Grooper | ACE

Architect · Consultant · Engineer

ARCHITECT TRAINING MANUAL VOL.2

ADVANCED INVOICE PROCESSING

13900 N. Harvey Ave ● Edmond, OK ● 73013 ● 405-507-7000

www.grooper.com

TABLE OF CONTENTS

In this Architect Training Manual Vol. 2 • Advanced Invoice Processing guide for **Grooper**, we will flesh out our understanding by continuing to work with a familiar set of documents. This understanding will be achieved by focusing mainly on creating a more complex Content Model with new extractors and data elements.

While the approach won't be quite as repeated and linear as what was covered in Overview and Concepts, it is still best practice to think in terms of the Five Phases of processing documents through **Grooper**.

- **Phase 1 - Acquire**
- **Phase 2 - Condition**
- **Phase 3 - Organize**
- **Phase 4 - Collect**
- **Phase 5 - Deliver**

It should also be noted that as the document progresses, the pace of the steps provided will advance to prevent repetition and save time and space.

PHASE 1 – ACQUIRE

This process will begin in a similar fashion to Overview and Concepts, by scanning files into **Grooper**. This will be done with a slight twist, however.

OBJECT PORTABILITY IN GROOPER

Follow the below link to obtain a zip file. Extract the zip files from within this file to a directory you have quick access to, as those files will be imported into **Grooper**.

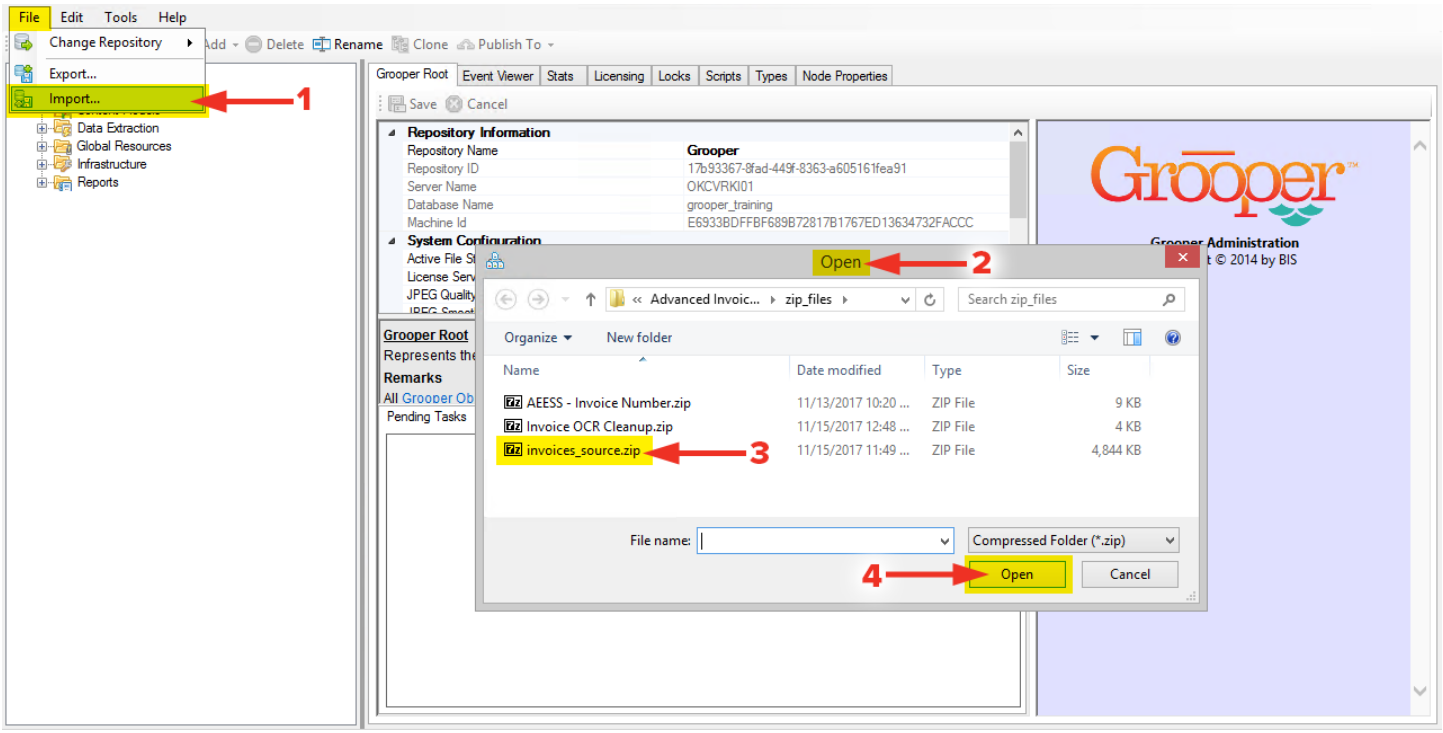
Grooper A.C.E. - Architect Training Vol.2 - Advanced Invoice Processing - Zip Files

IMPORTING ZIP OBJECTS INTO GROOPER

A key strength of **Grooper** is the flexibility gained by its object-oriented programming. Any object made in **Grooper** can be easily exported as a zip object, and subsequently just as easily imported into another Grooper environment.

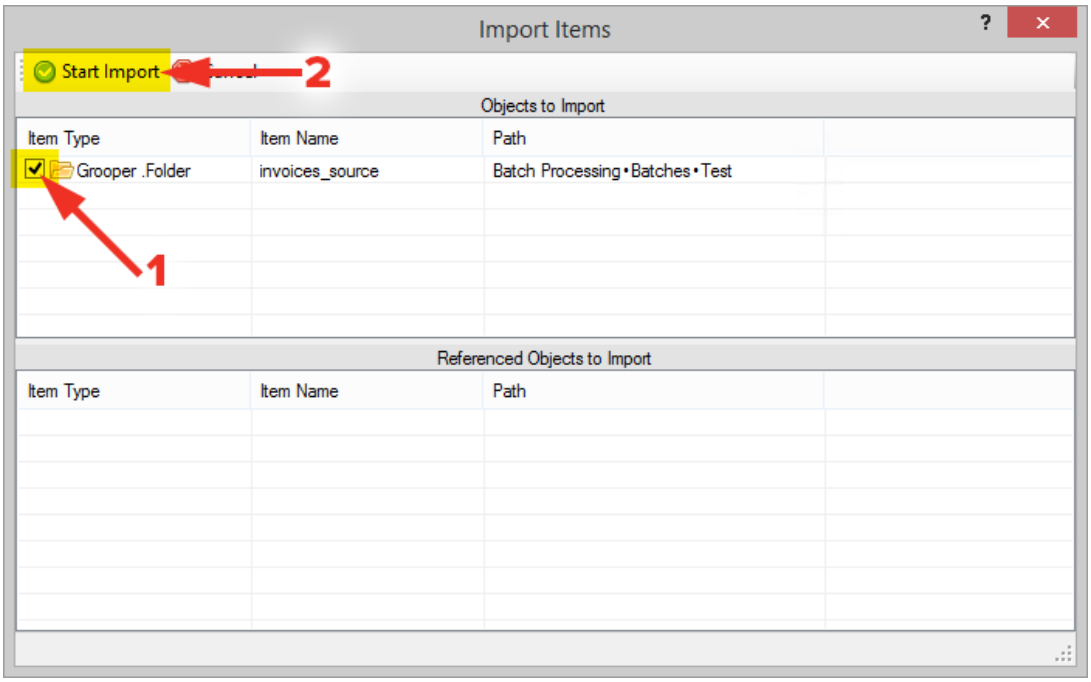
STEP 1 – FILE IMPORT

Start by launching Grooper Administration and **(1)** click File and select Import. **(2)** An Open window will appear. From here **(3)** select the invoices_source.zip file and **(4)** click Open.



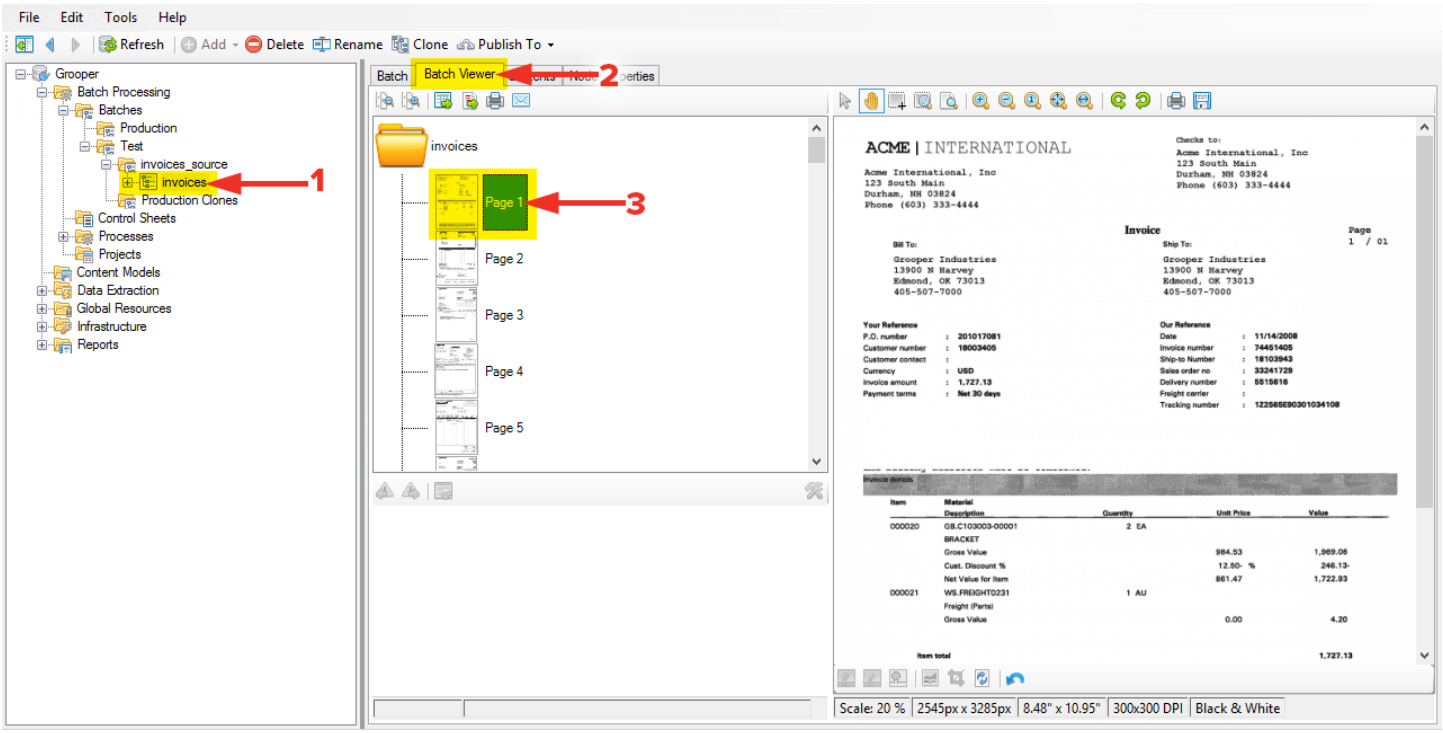
STEP 2 – SELECT IMPORT ITEMS

In the Import Items window that opens, **(1)** check the object to be imported and **(2)** click Start Import.



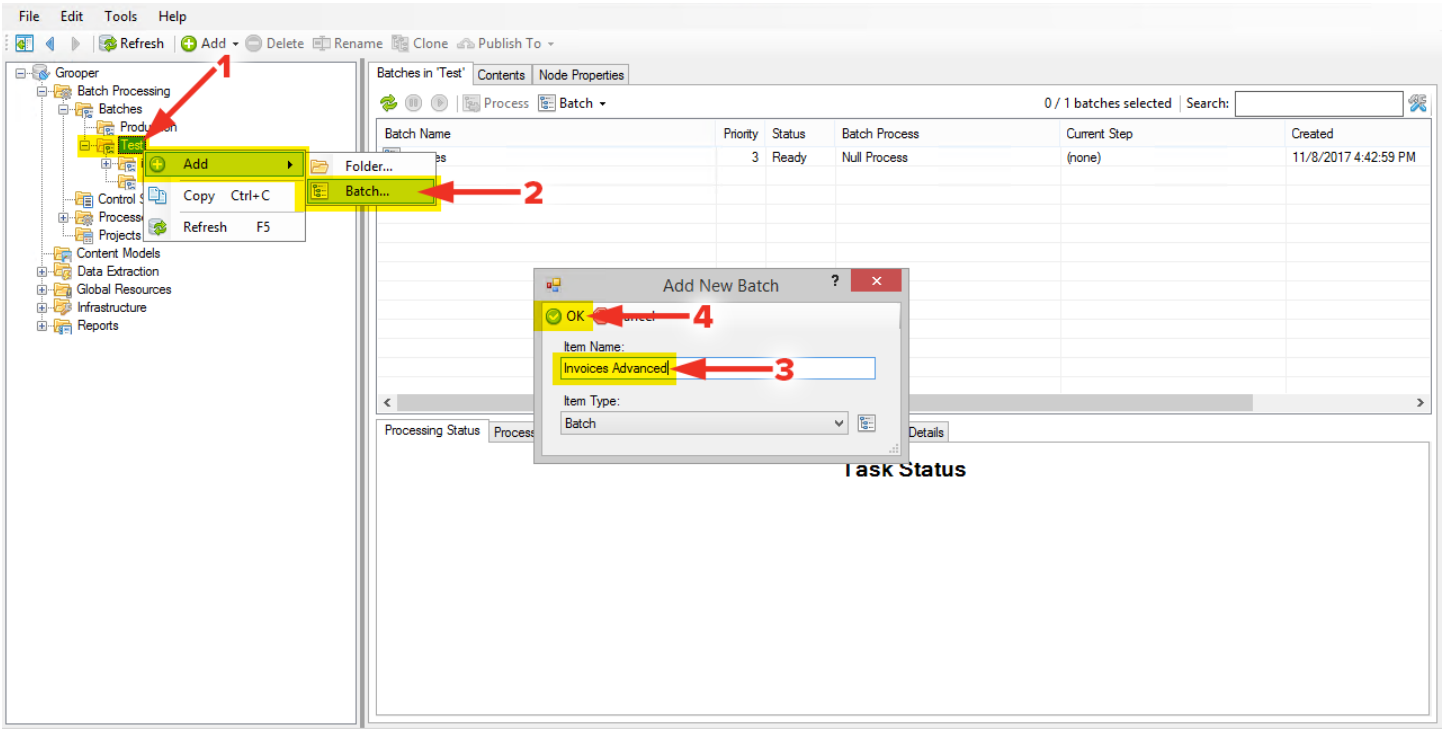
STEP 3 – VIEW NEWLY IMPORT BATCH

Expand the **GROOPER** node tree to **(1)** Grooper > Batch Processing > Batches > Test > invoices_source and select the invoices batch object. From there, **(2)** click the Batch Viewer tab and take a moment to **(3)** look at the page objects within the batch.



STEP 4 – CREATE EMPTY BATCH

The object just imported will be leveraged as a source for page objects, and as such, should not be affected by any processing. An empty batch will be created so testing can be done. **(1)** Navigate to Grooper > Batch Processing > Batches > Test, **(2)** right click and Add > Batch. In the Add New Batch window, **(3)** name the batch Invoices Advanced and **(4)** click OK.

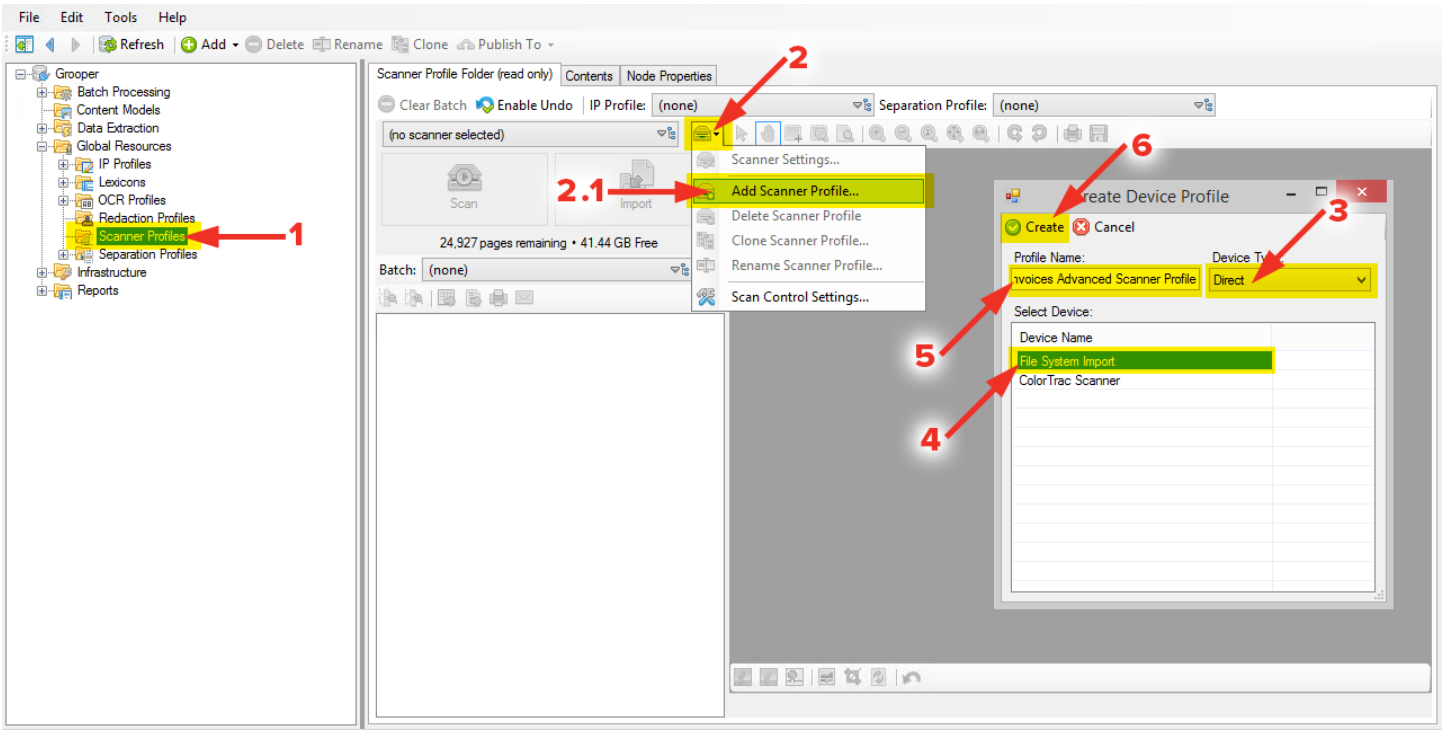


CREATING A SCANNER PROFILE TO LEVERAGE THE SOURCE BATCH

A Scanner Profile will now be created so that the page objects within the source batch can be injected into the newly created empty batch. This is done so that when a batch process is created later it can be run as an end to end process starting with Scan.

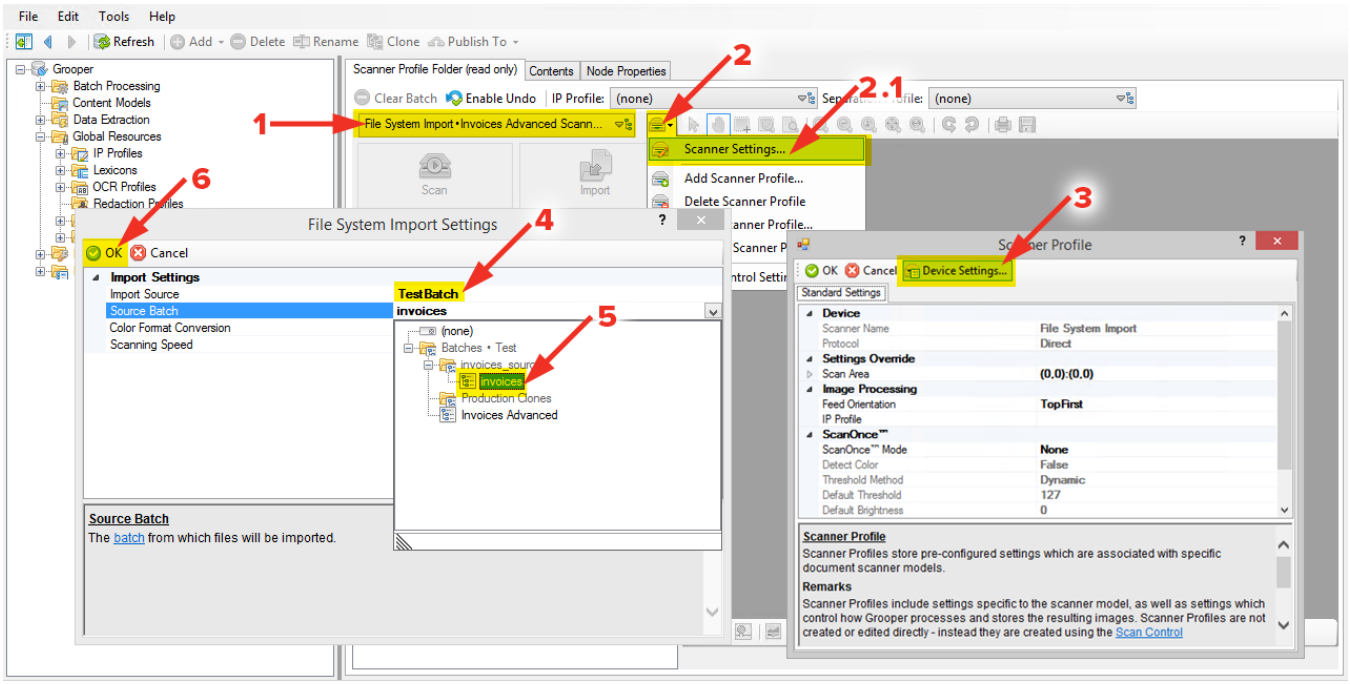
STEP 1 – CREATING A NEW SCANNER PROFILE

(1) Navigate to Grooper > Global Resources > Scanner Profiles. **(2)** Click the Scanner Profile drop-down button and select Add Scanner Profile... In the Create Device Profile window, **(3)** select Direct for the Device Type, and **(4)** select File System Import for the Device Name. **(5)** Name the profile Invoices Advanced Scanner Profile and **(6)** click Create.



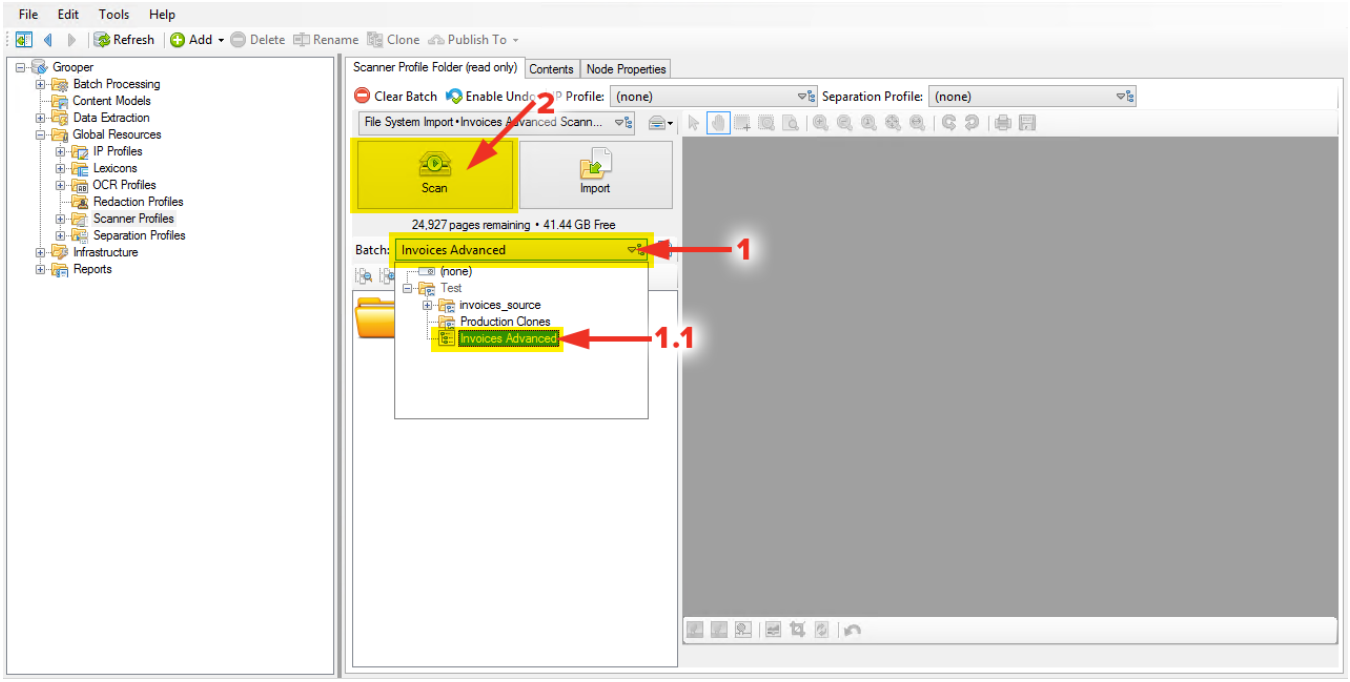
STEP 2 – SCANNER PROFILE SETTINGS

(1) With the newly created Scanner Profile selected in the drop-down, **(2)** click the Scanner Profile drop-down button and select Scanner Settings. In the Scanner Profile window that opens, **(3)** click Device Settings... From the File System Import Settings window, **(4)** set Import Source to TestBatch, and **(5)** set Source Batch to the Invoices batch object from within the invoices_source area, then **(6)** click OK to close both open windows.



STEP 3 – PERFORM SCAN

(1) From the Batch: drop-down select the Invoices Advanced batch object and **(2)** click the Scan button. The page objects from the source batch will be scanned into the Invoices Advanced batch.



PHASE 2 – CONDITION

The second phase in **Grooper** is just as critical as before as all information that will be worked with moving forward will come from conditioning the documents.

A FOCUS ON OCR

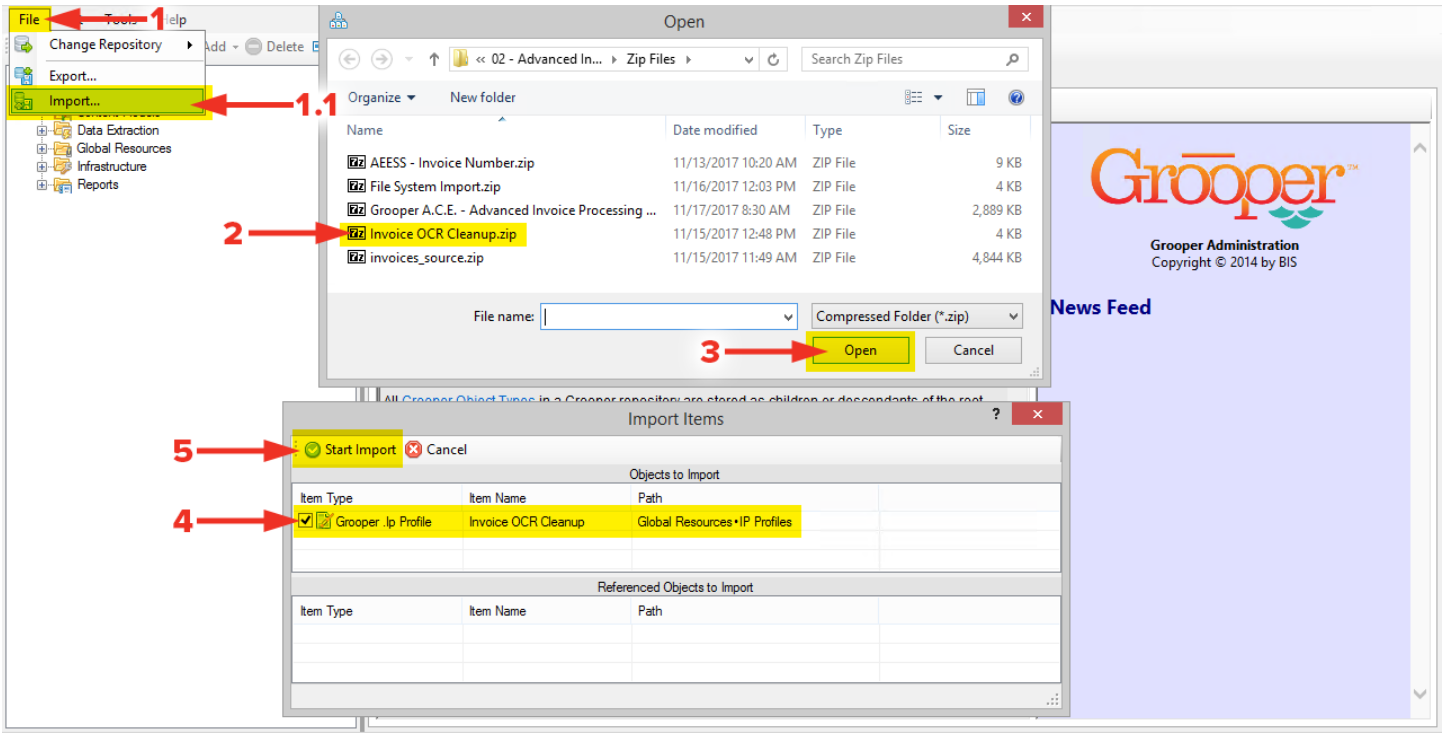
The previous documentation, Overview and Concepts, focused on the creation of an IP Profile during this Conditioning phase. The documents being worked with for these exercises don't require permanent cleanup, so the focus this time will be on creating an OCR Profile.

CREATING AND TESTING A NEW OCR PROFILE

An IP Profile will be leveraged for the non-permanent OCR Cleanup property, and that's been built and supplied.

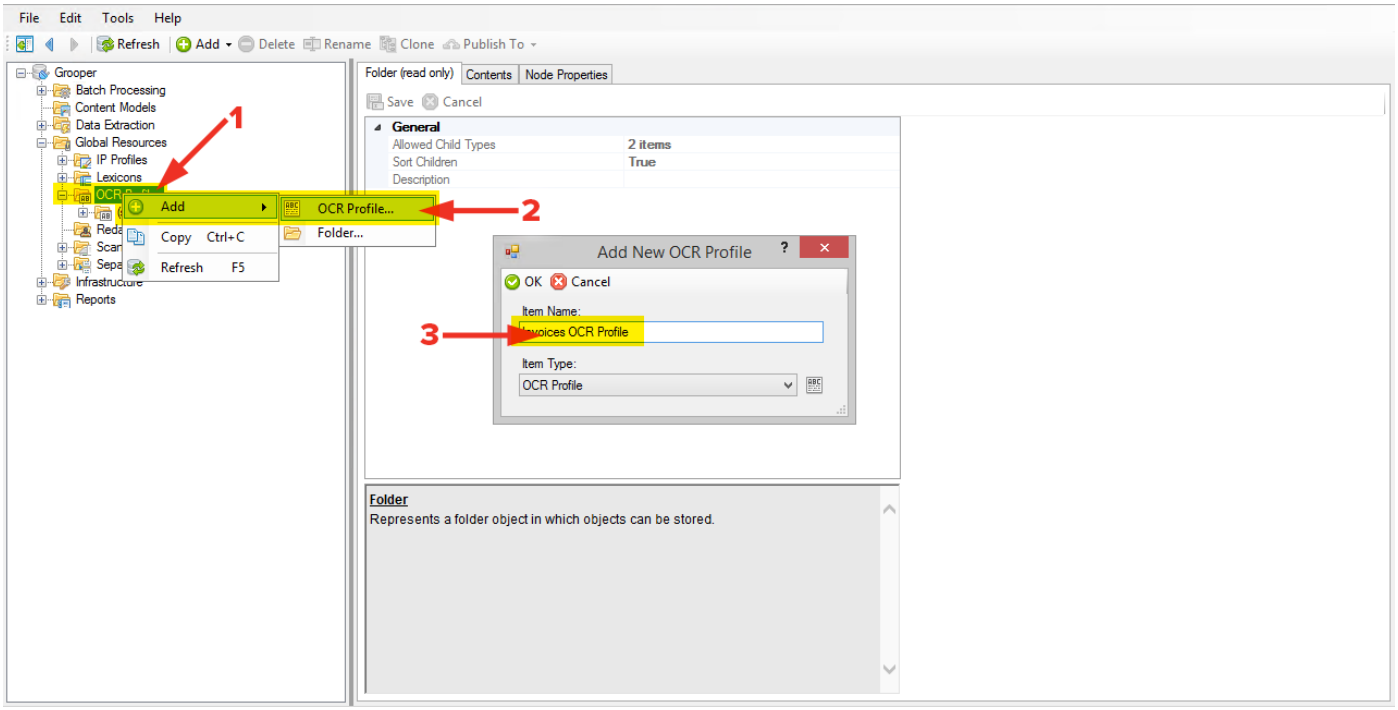
STEP 1 – IMPORT PRE-BUILT IP PROFILE

Start by **(1)** clicking File > Import, and from the Open window **(2)** select the Invoice OCR Cleanup.zip file and **(3)** click Open. In the Import Items window, **(4)** select the Grooper .Ip Profile object and **(5)** Start Import.



