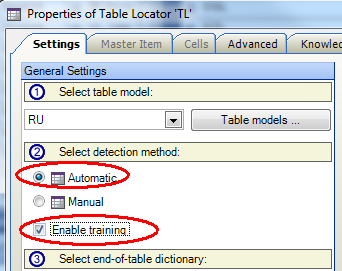
How to use Table Locators in a KTM Invoice Project

# Table Detection

Line Item matching is the most time consuming task for account processing clerks, and can easily provide the largest return on investment in a KTM solution.

KTM’s Table Locators use at least 7 different methods to find the line items of an invoice and can be difficult to configure optimally.

We recommend combining Manual and Automatic Methods together. This why trained tables use the Manual Method and unknown tables use the automatic method.



# Automatic Table Locators

The automatic table locator is used to locate the table on an **unknown invoice**. It attempts to find:

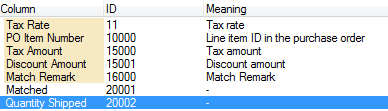
1. The table header
2. The rows and the columns
3. The identity of the columns
4. The end of the table

KTM 5.5 added the ability to the Table Locator to take the Line Item Matching Locator’s table as input.

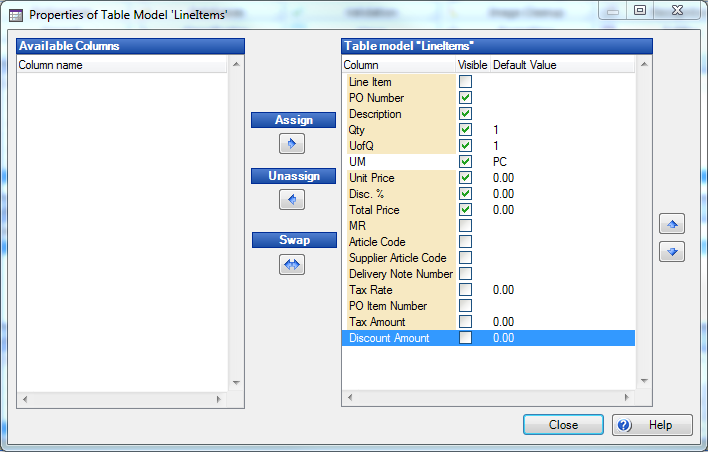
To detect the header of a table, you will need to train 50-100 invoices in “Table header packs” per language for this to be successful.

## Table Model

Make your table model as simple as possible. Your table model should match exactly the columns required for data entry into the ERP system. However use as many of the built-in KTM columns where possible, as they are used by the detection algorithms – custom columns will require header detection to be found.

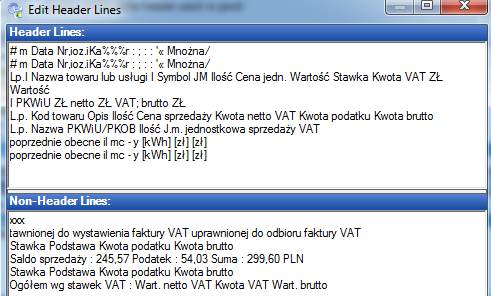
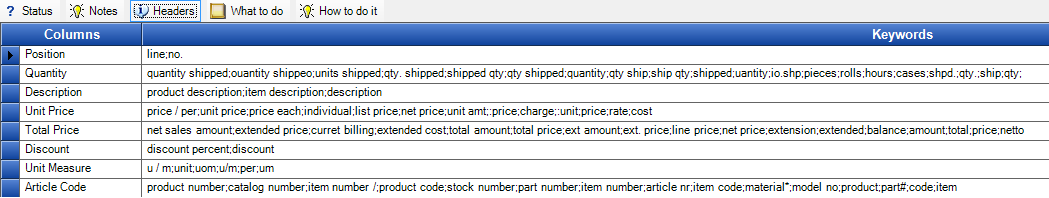


Note: the “Unit Measure” column is looking for scaling units “1”, “100” or “1000”. It not “kg”, “m²” etc.

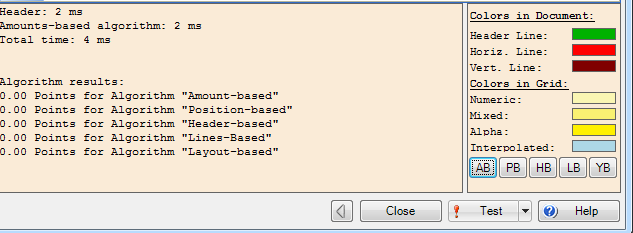
A sample invoice header pack

## Table Header Pack

A header pack consists simply of 3 text files. Try to make these files as complete as possible. Correct OCR errors manually. Header detection works best on column names that have no word-wrapping.

