

Kreditor-CSV

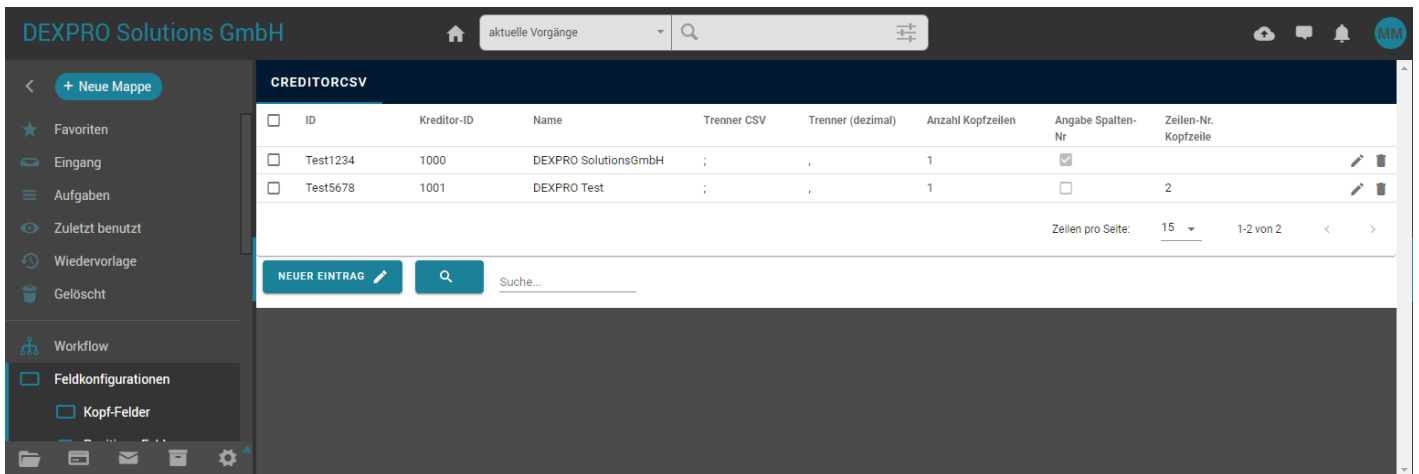
Teilweise können Kreditoren eine passende CSV mit den Rechnungspositionen bereitstellen. Für den Kreditor muss der Aufbau der CSV-Datei in der Konfiguration hinterlegt werden. Die CSV kann im Anschluss an ein separates Register hochgeladen werden und nun können die Rechnungspositionen geladen werden.

- [WEB-Konfiguration](#)
- [Hochladen und Auslesen einer CSV Datei](#)
- [User-Exit Funktionen](#)

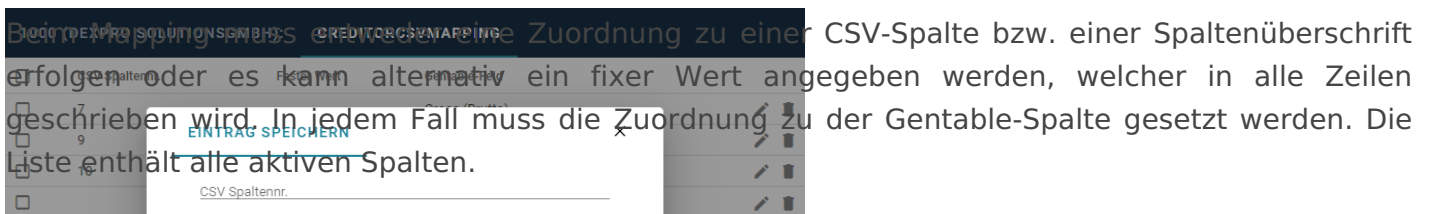
WEB-Konfiguration

Im ersten Schritt muss der Aufbau der CSV-Datei für den Kreditor definiert werden. Die CSV-Datei eines Kreditors muss einen gleichbleibenden Aufbau haben. Auf der Administrations-Outbar "**Invoice**" befindet sich unter "**Feldkonfigurationen**" der Ordner "**Kreditor CSV**". Pro Kreditor müssen folgende Angaben erfolgen:

