# Invoice Eskalation

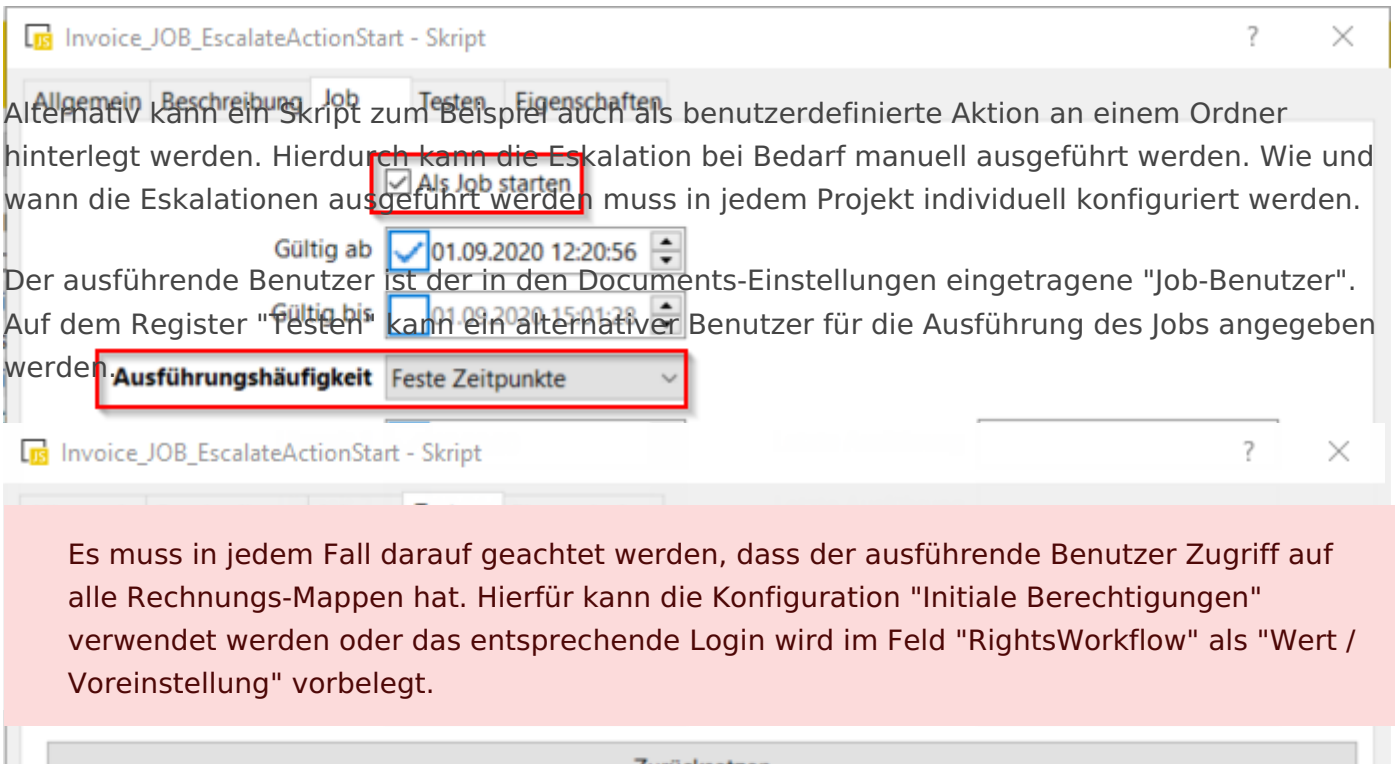
Dieses Buch beschreibt die Eskalations-Konfigurationsmöglichkeiten für das Invoice-Modul. In der Standardauslieferung werden Emails mit den gesammelten Informationen an die Benutzer versendet. Die Eskalationen können projektspezifisch beliebig erweitert und verändert werden.

# Job-Skript

Für jede Eskalation kann ein separates Portal-Skript als Job eingeplant werden. Hierdurch lassen sich die Ausführungszeitpunkte beliebig definieren. Es können auch projektspezifische Jobs erstellt werden.