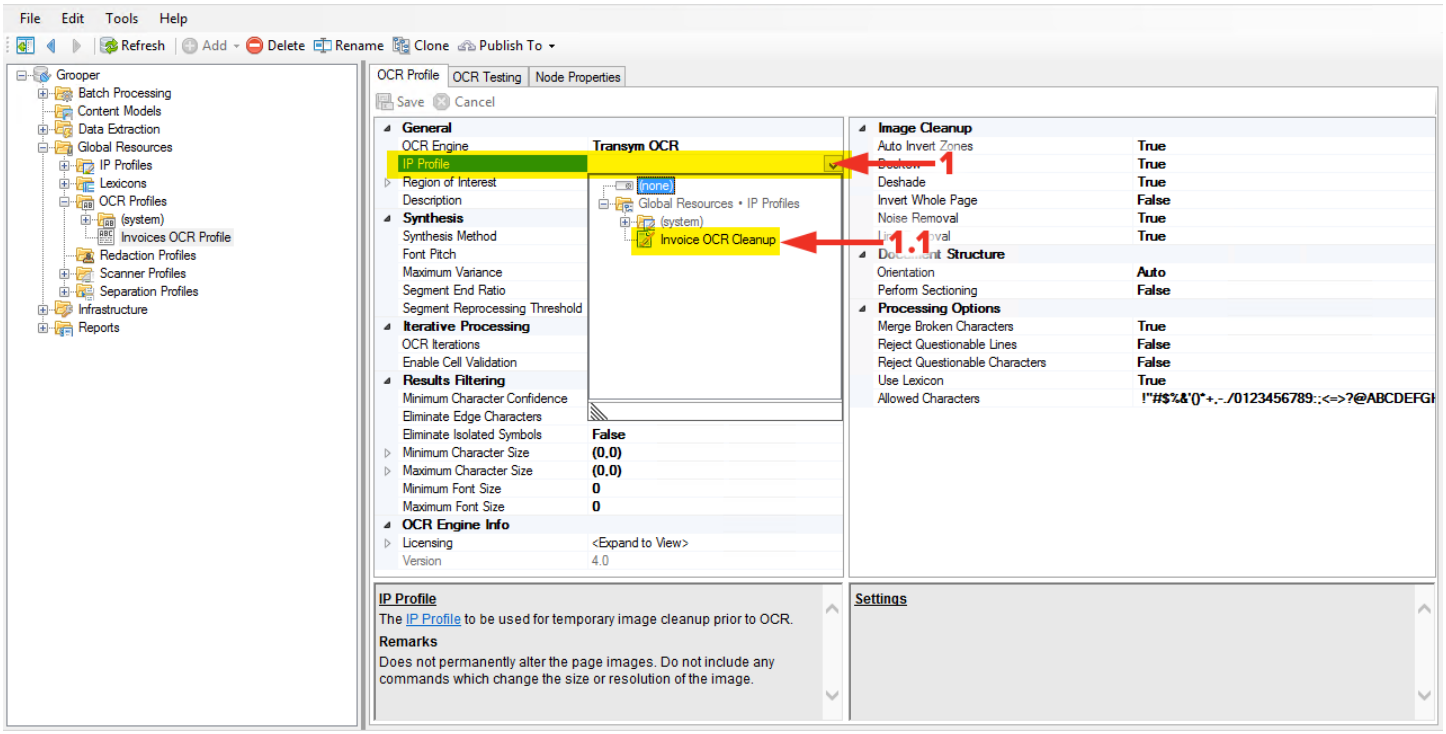
STEP 2 – CREATING A NEW OCR PROFILE

(1) Navigate to Grooper > Global Resources, **(2)** add a new OCR Profile, and **(3)** name it Invoices OCR Profile.



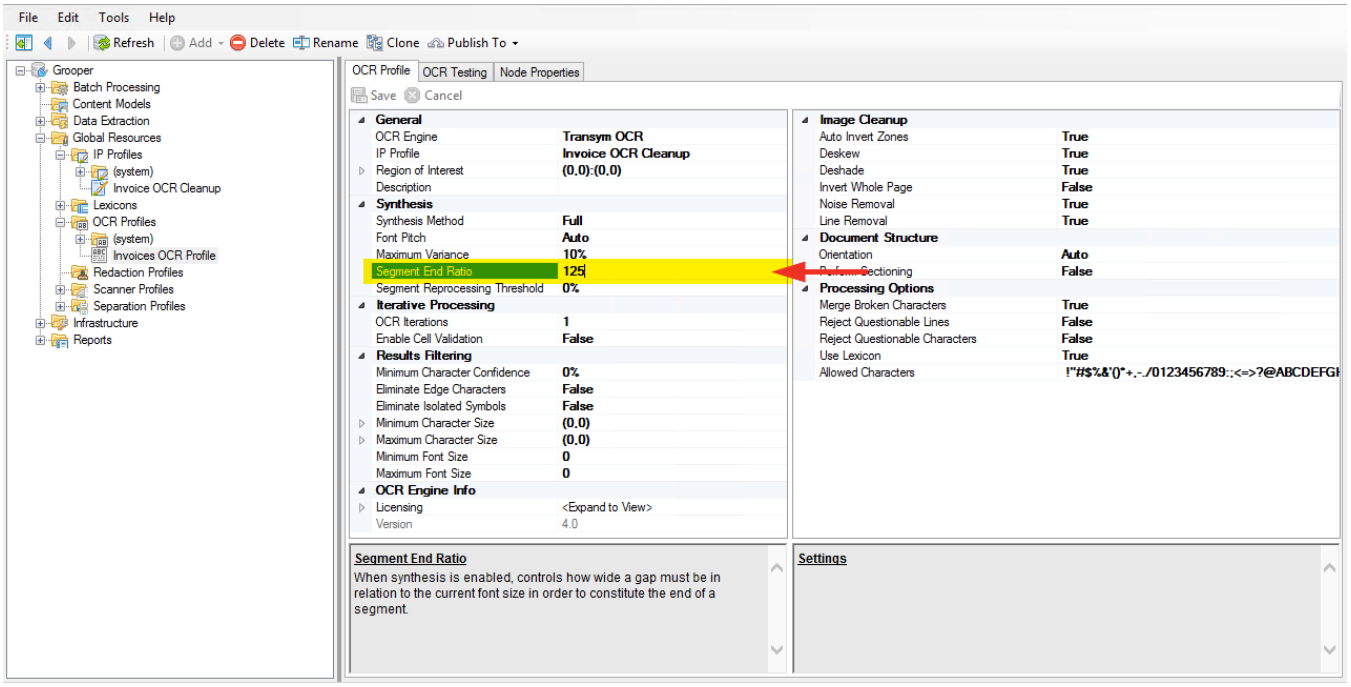
STEP 3 – SET IP PROFILE

With the newly created OCR Profile selected, **(1)** set the IP Profile to the one that was just imported.



STEP 4 – SYNTHESIS SETTINGS • SEGMENT END RATIO

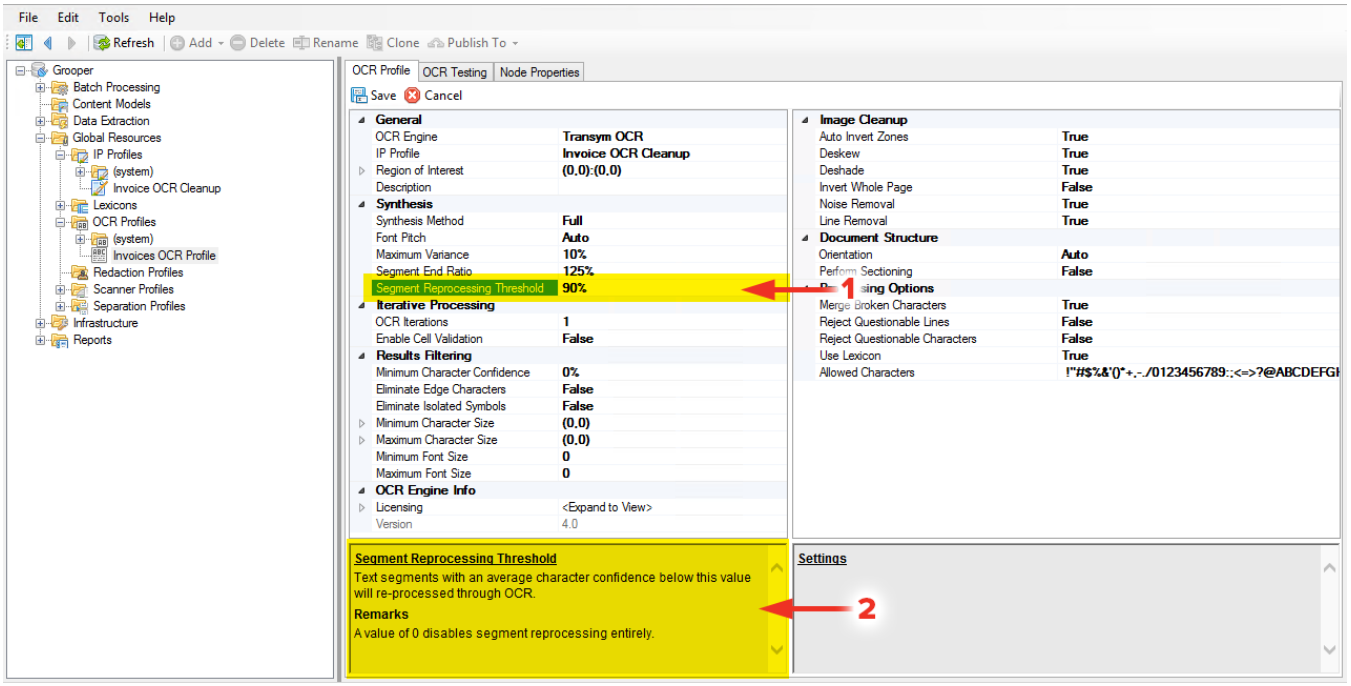
A segment in OCR'ed text is a set of synthesized characters on a line that are not separated by too large a space. The space considered "too large" is set by Segment End Ratio, in the Synthesis section. Set this to 125%, so a space must be 125% larger than the detected font size for it to be considered the end of a segment.



STEP 5 – SYNTHESIS SETTINGS • SEGMENT REPROCESSING THRESHOLD

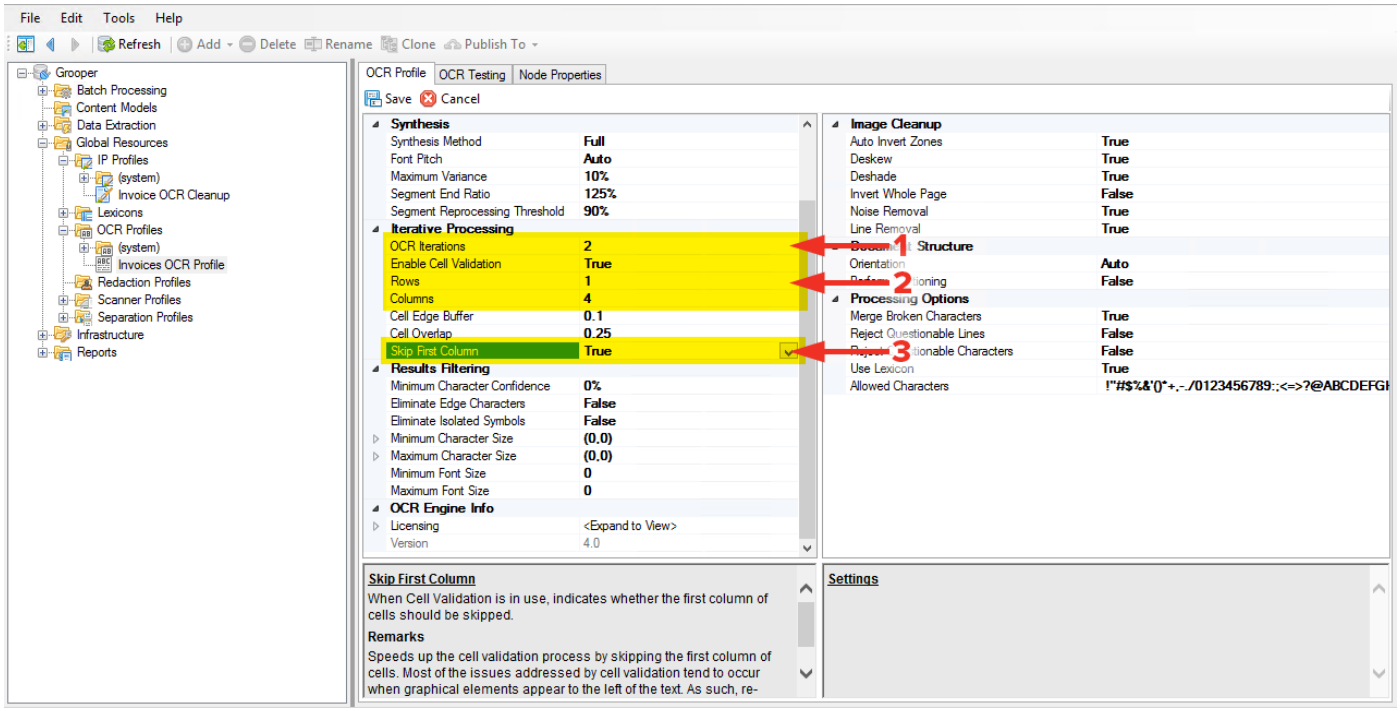
Now that a segment is understood, **(1)** set Segment Reprocessing Threshold to 90% and **(2)** take a moment to read over the tooltip:

Text segments with an average character confidence below this value will be re-processed through OCR.



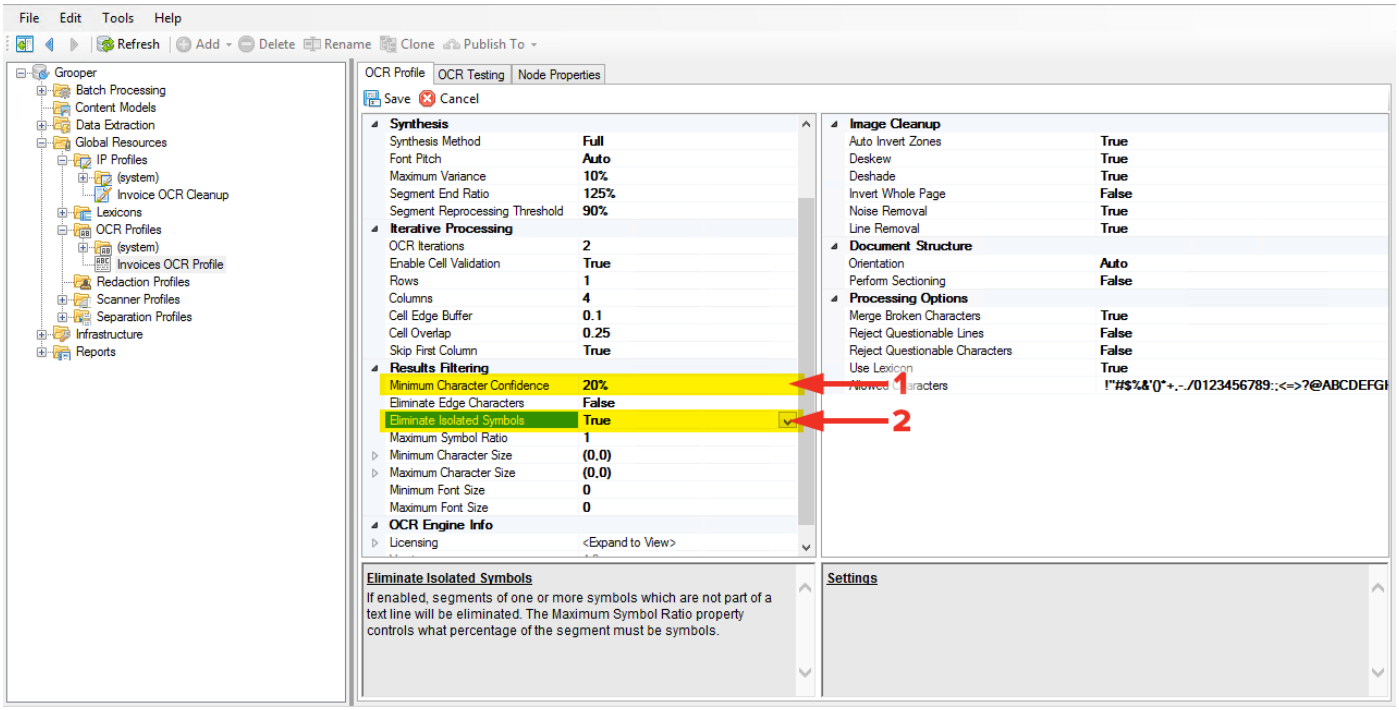
STEP 6 – ITERATIVE PROCESSING • OCR ITERATIONS AND CELL VALIDATION

In the Iterative Processing section, **(1)** set OCR Iterations to 2. **(2)** Enable Cell Validation (which will present a new set of properties) and set the Rows and Columns to 1 and 4 respectively. **(3)** Set Skip First Column to True.



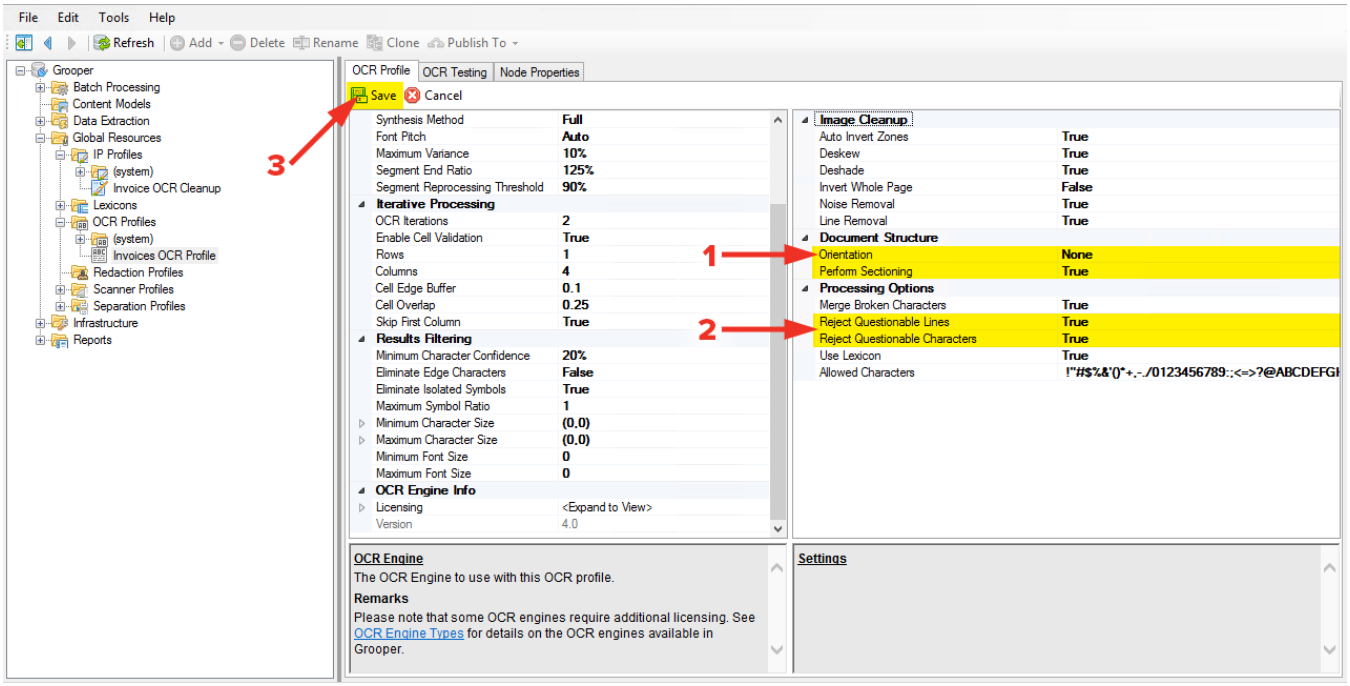
STEP 7 – RESULTS FILTERING

In the Results Filtering section, **(1)** set Minimum Character Confidence to 20% and **(2)** Eliminate Isolated Symbols to True.



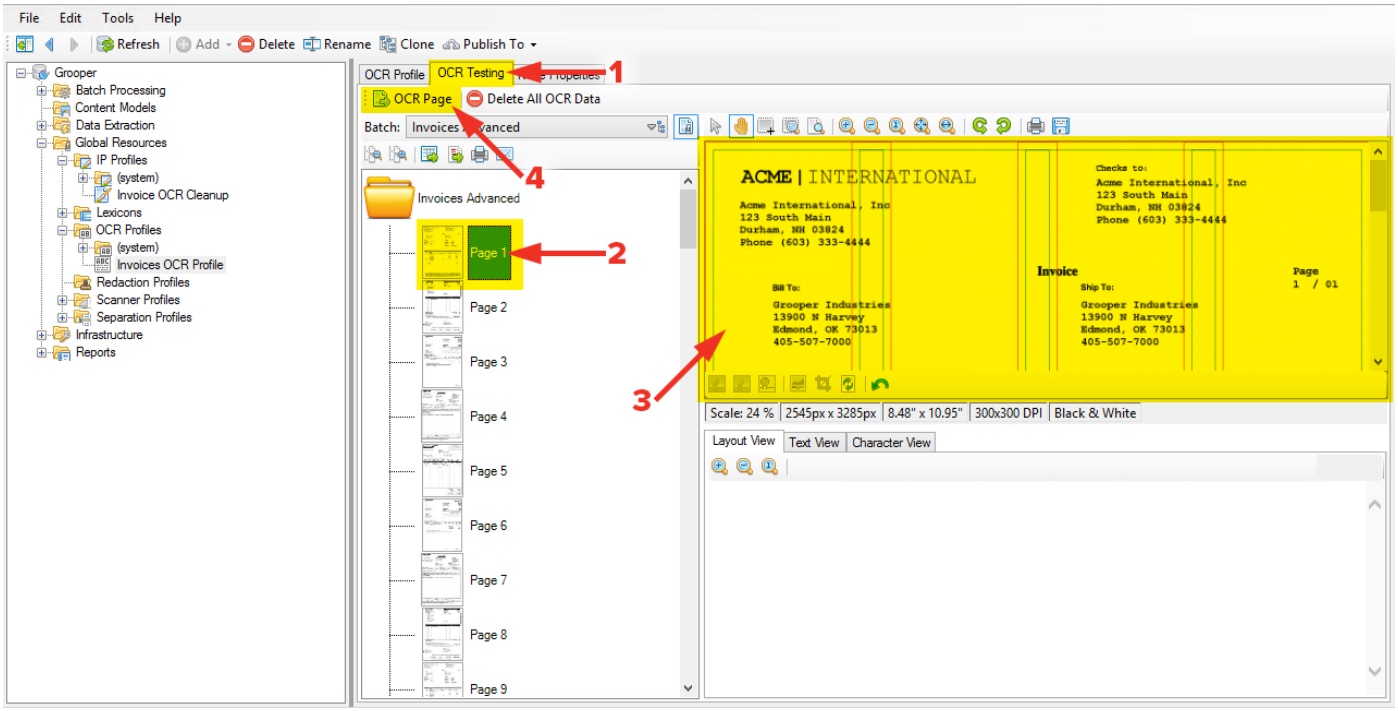
STEP 8 – FINAL ADJUSTMENTS

Finally, **(1)** set Orientation to None and Perform Sectioning to True, **(2)** set Reject Questionable Lines and Reject Questionable Characters to True. **(3)** Click the Save button.



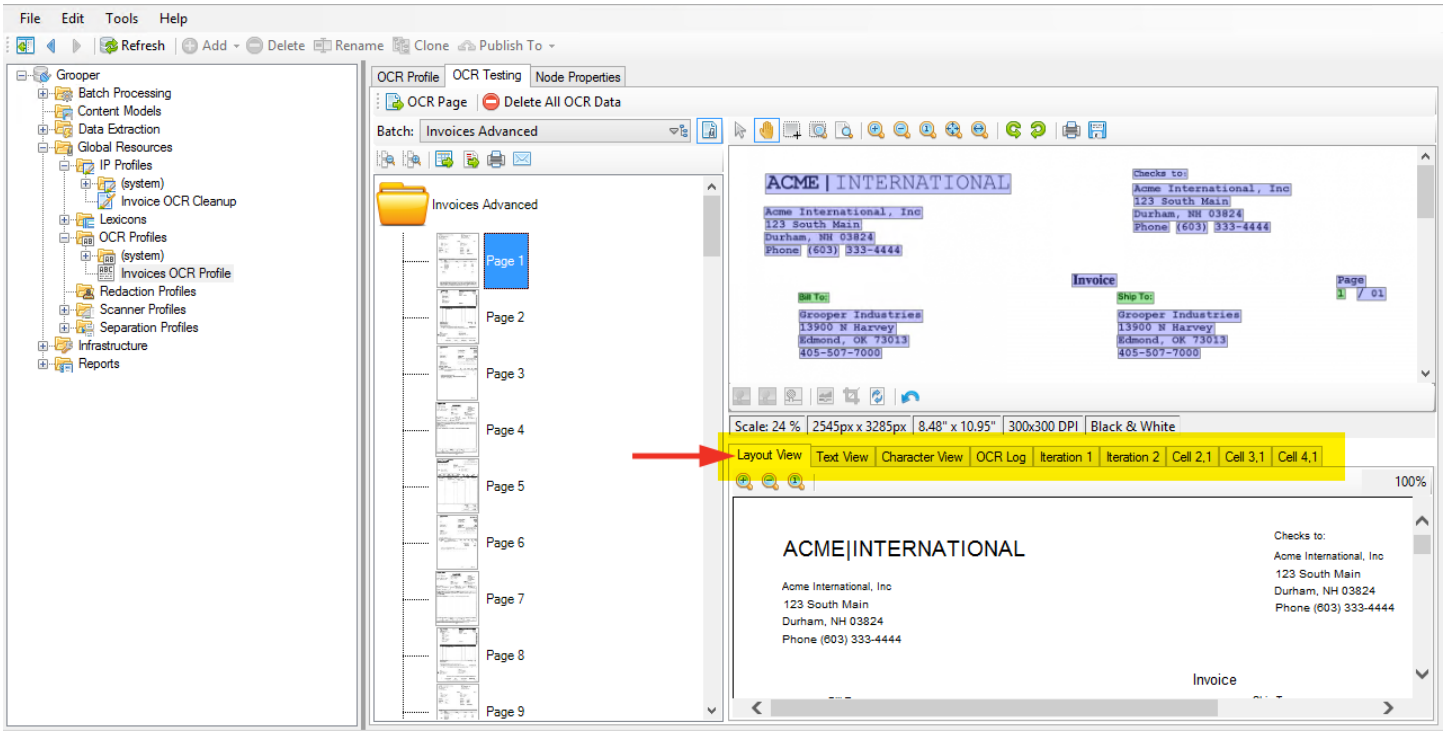
STEP 9 – TEST THE OCR PROFILE

(1) Click on the OCR Testing tab. **(2)** Select Page 1 from the Batch Viewer. **(3)** You'll notice in the Page Viewer how the selected page is divided into columns and that those columns overlap due to the column settings from the Iterative Processing section. **(4)** Click the OCR Page button.



STEP 10 – OBSERVE RESULTS

Look at the different tabs that are available as a result of the OCR Page test. Take note that this "test" permanently altered the page object as it applied OCR results.

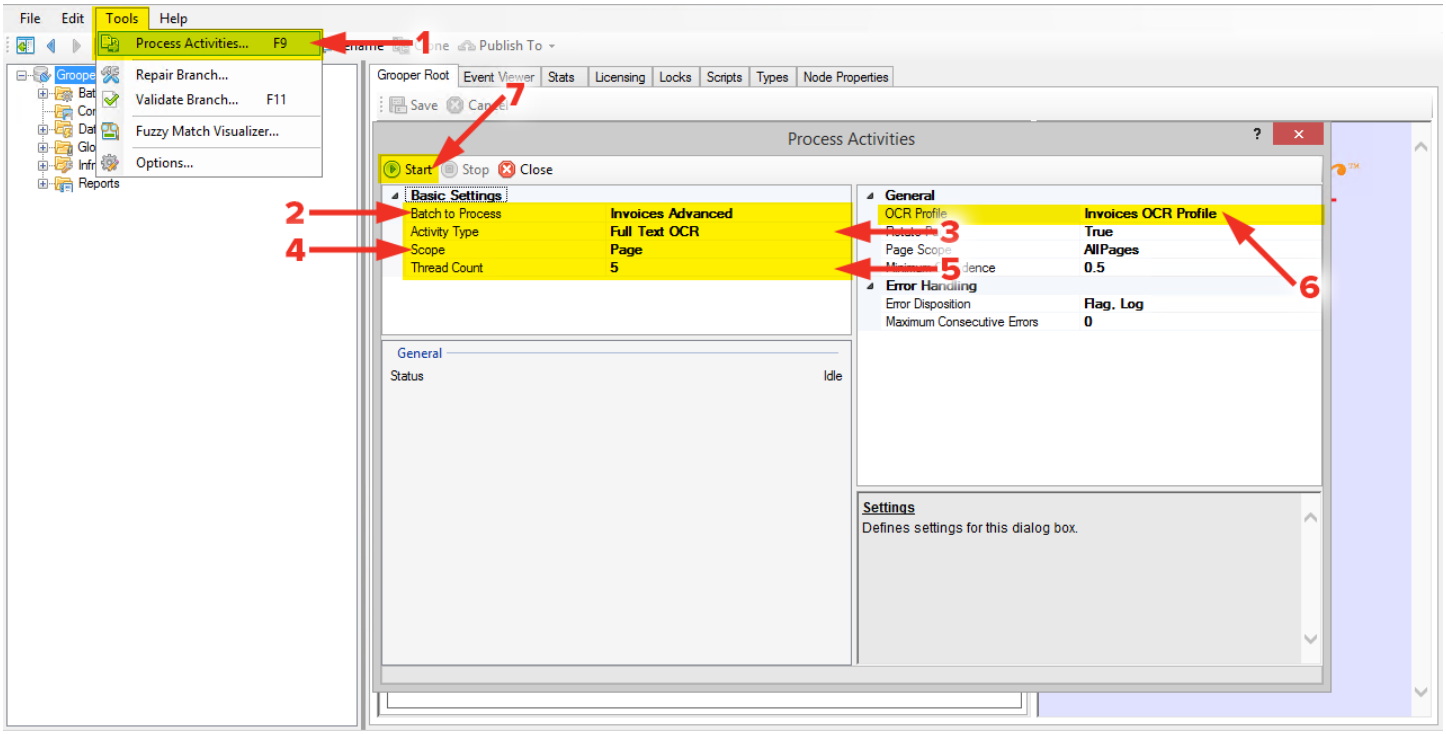


The Layout View tab shows a representation of the OCR'ed characters in a view that resembles the image.
The Text View tab shows the characters in their synthesized layout including carriage returns and line breaks.
The Character View tab shows a table with each row being each individual character OCR'ed, and columns representing specific results about that character from its XY coordinates on the page to percentage confidence results, etc.
The OCR Log tab displays information regarding specificities of the OCR process.
Finally, because iterations and cells were used in the OCR Profile, the remaining tabs show the characters read from the different iterations, as well as the images made from the column breakouts.

STEP 11 – PROCESS OCR ON THE BATCH

With the OCR Profile created and successfully exported, select the Grooper Root Node and then click the Refresh button that is directly above it. After this, **(1)** click on the Tools menu and select Process Activities. In the Process Activities window **(2)** select Invoices Advanced for the Batch to Process, **(3)** Full Text OCR for the Activity Type, **(4)** Page for Scope, **(5)** an amount that doesn't exceed your Grooper OCR license count for Thread Count, **(6)** Invoices OCR Profile for OCR Profile, and the other settings can be defaulted. **(7)** Click Start, and the OCR Profile will be run against all pages in the batch.

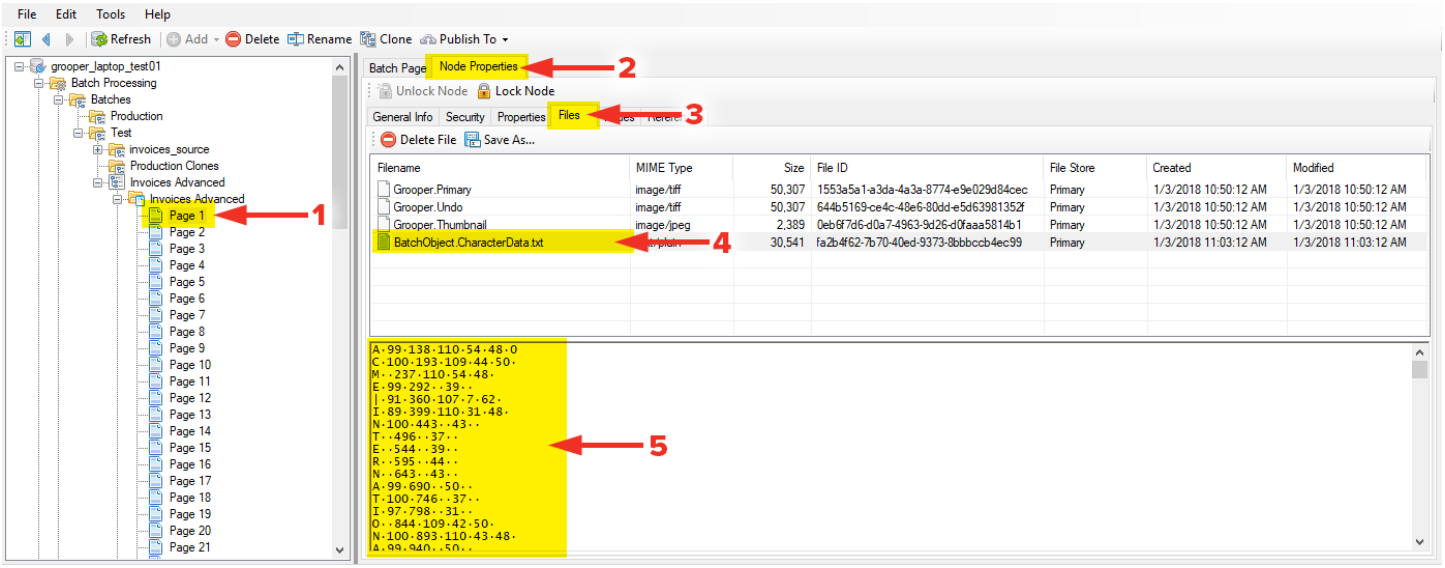
At this point a blue progress bar will begin to fill the bottom of the Process Activities window. Once it completes, a Process Completed window will pop up and you can close it. Once done, close the Process Activities window.