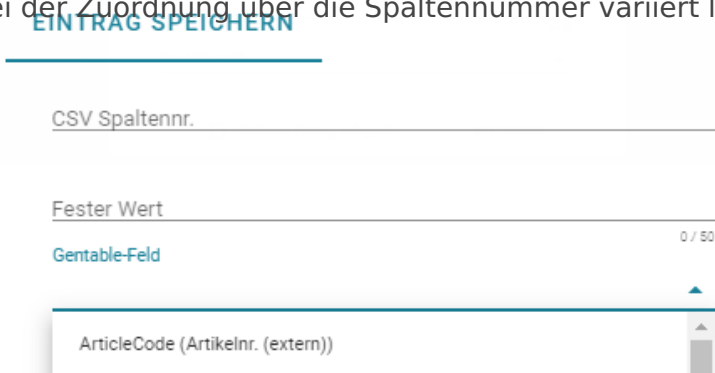
- Eindeutige Kreditor ID
- Kreditor Name
- Der Trenner zwischen den Werten in der CSV-Datei
- Die Anzahl an Kopfzeilen, welche beim Auslesen übersprungen werden sollen (zum Beispiel die Kopfzeile)
- Optional kann direkt die Spaltennummer angegeben werden
- Alternativ kann das Mapping über die Spaltenüberschrift erfolgen
Hierzu muss angegeben werden in welcher Zeile die Überschriften stehen













Im zweiten Schritt muss das Feldmapping angegeben werden. Bei einem Doppelklick auf den Eintrag öffnet sich eine neue Ansicht. Über den Button "**NEUER EINTRAG**" kann ein neues Mapping definiert werden. Je nachdem ob die Zuordnung über die Spaltennummer oder die die Spaltenüberschrift vorgenommen wird variiert die erste Spalte.




Bei der Zuordnung über die Spaltennummer variiert lediglich die Anzeige der ersten Spalte.



1000 (DEXPRO SOLUTIONSGMBH): CREDITORCSVMAPPING			
<input type="checkbox"/>	CSV Spaltennr.	Fester Wert	Gentable-Feld
<input type="checkbox"/>	7		Gross (Brutto)  
<input type="checkbox"/>	9		Net (Netto)  
<input type="checkbox"/>	10		VatRate (MwSt-Satz)  
<input type="checkbox"/>		DE19	VatCode (MwSt-Code)  

Zeilen pro Seite: 15  1-4 von 4  

NEUER EINTRAG 

Bei der Zuordnung über den Spaltennamen muss der korrekte Spaltenname in der korrekten Schreibweise inklusive Leerzeichen angegeben werden:

1001 (DEXPRO TEST): CREDITORCSVMAPPING			
<input type="checkbox"/>	CSV Spaltenname	Fester Wert	Gentable-Feld
<input type="checkbox"/>	Nettobetrag		Net (Netto)  
<input type="checkbox"/>	MWST		VatRate (MwSt-Satz)  
<input type="checkbox"/>		DE19	VatCode (MwSt-Code)  

Zeilen pro Seite: 15  1-3 von 3  

NEUER EINTRAG 

Sollten irgendwelche Angaben in der CSV nicht enthalten sein, welche allerdings fix vorgegeben werden können, dann kann dieser Wert über die Angabe "**Fester Wert**" fix gesetzt werden.

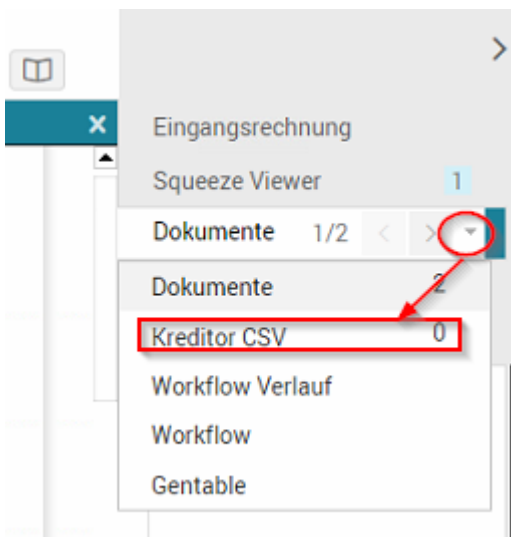
Beim Auslesen der CSV werden die bisherigen Positionszeilen verworfen und neu gelesen. Konfigurations-Anpassungen können somit jederzeit schnell getestet werden.