# Portal-Skript als Job einrichten

Dieser Abschnitt soll eine grobe Übersicht über den Aufbau und den Ablauf der Standard-Eskalation darstellen. Die Eskalations-Jobs werden in Documents als Portal-Skripte eingerichtet. Die in der Standard-Auslieferung enthaltenen Eskalations-Jobs führen jeweils nur einen Eskalations-Typen aus. Die Job-Einstellungen für das Portal-Skript werden auf dem Register "Job" konfiguriert. die Checkbox "Als Job starten" muss zwingend gesetzt sein. Für gewöhnlich soll eine Eskalationen einmal täglich zu festen Zeitpunkten ausgeführt werden.




Alternativ kann ein Skript zum Beispiel auch als benutzerdefinierte Aktion an einem Ordner hinterlegt werden. Hierdurch kann die Eskalation bei Bedarf manuell ausgeführt werden. Wie und wann die Eskalationen ausgeführt werden muss in jedem Projekt individuell konfiguriert werden.

Der ausführende Benutzer ist der in den Documents-Einstellungen eingetragene "Job-Benutzer". Auf dem Register "Testen" kann ein alternativer Benutzer für die Ausführung des Jobs angegeben werden.

Es muss in jedem Fall darauf geachtet werden, dass der ausführende Benutzer Zugriff auf alle Rechnungs-Mappen hat. Hierfür kann die Konfiguration "Initiale Berechtigungen" verwendet werden oder das entsprechende Login wird im Feld "RightsWorkflow" als "Wert / Voreinstellung" vorbelegt.

# Aufbau Job-Skript

Zu Beginn des Skripts werden alle Standard-Skript-Bibliotheken importiert. Hierdurch kann das Skript auf alle Standard-Objekte und User-Exits zugreifen. Im unteren Beispiel wird von einem Invoice-Modul ausgegangen.

Im Anschluss wird ein Eskalations-Objekt erzeugt. Dem Objekt werden der Mappentyp und der Eskalations-Typ als Parameter mitgegeben. Über das Objekt wird automatisch eine Log-Datei unter "**...\\Documents6\\DEXPRO\\Escalation\\...**" erstellt.

Pro Modul und Eskalationstyp werden automatisch Unterordner erzeugt und für jeden Benutzer, an dem eine Eskalationsmail versendet werden soll, wird eine html-Datei angelegt. Damit die Anwender möglichst wenige Emails erhalten, werden die Eskalations-Informationen zunächst gesammelt. Via "FileResultset()" werden alle Rechnungen auf eine Eskalation geprüft.

Wie erwähnt wird pro ermittelten Anwender eine HTML-Datei auf dem Server erstellt. In jeder Datei wird initial der Kopf-Text geschrieben - gefolgt von einer tabellarischen Übersicht der eskalierten Belege. Am Ende wird nur eine Mail pro Benutzer und Eskalations-Typ versendet. Die Dateien bleiben nach der Verarbeitung liegen, damit die Verarbeitung für den Support nachvollziehbar bleibt. Die alten Dateien werden zu Beginn eines neuen Job-Laufs gelöscht.

Erst wenn alle Vorgänge im FileResultset geprüft wurden, werden die Eskalations-Mails generiert und versendet.

**Beispiel:**

```
/* Job-Beispiel für Eskalationen */
// #import "Invoice__ImportLib";

var filetype = "Invoice";
var esc = new Escalation(filetype, "MeinEskalationsTyp");
    esc.cleanUpFilePath(); // alte Dateien löschen

context.setSuperMode(true);
var frs = new FileResultset(filetype, "", "");
for( var docFile=frs.first(); docFile; docFile=frs.next() ){
    /* Die escalate()-Funktion ruft die folgenden UserExits auf:
     * - Escalation.invoice_CheckEscalation()
     * - Escalation.invoice_GetEscalationUserArray()
```

```
     * - Escalation.invoice_CreateTableHead()
     * - Escalation.invoice_CreateTableEntry()
    */
    esc.escalate( docFile ); // Eskalation prüfen und Email-Html generieren
}

/* Beim Abschluss der Funktion wird das UserExit Escalation.invoice_CloseTable() aufgerufen.
 * Im Anschluss werden die Mailsversendet.
*/
esc.closeFilesAndSendMails(); // HTML abschließen und Mails versenden
esc.Log.stop(); // Logging beenden
context.setSuperMode(false);
context.returnValue = esc.Error;
```

