User-Exit Funktionen

Es werden diverse User-Exit Funktionen bereitgestellt, um die Daten zu gegebenen Zeitpunkten zu manipulieren.

CreditorCsv(DocFile, Log)

Technisch wird im Hintergrund ein Objekt vom Typen **CreditorCsv** erzeugt. Dem Objekt muss zwingend ein DocFile-Objekt übergeben werden. Wenn kein spezifisches Logging-Objekt übergeben wird, wird ein eigenes erzeugt. Das Objekt selber enthält folgende Attribute:

- **Error** {string} Fehlermeldung
- **Log** {Log} Das übergebene Log Objekt
- **DocFile** {DocFile} Das übergebene DocFile Objekt
- **ID** {string} ID des DocFile-Objekts
- **Gentable** {Gentable} Gentable des übergebenen DocFile Objekts
- **CsvArr** {Array} Array mit den CSV-Zeilen
- **Seperator** {string} CSV-Trennzeichen
- **DecSeperator** {string} Dezimaltrennzeichen
- **NoHeadLineCol** {int} Anzahl zu überspringende Kopfzeilen
- **CsvMapping** {Array} Array mit Spalten-Mapping-Objekten

Die User-Exit-Funktionen werden via prototype hinzugefügt. Dadurch erhält der Anwender Zugriff auf die Objekt-Attribute. In der Regel muss in den User-Exit-Funktionen zwischen den Kreditoren unterschieden werden. Der Kreditor kann über das DocFile-Objekt ermittelt werden.

ue_CreditorCsv_AdjustLineString(currentLine, lineNo)

Die Funktion wird nach dem Auslesen jeder einzelnen Zeile aufgerufen. Durch die Funktion kann der komplette Zeilen-String manipuliert werden. Zum Beispiel kann via **util.transcode()** ein falsches Encoding korrigiert werden.

```
/** CSV-Zeile manipulieren
 * @param {string} currLine CSV-Zeile als String
 * @param {int} lineNo Aktuelle Zeilennummer für Logausgaben
 * @returns {string} Angepasste CSV-Zeile als String
 **/
CreditorCsv.prototype.ue_CreditorCsv_AdjustLineString = function(currLine, lineNo){
var creditorID = this.DocFile.CreditorID;
  if( creditorID==="12345" ){
```

```
      return util.transcode("ISO-8859-15", currLine, "UTF-8");
    }
  return currLine;
}
```

ue_CreditorCsv_IgnoreLine(currentLineArr, currentLine, lineNo)

Nachdem die Zeile ausgelesen wurde wird die Zeile über den Trenner gesplittet und die Teil-Strings werden in ein Array geschrieben. Die CSV-Dateien können Summenzeilen enthalten, welche beim Auslesen übersprungen werden sollen. Oft können solche Zeilen einfach identifiziert werden, da bestimmte Spaltenwerte fehlen. Es kann auch sein, dass eine CSV die Daten von mehreren Rechnungen enthält. Über die Funktion können die Zeilen mit abweichender Rechnungsnummer übnersprungen werden.

```
/** CSV-Zeile überspringen
 * @param {Array} currLineArray CSV-Zeilen Array mit den einzelnen String-Werten
 * @param {string} currLine CSV-Zeile als String
 * @param {int} lineNo Aktuelle Zeilennummer für Logausgaben
 * @returns {boolean} 'true' um Zeile zu überspringen
 **/
CreditorCsv.prototype.ue_CreditorCsv_IgnoreLine = function(currLineArray, currLine, lineNo){
var creditorID = this.DocFile.CreditorID;
  if( creditorID==="12345" ){
    if( currLineArray[10]==="" ){
         return true;
        }
    }
  return false;
}
```

ue_CreditorCsv_AdjustLineArray(currentLineArr, lineNo)