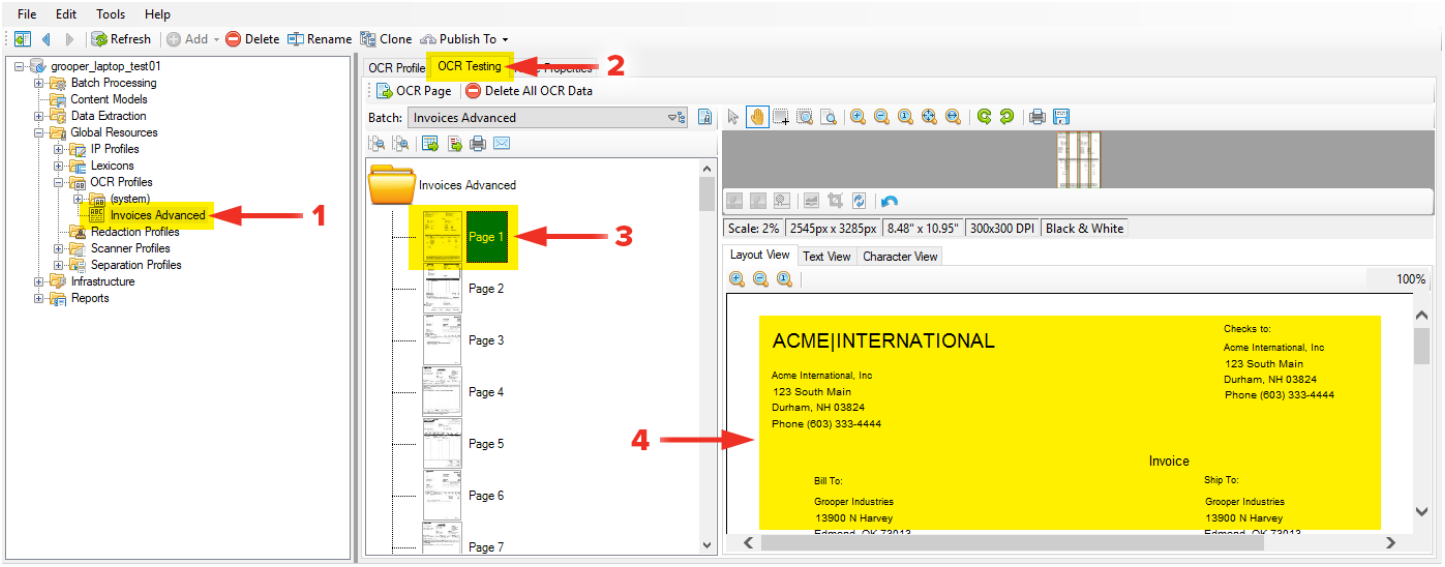
This ability to perform individual activities on the fly like this (via Process Activities) is a powerful feature in **Grooper** as it allows testing and results to be reached quickly and dynamically. You don't have to build out an entire Batch Process and test from end to end to see immediate results. As is probably obvious as well, this is not limited to Full Text OCR, as you can immediately process just about any activity in Grooper.

STEP 12 – CHECK FOR OCR RESULTS

You may find yourself at some point asking, "has this page/batch been OCRed?". There are a couple of places to check for this. **(1)** The first way is to, navigate all the way to the page object within a batch folder object. With a page object selected **(2)** click on the Node Properties tab. From there, **(3)** click on the Files tab. **(4)** Select the BatchObject.CharacterData.txt file (the existence of this file is enough…) and **(5)** notice all the character data in the Document Viewer panel.



The next place to place to determine if you have OCR results is within the OCR Testing tab of an OCR Profile. **(1)** Select any OCR Profile and **(2)** click on the OCR Testing tab. **(3)** Select a page in the Batch Viewer and **(4)** in the Layout View, if you have any characters visible, that page has been OCRed.



PHASE 3 – ORGANIZE

Separation and Classification are the next activities to be run against our batch. Control Sheets have been removed from the pages in this batch to show a new method of Separation. ESP Auto Separation could still be used to classify the documents once separated, but again, a different approach using extractors will be taken to show a different means of classifying than has been seen previously.

SEPARATION – CHANGE IN VALUE

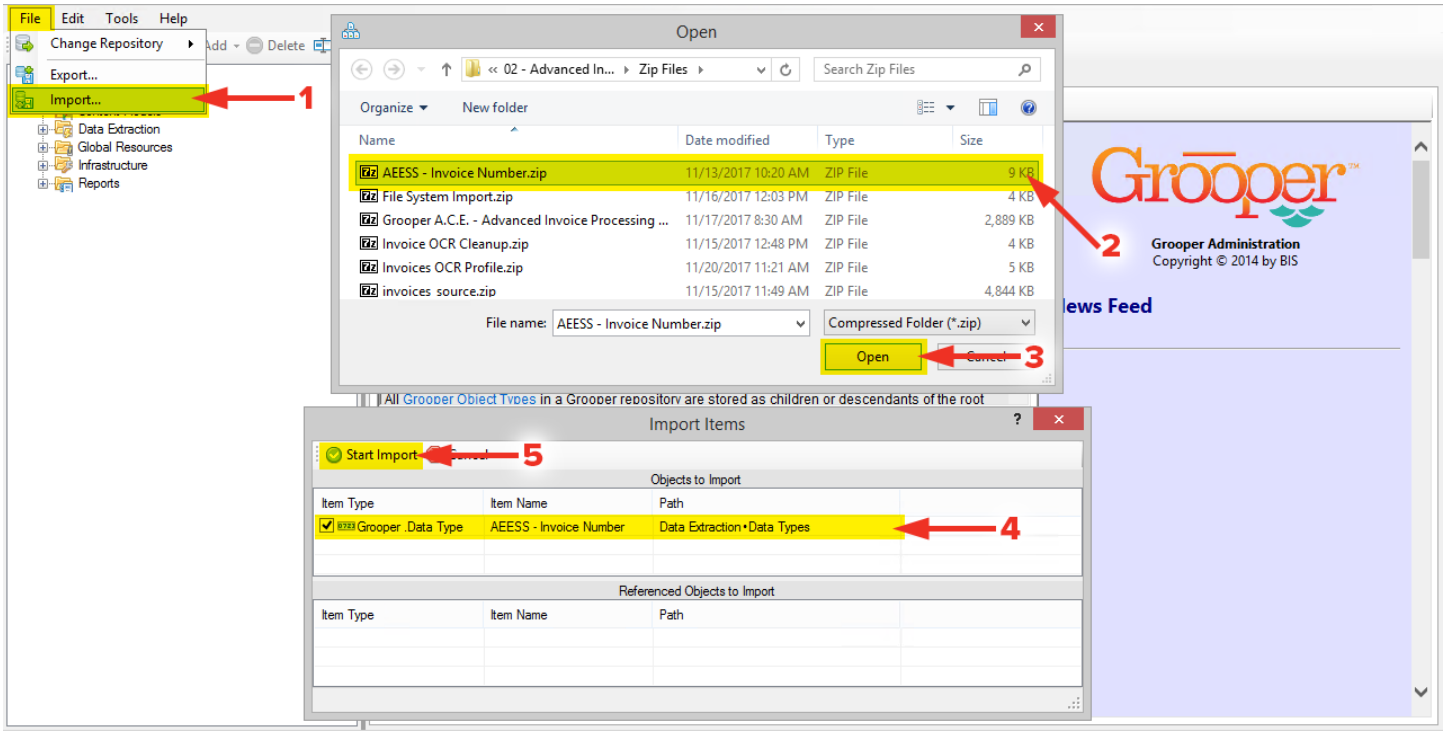
For this method of Separation to work, an extractor will be used in a Separation Profile to find a specific value on the pages. Until the value found changes, pages will be sorted together and treated as a single document. This Change in Value approach works well for invoices.

SETTING UP A SEPARATION PROFILE TO LEVERAGE A CHANGE IN VALUE

STEP 1 – IMPORT DATA TYPE

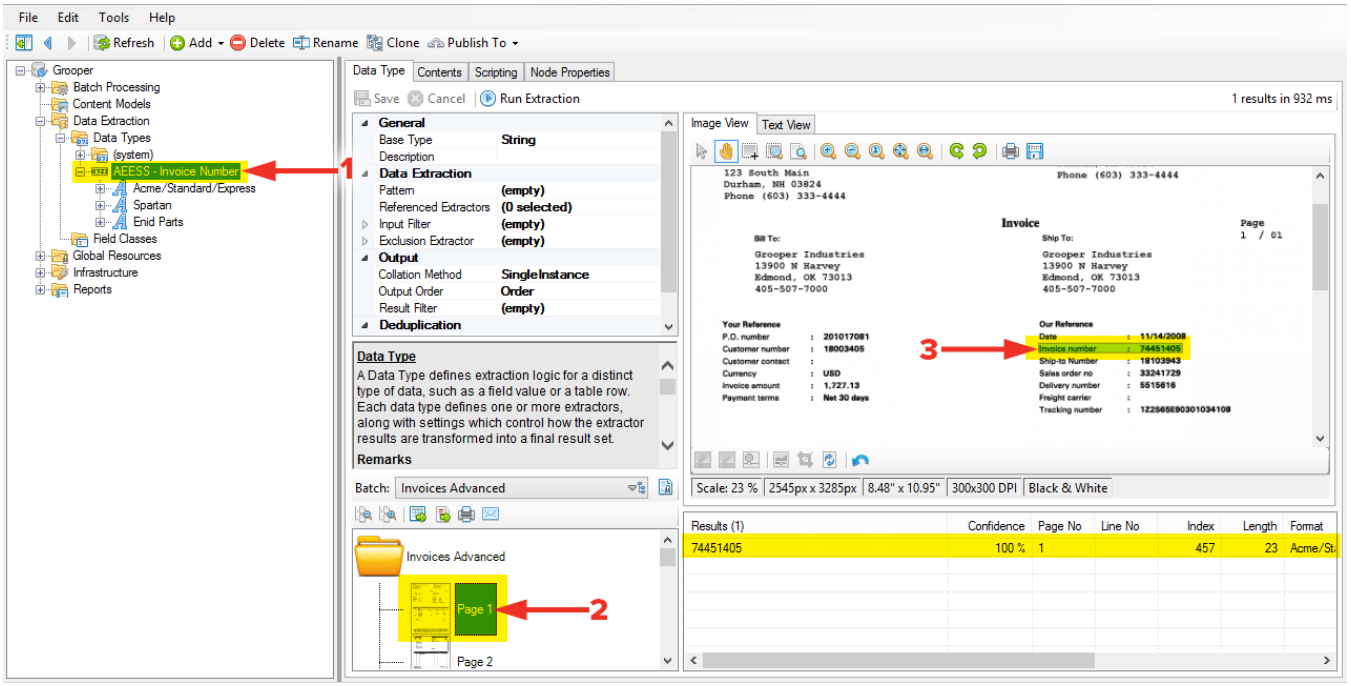
Instead of building the extractor necessary to perform this operation, the Data Type that was built during the Overview and Concepts exercises was exported and will be used here.

Start by **(1)** clicking File > Import and in the Open window **(2)** select the AEESS – Invoice Number.zip file **(3)** then click Open. In the Import Items window **(4)** select the Data Type and **(5)** click Start Import.



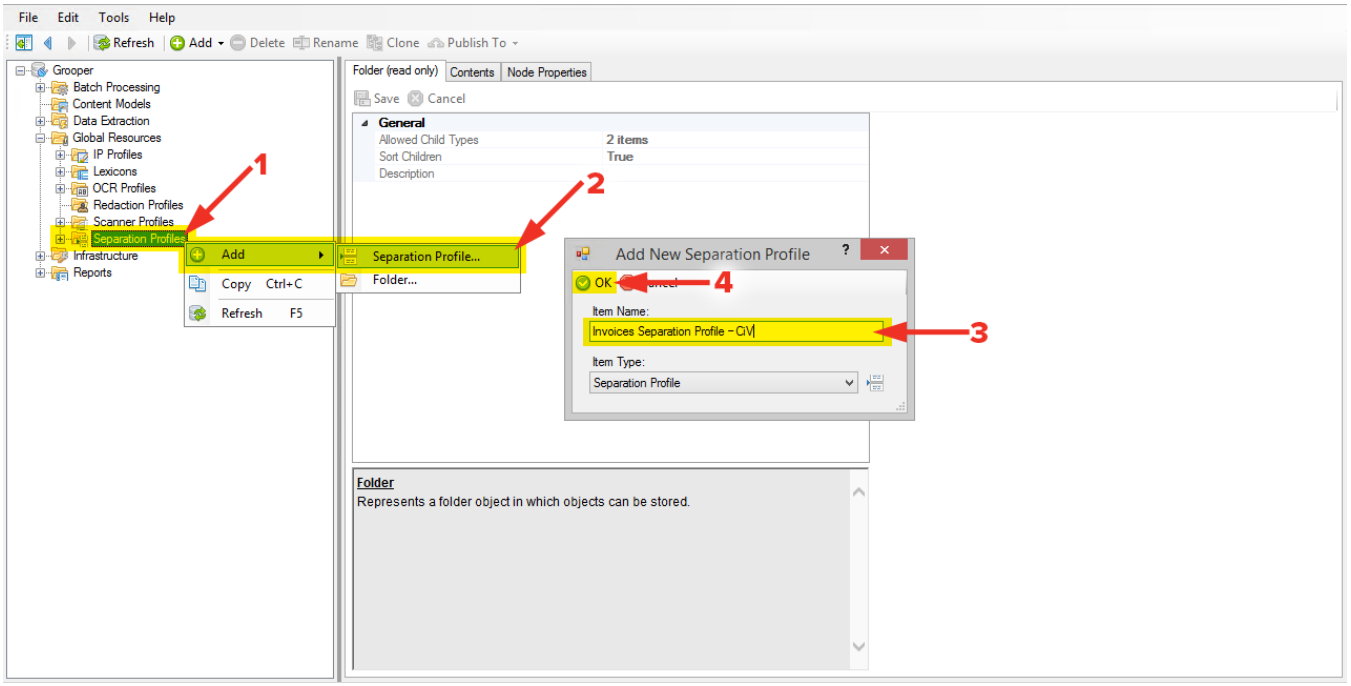
STEP 2 – REVIEWING THE DATA TYPE

To refresh your memory of what this Data Type did, **(1)** navigate to it in the node tree and **(2)** select a page object of the Batch Viewer. **(3)** The Invoice Number for this page will be highlighted, as will that of the other pages if you click through them as well.



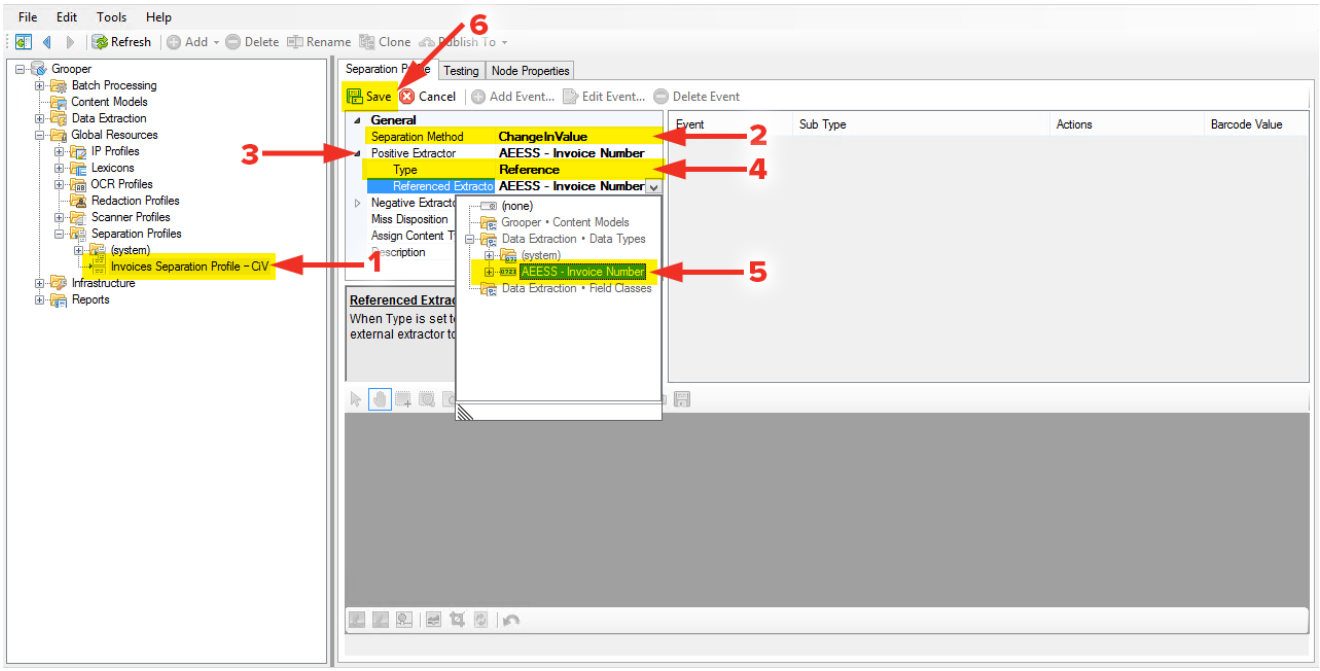
STEP 3 – SETTING UP THE SEPARATION PROFILE

(1) Navigate to Grooper > Global Resources and select Separation Profiles. **(2)** Right click and Add > Separation Profile... In the Add New Separation Profile window **(3)** name it Invoices Separation Profile – CiV and **(4)** click Ok.



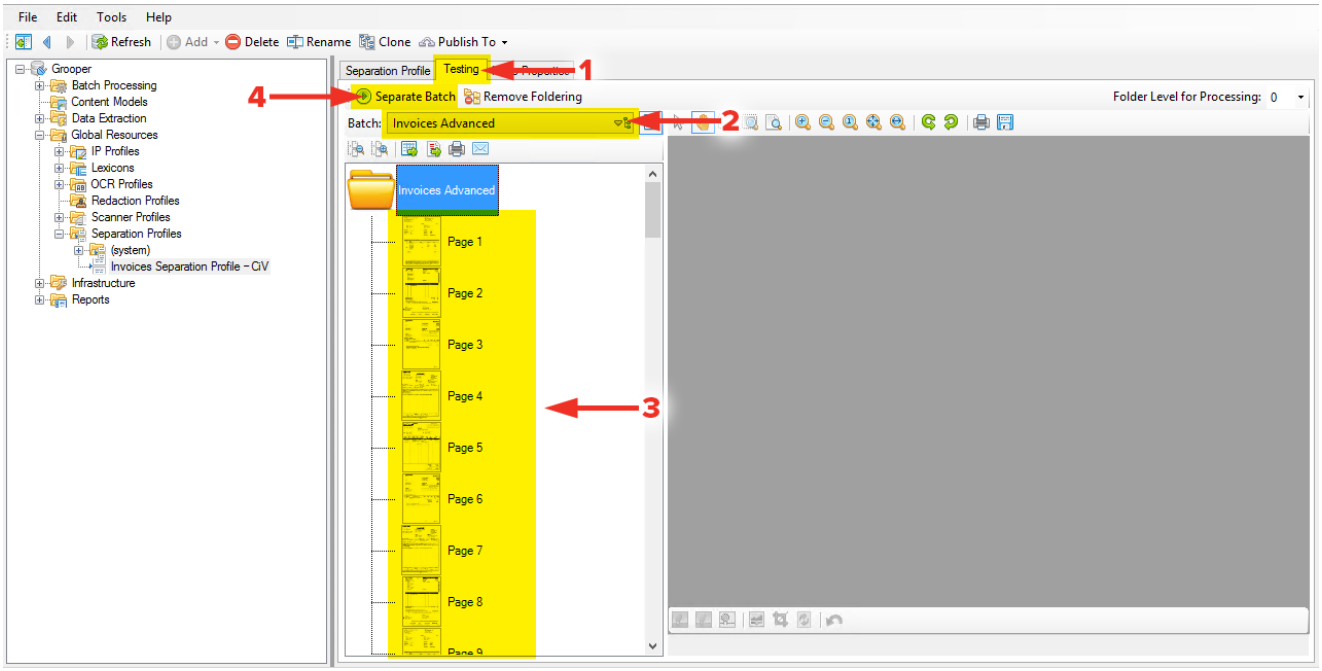
STEP 4 – SETTING PROPERTIES FOR NEW SEPARATION PROFILE

(1) With the newly created Separation Profile selected, **(2)** set the Separation Method to ChangeInValue. **(3)** Expand Positive Extractor and **(4)** set its Type to Reference, and **(5)** set the Referenced Extractor to the AEESS – Invoice Number extractor that was imported. **(6)** Click Save.



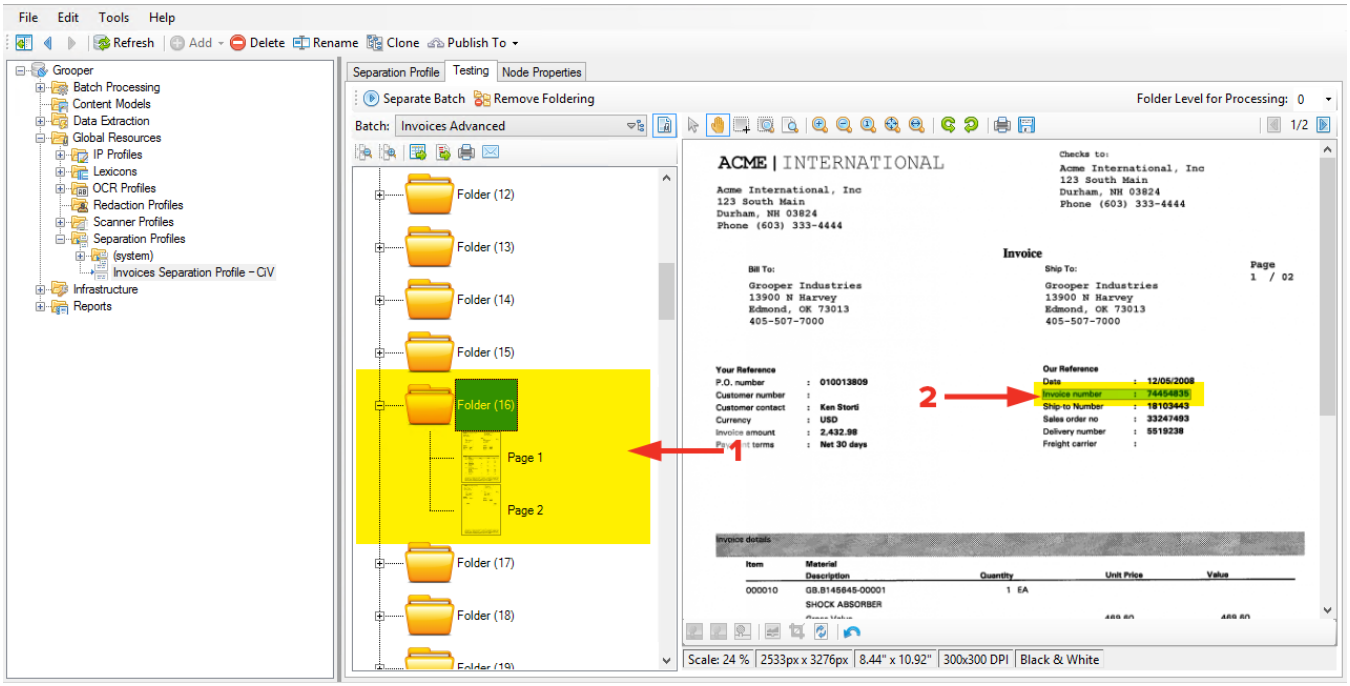
STEP 5 – TESTING SEPARATION

(1) Click the Testing tab and **(2)** make sure the Invoices Advanced batch is selected in the Batch: drop-down. **(3)** Observe the Batch Viewer and notice the pages are loose and not separated. **(4)** Click the Separate Batch button and separation will execute each time the Data Type that was imported detects a change in the value extracted. As a result, the loose pages will be in folders which can now be classified.



STEP 6 – TWO PAGE EXAMPLE

(1) In the Batch Viewer, scroll down to Folder (16) and notice there are two pages in this document.
(2) Observe the value extracted on page 1 and on page 2 and this should help illustrate that since the value didn't change between these pages, they were combined into one document.



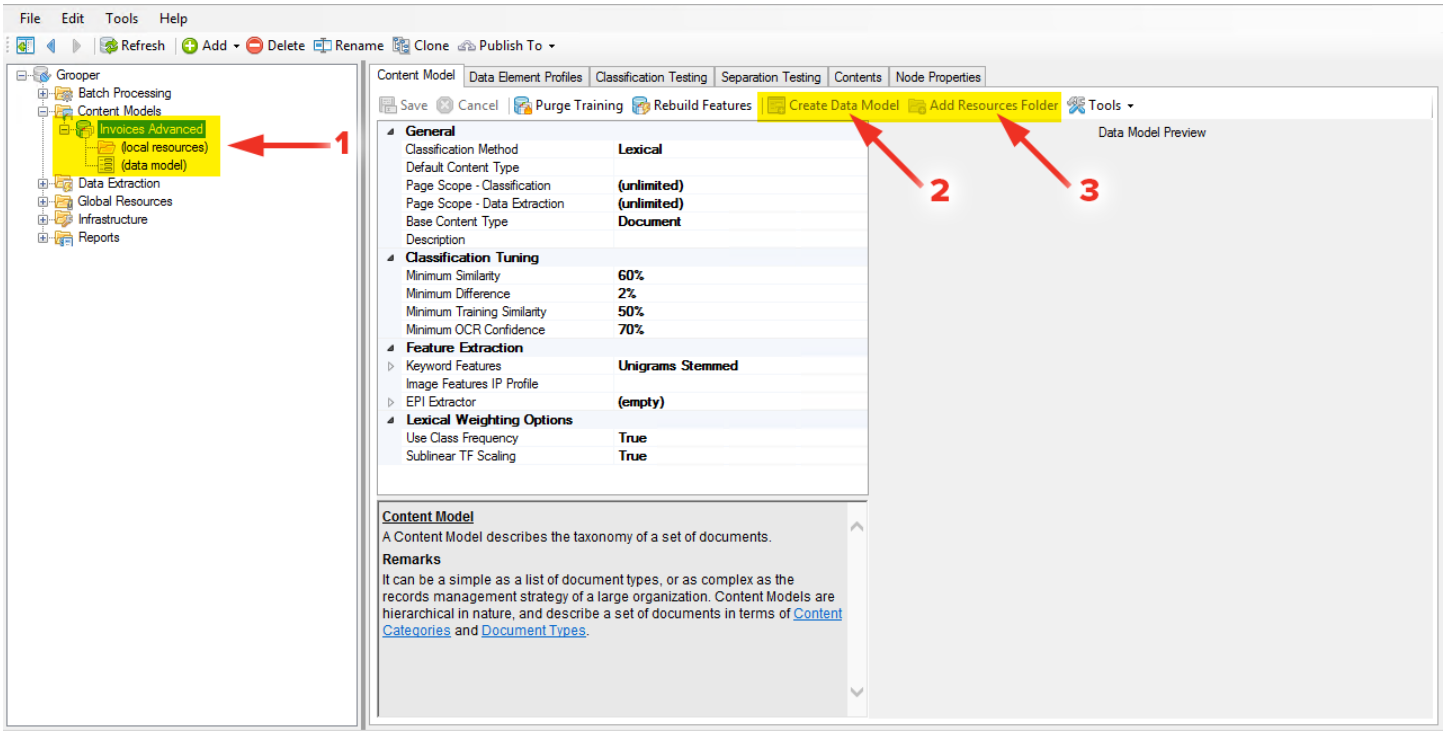
CLASSIFICATION – POSITIVE EXTRACTOR ON DOCUMENT TYPE

In previous exercises, Classification was performed because of Lexical analysis and training of documents. That approach would still work here, but learning a new method would be prudent. This new means of Classification allows a Document Type to leverage an extractor to find a value. If/when that value is found, the document is classified as that Document Type.

SETTING UP POSITIVE EXTRACTORS FOR DOCUMENT TYPES

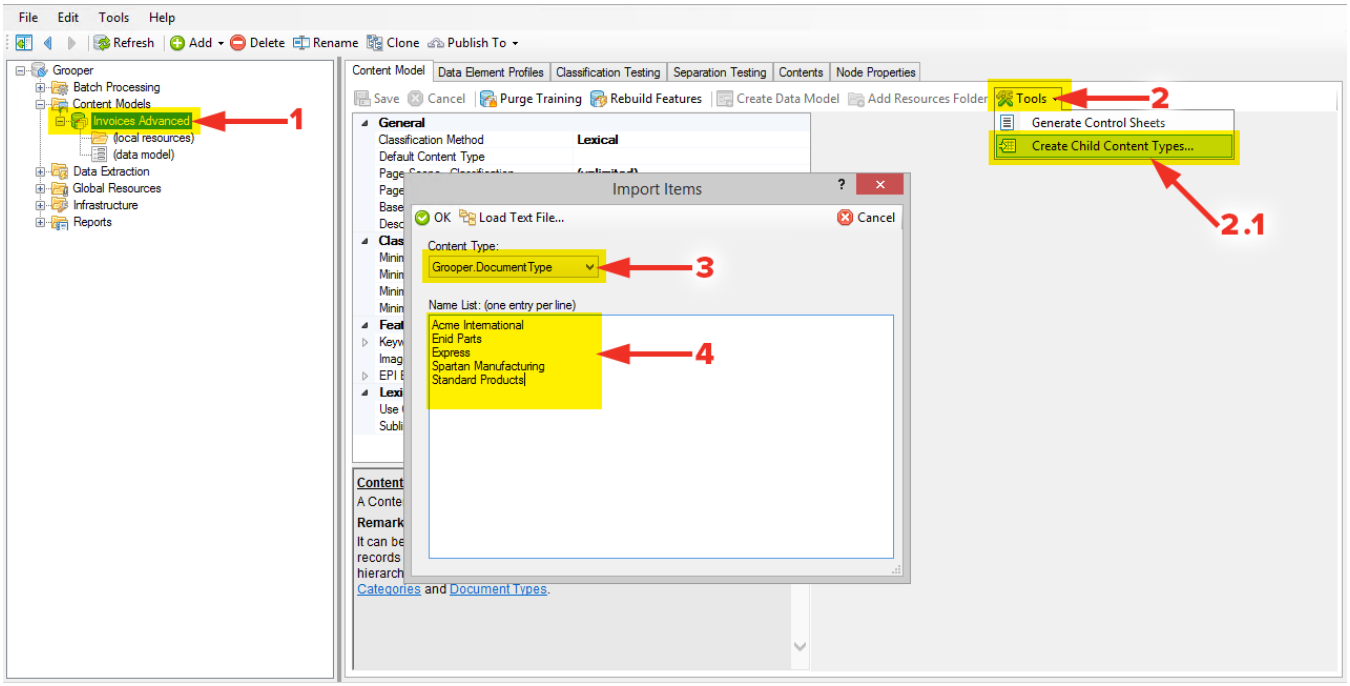
STEP 1 – CREATING A CONTENT MODEL

(1) Create a new Content Model and name it Invoices Advanced. **(2)** Add a (local resources) folder **(3)** and a (data model).