# Arbeits-Ordner

Die Eskalations-Informationen werden zunächst pro Anwender in Text-Dateien gesammelt. Der Eskalations-Ordner für Rechnungs-Belege befindet sich im Ordner "DEXPRO" der Documents-Installation unter "Escalation" und "Invoice".

"**...\\Documents5\\DEXPRO\\Escalation\\Invoice\\...**"

> Der Documents-Dienste-Benutzer muss Schreibrechte auf den Verarbeitungs-Ordner haben. Im Standard befindet sich der Ordner unterhalb "..\Documents5\DEXPRO\Escalation\".

Pro Eskalations-Typ wird ein separater Unter-Ordner erstellt. Zu Beginn einer Job-Ausführung wird der bestehende Arbeitsordner geleert. Für jeden Anwender der eine Eskalations-Mail erhalten soll wird eine neue Text-Datei erstellt. Durch das Sammeln der Informationen wird erreicht, dass jeder Anwender nur eine Mail pro Eskalation erhält.

Erst nachdem alle Rechnungen kontrolliert wurden werden die Mails versendet. Die Dateien bleiben nach der Verarbeitung für den Support zur Nachvollziehbarkeit liegen und werden erst bei der nächsten Job-Ausführung gelöscht.

# Logging

Das Log wird automatisch durch das Objekt "Escalation" erstellt. Pro Eskalations-Typ wird ein Log mit Tages-Stempel geschrieben. Alle Eskalations-Logs werden in der Standardauslieferung im Ordner "Escalation" unterhalb des "DEXPRO"-Ordners abgelegt.

"**...\\Documents5\\DEXPRO\\Escalation\\...**"

Der Ablage-Ordner für die Log-Dateien kann über den Parameter "Escalation_Paths" ebenso angepasst werden wie das Intervall, wann neue Log-Dateien erstellt werden (pro Ausführung bzw. Zeitstempel / täglich (Standard) / monatlich / jährlich). Die Log-Ausgaben innerhalb der User-Exits können beliebig angepasst werden.

# Eskalation Funktionsbibliothek und User-Exits

In diesem Kapitel wird der Aufbau und die generelle technische Abarbeitung der Eskalationen beschrieben.

# "Escalation()"-Objekt

Für die Eskalation werden unter anderem die folgenden Portal-Skript-Bibliotheken benötigt:

- **DEXPRO__EscalationLib**
- **DEXPRO__UserExit_Escalation // Invoice**
- **DEXPRO__UserExit_Escalation_Mailroom**
- **DEXPRO__UserExit_Escalation_Procurement**

Für den eigentlichen Eskalations-Aufruf wird ein Portal-Skript als Job eingeplant. Im Job-Skript wird ein "Escalation()" Objekt erstellt. Dem Objekt muss der Mappentyp-Name als erster Parameter mitgegeben werden. Als zweiter Parameter wird der Eskalations-Typ mitgegeben. Im Standard sind die Typen "Deadline", "Action" und "PPC" definiert - es können aber beliebige Typen projektspezifisch definiert werden. Das folgende Skript zeigt ein Beispiel für einen "Deadline"-Eskalations-Job.

```
// #import "Invoice__ImportLib"
var esc = new Escalation("Invoice", "Deadline");
if( esc.Result===true ){
    // Arbeits-Ordner "aufräumen"
    esc.cleanUpFilePath();
    // Alle Rechnungs-Mappen iterieren und Zugriffsberechtigungen ignorieren
    context.setSuperMode(true);
    var frs = new FileResultset("Invoice", "", "");
    for( var docFile=frs.first(); docFile; docFile=frs.next() ){
        // Eskalations-Dateien pro Benutzer erstellen
        // -> esc.invoice_checkEscalation(docFile)
        // -> esc.invoice_getEscalationUserArray(docFile)
        // -> esc.invoice_addBodyContent
        //      -> esc.invoice_createTableHead();
        //      -> esc.invoice_createTableEntry();
        esc.escalate(docFile);
    }
    // Versenden der Eskalations-Mails
    // -> esc.invoice_closeTable(su)
    // -> esc.invoice_sendMail(su, html)
    esc.closeFilesAndSendMails();
    esc.Log.stop();
```

```
    context.setSuperMode(false);
    if( esc.Result===true ){
        return 1;
    }
}
return esc.Error;
```