Direkt nach der Funktion ue_CreditorCsv_IgnoreLine() wird gleich das nächste UserExit ausgeführt, um bestehende Zeilen-Werte zu manipulieren. Hier ist zu berücksichtigen, dass alle Werte im Array als String vorliegen!

```
/** CSV-Zeilen-Werte manipulieren
 * @param {Array} currLineArray CSV-Zeilen Array mit den einzelnen String-Werten
 * @param {int} lineNo Aktuelle Zeilennummer für Logausgaben
```

```
 * @returns {Array} Angepasstes Zeilen-Array
 **/
CreditorCsv.prototype.ue_CreditorCsv_AdjustLineArray = function(currLineArray, lineNo){
  var creditorID = this.DocFile.CreditorID;
    if( creditorID==="12345" ){
      /* ... */
      }
    return currLineArray;
}
```

ue_CreditorCsv_Parse_String(value, techFieldName)

ue_CreditorCsv_Parse_Bool(value, techFieldName)

ue_CreditorCsv_Parse_Date(value, techFieldName)

ue_CreditorCsv_Parse_Timestamp(value, techFieldName)

ue_CreditorCsv_Parse_Numeric(newValue, value, noDecPlaces, techFieldName)

Die einzelnen Angaben in der CSV werden als String ausgelesen und müssen ggf. in andere Typen formatiert werden. Die verwendeten Funktionen bieten wiederum diese User-Exit-Funktionen an, um die Angaben zu manipulieren.

```
/** Einzelnen String-Wert manipulieren
 * @param {string} value Eingehender String-Wert
 * @param {string} techFieldName Technischer Name der Gentable-Spalte
 * @returns {string} Angepasster String-Wert
 **/
CreditorCsv.prototype.ue_CreditorCsv_Parse_String = function(value, techFieldName){
  var creditorID = this.DocFile.CreditorID;
    if( creditorID==="12345" ){
      switch(techFieldName){
          case "GLAccount":
              /* ... */
                  break;
          }
      }
    return value;
}
```

```
/** Einzelnen Numerischen-Wert manipulieren
 * @param {float} newValue Geparster numerischen Wert geparster
 * @param {string} value Ursprünglicher String-Wert
 * @param {int} noDecPlaces Anzahl Dezimalstellen
 * @param {string} techFieldName Technischer Name der Gentable-Spalte
 * @returns {float} Angepasster Wert
 **/
CreditorCsv.prototype.ue_CreditorCsv_Parse_Numeric = function(newValue, value, noDecPlaces,
techFieldName){
var creditorID = this.DocFile.CreditorID;
  if( creditorID==="12345" ){
    switch(techFieldName){
        case "Net":
            /* ... */
                break;
        }
    }
  return value;
}
```

ue_CreditorCsv_AdjustRow(row, lineNo)

Über das Zeilen-Array wird ein Gentable-Zeilen-Objekt erstellt und diese Gentable-Zeile kann im Anschluss manipuliert werden. Über Stammdaten-Abgriffe können zum Beispiel weitere Werte abgegriffen werden.

```
/** Gentable-Zeile manipulieren
 * @param {Object} row Gentable row object
 * @param {int} lineNo Aktuelle Zeilennummer für Logausgaben
 * @returns {Object} Angepasstes Gentable-Zeilen-Objekt
 **/
CreditorCsv.prototype.ue_CreditorCsv_AdjustRow = function(row, lineNo){
var creditorID = this.DocFile.CreditorID;
  if( creditorID==="12345" ){
    if( row.GLAccount==="XYZ" ){
         row.GLAccount_Desc = "description";
        }
    }
```

```
  return row;
}
```

ue_CreditorCsv_AdjustGentable(gentable)

Aus den Gentable-Zeilen wird ein Gentable-Objekt erstellt und auch dies kann wiederum manipuliert werden. Manchmal ist es erwünscht, dass Zeilen mit übereinstimmenden Angaben automatisch zusammengefasst werden.

```
/** Gentable-Zeile manipulieren
 * @param {Gentable} rows Gentable object
 * @returns {Gentable} Angepasstes Gentable
 **/
CreditorCsv.prototype.ue_CreditorCsv_AdjustGentable = function(rows){
var creditorID = this.DocFile.CreditorID;
  if( creditorID==="12345" ){
    for( var i=0; i<rows.length; i++ ){
         var row = rows[i];
          if( row.GLAccount==="XYZ" ){
             /* ... */
            }
        }
    }
  return rows;
}
```

Revision #3
Created 2022-05-13 05:53:52 UTC by Markus Meisner
Updated 2024-07-24 08:14:55 UTC by Markus Meisner