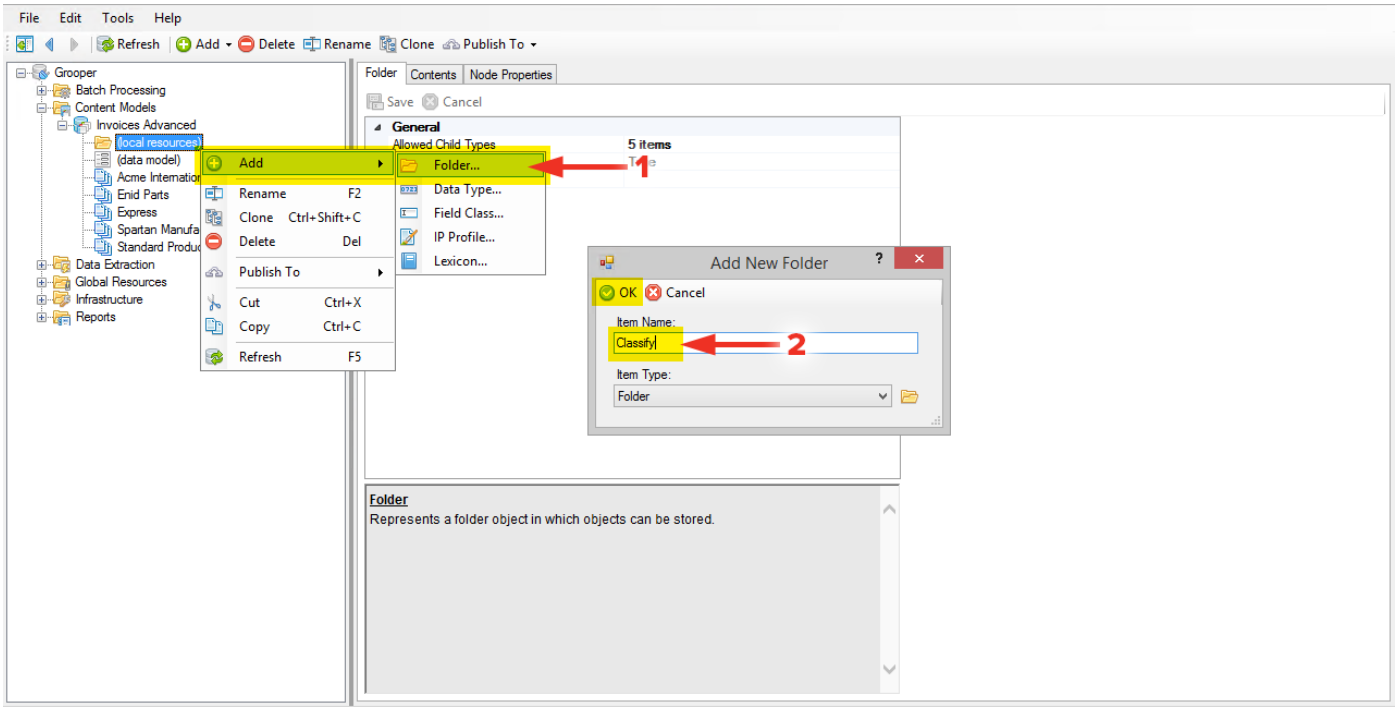
STEP 2 – ADD DOCUMENT TYPES

(1) With the newly created Content Model selected, **(2)** click on the Tools drop-down menu and select Create Child Content Types… In the Import Items menu, **(3)** make sure Content Type: is set to Grooper.DocumentType and in the Name List: **(4)** add the following names: Acme International, Enid Parts, Express, Spartan Manufacturing, and Standard Products. Click Ok.



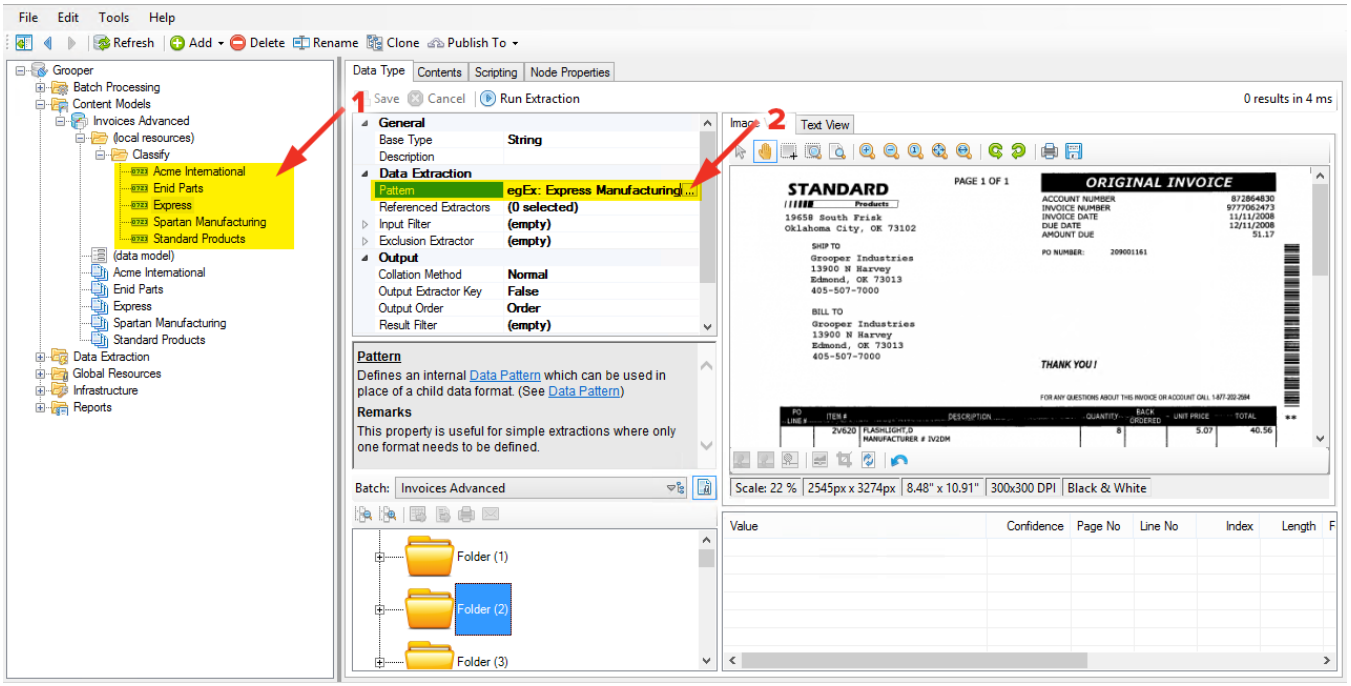
STEP 3 – ADD CLASSIFY FOLDER TO LOCAL RESOURCES

Select the (local resources) folder and **(1)** add a new folder **(2)** named Classify.



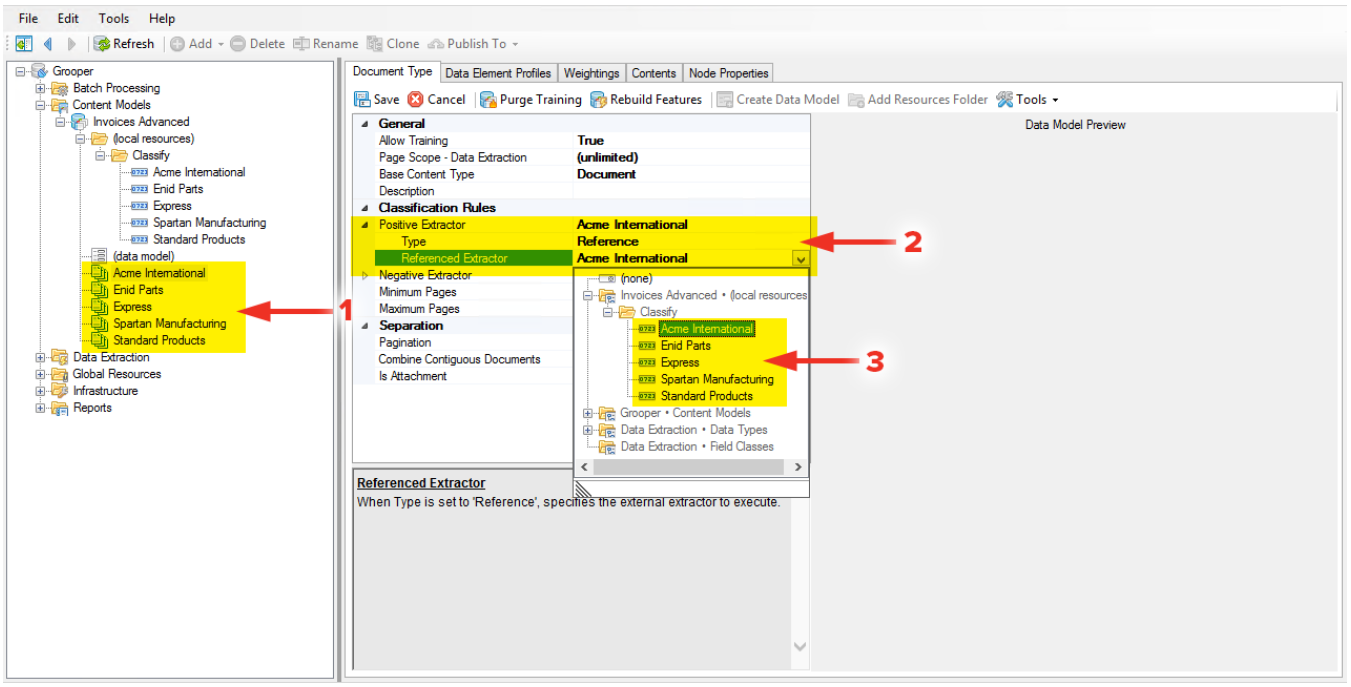
STEP 4 – ADD DATA TYPES

In the newly created Classify folder, **(1)** create 5 Data Types and **(2)** set their Pattern to reflect the title of each of the 5 company invoices: Acme International, Enid Parts, Express Manufacturing, Spartan Manufacturing, Standard Products.



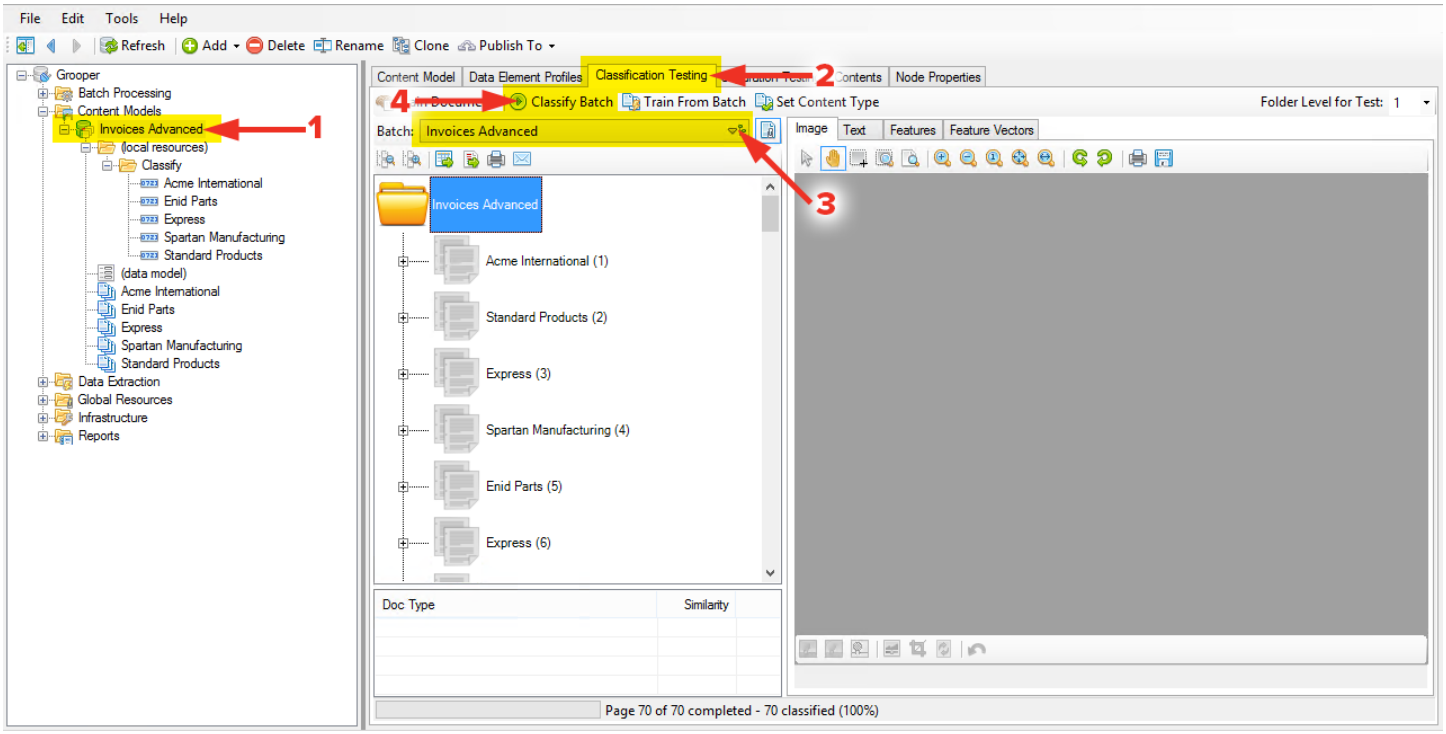
STEP 5 – DOCUMENT TYPE POSITIVE EXTRACTORS

(1) Go into each Document Type of the Content Model and **(2)** set their Positive Extractor Types to Reference, and **(3)** point the Referenced Extractor to their corresponding Data Type that was made in the Classify folder.



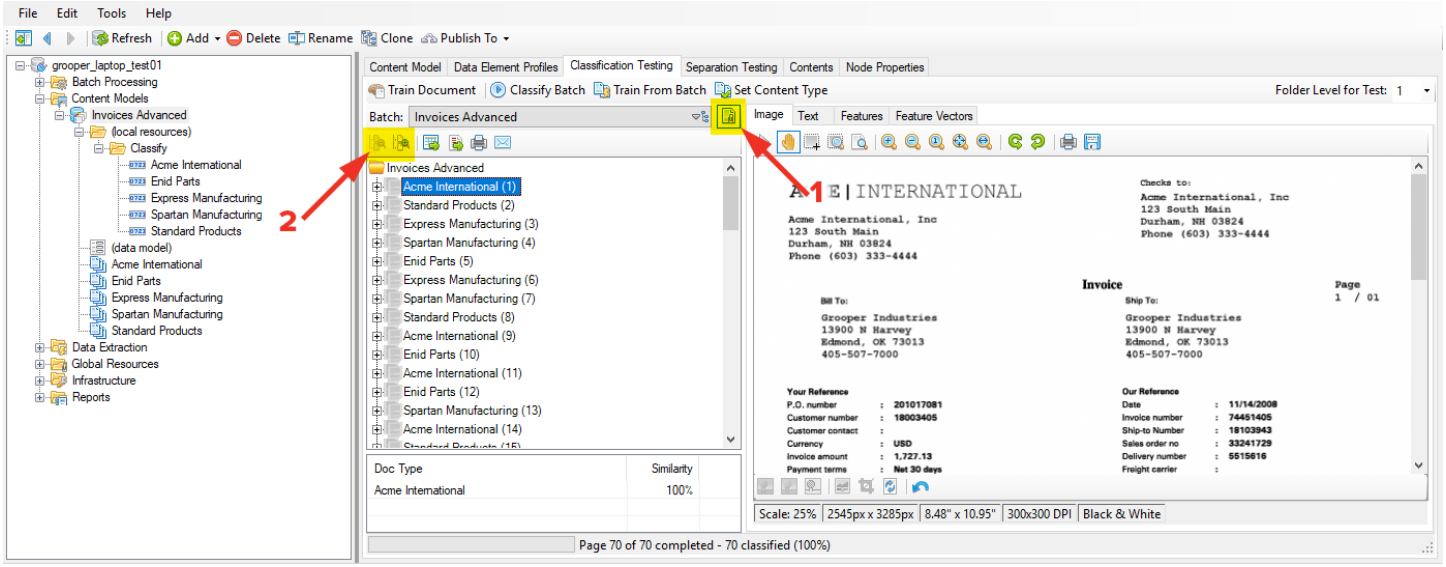
STEP 6 – CLASSIFY THE BATCH

(1) With the Content Model selected, **(2)** click on the Classification Testing tab. **(3)** Make sure the Invoices Advanced batch is selected in the Batch: drop-down. **(4)** Click the Classify Batch button (in the Classification Test Settings window that appears, leave it to DocType and click OK) and observe all the documents get accurately classified.



STEP 7 – BATCH VIEWER TOOLBAR INTERFACE BUTTONS

Finally, I'd like to call attention to some buttons in the Batch Viewer. **(1)** This button is very handy as you navigate in and out of different Batch Viewers as it will, when toggled, keep the last selected document the active/selected document. **(2)** While I have hi-lighted the zoom buttons as I use these the most frequently, there are others here for exporting, printing, and emailing selected documents.



PHASE 4 – COLLECT

As was mentioned in the introduction, the bulk of the time spent with this document will be here in **Phase 4 - Collect**. A new extractor, the Field Class, will be introduced, as well as a new Data Element, the Data Table. Properties that were glazed over in previous exercises will be critical in getting desired results now. More, and new ways of practice with RegularExpression will be inherent and necessary to successful extraction.

ESTABLISHING A DATA MODEL AND WORKING WITH A DATA TABLE

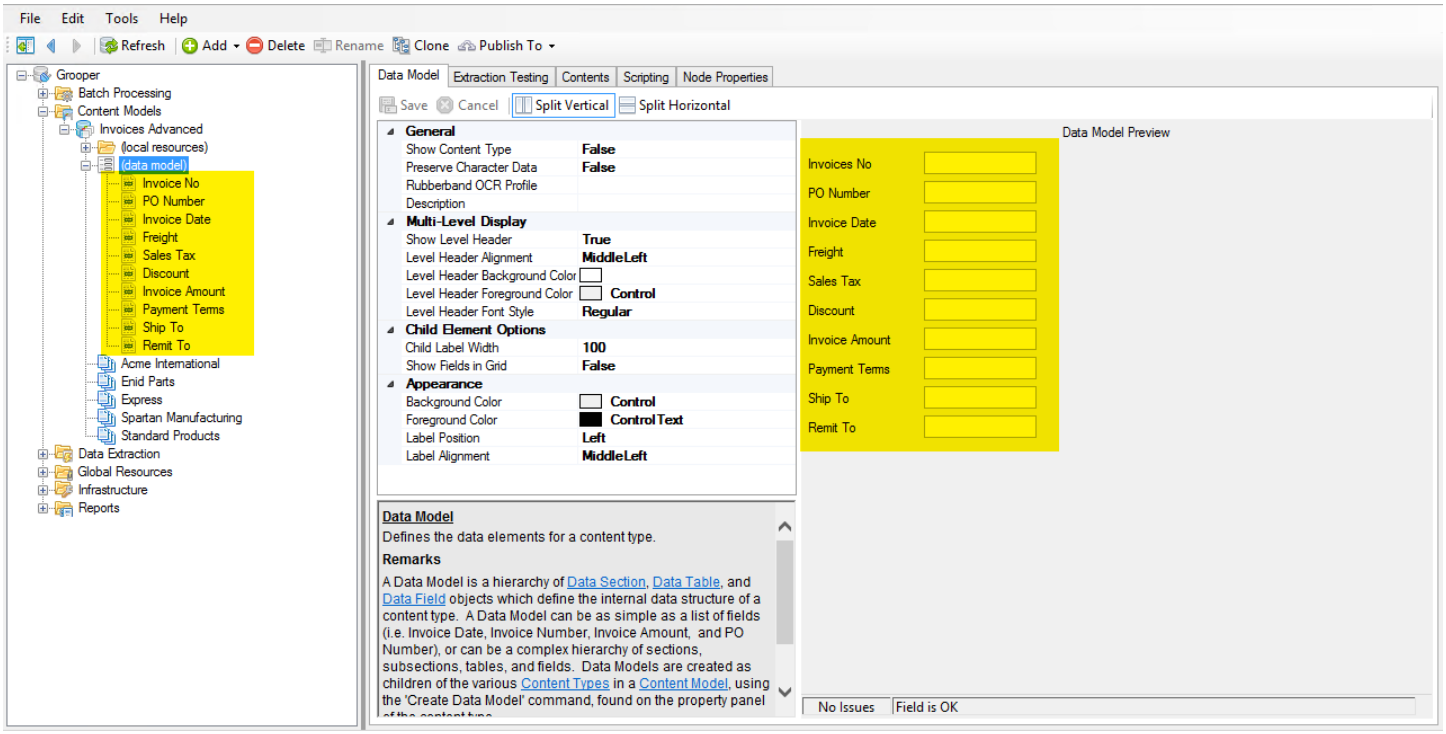
It's best to have a solid understanding of what is required from the documents so that building out the entire Data Model from the beginning can be achieved. This will add to efficiencies down the road.

SETTING UP NEW DATA FIELDS AND DATA TABLE

To begin, all the Data Fields will be created. Once the desired fields of information are firmly established, how that information can be extracted is better understood, and the approach to building out the extractors will be easier.

STEP 1 – ADDING ALL THE DATA FIELDS

Add each of the following as Data Field objects to the Data Model:
Invoice No • PO Number • Invoice Date • Freight • Sales Tax
Discount • Invoice Amount • Payment Terms • Ship To • Remit To

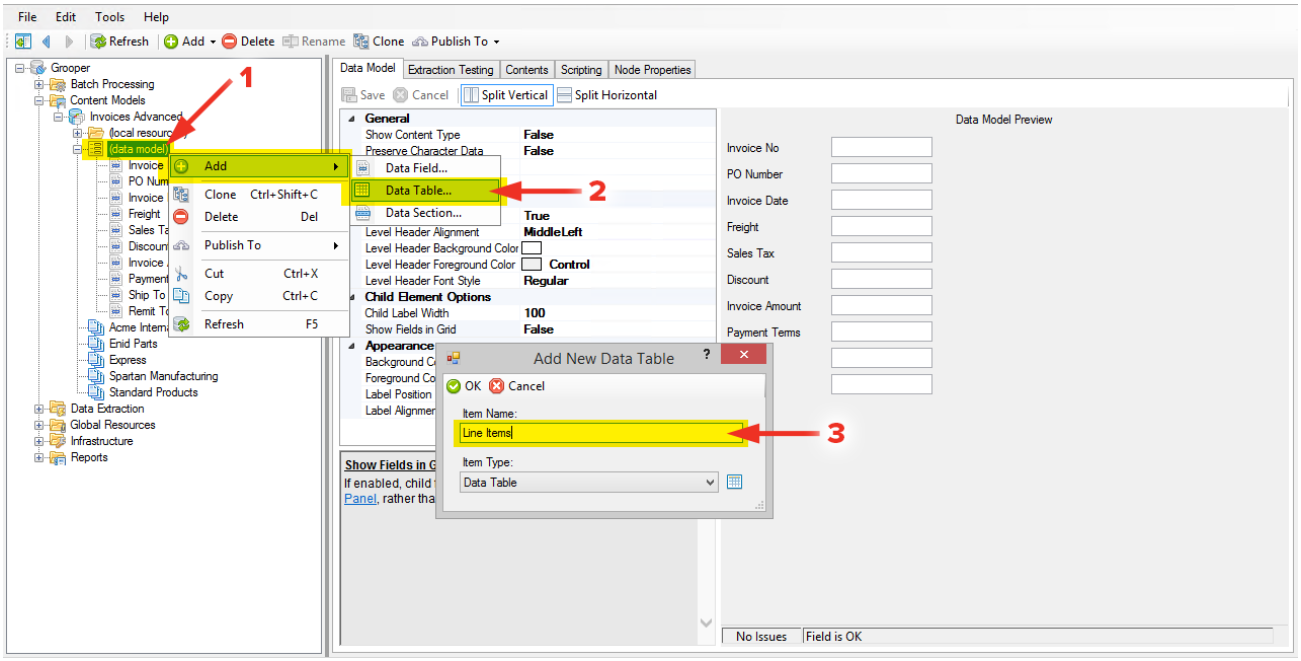


If at any point you want to change the order of (most, not all) objects in the Grooper Node Tree, simply select one, and hold the Ctrl button and use either the ↑ or ↓ arrow keys on the keyboard.

STEP 2 – ADDING NEW DATA ELEMENT – DATA TABLE

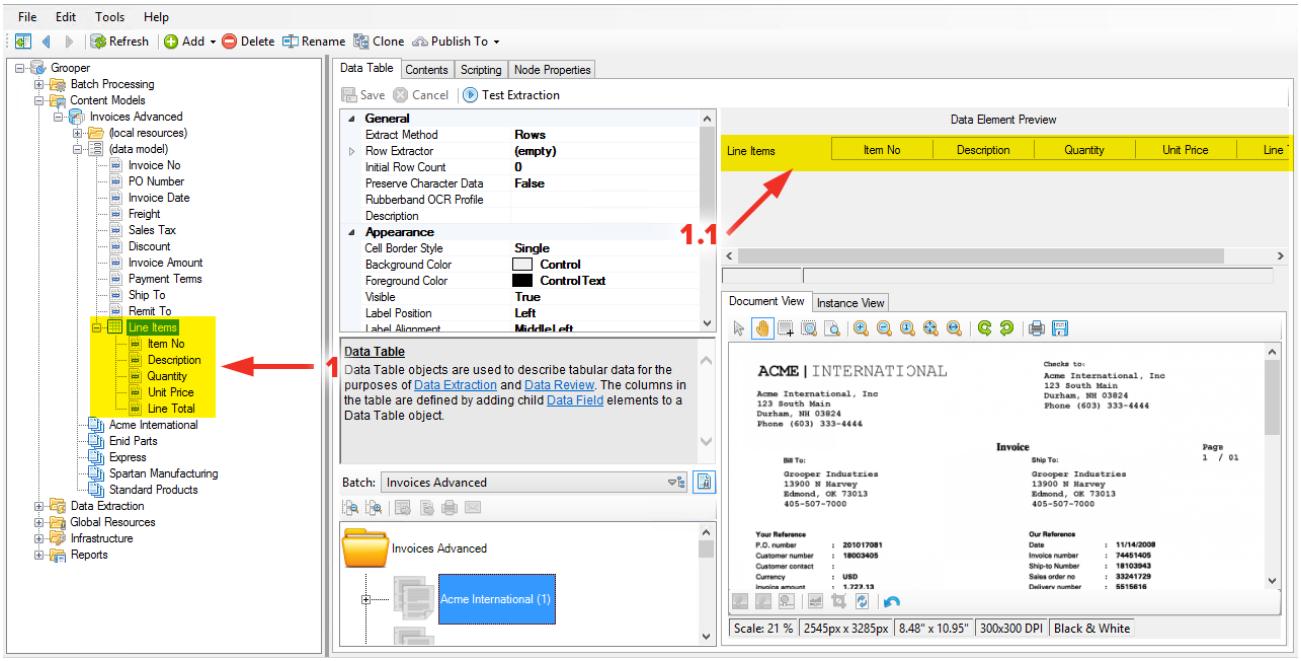
While individual Data Fields are purposed for displaying individual elements of data, there are instances where multiple elements represent one overall idea of information. For this, a Data Table works well. The Data Table is a child to the Data Model, but itself contains Data Fields that will be represented succinctly by their parent Data Table. It also houses some critical control properties that will dictate behaviors of the child Data Fields.

(1) Right Click on the Data Model **(2)** and Add > Data Table... **(3)** Name it Line Items.



STEP 3 – ADDING DATA FIELDS TO THE DATA TABLE

(1) Add each of the following as Data Field objects to the Data Table (pay very close attention to spelling):
Item No • Description • Quantity • Unit Price • Line Total



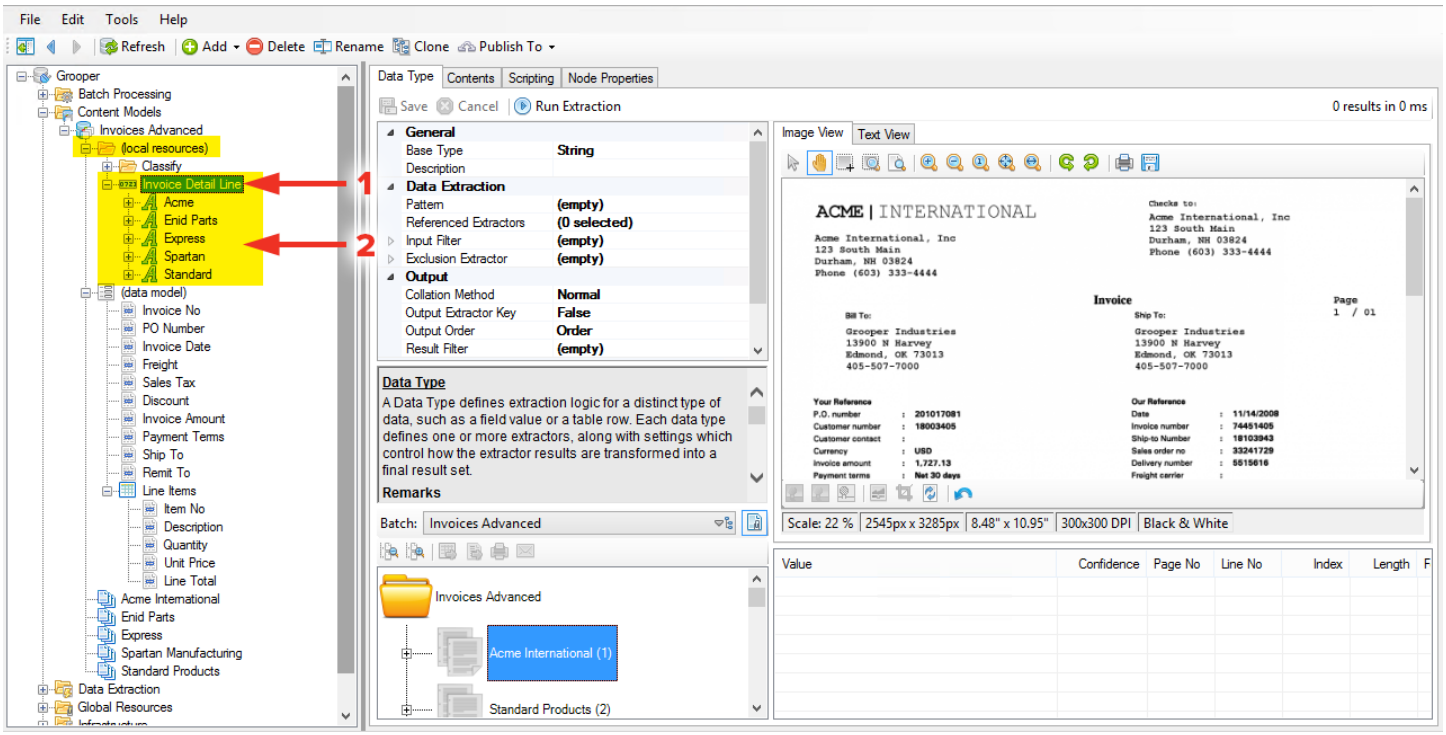
SETTING UP THE TABLE ROW EXTRACTOR

It is possible to create an extractor for each Data Field of the Data Table, however, there is a better approach. When writing a RegEx pattern for an extractor it is possible to label elements of the pattern with groups. Groups in a RegEx pattern are referred to as sub-elements. If the sub-elements are named exactly the same (but underscores instead of spaces) as the Data Field child objects of the Data Table, one Data Format can populate all the Data Fields of the Data Table.

STEP 1 – ADD NEW DATA TYPE AND CHILD DATA FORMATS

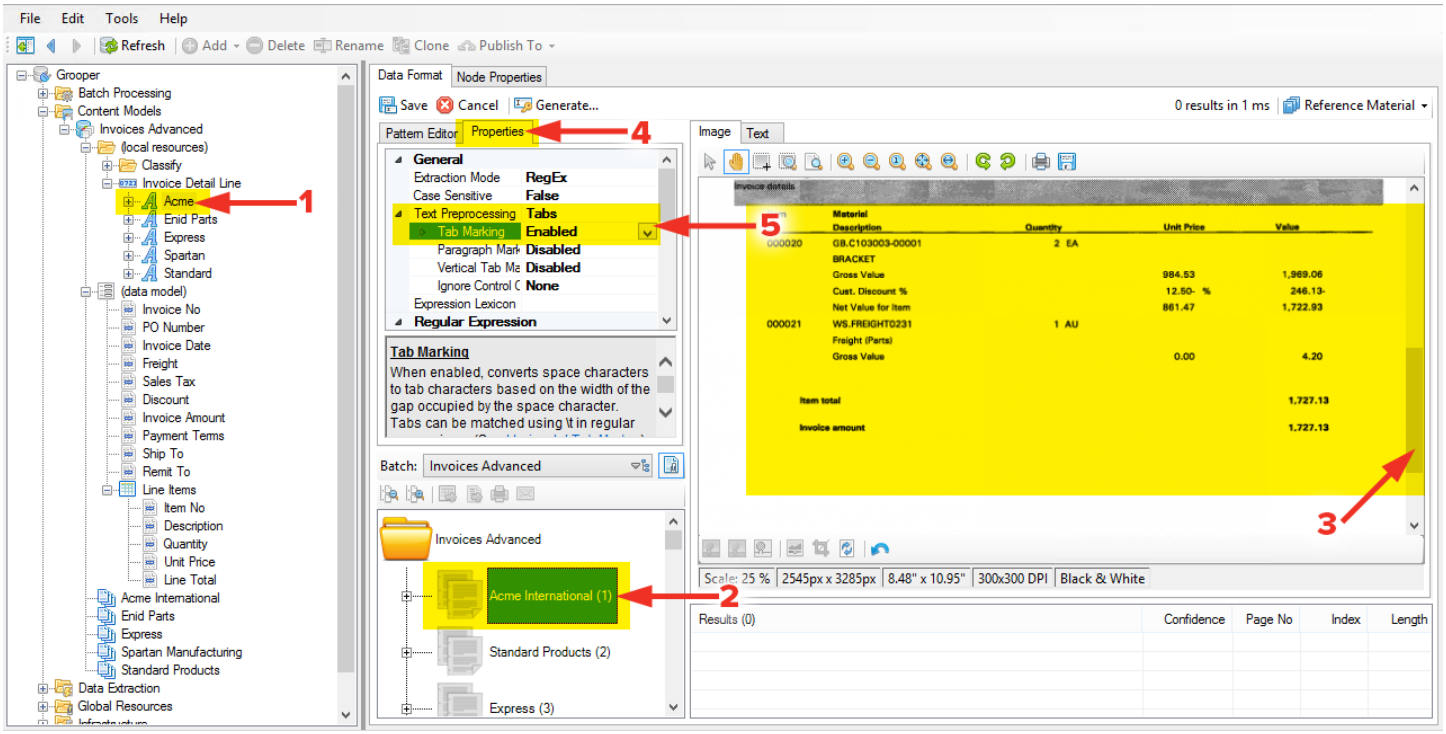
One Data Type will be passed as the Row Extractor of the Data Table, but it will need different formats to handle each of the different types of invoices. These different formats will be found with different Data Format objects.