Das "**Escalation**"-Objekt erhält 2 Parameter und diverse Property.

| | | | |
|---|---|---|---|
| **@param** | **{string}** | **template** | **Name des aufrufenden Mappentypen ("Invoice", "Mailroom")** |
| **@param** | **{string}** | **escType** | **Eskalations-Typ (Standard: "Deadline", "Action", "PPC")** |
| **@property** | **{string}** | **Error** | **Fehlermeldung** |
| **@property** | **{boolean}** | **Result** | **Ergebnis (true/false)** |
| **@property** | **{string}** | **Template** | **Name des aufrufenden Mappentypen ("Invoice", "Mailroom")** |
| **@property** | **{string}** | **EscalationType** | **Eskalations-Typ (Standard: "Deadline", "Action", "PPC")** |
| **@property** | **{string}** | **LogPath** | **Pfad für die Ablage der Logdatei** |
| **@property** | **{string}** | **Path** | **Arbeits-Pfad, in dem pro Benutzer eine Datei mit den Informationen der eMail abgelegt wird.** |
| **@property** | **{string}** | **LogInt** | **Intervall für den Zeitpunkt, zu dem eine neue Logdatei erstellt wird ("timestamp", "date" (Standard), "month", "year").** |
| **@property** | **{Log}** | **Log** | **new Log(context.scriptName, this.LogPath, this.LogInt)** |

**Funktionsliste (DEXPRO_EscalationLib):**

boolean **setEscalationType( String newEscalationType )**
Setzt einen neuen Eskalations-Typen.
Return: true / false

boolean **cleanUpFilePath()**
Löscht alle Dateien aus dem Verarbeitungs-Ordner zum aktuellen Eskalations-Typen.
Return: true / false

boolean **escalate( DocFile docFile )**
Prüft die Eskalation für die aktuelle Mappe. Ggf. wird ein Eskalations-Eintrag pro Anwender hinzugefügt. Bei Gruppen erhält jeder Benutzer aus der Gruppe eine Eskalations-Mail. Die Funktion ruft weitere interne Funktionen auf.
- this.checkEscalation( DocFile docFile )
- this.getEscalationUserArray( DocFile docFile )
- this.addBodyContent( DocFile docFile )
Return: true / false

boolean **addBodyContent( DocFile docFile, SystemUser userobj )**
Wenn zum aktuellen Benutzer noch keine Info-Datei existiert wird eine neue Datei erstellt. Andernfalls wird die bestehende Datei verwendet. Die Funktion fügt einen Tabellen-Eintrag für die aktuelle Rechnung hinzu und ruft hierfür weitere interne User-Exit Funktionen auf.
- this.createTableHead( DocFile docFile, SystemUser userobj, String lang )
- this.createTableEntry( DocFile docFile, SystemUser userobj, String lang )
Return: true / false

boolean **closeFilesAndSendMails( DocFile docFile, SystemUser userobj )**
Die Funktion iteriert alle erstellten Dateien aus dem Arbeits-Ordner. Die Funktion schließt die Html-Tabelle und versendet die HTML als Mail. Hierfür werden die folgenden User-Exit-Funktionen aufgerufen:
- this.closeTable( SystemUser su )
- this.sendMail( SystemUser su, String html )
Return: true / false

boolean **parseDateToLocaleString( Date dateObj, String language )**
Die Funktion wandelt ein Datums-Objekt in einen String um. Das Datumsformat wird automatisch