1. Header Row Examples
2. Non-Header Row Examples  
   
3. Header Keywords  
   

## Row & Column Detection

The next step applies 5 different algorithms, which attempt to read the lines underneath the header, and interpret the columns. (You can switch the Table Locator configuration panel into Debug Mode by pressing CTRL-F12, where you can test individual algorithms. The algorithm with the best score wins – the other 4 are ignored.)  


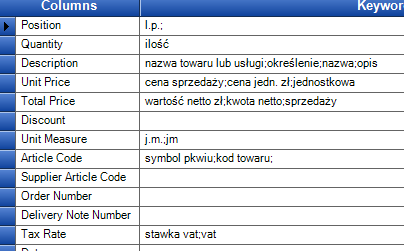
## Amount Based Algorithm

Attempts to find mathematical relationships in rows (e.g. A\*B=C).  
To improve it, make sure that your format locator that feeds the table locator is finding the amounts in the table.

## Position Based Algorithm

Attempts to find an increasing number, probably on the left of the document, which represents the position number in the invoice. Ideally it is 1, 2, 3, 4, 5, 6, …..

## Header Based Algorithm

Attempts to use the header words to understand where the columns are.  
To improve it, make sure that your Table Header Pack is complete with all possible keywords for the columns. The example below would fail to find the discount and supplier article code columns. There are only 4 description column names, which are also probably too few. Correct the spelling of the keywords to improve fuzzy scoring across all of your invoices. 

## Line Based Algorithm

Attempts to find the table cells based on horizontal and vertical lines that are on the pages. You cannot improve how this algorithm works.

## Layout Based Algorithm

Attempts to use white space between columns to understand where the columns are. You cannot improve how this algorithm works.

# Manual Table Locator

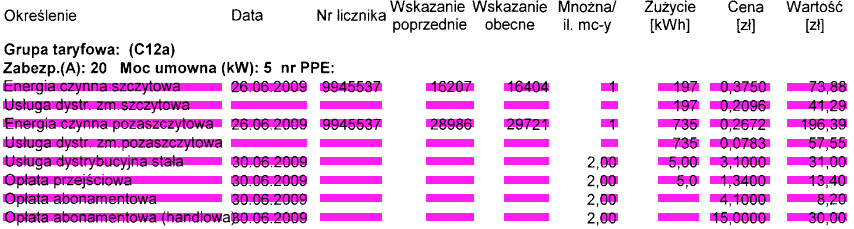
This second mode of the Table Locator uses sample documents to identify a previously seen table.

You can either make subclasses for important vendors with their own Manual Table Locator, or use the Online Learning Mechanism’s specific knowledge bases to store these sample tables – the result is the same.

The easiest way to understand the table locator is that of a stencil (<http://en.wikipedia.org/wiki/Stencil>) being moved down the page looking for matches. It does not look for the table header.

Tips

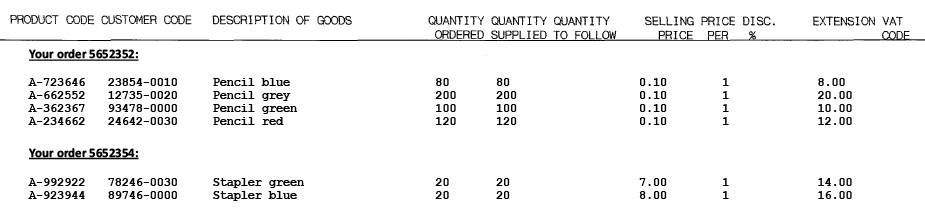
* The “master item” should be a table row containing optional row and long descriptions.
* The “master item” should not have missing cells.
* Mark the optional rows in the master item.
* Expand the left and right edges of the cells.
* Make sure that you mark optional cells.
* Cell Types are internal algorithms that attempt to work it if the cells are numeric or alphabetic
* After online learning, check and correct specific training samples in the “Edit Document” Dialog (press F10 on the Xdoc)
* Anchors are words/phrases that appear on each table row, but are not needed to be read – they help with matching the stencil.
* In the example below we see that columns 2-6 are optional, and that column 6 with “1” and “2,00” is too narrow. “2,00” will be corrected detected. However, if the OCR engine sees “2 00” then only “00” will be found. The last row will have “(handlowa)80.06.2009” in column 1.