(1) Add a Data Type to the (local resources) folder and name it Invoice Detail Line. **(2)** Add children Data Format objects to the Invoice Detail Line Data Type naming them after the different invoice providers: Acme, Enid Parts, Express, Spartan, Standard (the names of the Data Formats don't have to be named exactly as the DocTypes they're for.)



STEP 2 – ACME PATTERN 01

(1) Select the Acme Data Format and **(2)** make sure the Acme International (1) document is selected in the Batch Viewer. **(3)** In the Page Viewer, scroll the page down to view the tabular information on the document. **(4)** Click on the Properties Tab and expand Text Preprocessing, then **(5)** set Tab Marking to Enabled.



STEP 3 – ACME PATTERN 02

Back in the Pattern Editor tab, type the following into the Value Pattern area:

```
[@Number]{6}\t(?!WS.FREIGHT)
(?<Item_No>[^\t]*)\t
```

This is only the beginning of the pattern, but it's a good place to pause for a moment to get a better understanding of what it's doing.

• **[@Number]{6}** – The square brackets **[ ]** are defining a character set and the **@Number** is an expression variable built into **Grooper** that defines a range of number characters, but accounts for possible OCR errors. You may notice as you begin to type that **Grooper**'s built in IntelliSense will detect what you are typing. Press Enter to allow the IntelliSense to complete the expression for you. The number in curly bracers **{6}** following the character set is a quantifier that defines how many of what in the character set in a row you're looking for.

• **\t** – The escape character **** followed by the **t** is defining a tab character. If you look at the Text tab you'll see the characters that were OCR'ed, including carriage returns **\r** and line new line feeds **\n**. Tab marking is off by default, and simple spaces are instead read. When tab marking is enabled (feel free to go back to the properties tab and look at the settings for tab marking, as there is more to it than simply enabling it) spaces between OCRed characters that are larger than a defined threshold (based on font size) are read as tabs.

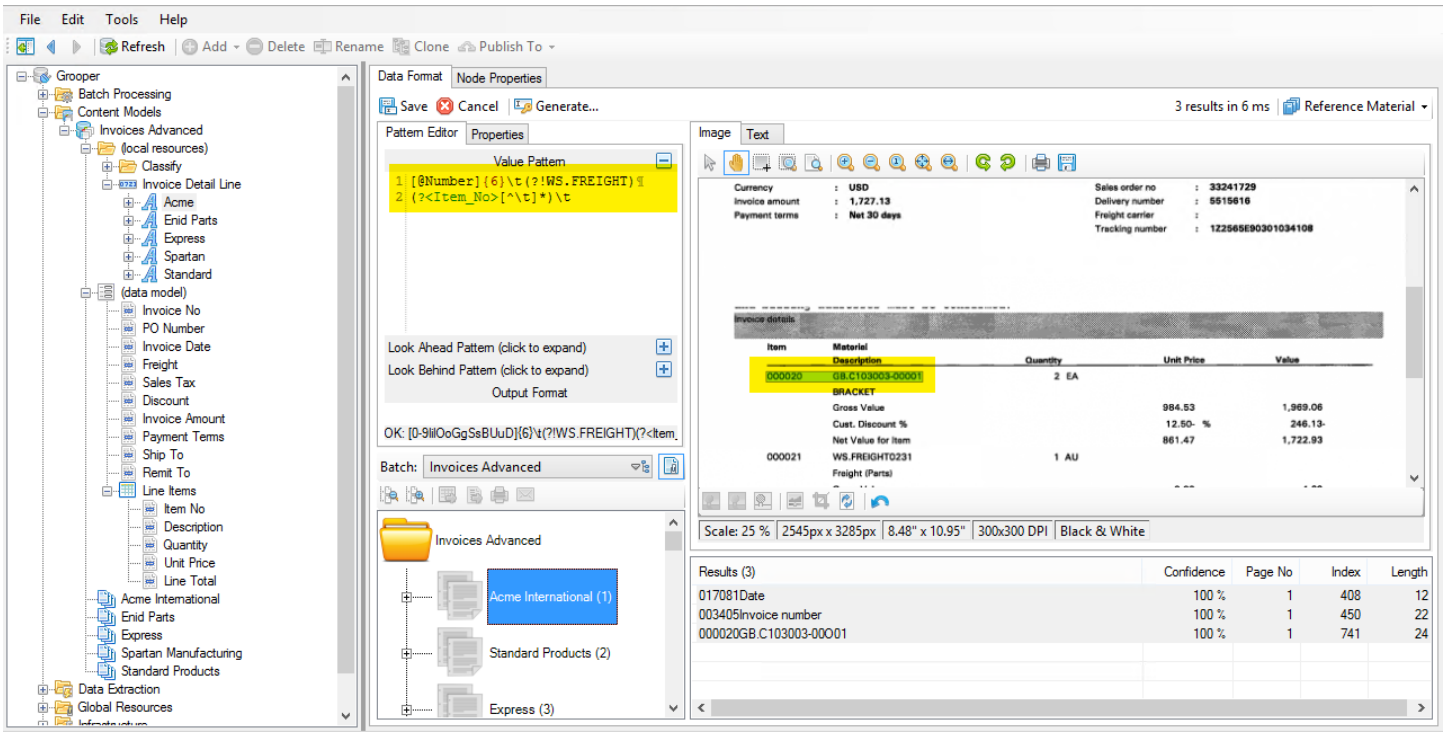
• **(?!WS.FREIGHT)** – This defines what is called a negative look ahead. Everything between the **(?!** and the **)** is considered as being unwanted following the previous pattern. So if any combinations of 6 number characters is followed by the string **WS.FREIGHT**, that string of 6 numbers will be considered invalid.

• The pilcrow ¶ character is not actually typed in this case, but in **Grooper** is automatically inserted when you return the line in your pattern. This allows you to write neat patterns by having them exists on more than one line.

• **`(?<Item_No>[^\t]*)`** – Strings of characters surrounded by parenthesis **`(example)`** are considered to be in a group. When the open parenthesis **`(`** is followed immediately by a question mark **`?`** and then immediately by a less than **`<`** the group is being named. The string of characters following the less than **`<`** are naming the group, up until the greater than **`>`**. Therefore **`(?<Item_No>)`** is a group named **`Item_No`**.

Naming the group isn't useful if you're not defining what the group contains however. Everything between the greater than **`>`** and the close parenthesis **`)`** will be what is contained in the named group. In this case, **`[^\t]*`**. Starting a character set **`[ ]`** with a caret **`^`** tells the system what to NOT get. In this case, any non-tab character. The quantifier **`*`** outside the character set says get zero or more not-tab **`[^\t]`** characters. After the capture group, a tab **`\t`** character is captured to continue the pattern.

• To recap: 6 number characters **`[@Number]{6}`** are followed by a tab **`\t`**, but not where the string **`WS.FREIGHT`** exists **`(?!WS.FREIGHT)`**. A named group called **`Item_No`** contains zero or more non-tab characters **`(?<Item_No>[^\t]*)`** followed by a tab character **`\t`**.



STEP 04 - ACME PATTERN 03

Return the line and enter the following pattern:

```
(?<Quantity>[@Number]{1,3})\s
```

This creates another named group called **Quantity** that looks for 1 to 3 number characters followed by a space.

STEP 05 – ACME PATTERN 04

Return the line and enter the following pattern:

```
[A-Z]{2}\r\n
```

This will capture any letter in the alphabet twice, followed by a carriage return and newline feed.

STEP 06 – ACME PATTERN 05

Return the line and enter the following pattern:

```
(?<Description>[^\r]*)\r\n
```

This will create another named group called **Description** that will capture all non-carriage return characters, followed by a carriage return, and a new line feed.

STEP 07 – ACME PATTERN 06

Return the line and enter the following pattern:

```
([^\t]*\t)?
```

This creates a non-named group that will capture all non-tab characters, followed by a tab. The question mark quantifier (zero or one) after the group will make it optional.

STEP 08 – ACME PATTERN 07

Return the line and enter the following pattern:

```
(?<Unit_Price>[@Number.,]*)\t
```

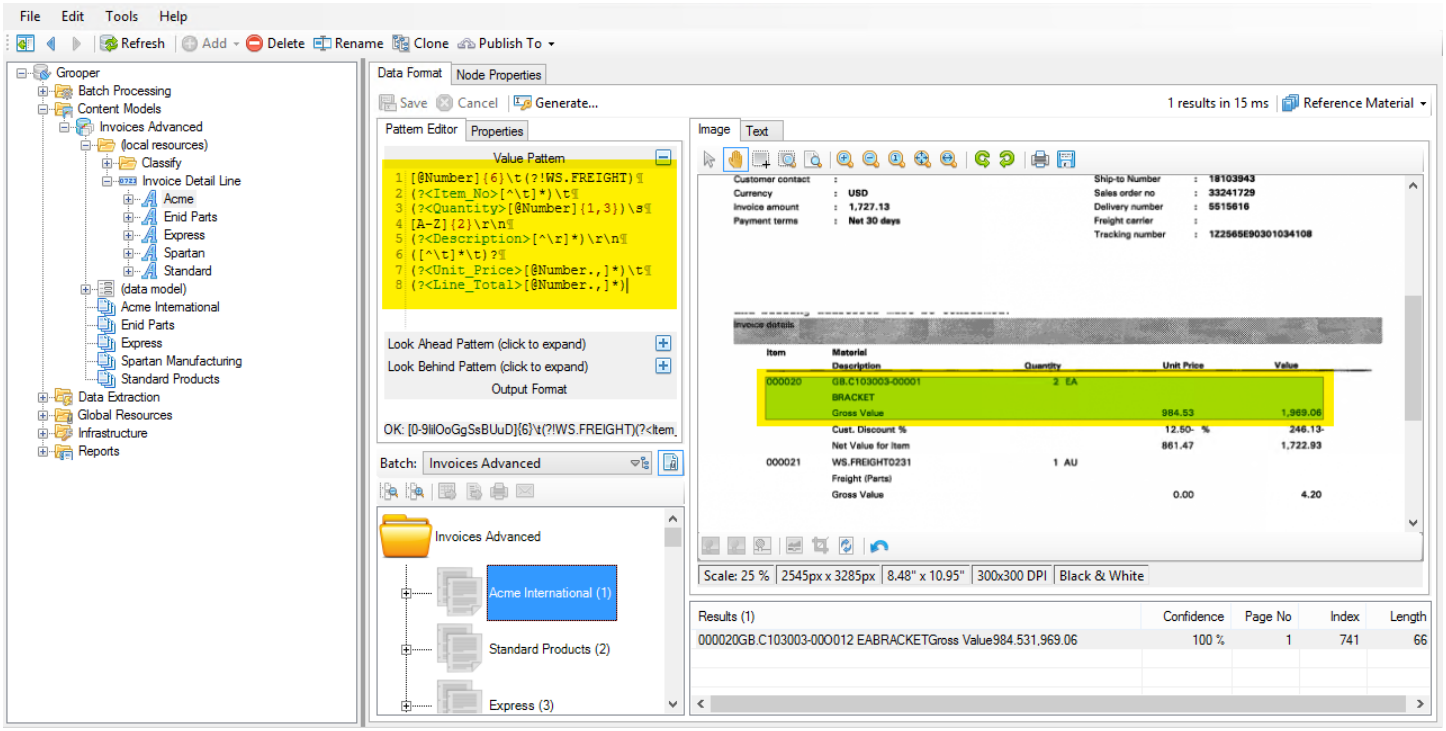
This creates a group named **Unit_Price** that will capture zero or more number characters, periods, and commas, followed by a tab.

STEP 09 – ACME PATTERN 08

Return the line and enter the following pattern:

```
(?<Line_Total>[@Number.,]*)
```

This will create the final named group for this pattern called Line_Total. It will capture zero or more number characters, periods, and commas, then stop.



STEP 10 – REMAINING PATTERNS

The remaining Data Formats that are child objects of the Invoice Detail Line Data Type will all use tab marking and have the following patterns:

Enid Parts:

```
(?!020-0027)
(?<Item_No>[A-Z0-9.\-]+)\t
([A-Z]{1,2}\t)?
(?<Description>[^\r]+)\r\n
[@Number.,]+\t
(?<Quantity>[@Number.,\s]+)\t
[A-Z]{1,4}\t
(?<Unit_Price>[@Number.,]+)\s
[A-Z\s/]+\t
(?<Line_Total>[@Number.]+)
```

For this pattern use a look ahead of:
\n

Express:

```
[0-9]{1,3}[\s\-]
(?<Item_No>[0-9]{4}[0-9A-Z]{3,4})\t
(?<Description>[^\t]+)\t[0-9]{1,4}\t
(?<Quantity>[0-9]{1,4})\t[0-9]{1,4}\t
(?<Unit_Price>[0-9.,]+)\t
(?<Line_Total>[0-9.,]+)
```

Spartan:

```
\d{5}\s[A-Z]\s
(?<Item_No>\d{5})\t
\d{1,4}\t\d{1,4}\t
(?<Quantity>\d{1,4})\t
(?<Unit_Price>[\d.,]+)\t
([A-Z]{1,5}\t)?
(?<Line_Total>[\d.,]+)
[^\n]+\n
(?<Description>[^\t\r]+)
```

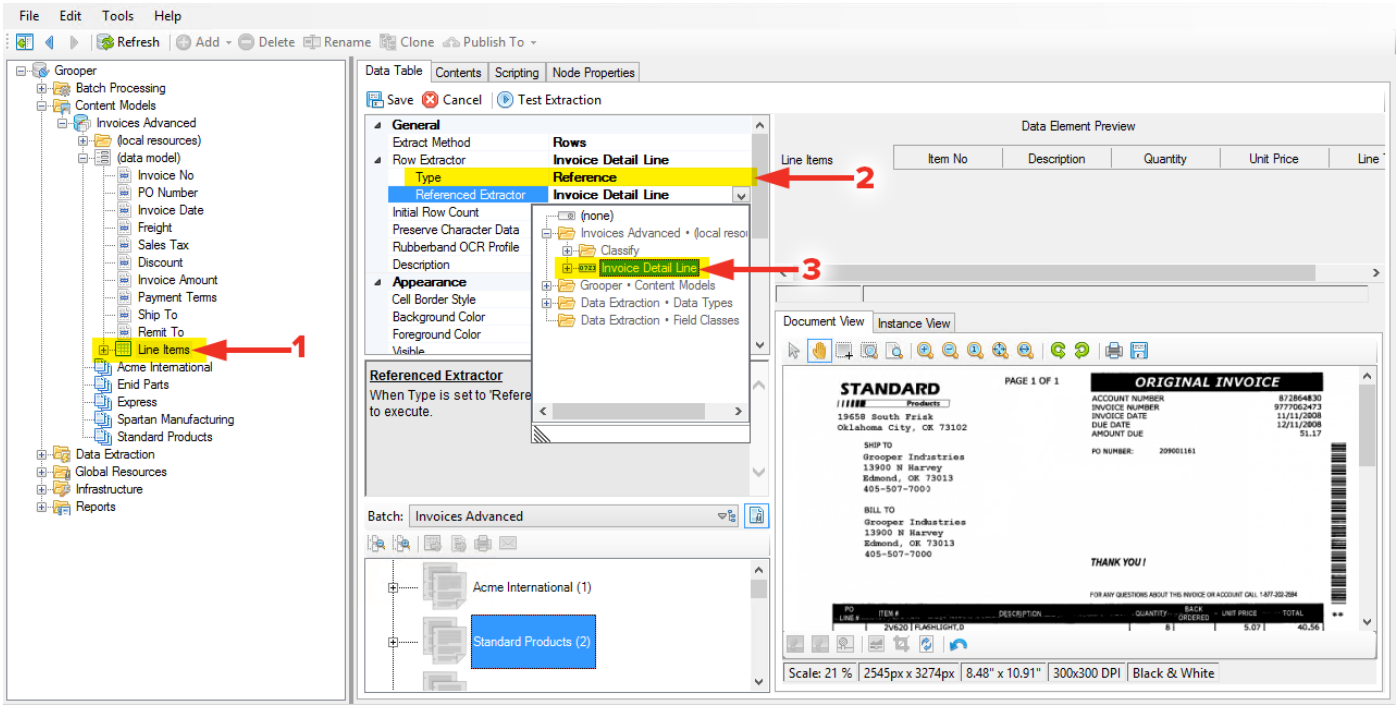
Standard:

```
([0-9]{6}\s)?
(?<Item_No>[0-9A-Z]{5})\s
(?<Description>[^\t\r]*)\t
(?<Quantity>[0-9]{1,5})\s
(?<Unit_Price>[0-9.,]+)\s
(?<Line_Total>[0-9.,]+)
```

For this pattern use a look ahead of:
\n
And a look behind of:
\r

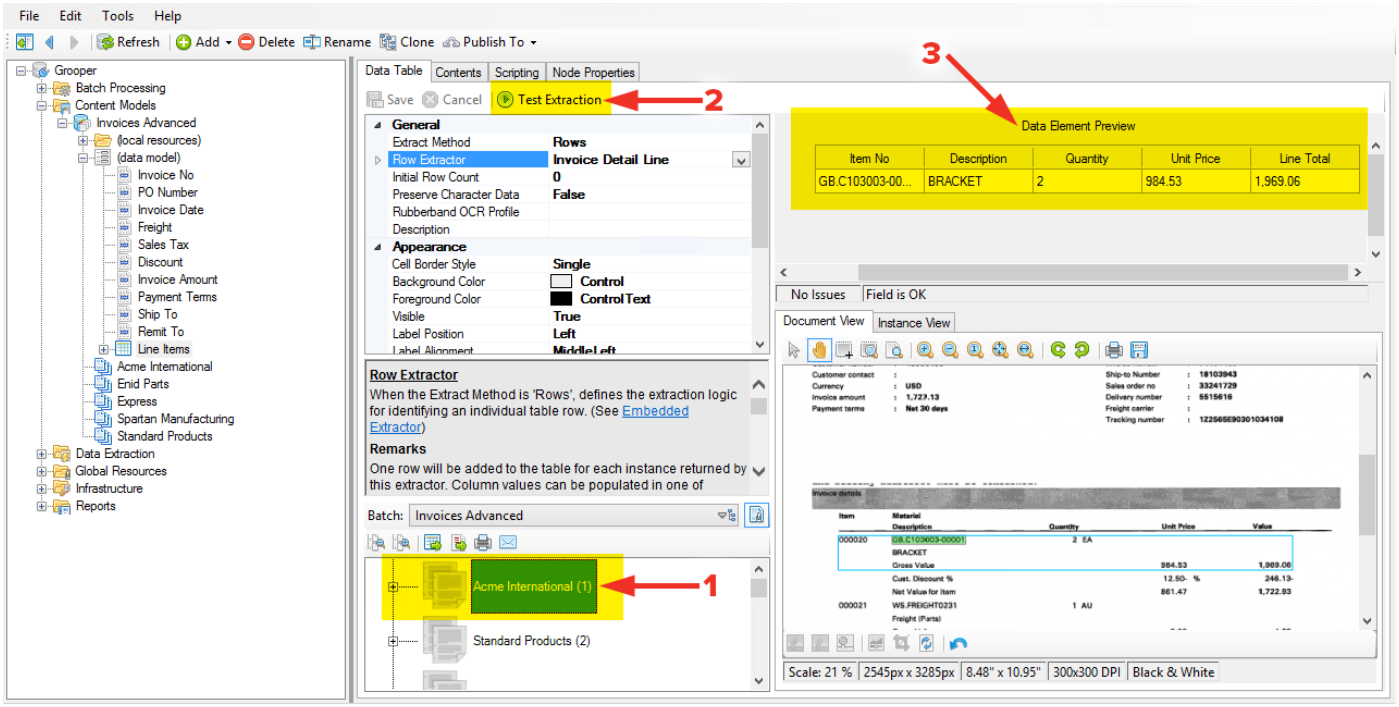
STEP 11 – SETTING THE ROW EXTRACTOR

With the Data Formats for our Invoice Detail Line Data Type complete, it's time to reference this Data Type within the Data Table. Expand the (data model) and **(1)** select the Line Items Data Table. **(2)** Set the Row Extractor as a Type > Reference and **(3)** select the Invoice Detail Line as the Data Type for the Referenced Extractor.



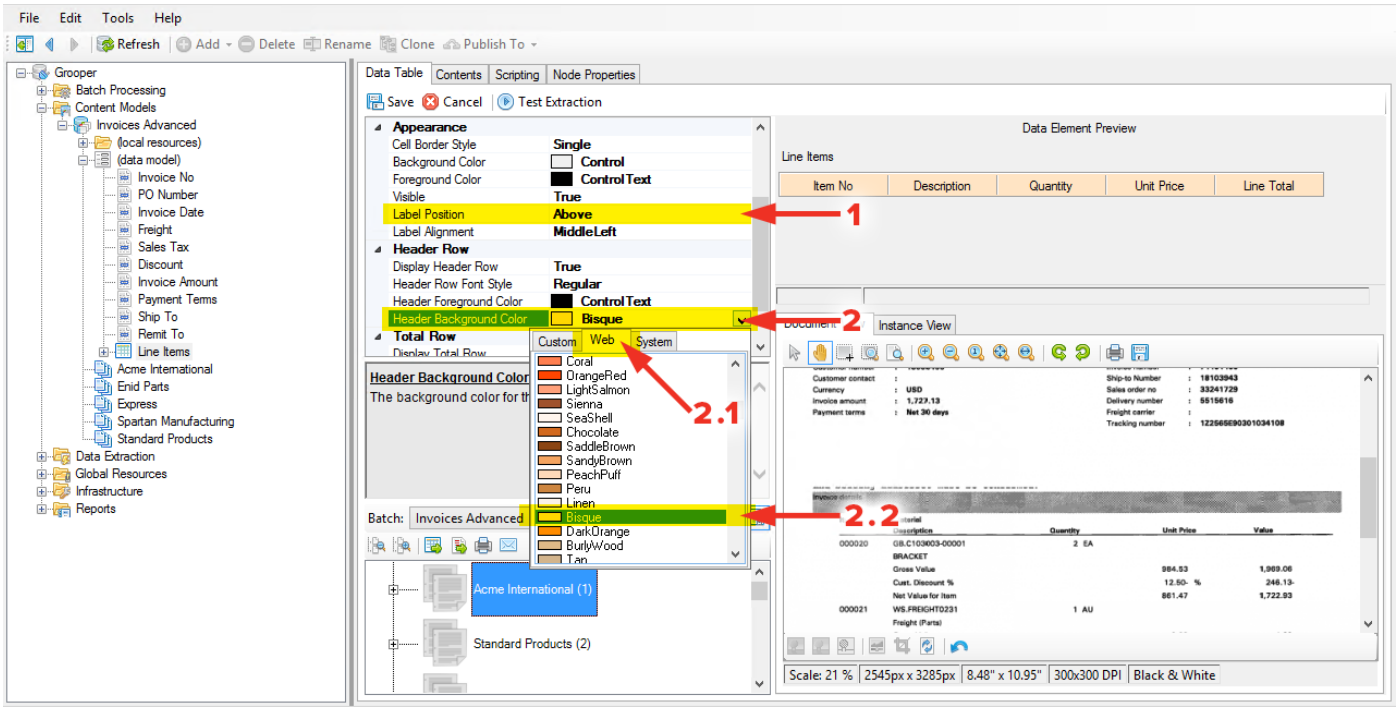
STEP 12 – TESTING EXTRACTION

(1) Select any document from the Batch Viewer, and **(2)** click the Test Extraction button. **(3)** In the Data Element Preview, the results of the test extraction should be visible and displayed in a table.



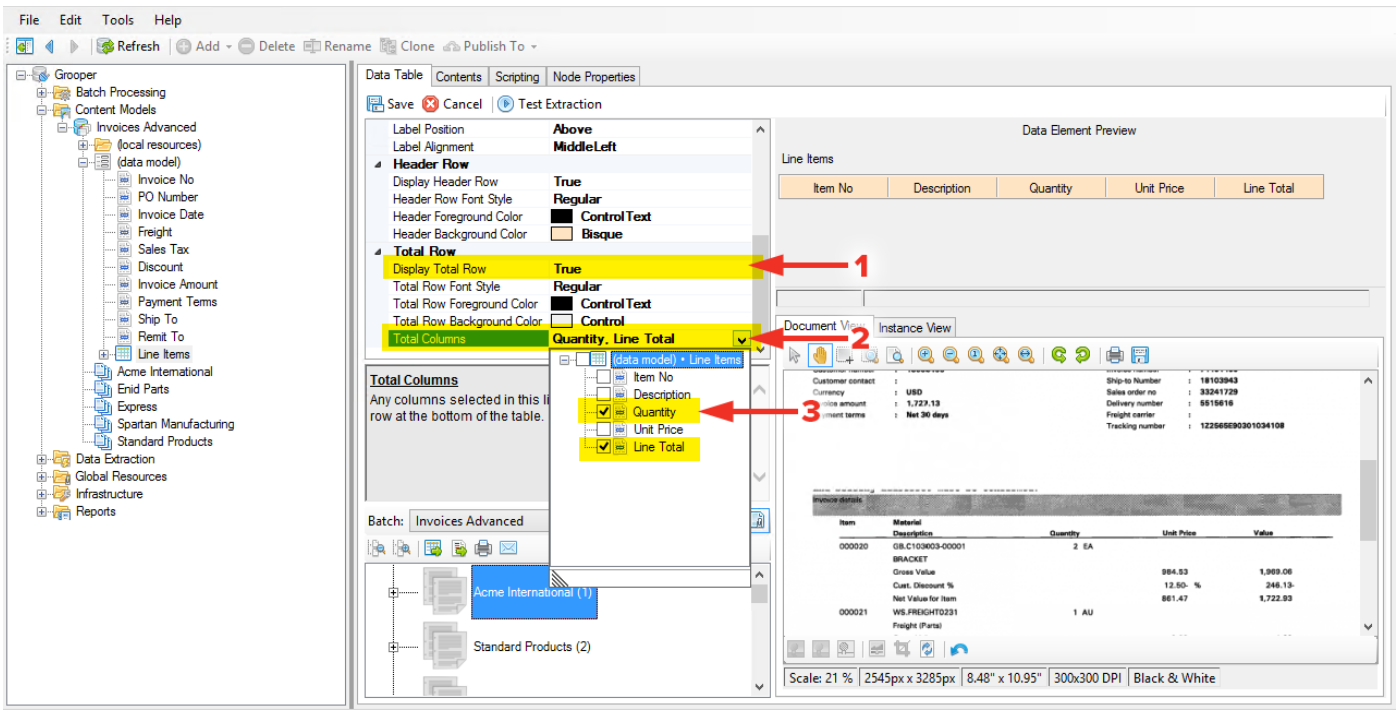
STEP 13 – ADJUSTING PROPERTIES OF THE DATA TABLE

in the Appearance properties section, **(1)** set the Label Position to Above. In the Header Row section, **(2)** set Header Background Color to Bisque.



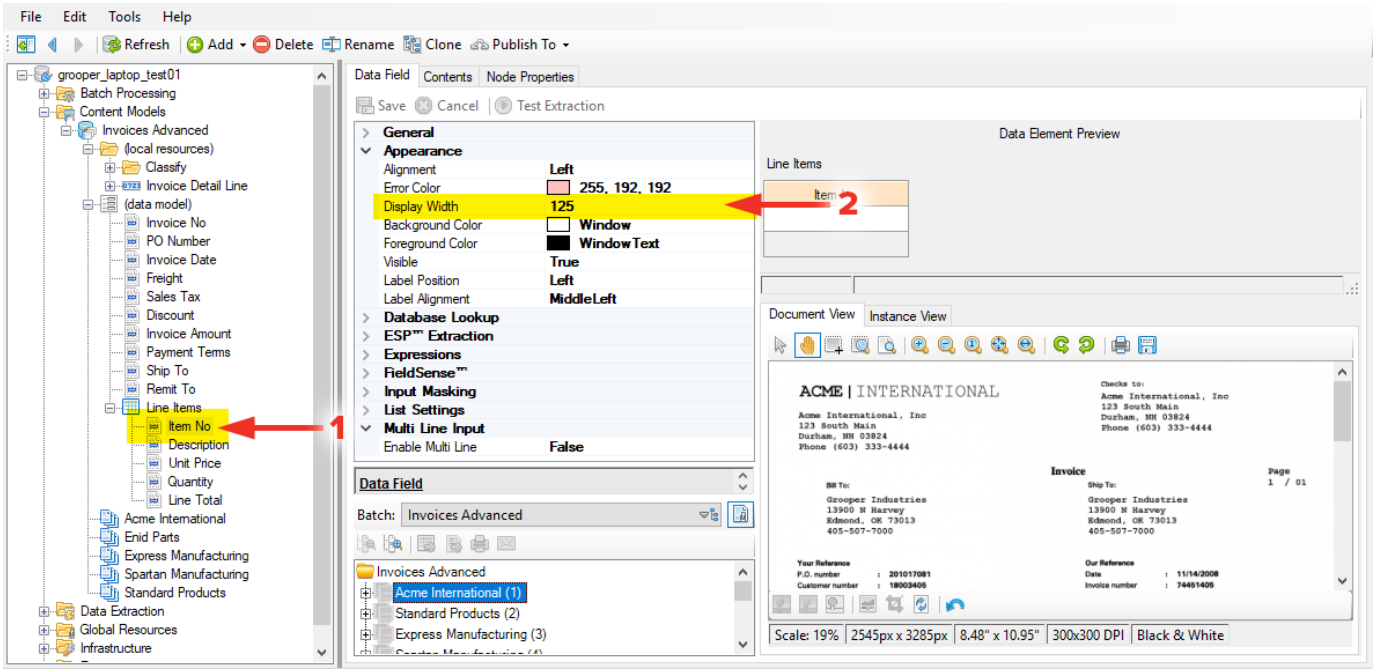
STEP 14 – ENABLE TOTAL ROW

The final section of the Data Table, Total Row, houses settings for a dynamically generated row that will create sum totals of selected columns. **(1)** Set Display Total Row to True, and **(2)** select Quantity and Line Total as the Total Columns. Feel free to Test Extraction again to observe the changes made.



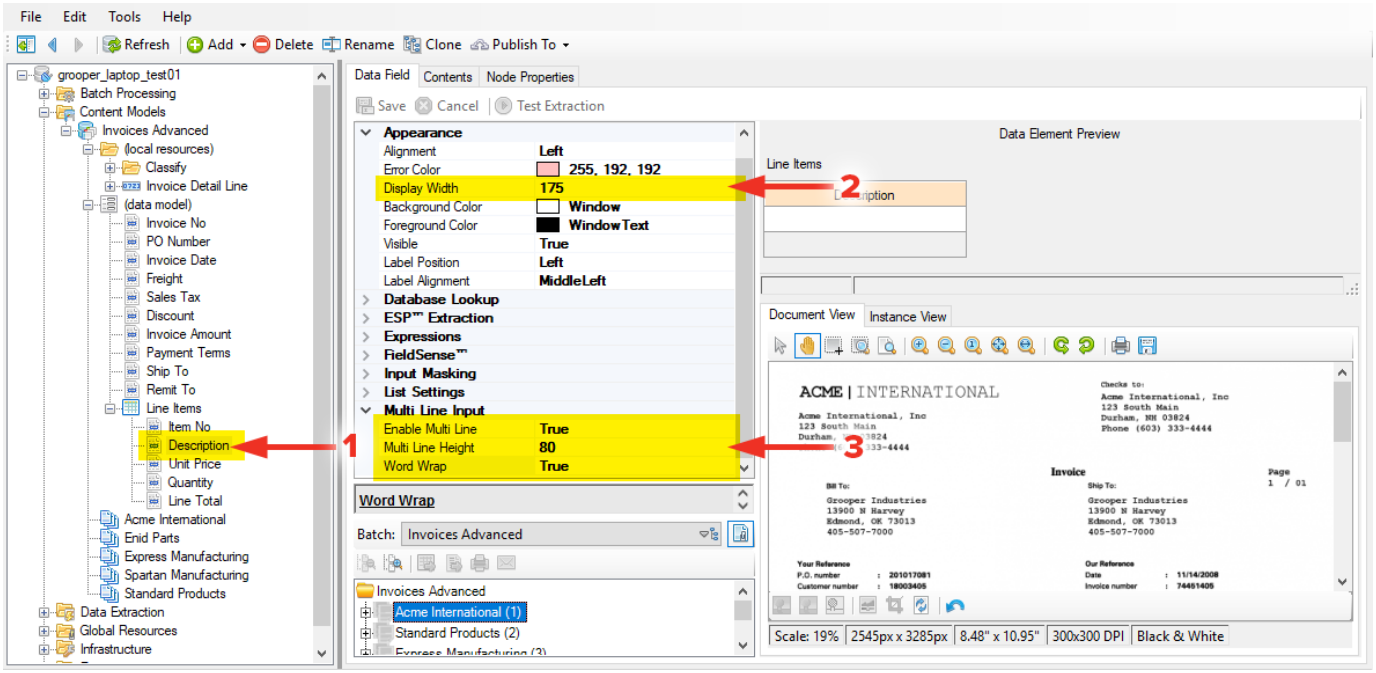
STEP 15 – TABLE APPEARANCE ADJUSTMENTS

The fields of the table are uniform in their appearance, without respect to what they're capturing, so some information has too much room, while others (like Description) don't have enough. **(1)** Click the Item No Data Field and **(2)** set its Display Width to 125. Do the same for Unit Price, Quantity, and Line Total with values of 75, 100, and 85 respectively.