über die Sprache ermittelt. Bei einem ungültigen Datumsobjekt wird ein Leerstring zurückgegeben. Wenn der Parameter bereits vom Typ "String" ist wird der String zurückgegeben.
Return: Date as String

**Neben den Standard-Funktion werden einige Funktionen als User-Exit-Funktionen ausgeliefert. Die Funktionen sind in der Bibliothek DEXPRO__EscalationLib enthalten und können projektspezifisch angepasst werden.**

**Ab der Invoice-Version 1.1.110 werden eigene Funktionen pro Modul (Invoice/Mailroom/Procurement) aufgerufen und pro Modul wird eine separate UserExit-Lib ausgeliefert. Wenn bei einem Invoice-Update weiterhin die alten Funktionen verwendet werden sollen, dann kann einfach weiterhin die alte Lib "DEXPRO__UserExit_EscalationLib" verwendet werden. Wenn die neuen Funktionen nicht existieren, dann werden weiterhin die alten Funktionen verwendet.**

boolean **checkEscalation( DocFile df ) // Alt**
boolean **invoice_CheckEscalation( DocFile df )**
boolean **mailroom_CheckEscalation( DocFile df )**
boolean **procurement_CheckEscalation( DocFile df )**
Die Funktion wird pro DocFile-Objekt aufgerufen und gibt durch die Rückgabe zurück ob eine Eskalation stattfinden soll oder nicht. Im ausgelieferten Beispiel werden bereits die Eskalations-Typen und die Mappentypen unterschiedlich behandelt.
Return: true / false

boolean **getEscalationUserArray( DocFile df ) // Alt**
boolean **invoice_GetEscalationUserArray( DocFile df )**
boolean **mailroom_GetEscalationUserArray( DocFile df )**
boolean **procurement_GetEscalationUserArray( DocFile df )**
Die Funktion ermittelt pro DocFile-Objekt die zu informierenden Benutzer. Für jeden ermittelten Benutzer wird ein eMail-Eintrag in die Datei zum Benutzer erzeugt.
Return: Array mit SystemUser-Objekten

boolean **createTableHead( DocFile df, SystemUser su, String lang )**
boolean **invoice_CreateTableHead( DocFile df,SystemUser su,String lang )**
boolean **mailroom_CreateTableHead( DocFile df,SystemUser su,String lang )**
boolean **procurement_CreateTableHead( DocFile df,SystemUser su,String lang )**
Wenn zu einem Benutzer noch keine Eskalations-Text-Datei existiert, dann wird zunächst eine neue Datei angelegt. Die Mail wird als HTML-Mail erzeugt und soll in der Regel einen einleitenden Info-Text zur Eskalation enthalten. Die ermittelten Rechnungen werden anschließend tabellarisch

gelistet und hierfür muss zunächst die Zeile mit den Tabellen-Überschriften generiert werden. Die Tabelle kann je nach Mappentyp und Eskalations-Typ unterschiedliche Spalten mit unterschiedlich aufbereiteten Informationen enthalten.
Return: true / false

boolean **createTableEntry( DocFile df, SystemUser su, String lang )**
boolean **invoice_CreateTableEntry( DocFile df, SystemUser su, String lang )**
boolean **mailroom_CreateTableEntry( DocFile df, SystemUser su, String lang )**
boolean **procurement_CreateTableEntry( DocFile df, SystemUser su, String lang )**
Die Funktion fügt einen Tabellen-Eintrag zu einer existierenden Eskalations-Text-Datei hinzu. Die Tabelle kann je nach Mappentyp und Eskalations-Typ unterschiedliche Spalten mit unterschiedlich aufbereiteten Informationen enthalten.
Return: true / false

boolean **closeTable( SystemUser su )**
boolean **invoice_CloseTable( SystemUser su )**
boolean **mailroom_CloseTable( SystemUser su )**
boolean **procurement_CloseTable( SystemUser su )**
Nachdem alle Mappen auf Eskalation geprüft wurden muss bei allen Dateien sowohl die HTML-Tabelle als auch der "body" und der "html" Tag geschlossen werden. Ggf. kann noch ein abschließender Text angefügt werden.
Return: true / false

boolean **sendMail( SystemUser su, String body )**
boolean **invoice_SendMail( SystemUser su, String body )**
boolean **mailroom_SendMail( SystemUser su, String body )**
boolean **procurement_SendMail( SystemUser su, String body )**
Zum Schluss muss die Eskalation als Email versendet werden. Die Funktion erhält den Benutzer als SystemUser-Objekt und den gesamten eMail-Text im HTML-Format aus der Datei. Absender und Betreff können über die properties-Sprachdateien definiert werden.
Return: true / false