Both manual and automatic table detection can search for interleaved and embedded columns.

## Interleaved Columns

An interleaved column is a value that applies to all the rows below it, like the order number below.



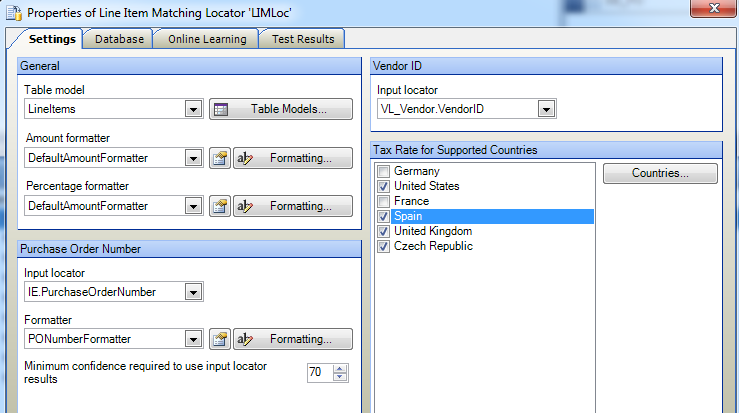
## Embedded Columns

An embedded column is typically an article number or date appended to a description.

# Line Item Matching Locator

LIMLoc matches the words on the invoices with the open purchase orders from the ERP system. It requires an SQL connection to a database containing open purchase orders.

If LIMLoc does not have a vendor ID it cannot work.

If no PO numbers are provided, LIMLoc matches against the first 20 open orders in the database.  


LIMLoc has a self-contained learning mechanism to improve its extraction.

LIMLoc returns tolerance messages for each line item.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Valid | Message |  | Name |
| 1 | True | OK |  | IDS\_MatchRemark\_Match |
| 2 | True | Under Or over delivery |  | IDS\_MatchRemark\_UnderOverDelivery |
| 3 | True | Change In units of measurement |  | IDS\_MatchRemark\_UomChanges |
| 4 | True | Under Or over price |  | IDS\_MatchRemark\_UnderOverPrice |
| 5 | False | Missing unit price |  | IDS\_MatchRemark\_MissingUnitPrice |
| 6 | True | Unit price scaled by power of ten |  | IDS\_MatchRemark\_ScaledUnitPrice |
| 7 | True | Total amount scaled by power of ten |  | IDS\_MatchRemark\_ScaledTotalAmount |
| 8 | False | OCR errors |  | IDS\_MatchRemark\_OcrErrors |
| 9 | False | Only total amount Matched |  | IDS\_MatchRemark\_OnlyTotalAmount |
| 10 | False | Several matches possible |  | IDS\_MatchRemark\_Ambiguous |
| 11 | False | Only quantity, unit price And total amount Matched |  | IDS\_MatchRemark\_UnmatchedConsistent |
| 12 | False | Could Not Match |  | IDS\_MatchRemark\_Unmatched |
| 13 | True | Validated by user |  | IDS\_MatchRemark\_UserValidated |

# Combining Locators

You will obtain the best results with table detection by combining together the following 10 locators.

|  |  |  |
| --- | --- | --- |
| Locator Name | Locator Type | Function |
| IGL | Invoice Group | Finds the VendorID from Online Learning |
| OGL | Order Group | Finds the Order Number from Online Learning |
| FL\_PO | Format | Finds PO Number candidates |
| IE | Invoice Evaluator | Combines OGL.PO and FL\_PO |
| DB\_Vendor | Database | Finds vendor candidates |
| FL\_VAT | Format | Finds VATNumber candidates |
| VL\_Vendor | Vendor | Uses IGL.VendorID, DB\_Vendor and FL\_VAT to find the Vendor |
| LIM | Line Item Matching | Uses IE.PO and VL\_Vendor.VendorID to find the line Items |
| FL\_Amounts | Format | Finds any amounts on the document |
| TL | Table Locator | Uses LIM and FL\_Amounts to find the table. |

# Validation

You can use the table interpolation feature in KTM Validation to quickly extract tables and also to train tables. You will need to examine these specific training samples in Project Builder by pressing F10 on the Xdocs.

The Interpolation Button can be pressed more than once. Interpolate a row, select another row and then interpolate that. This will successively for more of the table.