STEP 16 – APPEARANCE ADJUSTMENTS CONTINUED

(1) Select the Description Data Field and **(2)** set its Display Width to 175. **(3)** Set Enable Multi Line Height to True, Multi Line Height to 80, and Word Wrap to True. With these adjustments made, feel free to Test Extraction on the Line Items Data Table to see the results.



THE GROOPER FIELD CLASS EXTRACTOR

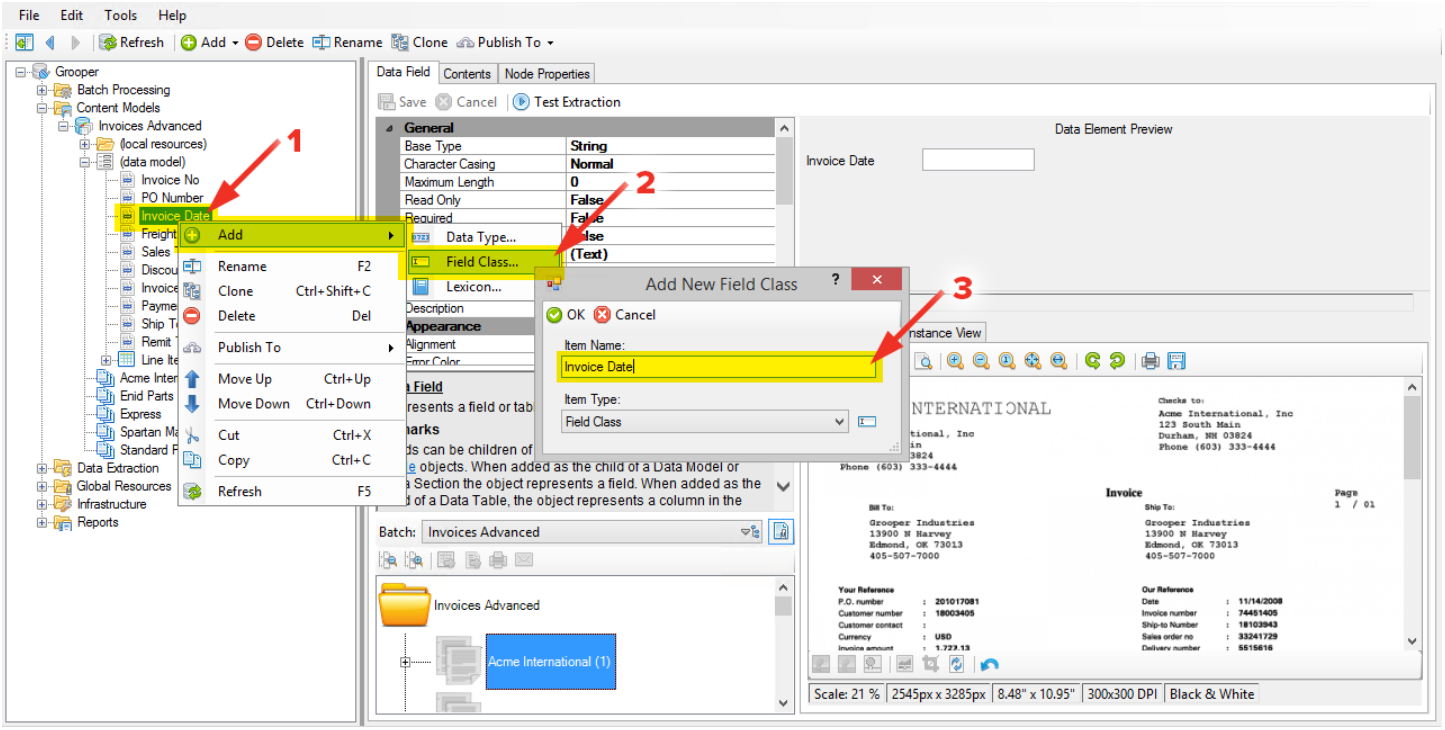
Throughout all the training that's been done so far, only two of the three extractors in **Grooper** have been leveraged: the Data Type and the Data Format. The final extractor to learn about is the Field Class. The Field Class is one of **Grooper**'s most powerful tools and steps up where simple patterns can fail. The reason for this is it is not reliant on uniform structure of a document to perform its extraction. It instead leverages the context around a desired value to understand what is to be extracted. This context is understood because of an end user training the Field Class to understand what features around a desired value it considers positively, or negatively.

SETTING UP A FIELD CLASS FOR THE FIRST TIME

The best way to learn about the Field Class is to use one. This process will begin with the Invoice Date Data Field, and step by step walk through creating (or referencing) all the requisite components to get accurate, consistent extraction.

STEP 1 – ADDING A FIELD CLASS TO THE INVOICE DATE DATA FIELD

(1) Select the Invoice Date Data Field then **(2)** right-click and select Add > Field Class... **(3)** Name it Invoice Date.



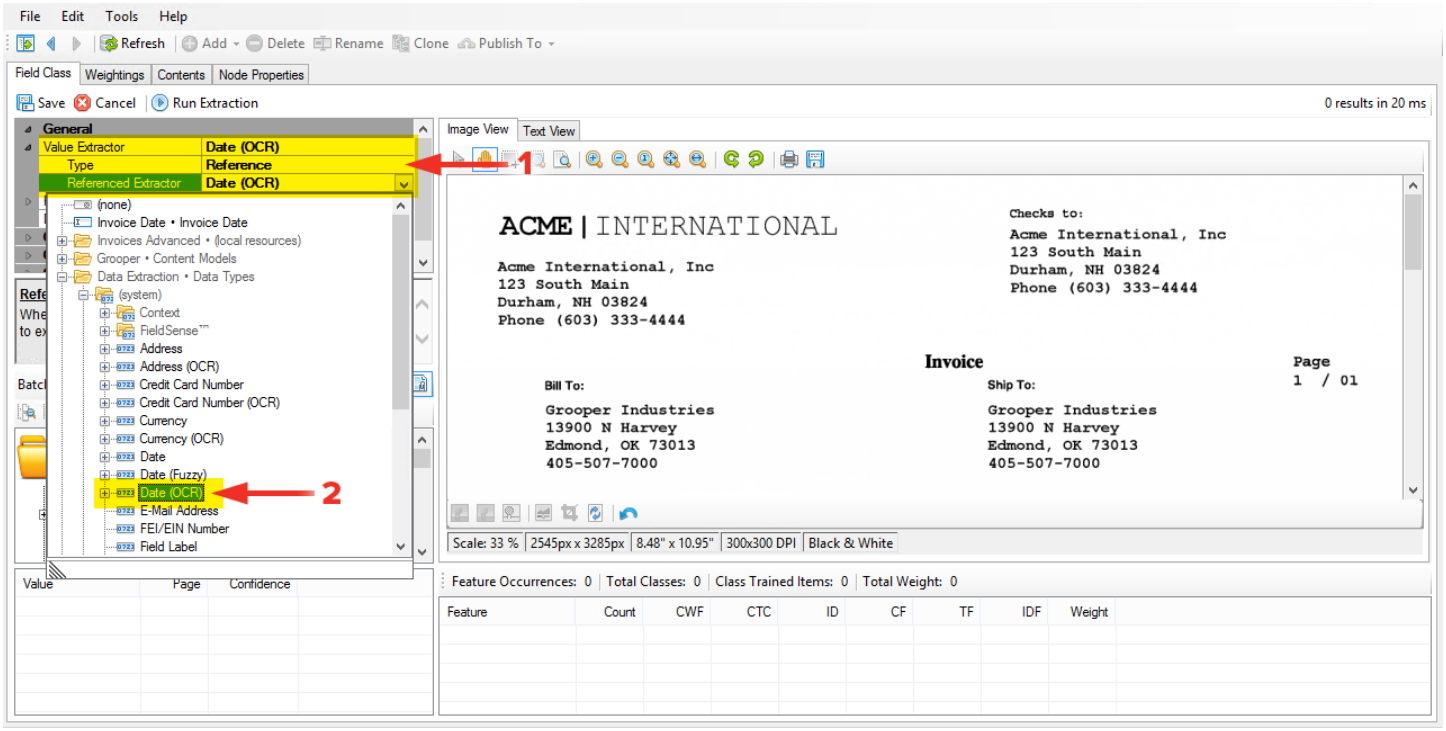
STEP 2 – SETTING THE VALUE EXTRACTOR

The first thing to understand about a Field Class is that it leverages two extractors to accomplish its end goal. To keep things simple in the beginning extractors will be referenced, instead of built from scratch.

The first property to set for a Field Class is the Value Extractor. This is the extractor that will define what information is to be found, and in the case of this Data Field, a date.

(1) Set the Value Extractor Type to Reference, and the Referenced Extractor **(2)** to Data Extraction • Data Types > (system) > Date (OCR).

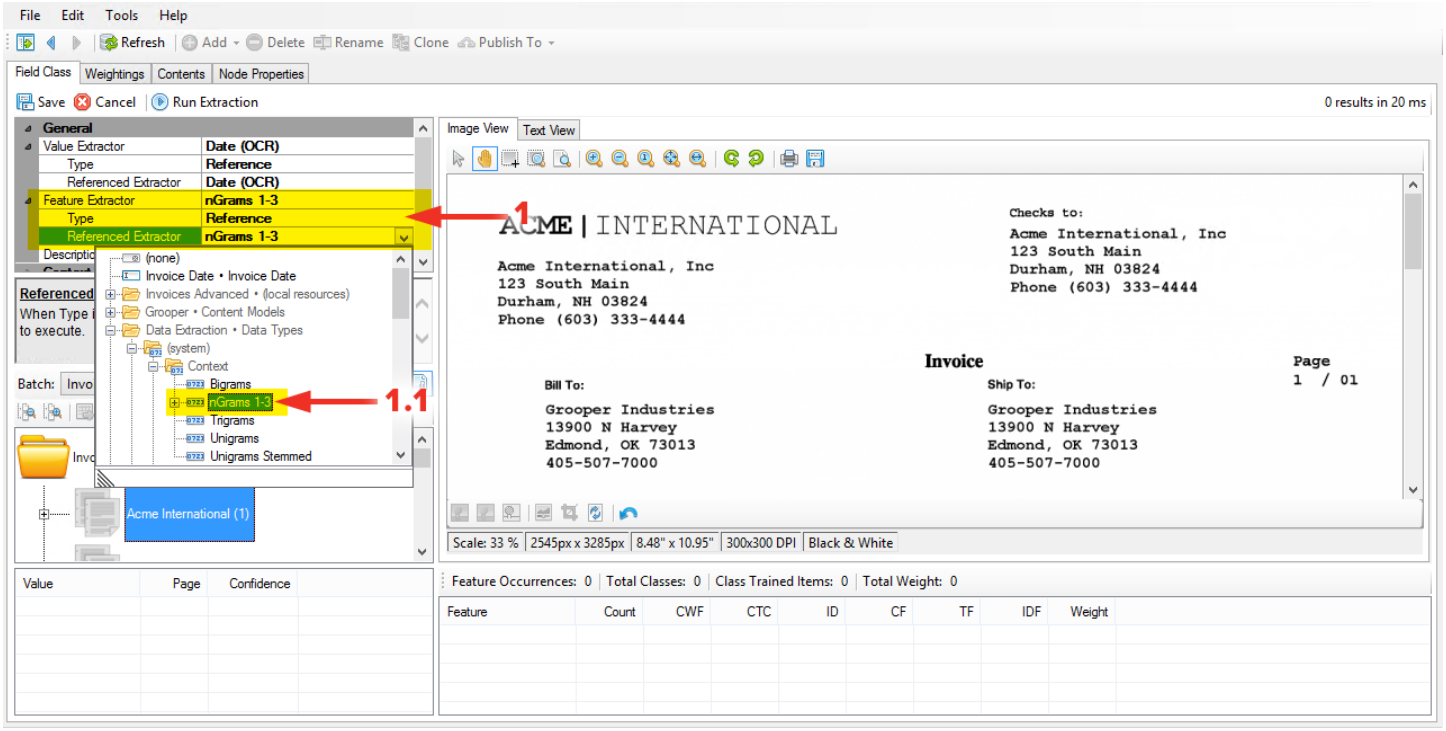
As a side point, feel free to explore this Data Type on your own to discover how it is returning the values it does.



STEP 3 – UNDERSTANDING THE FEATURE EXTRACTOR

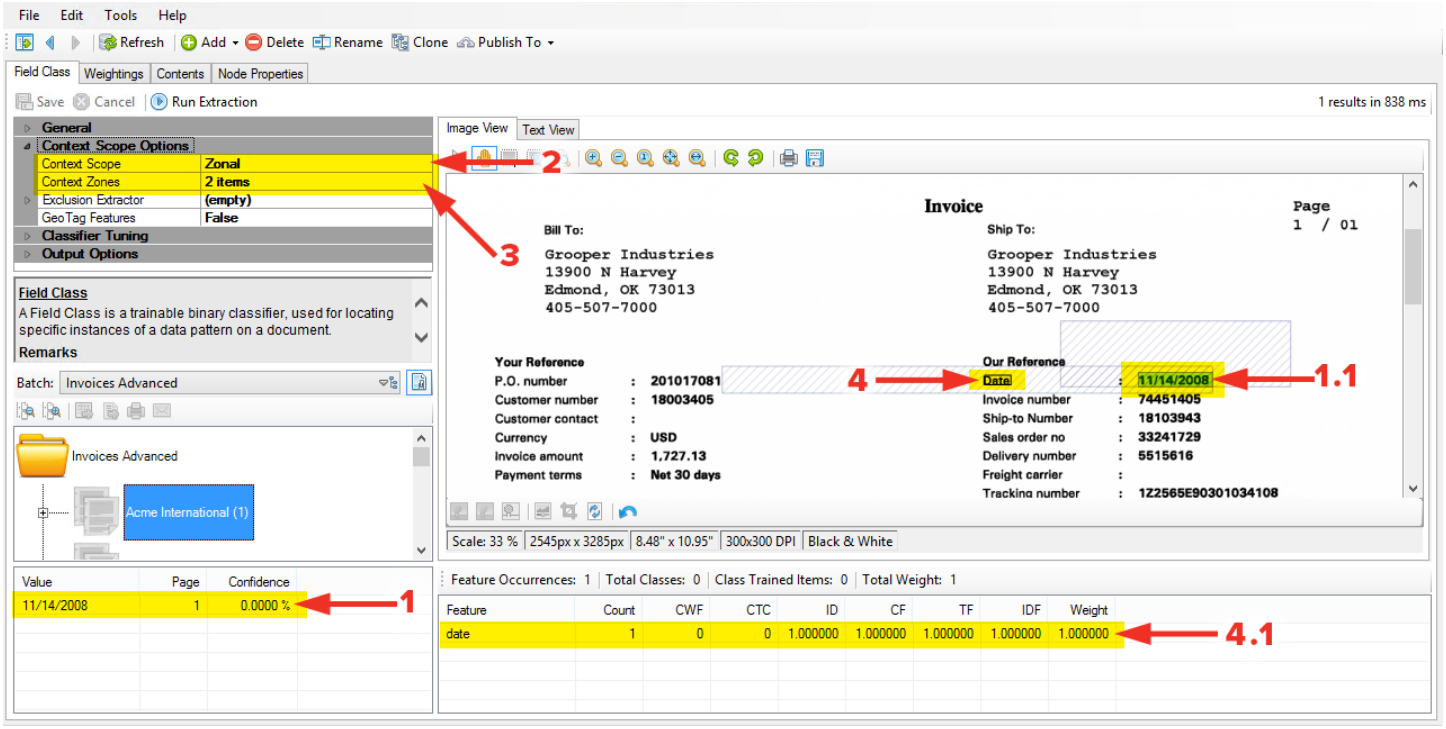
The second extractor that is set for a Field Class defines what around the desired value defines it. If there are multiple date patterns on an invoice, it is clear which date is the desired one due to contextual information surrounding that value.

(1) Field Classes default properties have the system Data Type – nGrams 1-3 set as the Feature Extractor. This will work fine for now. Save the changes made and click the Run Extraction button on the Acme International (1) document.



STEP 4 – UNDERSTANDING CONTEXT SCOPE AND ZONES

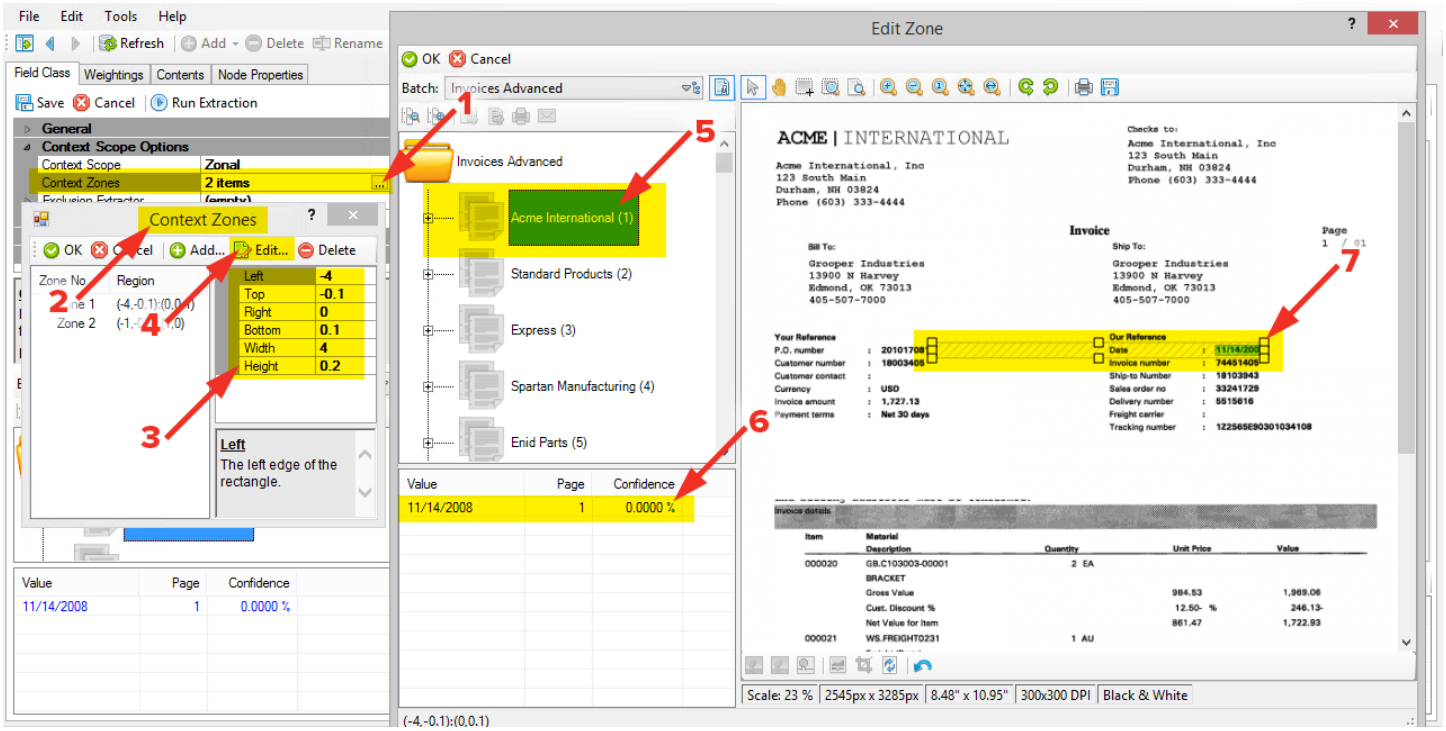
With extraction run there will be new information presented that needs to be explained. **(1)** First off, the Value Extractor will return a value and that will be displayed in the Value List View, as well as draw a green box on the value in the Page Viewer. **(2)** The default Zonal Context Scope will define **(3)** 2 items for the Context Zones which are rectangles drawn around the value. **Grooper** will look within the boundaries of these rectangles to find features returned from the Feature Extractor. **(4)** Features it finds will have a blue box drawn around them.



STEP 5 – EDITING A CONTEXT ZONE

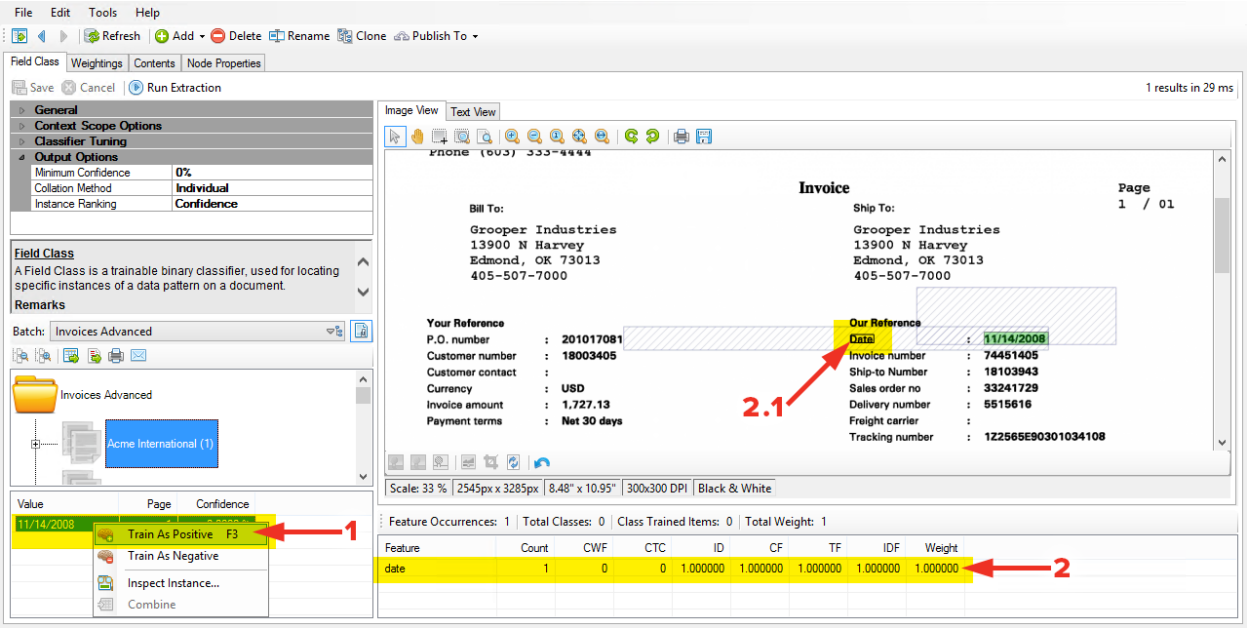
With the Context Zones property selected, **(1)** click the ellipsis button **(2)** to bring up the Context Zones window. **(3)** From here you can select a zone and manually adjust the boundary numeric values, or click **(4)** Edit... which will bring up the Edit Zone window. In this window, **(5)** select a document **(6)** and a value, and **(7)** use the handles of the marquee to adjust the shape.

There is no specific need to adjust the Context Zones for now, so you can leave them defaulted. This was merely an exercise in understanding how one can adjust the Context Zones.



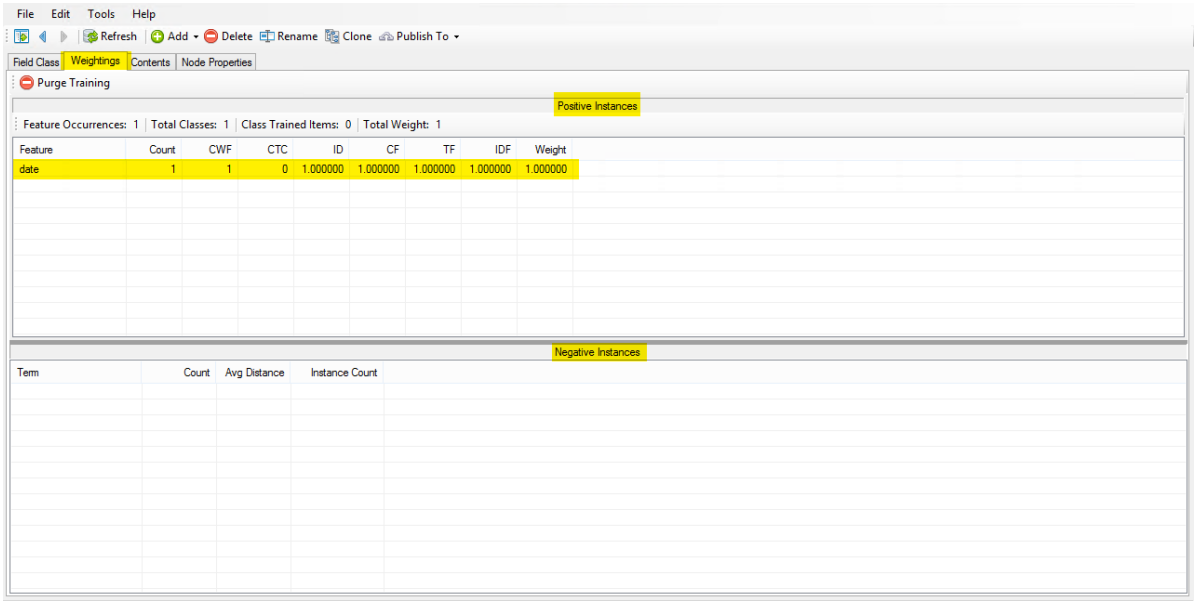
STEP 6 – TRAINING A FEATURE INSTANCE

The Field Class now must be trained as to what feature it will consider positively so it can identify an appropriate value. For the Acme International (1) document, there is only one date value returned, therefore only the feature, date, will be considered. **(1)** In the Value List View, select and right click the only listed value and select Train As Positive. Keep in mind this positive training happened for **(2)** the feature date, not for the value 11/14/2008.



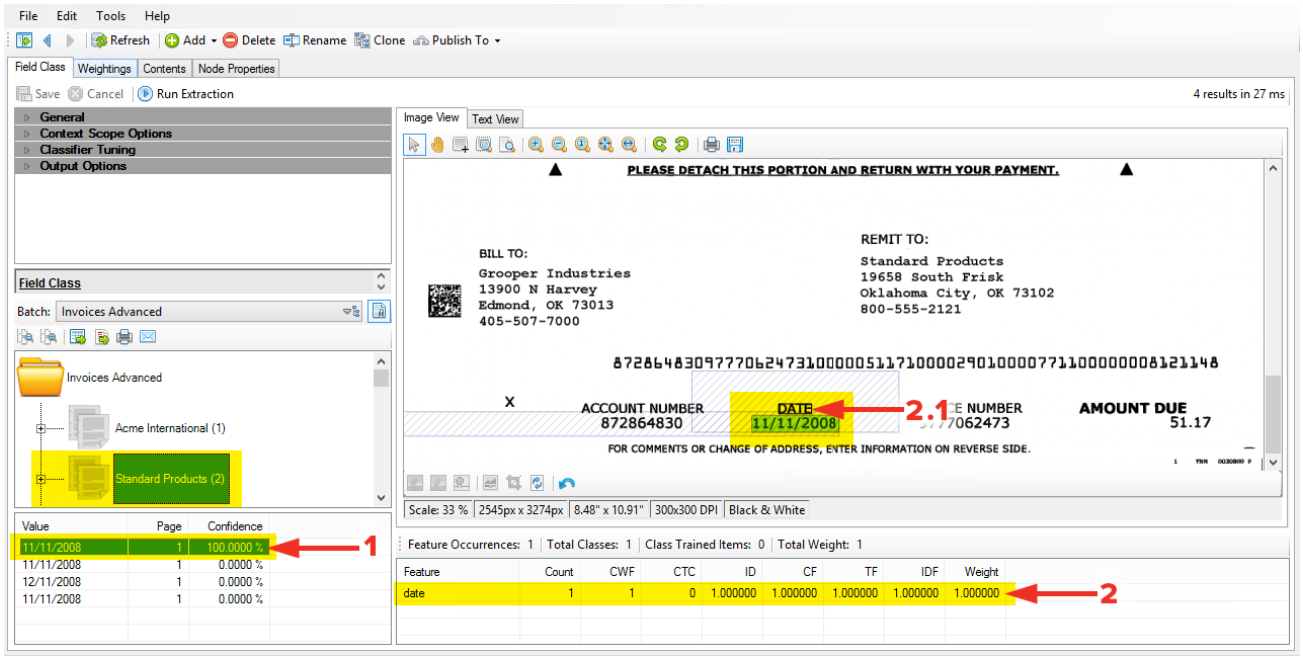
STEP 7 – CHECKING WEIGHTINGS

Click on the Weightings tab. This will display the features that have been trained. When a feature is highlighted in a blue box, and the corresponding value trained against it, the value(s) within boxes will be stored in the Positive Instances. These positive instances will create a weight by which a confidence is built to understand which value to consider on future documents. If other values are present, and as a result, other features highlighted by those values, but those are not selected to be trained positively, those features will be stored as Negative Instances. For the sake of this simple example with the Acme International (1) document, only one word was stored positively.



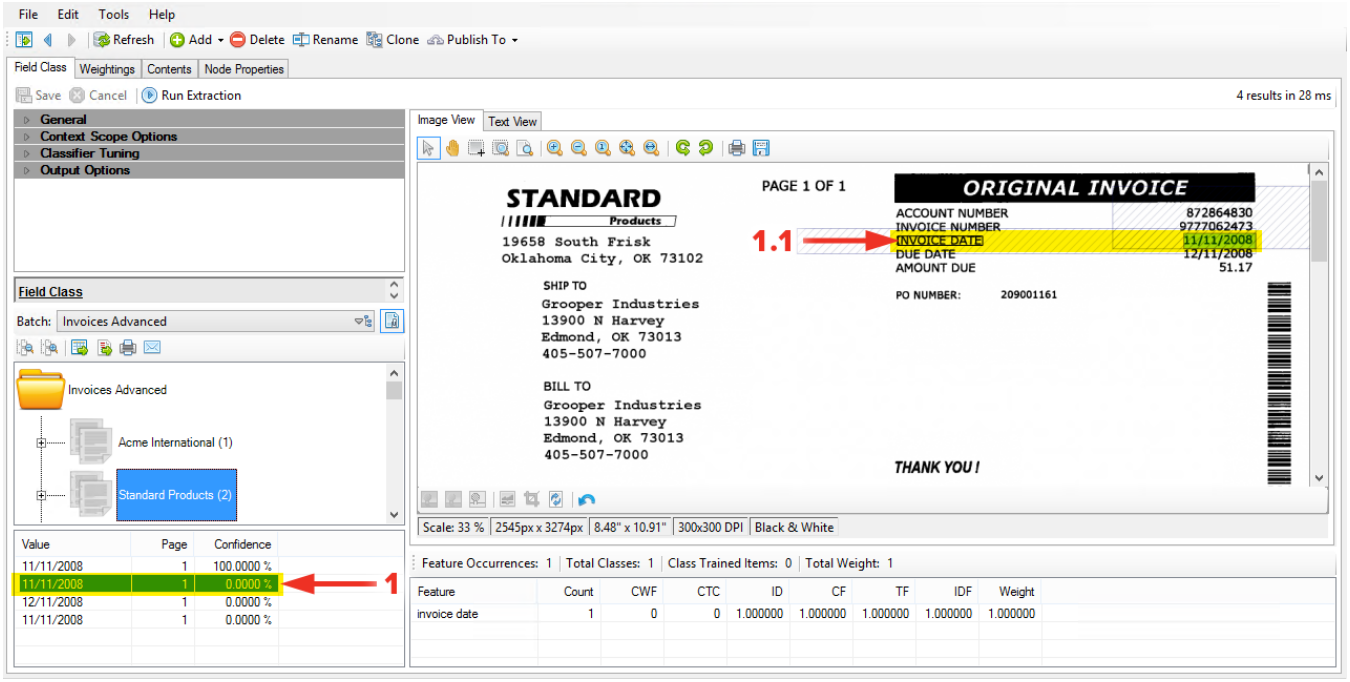
STEP 8 – CHECKING THE NEXT DOCUMENT

Click the Standard Products (2) document in the Batch Viewer. **(1)** Notice that several date values are being returned by the Value Exractor, but only one is listed with confidence (100% in this case.) **(2)** The feature date was found within a context zone around the extracted value, and since this one word has been trained positively, **Grooper** is 100% confident this value is what we want.