# Eskalations-Typen

# Eskalation nach Bearbeitungsfrist ("Deadline")

Jede Rechnung sollte innerhalb einer definierten Bearbeitungsfrist gebucht werden. Sobald diese Frist überschritten wird versendet das System eine Eskalations-Email an den aktuell sperrenden Benutzer.

Die Rechnungs-Mappe muss im Umlauf sein. Wenn die Mappe bei einer technischen Aktion für die Buchungs-Schnittstelle oder der Archivierung liegt muss ebenfalls keine Eskalation erfolgen. In der Regel können alle technischen Aktionen ignoriert werden. Die hinterlegte Prüfung kann beliebig geändert order erweitert werden. Zum Beispiel können Rechnungen die auf Wiedervorlage liegen von der Eskalation ausgeschlossen werden.

Die Prüfung aus Eskalation wird in der Funktion "checkEscalation()" durchgeführt. Die Funktion wird pro DocFile-Objekt - also pro Rechnung aufgerufen.

```
12      switch( this.EscalationType ){
13          case "Deadline":
14              if( df.ActionStatus==="WaitSplits" || df.ActionStatus==="TechActionJob" || df.ActionStatus==="TechActionReceive" ){
15                  this.Log.info("["+id+"] checkEscalation() .. df.ActionStatus("+df.ActionStatus+") .. return false");
16                  return false;
17              }
18
19              if( this.Template==="Invoice" ){
20                  var wfStart = df.WorkflowStart;
21                  if( wfStart instanceof Date ){
22                      var wfStartDateStr  = util.convertDateToString(wfStart, "dd.mm.yyyy");
23                      var wfStartPlusX    = context.addTimeInterval(wfStart, 10, "days", true);
24                      var wfStartPlusXStr = util.convertDateToString(wfStartPlusX, "dd.mm.yyyy");
25                      if( wfStartPlusX < new Date() ){
26                          this.Log.info("["+id+"] checkEscalation() .. context.addTimeInterval("+wfStart+", 10, \"days\", true)=="+wfStartPlusXStr+" < new Date() "
27                                        + ".. return true");
28                          return true;
29                      }
30                      this.Log.info("["+id+"] checkEscalation() .. context.addTimeInterval("+wfStart+", 10, \"days\", true)=="+wfStartPlusXStr+" < new Date() .. return true");
31                      return false;
32                  }
33                  else{
34                      df.WorkflowStart = new Date();
35                      if( df.sync() ){
36                          this.Log.info("["+id+"] checkEscalation() .. set df.WorkflowStart := new Date() .. return false");
37                          return false;
38                      }
39                      else{
40                          this.Log.info("["+id+"] checkEscalation() .. could not set set df.WorkflowStart: "+df.getLastError()+" .. return false");
41                          return false;
42                      }
43                  }
44              }
```

Für die Eskalations-Prüfung wird im Standard das Datumsfeld "WorkflowStart" verwendet, welches direkt bei der Erstellung der Mappe auf einen aktuellen Zeitstempel gesetzt wird. Es kann aber jedes beliebige Datum aber auch jede anders geartete Prüfung ausgeführt werden. Auf das