Über die Konfiguration kann lediglich das Mapping konfiguriert werden. Werte die nicht in der CSV-Datei enthalten sind werden sind können über die WEB-Konfiguration nicht automatisch generiert werden! Über die unterschiedlichen UserExit-Funktionen können die einzelnen Werte sowie das gesamte Ergebnis beliebig angepasst werden.

Hochladen und Auslesen einer CSV Datei

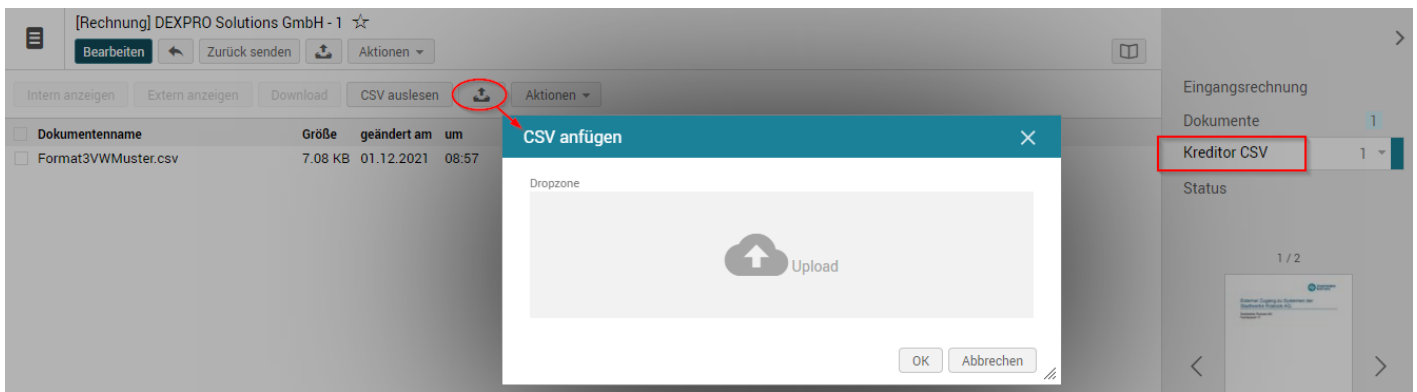
Die Voraussetzung für das Einlesen einer CSV-Datei ist, dass am Rechnungs-Vorgang ein Kreditor hinterlegt ist und zudem muss für diesen Kreditor eine CSV-Konfiguration existieren. Die CSV-Datei muss passend zur Konfiguration sein.

Nachdem die CSV-Konfiguration für den Kreditor eingerichtet wurde kann die CSV-Datei an eine Rechnungs-Akte hochgeladen werden. Hierfür muss der Anwender auf das Dokumenten-Register "**Kreditor CSV**" wechseln. Die Anzeige des Registers wird über ein Skript gesteuert und kann somit von Projekt zu Projekt variieren. Die Anzahl der direkt angezeigten Register ist in der Standardauslieferung eingeschränkt. Weitere Register verbergen sich hinter dem Pfeil am Register.



Wenn Sie das Register weiterhin nicht sehen wenden Sie sich an den Administrator.

Am Register werden 2 benutzerdefinierte Aktion als Button angezeigt. Über den Button mit dem Icon öffnet sich ein Pop-Up-Fenster mit einer sogenannten Dropzone. Die CSV-Datei kann via Drag-N-Drop auf diese Dropzone gezogen werden. Durch die Bestätigung "OK" wird die CSV-Datei an das Register hochgeladen.



Über den zweiten Button "**CSV auslesen**" wird die CSV-Datei verarbeitet und pro Zeile in der CSV-Datei wird eine Rechnungsposition erstellt. Der Button wird im Bearbeitungs-Modus ausgeblendet. Wenn Sie den Button nicht sehen müssen Sie die Rechnungs-Akte über den Button "Speichern" zuerst speichern. Sollte bei der Ausführung noch keine Konfiguration zum Kreditor existieren wird eine entsprechende Fehlermeldung ausgegeben.

Die bestehenden Positionszeilen samt der evtl. bereits hinzugefügten Kontierung gehen unwiderruflich verloren und werden durch die Positionsangaben aus der CSV ersetzt!

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 übersprungen 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;
}

```