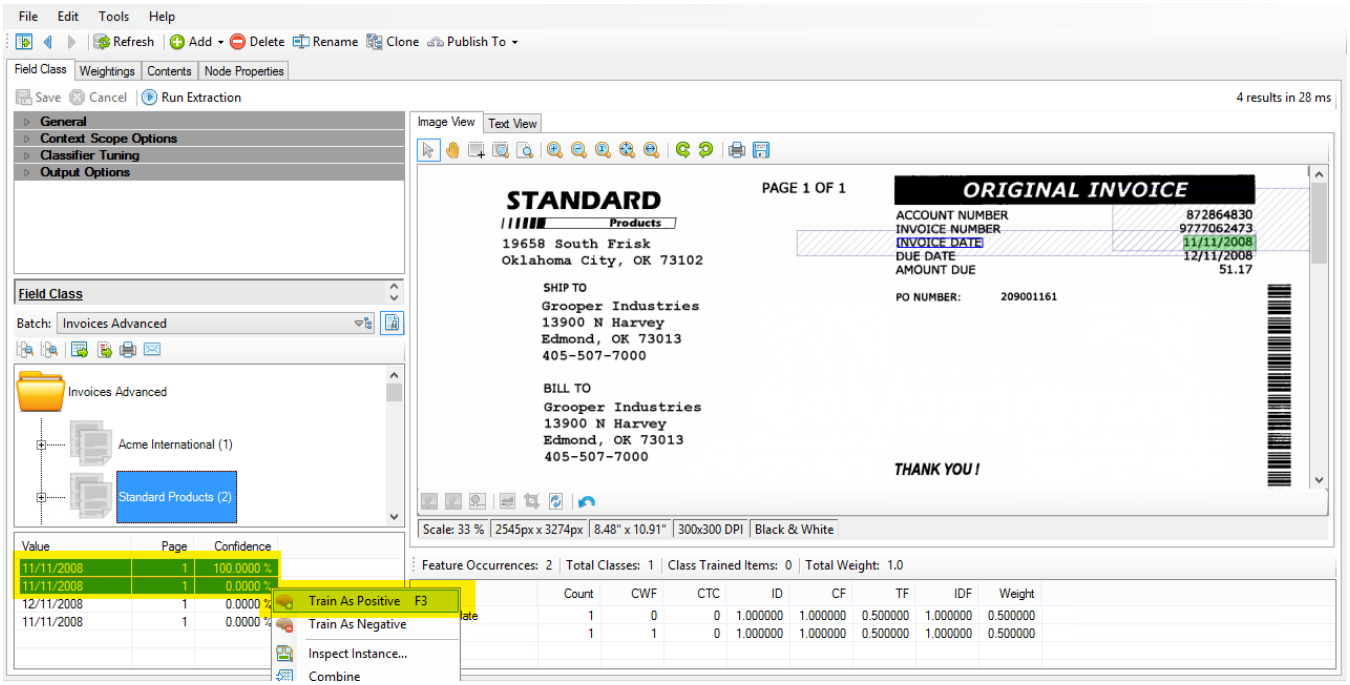
STEP 9 – FINDING ANOTHER CORRECT VALUE

While this value is the desired value, there is another instance of a correct value on the document. **(1)** Select the second 11/11/2008 value, which will be an instance at the top of the document. This is also a value we want, but the feature (in this case a two-word phrase) invoice date hasn't been trained positively.



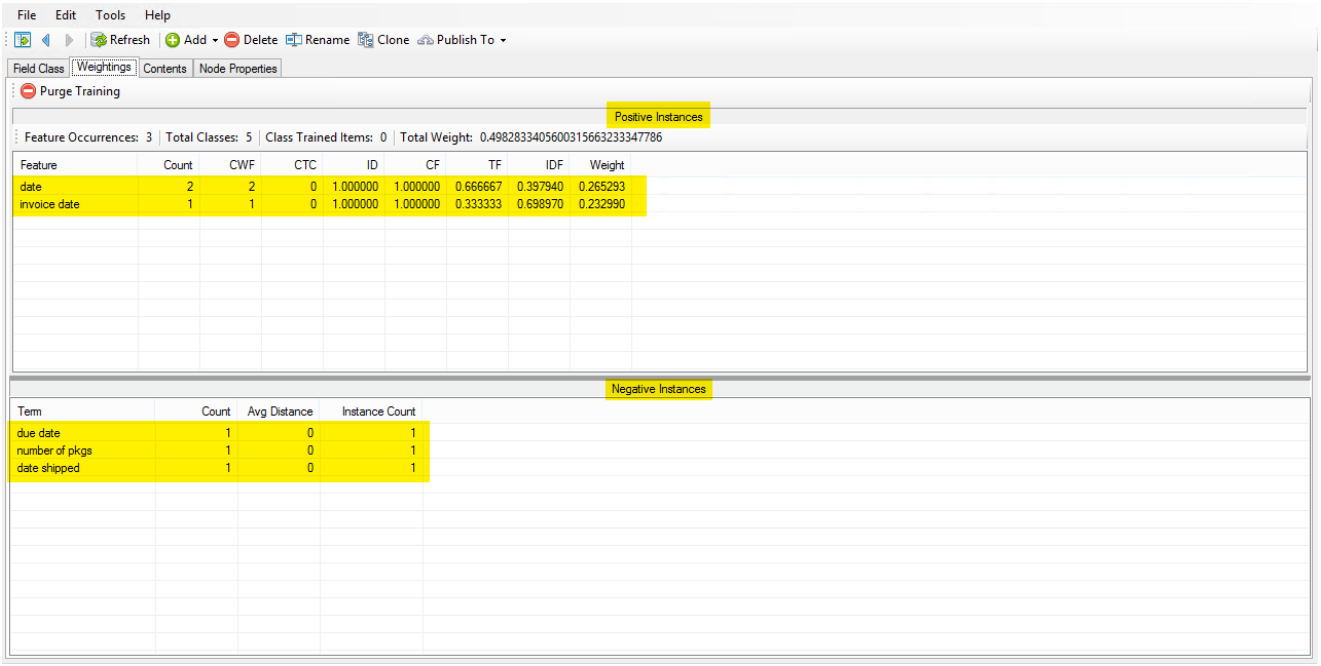
STEP 10 – TRAINING TWO FEATURE INSTANCES

Both features, date and invoice date need to be positively reinforced, so both should be selected and trained positively. Keep in mind, features found on the document but not selected at the time of positive training will be considered negatively, and at this point, training the feature date negatively would throw off our weightings.



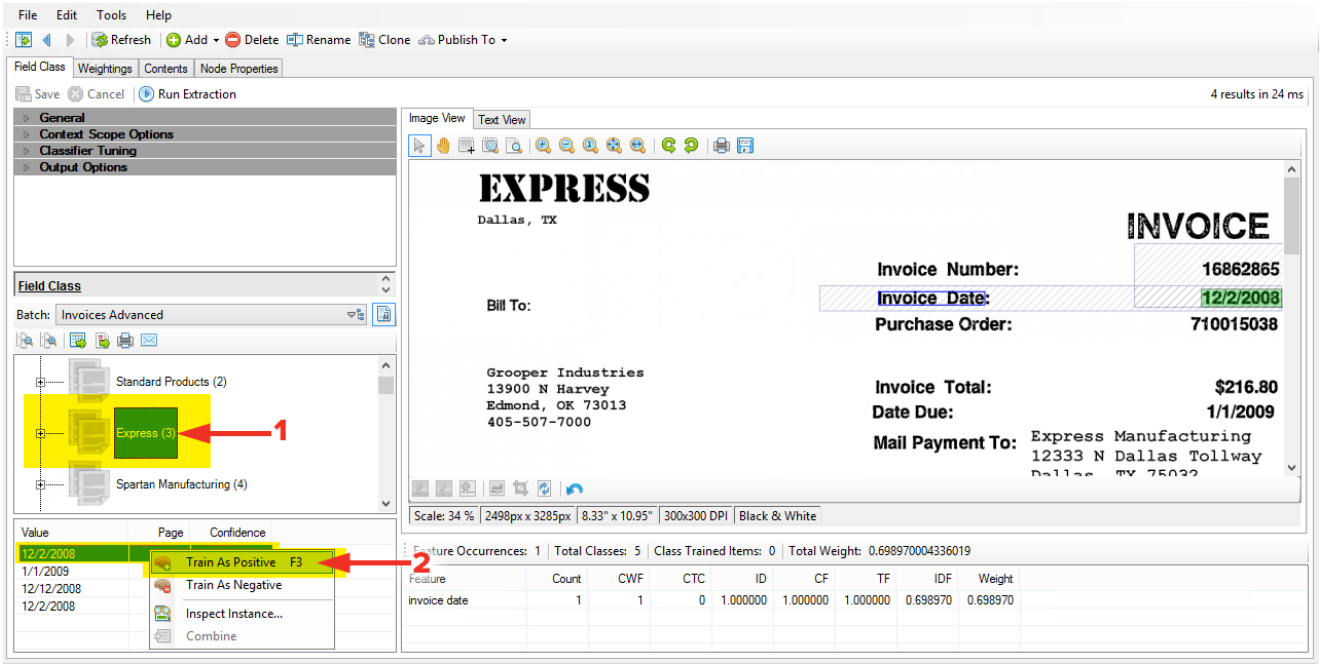
STEP 11 – CHECKING WEIGHTINGS ONCE AGAIN

Click the Weightings tab again. Notice now there are two Positive Instances: date and invoice date. Notice also that the feature date has a count of two, while invoice date only has a count of one. See also that there are now Negative Instances. These were features that were captured within the Context Zones of the other listed values for this document, but weren't selected when the positive training was invoked.



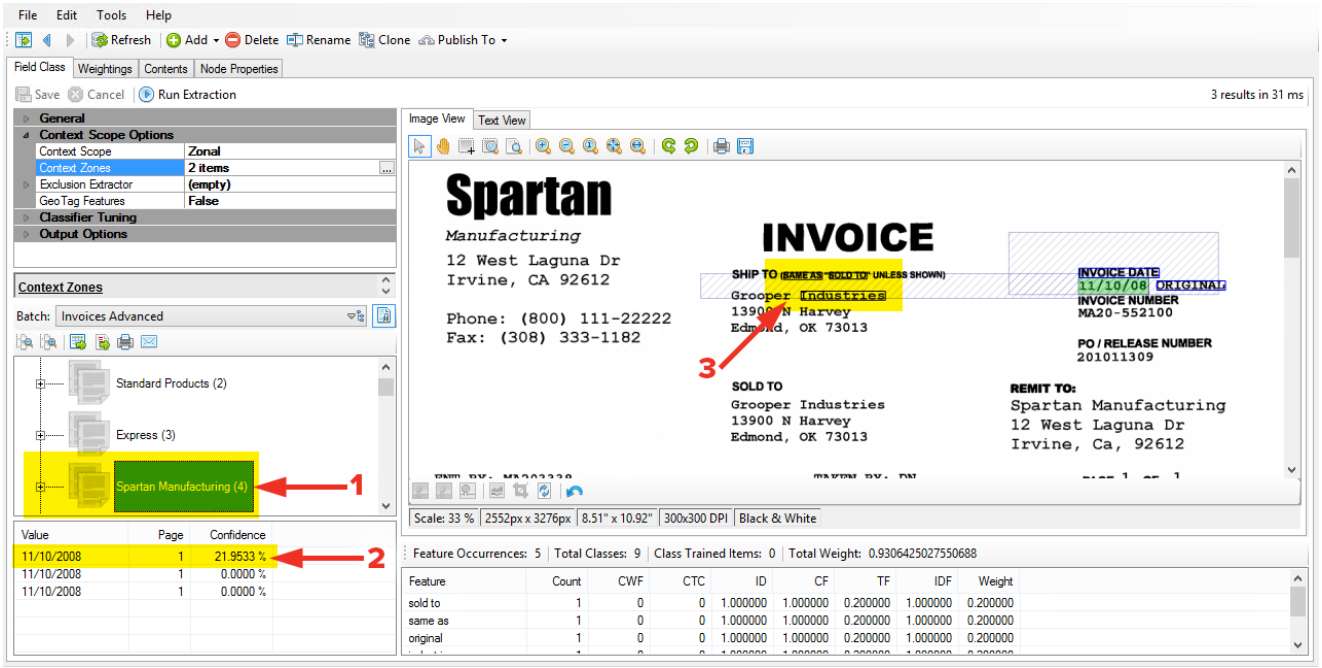
STEP 12 – REINFORCING TRAINING

(1) Select the Express (3) document then **(2)** select the 12/2/2008 value in the List View with ~65% and right-click Train As Positive. This will reinforce the feature invoice date as a Positive Instance, and add features around the other listed values as Negative Instances.



STEP 13 – ZONAL PROBLEMS

(1) Click the Spartan Manufacturing (4) document. **(2)** Notice the low confidence of the correct value in the List View. **(3)** This low confidence is due to the erroneous features being captured by the context zones surrounding the desired value. Considering this, the Context Zones need to be adjusted to mitigate capturing unwanted features.



STEP 14 – ZONAL ADJUSTMENTS

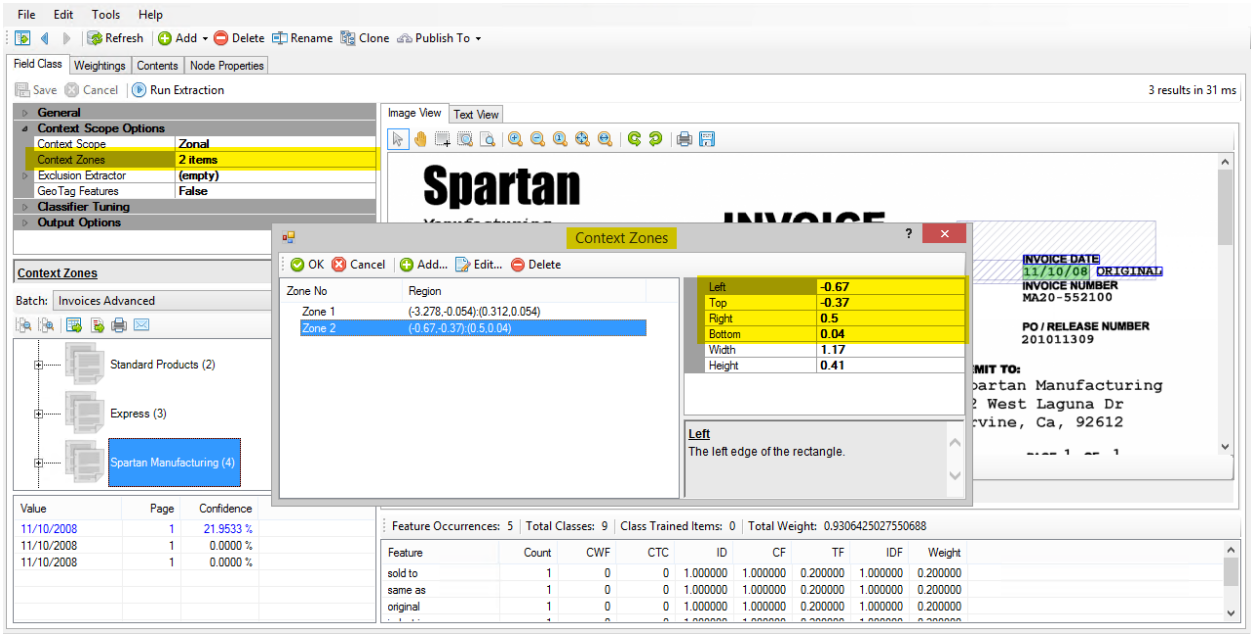
Bring up the Context Zones window and enter the following values:

Zone 1 – Left: **-3.278** Top: **-0.054** Right: **0.312** Bottom: **0.054**

Zone 2 – Left: **-0.67** Top: **-0.37** Right: **0.5** Bottom: **0.04**

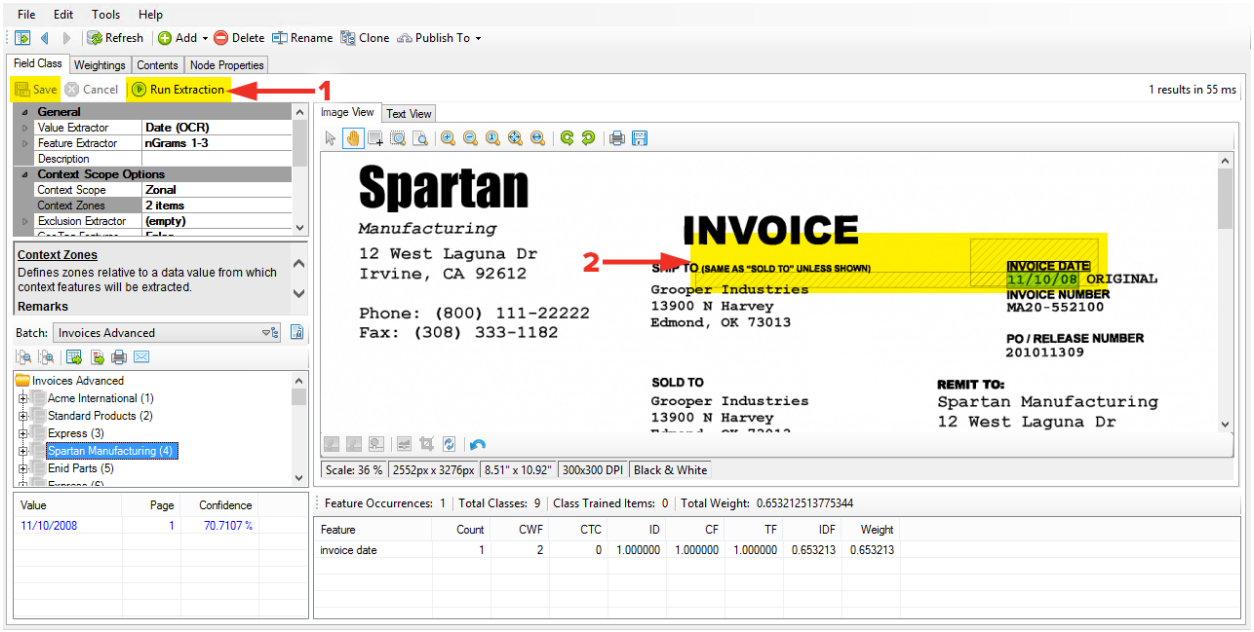
Click OK to accept the changes and close the window.

These values will adjust the Context Zones such that the outer most boundaries of the boxes are not touching the center point of the blue boxes that are drawn around the unwanted features. Keep this in mind when adjust Context Zones. You only need your zones to touch (or NOT in this case) the mid-points, not the entire word.



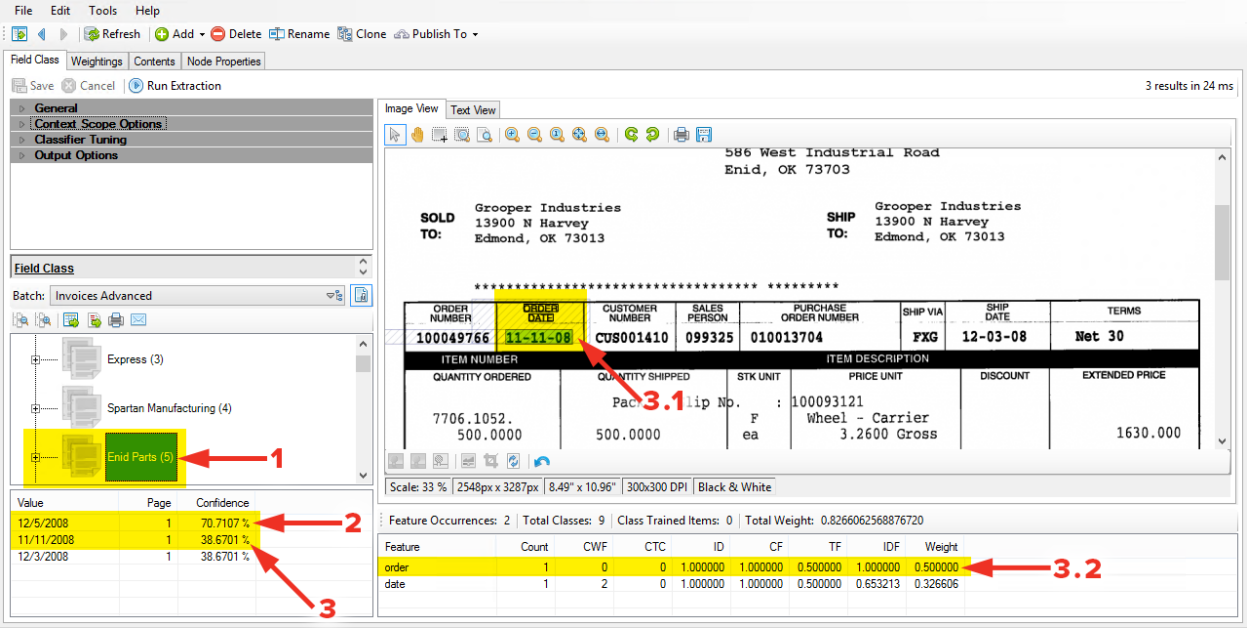
STEP 15 – CHECKING ZONAL ADJUSTMENTS

(1) Click Save and Run Extraction. **(2)** Notice the Context Zones are no longer highlighting unwanted features, and the Feature List View is returning a desired feature. This instance will not need to be trained.



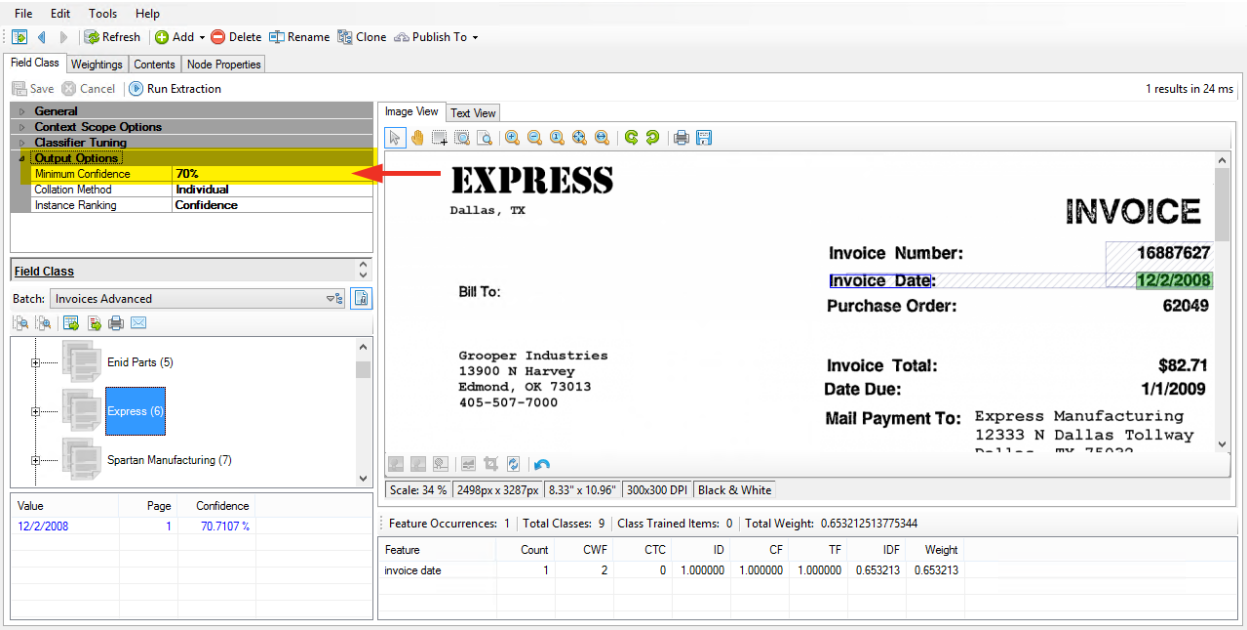
STEP 16 – UNDERSTANDING WEIGHTINGS FURTHER

(1) Click the Enid Parts (5) document. **(2)** The correct value will be listed at the top of the Value List View, but you should notice a couple of other date values listed with non-zero values for their confidence. **(3)** Select one of the ~38% values, and check the Feature List View and observe that a positively trained feature, in this case date, is being found within a Context Zone, but also an untrained feature, order. Due to this, **Grooper** is less confident this is a correct value, which is correct for this case.



STEP 17 – SETTING MINIMUM CONFIDENCE

With a well-established confidence of ~70%, unwanted returned values can be eliminated by setting an acceptance threshold. Set the Minimum Confidence property to 70%. Any value results returned that have a confidence less than this will be dropped. Feel free to scroll through and select documents from the Batch Viewer and see the results of this trained Field Class.

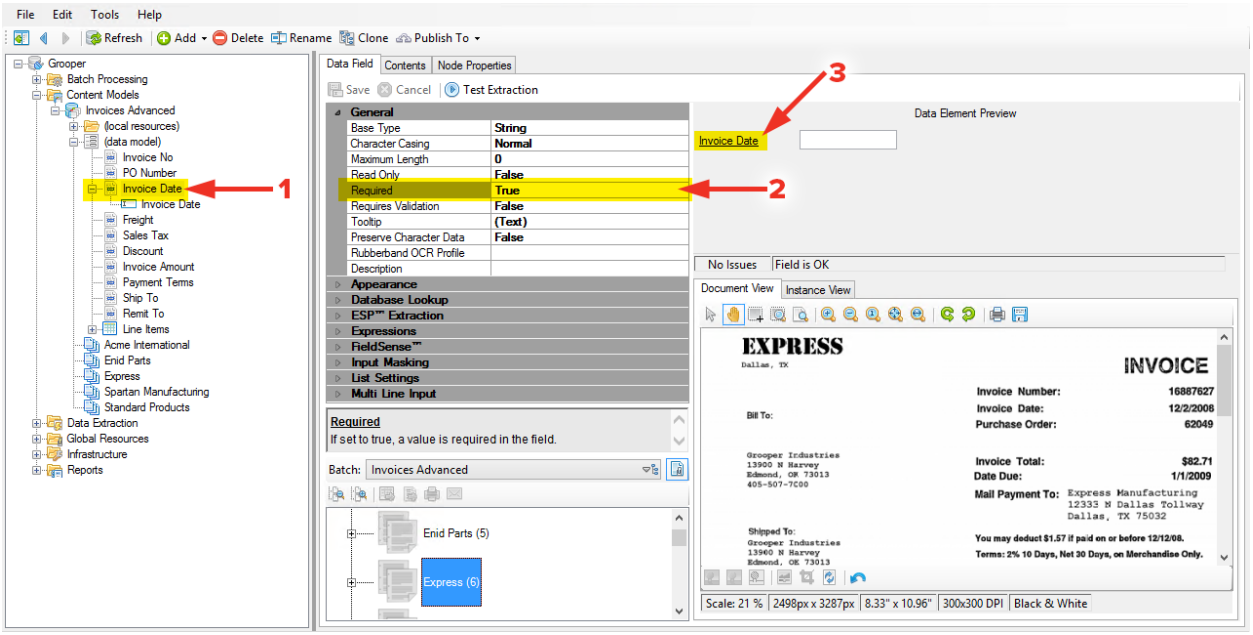


SETTING UP THE INVOICE DATE DATA FIELD

The extractor for the Invoice Date Data Field has been built, but it needs to be applied to the Data Field and its properties adjusted.

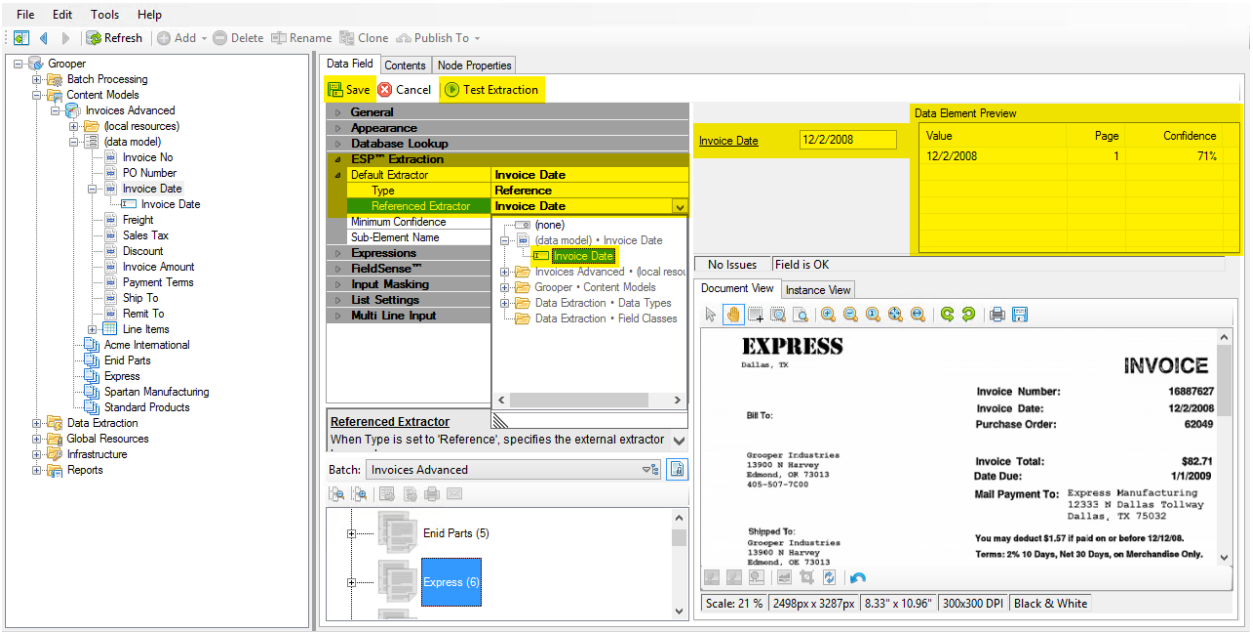
STEP 1 – SETTING THE DATA FIELD TO BE REQUIRED

(1) Select the Invoice Date Data Field and in the General section, **(2)** set its Required property to True. **(3)** Notice the name of the field become underlined.



STEP 2 – APPLYING THE EXTRACTOR

In the ESP Extraction section, set the Default Extractor Type to Reference. Point the Referenced Extractor property to the Invoice Date Field Class. Save and Test Extraction to see results.

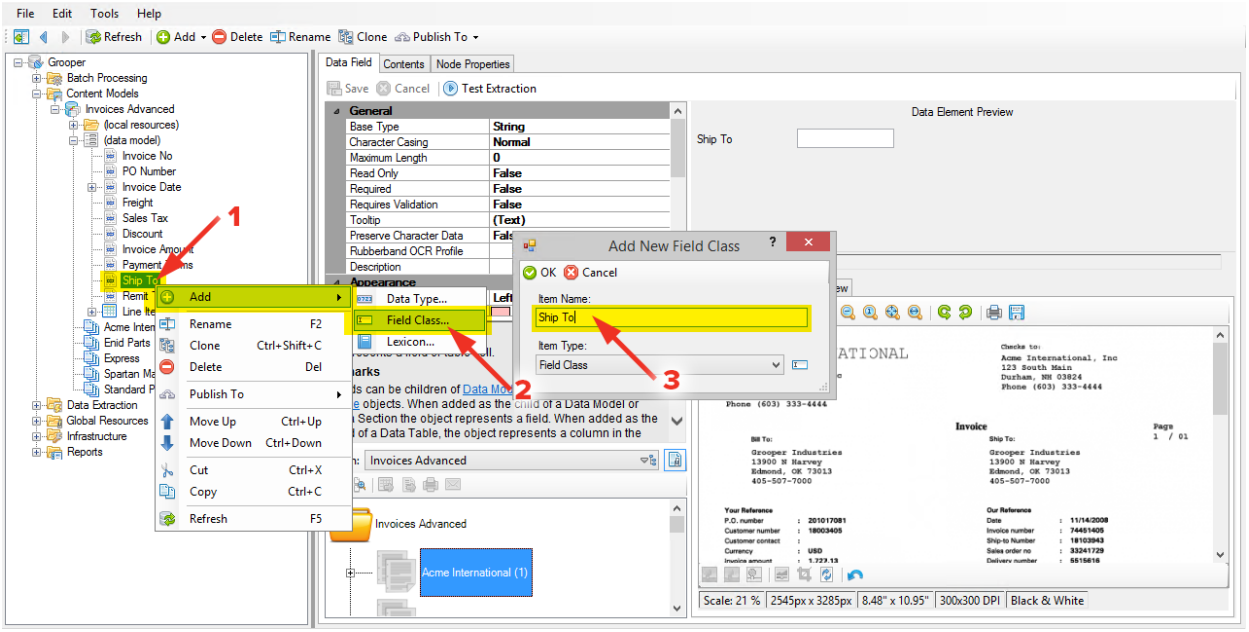


THE NEXT FIELD CLASS – BUILDING A FEATURE EXTRACTOR

Let's continue working with the Field Class, but this time build out one of the two extractors it leverages. This Field Class will be looking for the Ship To address, and will use a (system) Data Type for the address pattern (the Value Extractor). The Feature Extractor will use a simple Data Type that will look for specific word patterns, but return a single value each time. The usefulness of this will be recognized soon.