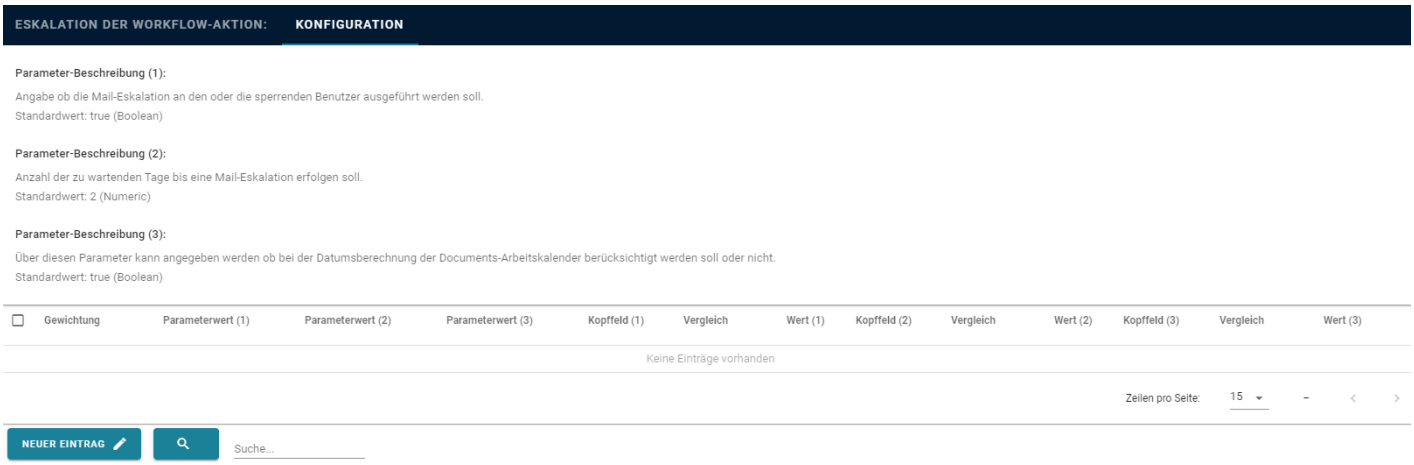
Workflow-Start-Datum werden standardmäßig 10 Arbeitstage aufaddiert. Wenn das berechnete Datum in der Vergangenheit liegt, soll die Rechnung eskalieren ("return true").

Im Anschluss werden die zu informierenden Benutzer über die User-Exit-Funktion "getEscalationUserArray()" ermittelt. Für jeden ermittelten Benutzer wird eine HTML-Datei erstellt und für jede Rechnung wird ein Eintrag hinzugefügt. Am Ende werden die HTML-Tags geschlossen und die Mails werden versendet.

Die Eskalation kann über das Skript "Invoice_Job_EscalatePPC" gesteuert werden.

# Eskalation zur Workflow-Aktion ("Action")

Anders als die Eskalation "Deadline" bezieht sich diese Eskalation nicht auf den gesamten. Beim Eskalations-Typen "Action" erfolgt eine Eskalation, wenn eine Rechnung eine definierte Anzahl an Tagen in einer Workflow-Aktion liegt. Die Prüfung auf eine Eskalation erfolgt durch die User-Exit-Funktion "checkEscalation()". Die Funktion verwendet in der Standardauslieferung den Parameter "Escalation_WorkflowAction".




Über den Parameter können 3 Werte angegeben werden:

1. Der erste Parameter-Wert gibt an, ob eine Eskalation via Mail überhaupt stattfinden soll. Der Standardwert für den Parameter ist "true". Die Eskalation kann generell oder abhängig von Feldwerten deaktiviert werden.
2. Über den zweiten Parameter-Wert kann definiert werden, nach wie viel Tagen die Eskalation ausgeführt werden soll. Im Standard wird eine Eskalation ausgeführt, wenn eine Mappe 2 Tage in einer Aktion liegt. Die hier angegebene Anzahl an Tagen wird auf das Eingangsdatum in die Aktion ("ActionStart") aufaddiert.
3. Bei der Addition der Tage wird im Standard der Arbeitskalender berücksichtigt. Andernfalls würden Rechnungen die an einem Freitag in die Aktion laufen direkt nach dem Wochenende eskalieren.

```
57          case "Action":
58              var escParam   = df.getParamObject("Escalation_WorkflowAction");
59              var doEscalate = df.getParamValue(escParam, 1, true);
60              if( doEscalate === false ){
61                  this.Log.info("["+id+"] checkEscalation() .. do not escalate workflow action .. return false");
62                  return false;
63              }
64              else{
65                  var noDays        = df.getParamValue(escParam, 2, 2);
66                  var useWorkCal    = df.getParamValue(escParam, 3, true);
67                  var actionStartTS = df.ActionStart;
68                  if( (actionStartTS instanceof Date)===true ){
69                      var startStr = util.convertDateToString(actionStartTS, "dd.mm.yyyy HH:MM:SS");
70                      var escDate  = context.addTimeInterval(actionStartTS, Number(noDays), 'days', useWorkCal);
71                      if( escDate < new Date() ){
72                          this.Log.info("["+id+"] checkEscalation() .. action started("+startStr+") .. return true");
73                          return true;
74                      }
75                      else{
76                          this.Log.info("["+id+"] checkEscalation() .. action started("+startStr+") .. return false");
77                          return false;
78                      }
79                  }
80              }
81              break;
82      }
```

