über den Parameter "**3WayMatchCheck_Wait4GoodsReceipt**" kann festgelegt werden, wie lange eine Rechnung maximal auf einen Wareneingang warten soll. Wenn die definierte Anzahl an Tagen erreicht ist, wird die Rechnung ebenfalls automatisch weitergeleitet. Vor der Weiterleitung wird jetzt die Funktion **DocFile.ue_AfterWait4GoodsReceived()** ausgeführt.

```
/** Diese UserExit-Funktion wird bei den Rechnungen ausgeführt,
 * die länger im Workflow-Schritt "Warten auf Wareneingang" liegen
 * und nach einer definierten Zeitspanne automatisch weitergeleitet werden sollen.
 * @param {Log} log Logging-Objekt
 * @since Invoice 1.1.300
**/
DocFile.prototype.ue_AfterWait4GoodsReceived = function(log){
  ;
}
```