Die Eskalation erfolgt in der Standardauslieferung immer an die aktuell sperrenden Benutzer. Für eine Eskalation zum Beispiel an Vorgesetzte müsste ein neuer Eskalations-Typ definiert werden.

Die Eskalation kann über das Skript "Invoice_Job_EscalateActionStart" gesteuert werden.

# Eskalation zur Zahlungsbedingung ("PPC")

Über den Job "Invoice_JOB_EscalatePPC" werden die Zahlungsbedingungen eskaliert. Das Skript iteriert alle Rechnungs-Mappen und ermittelt zunächst für alle Rechnungen das Datumswerte zur hinterlegten Zahlungsbedingung. Im Anschluss werden die Datumswerte "ConditionsOfPayment_Net" und "ConditionsOfPayment_Date1" geprüft. Auf die Datumswerte werden in der Standard-Auslieferung 2 Tage ohne Berücksichtigung des Arbeitskalenders addiert. Sollte eines der berechneten Datumswerte nicht mehr in der Vergangenheit liegen muss eskaliert werden.

```
84          case "PPC":
85              var checkDate    = context.addTimeInterval(new Date(), 4, 'hours', false);
86              var checkDateStr = util.convertDateToString(checkDate, "dd.mm.yyyy HH:MM");
87              df.setPaymentConditions(true);
88              var escDays = 2;
89              var netDate = df.ConditionsOfPayment_NetDate;
90              if( netDate instanceof Date ){
91                  var netDateStr = util.convertDateToString(netDate, "dd.mm.yyyy");
92                  if( netDate <= checkDate ){
93                      var escDate = context.addTimeInterval(netDate, escDays, 'days', false);
94                      if( escDate >= checkDate ){
95                          this.Log.info("["+id+"] checkEscalation() .. netDate("+netDateStr+") + "+escDays+"days >= checkDate("+checkDateStr+")");
96                          return true;
97                      }
98                      else{
99                          this.Log.info("["+id+"] checkEscalation() .. netDate("+netDateStr+") + "+escDays+"days < checkDate("+checkDateStr+")");
100                     }
101                 }
102                 else{
103                     this.Log.info("["+id+"] checkEscalation() .. netDate("+netDateStr+") > checkDate("+checkDateStr+")");
104                 }
105             }
106             else{
107                 /* this.Log.info("["+id+"] checkEscalation() .. df.ConditionsOfPayment_NetDate is not an instance of Date"); */
108             }
109             var percDate = df.ConditionsOfPayment_Date1;
110             if( percDate instanceof Date ){
111                 var percDateStr = util.convertDateToString(percDate, "dd.mm.yyyy");
112                 if( percDate <= checkDate ){
113                     var escDate = context.addTimeInterval(percDate, escDays, 'days', false);
114                     if( escDate >= checkDate ){
115                         this.Log.info("["+id+"] checkEscalation() .. percDate("+percDateStr+") + "+escDays+"days >= checkDate("+checkDateStr+")");
116                         return true;
```

Die Eskalation erfolgt an den bzw. die sperrenden Benutzer.