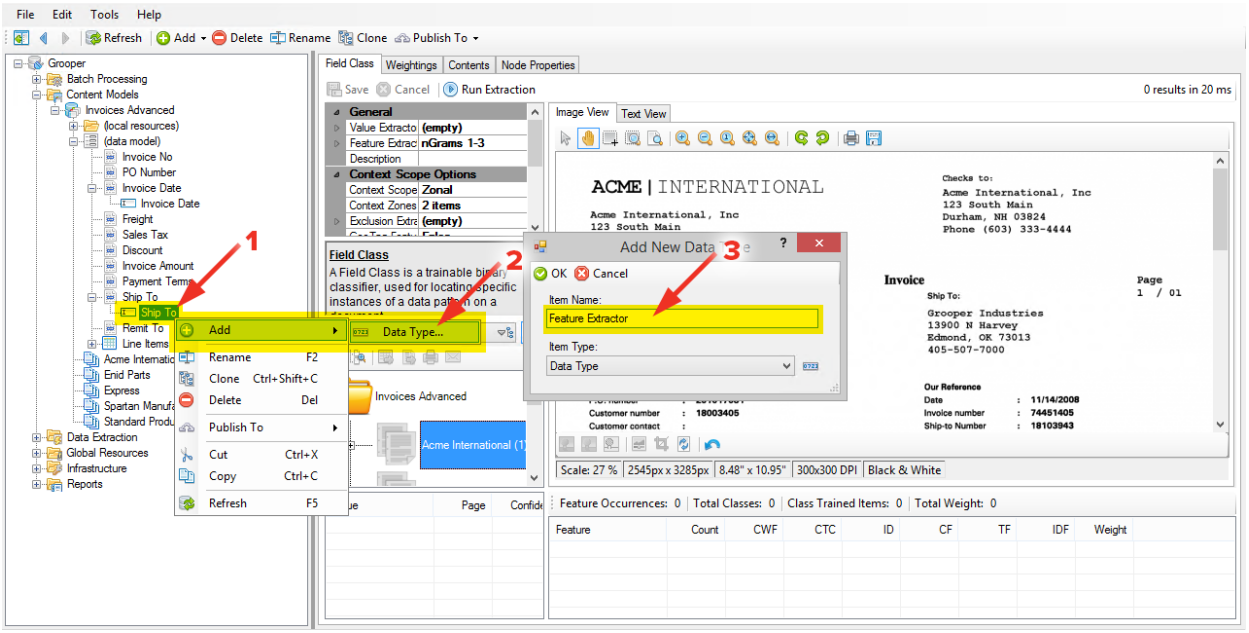
STEP 1 – ADDING A FIELD CLASS TO THE SHIP TO DATA FIELD

Right-click the Ship To Data Field and Add > Field Class... Name it Ship To.



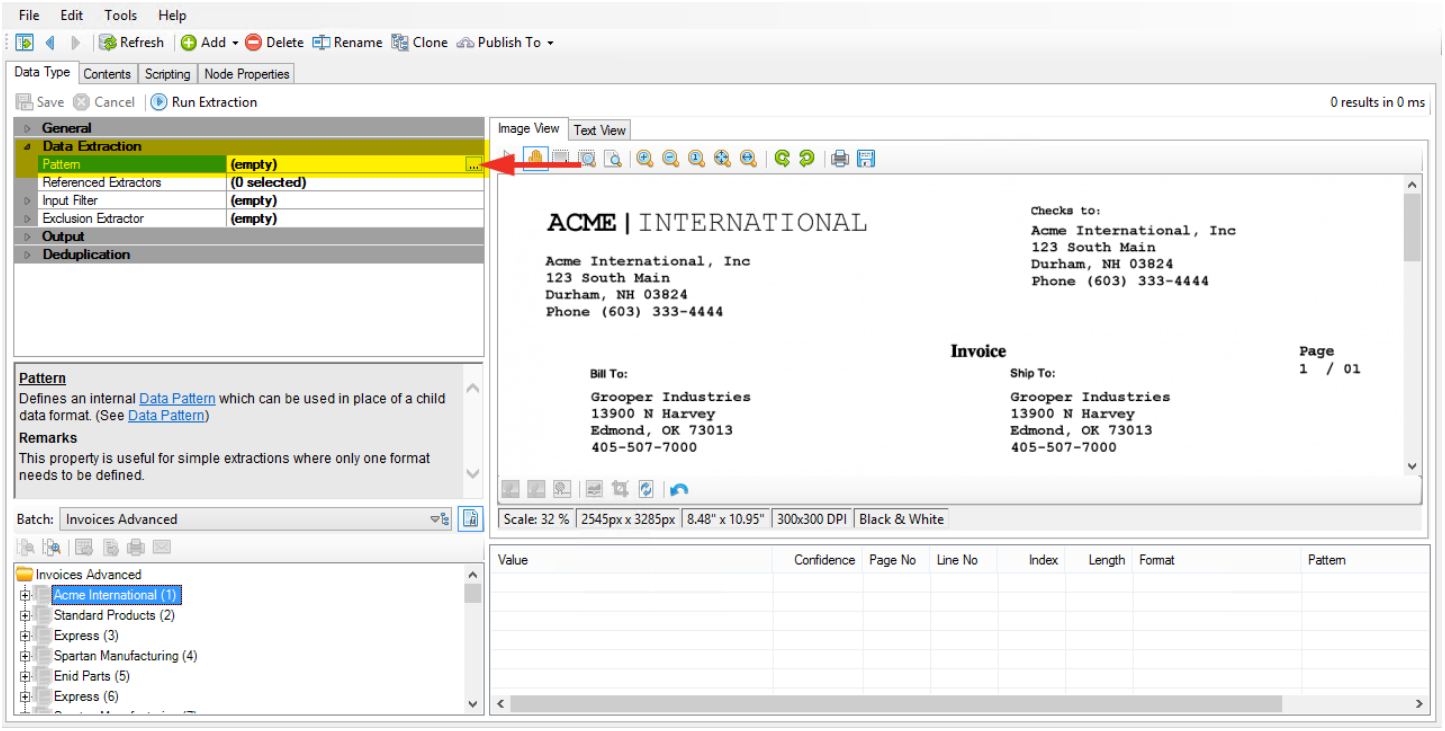
STEP 2 – ADDING A DATA TYPE TO THE SHIP TO FIELD CLASS

(1) Right-click the Ship To Field Class **(2)** and Add > Data Type... **(3)** Name it Feature Extractor.



STEP 3 – ADDING AN INTERNAL PATTERN TO A DATA TYPE

Previously, the patterns added to Data Types that have been worked with were via Data Format objects. This pattern will be written internally on the Data Type by selecting the Pattern property in the Data Extraction section and clicking the ellipsis button.



STEP 4 – WRITING THE PATTERN / FORCING A SPECIFIC OUTPUT

(1) In the Value Pattern use the following pattern:

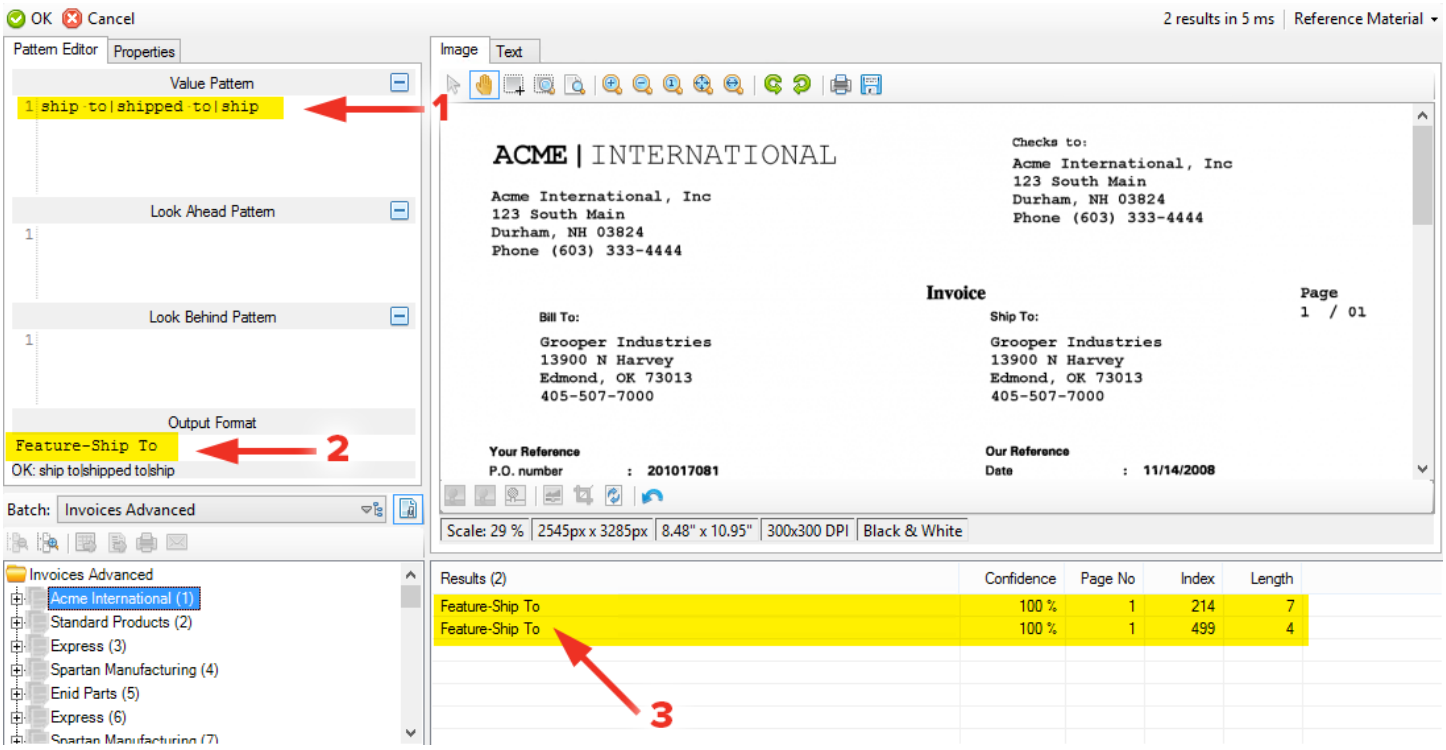
```
ship to|shipped to|ship
```

The vertical pipe characters in RegEx determine OR, so in this case find **`ship to`** or **`shipped to`** or **`ship`**.

(2) In the Output Format use the following pattern:

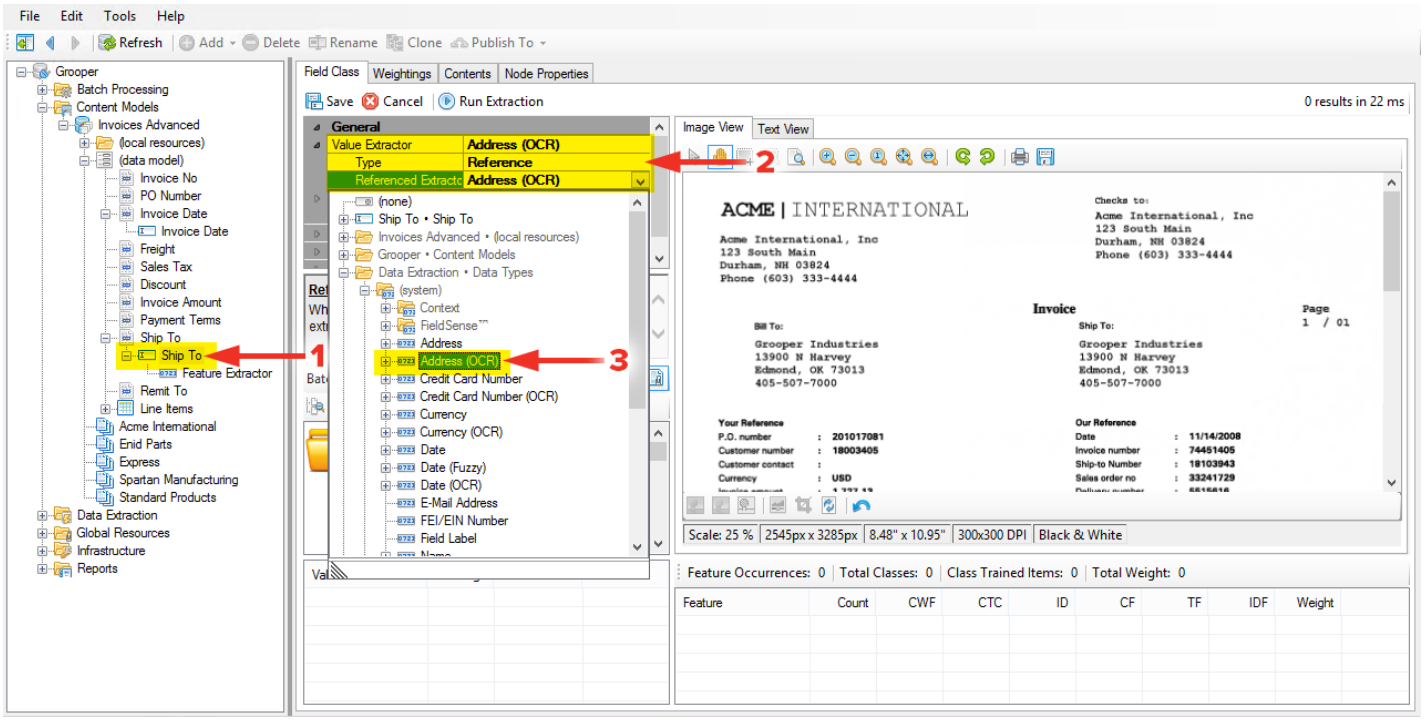
```
Feature-Ship To
```

(3) This will force all instances of returned results found by the RegEx pattern to be returned as the string Feature-Ship To. Click through some documents in the Batch Viewer and see the results that are returned. When the Field Class features are trained, the usefulness of this technique will become apparent.



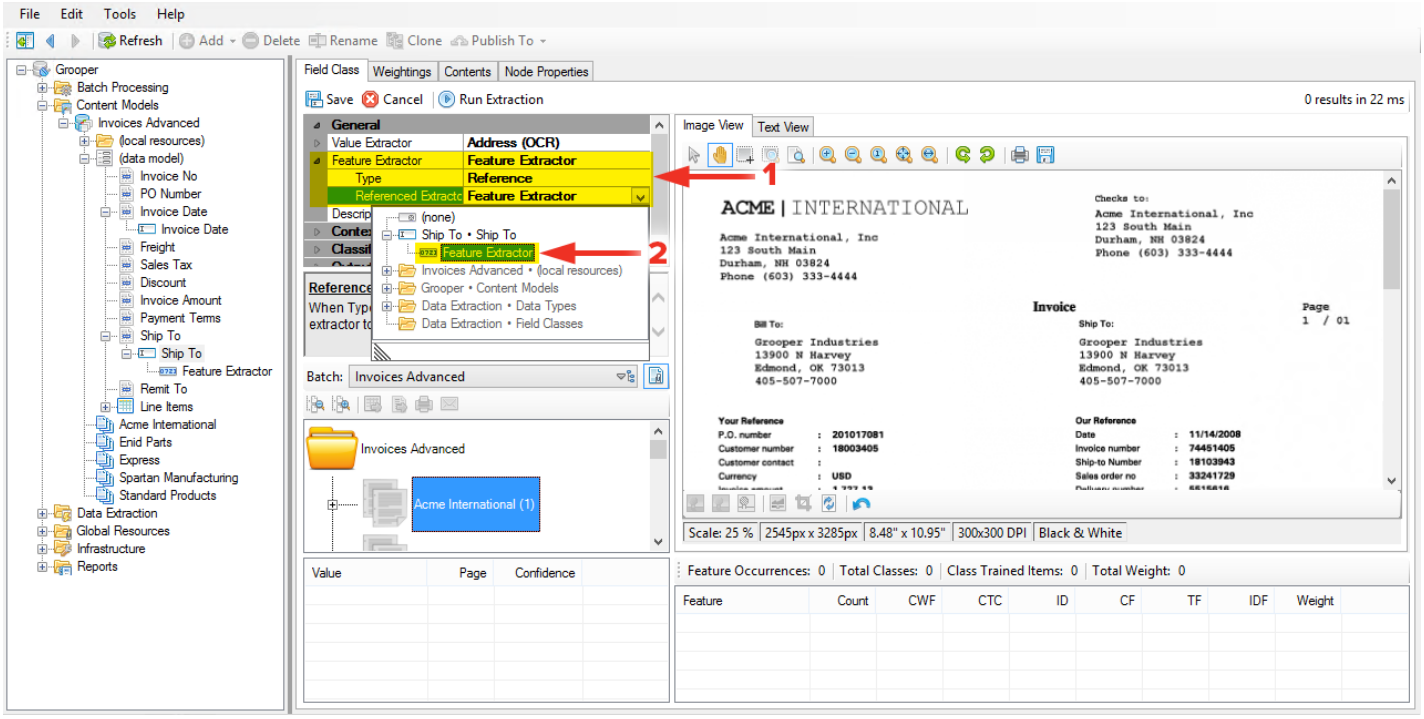
STEP 5 – SETTING THE VALUE EXTRACTOR

(1) Click back on the Ship To Field Class. **(2)** Set the Value Extractor type to Reference **(3)** and the Referenced Extractor to the Data Extraction • Data Types > (system) > Address (OCR) Data Type.



STEP 6 – SETTING THE FEATURE EXTRACTOR

(1) Set the Feature Extractor Type to Reference and **(2)** set the Referenced Extractor to the Feature Extractor Data Type that is a child object of the Ship To Field Class.

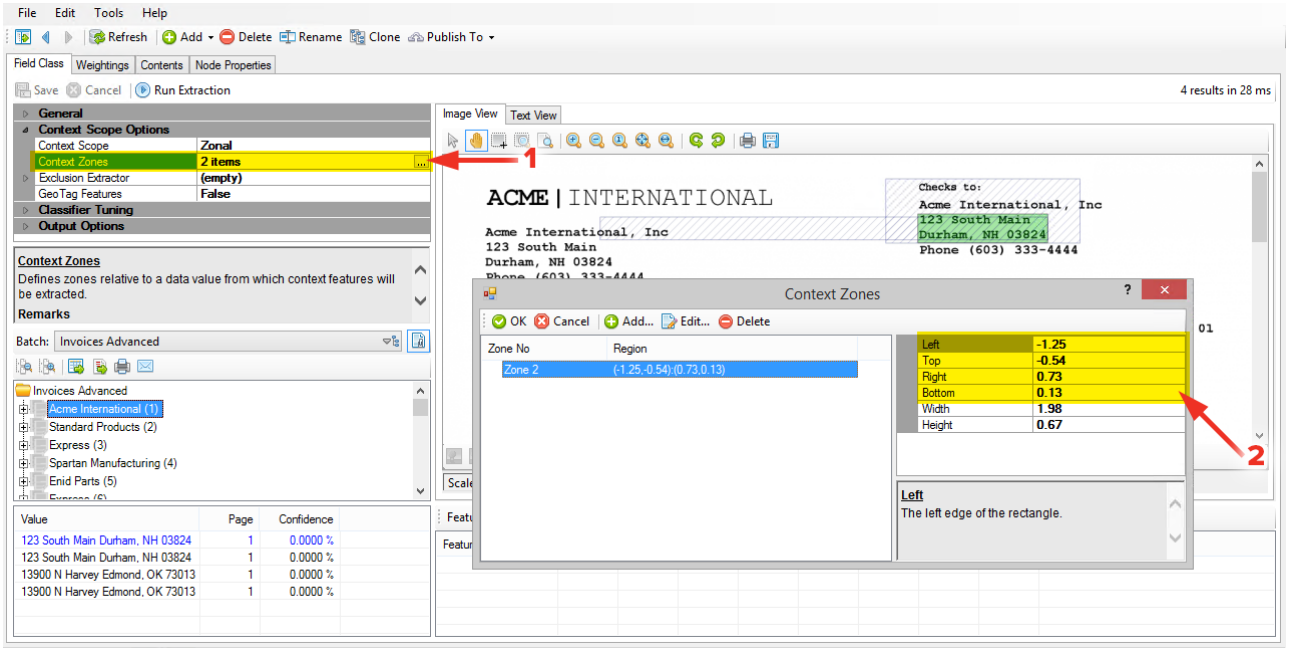


STEP 7 – ADJUSTING THE CONTEXT ZONES

In the Context Scope Options section, **(1)** select the Context Zones property and click the ellipsis button to bring up the Context Zones properties window. **(2)** Delete Zone 1 and set the Zone 2 dimensions to:

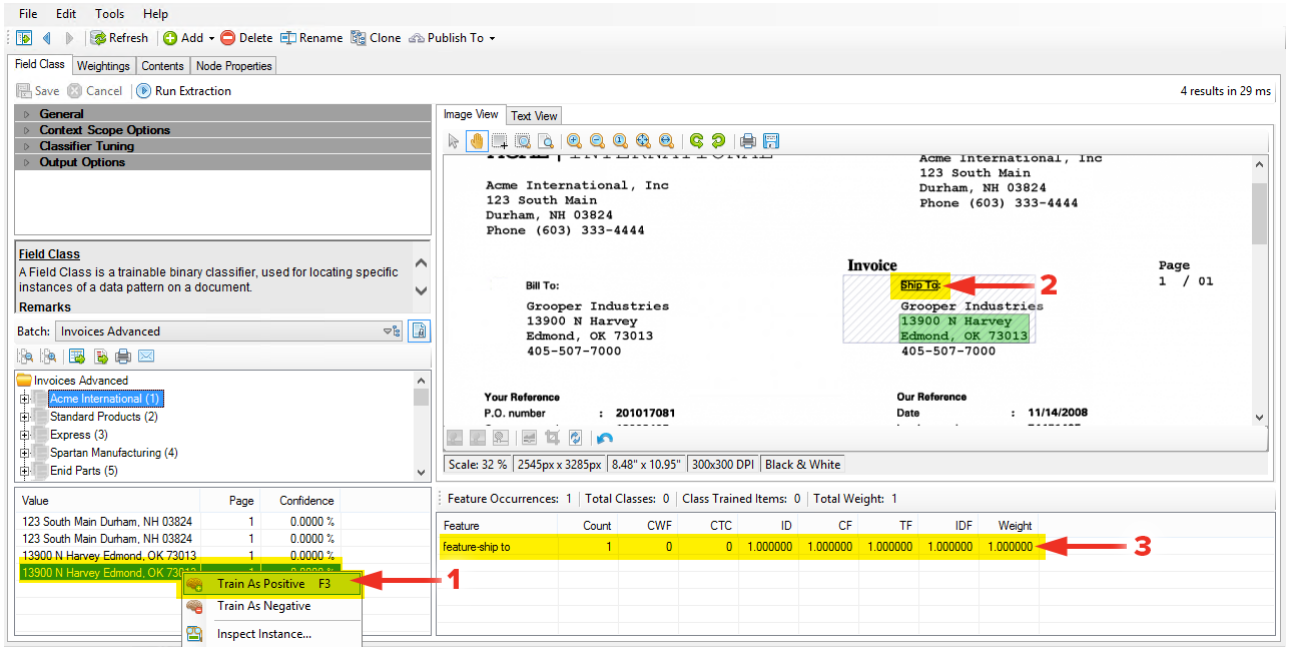
Zone 2 – Left: **-1.25** Top: **-0.54** Right: **0.73** Bottom: **0.13**

Click OK to close the Context Zones properties window, then Save and Run Extraction on the Field Class against the Acme International (1) document.



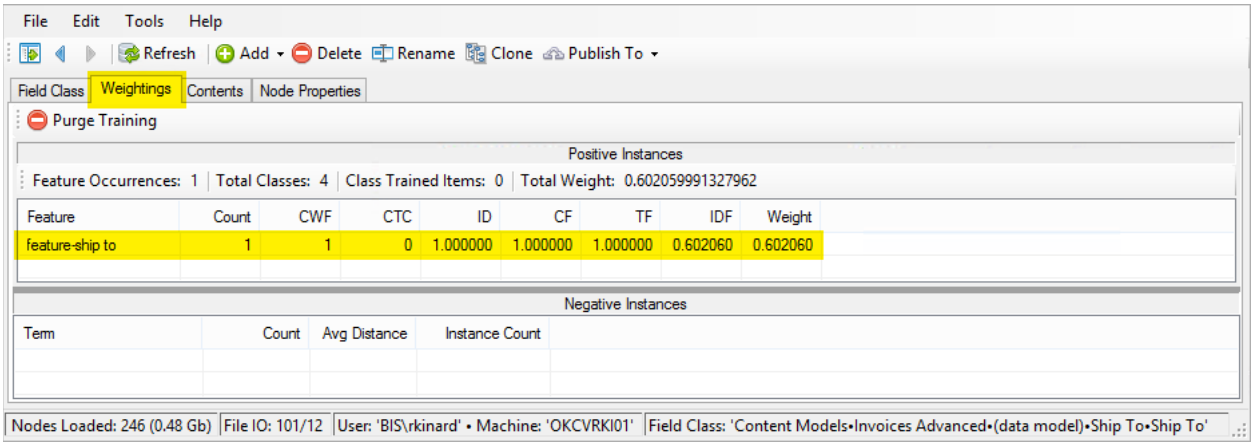
STEP 8 – TRAINING A FEATURE INSTANCE

Four values should be returned, but only one of them will capture a feature. **(1)** Select the value that returns a feature, and **(2)** notice that Ship To in the image has the blue box drawn around it, but **(3)** the Feature Occurrence that's returned is feature-ship to. Train this instance positively.



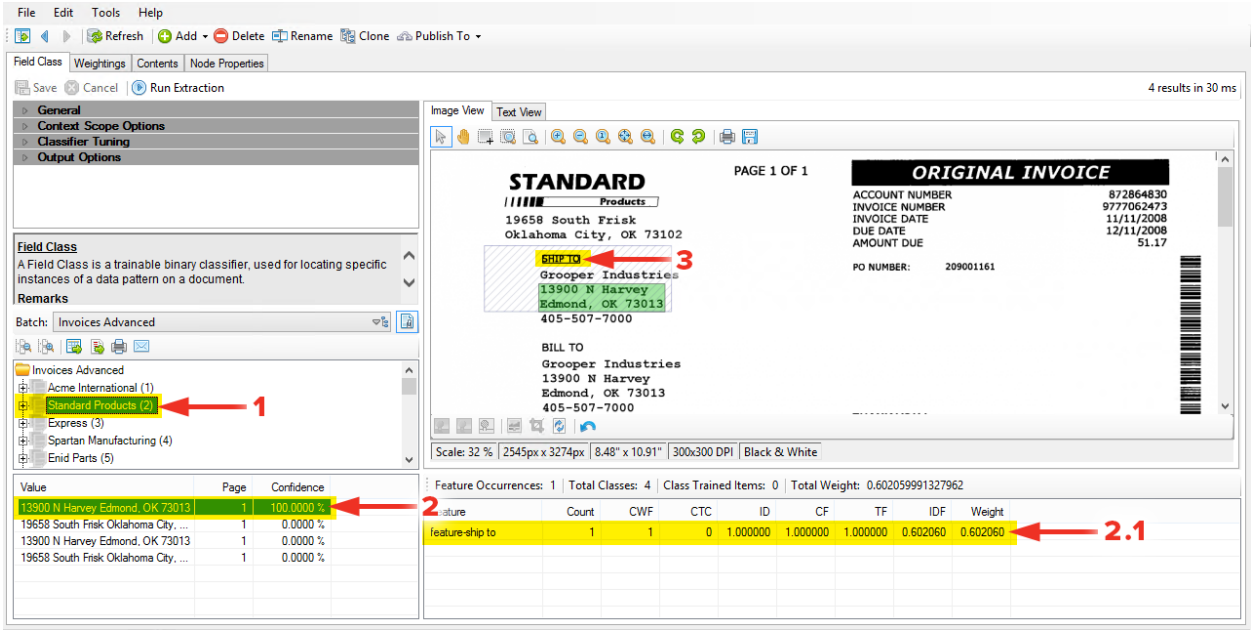
STEP 9 – CHECK WEIGHTINGS

Check the Weightings tab to verify that only one feature has been trained, and it is feature-ship to. No negative instances will be listed because the Context Zone around the other values from the Acme Internationa (1) did not capture a feature, and as such, did not return negative results during the positive training of the correct value.



STEP 10 – TESTING OTHER DOCUMENTS

(1) Select the Standard Products (2) document from the Batch Viewer and **(2)** notice the top value return the feature feature-ship to and a confidence of 100%.

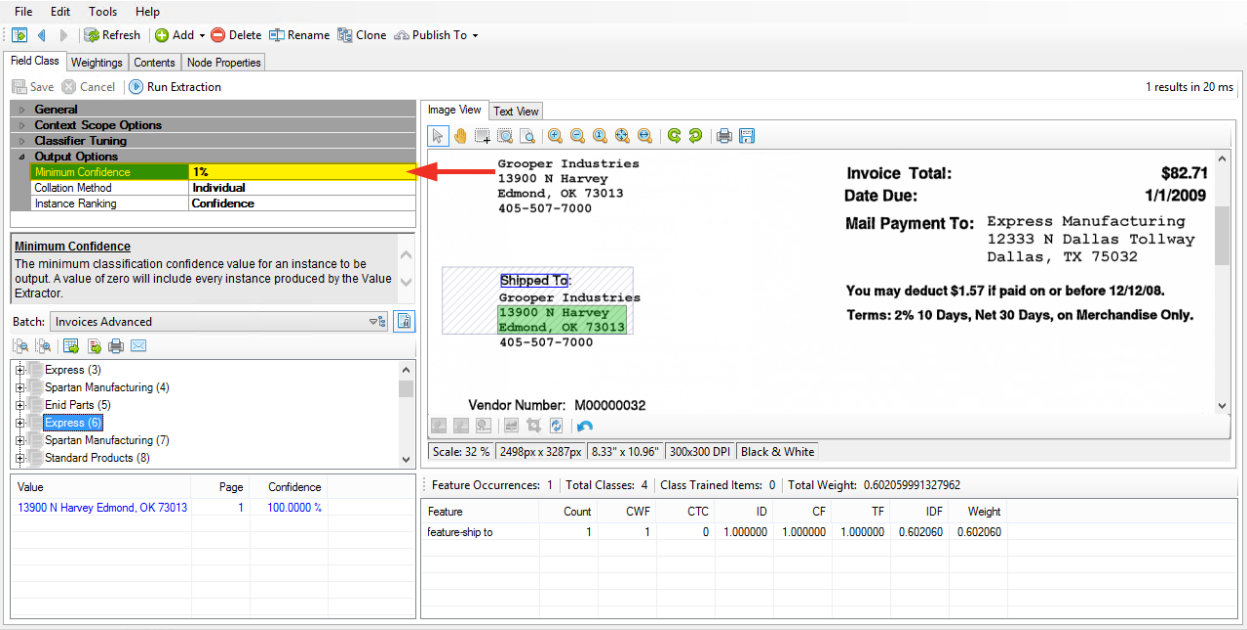


Now test either Express (3) or Enid Parts (5) and notice that the top value result is also returning the feature feature-ship to, **(3)** but the highlighted feature in the Page Viewer is Shipped to and Ship (respectively.)

If you recall, the Data Type that was built to extract features is looking for: ship, shipped to, and ship to, but results are always returned as: feature-ship to. This feature, as a result, only needed to be trained once, and the Field Class will operate at 100% confidence. If the Data Type were built to find those words, but only return what was found, each instance of ship, shipped to, or ship to would have to be trained separately.

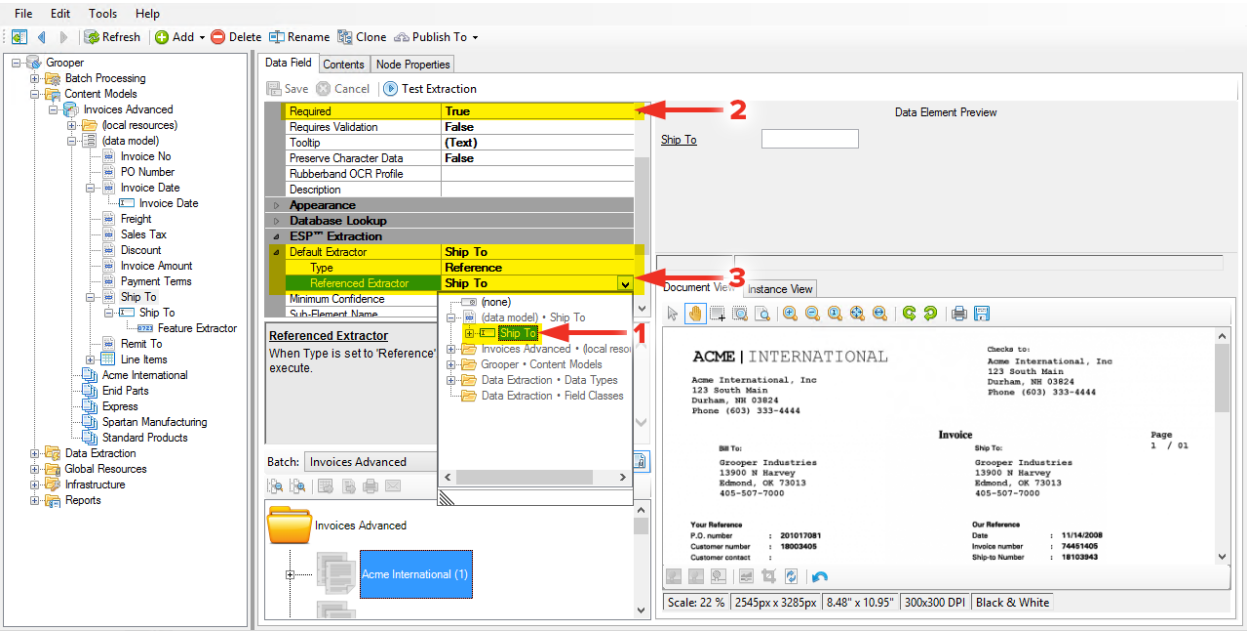
STEP 11 – SETTING MINIMUM CONFIDENCE

Because there is only one feature trained, and it is being returned at 100% confidence, the Minimum Confidence can be set to anything above 0% and all unwanted results will be trimmed.



STEP 12 – SETTING PROPERTIES FOR THE SHIP TO DATA FIELD

(1) Select the Ship To Data Field and **(2)** set its Required property to True. **(3)** Set the Default Extractor Type to Reference and set the Referenced Extractor to the Ship To Field Class.

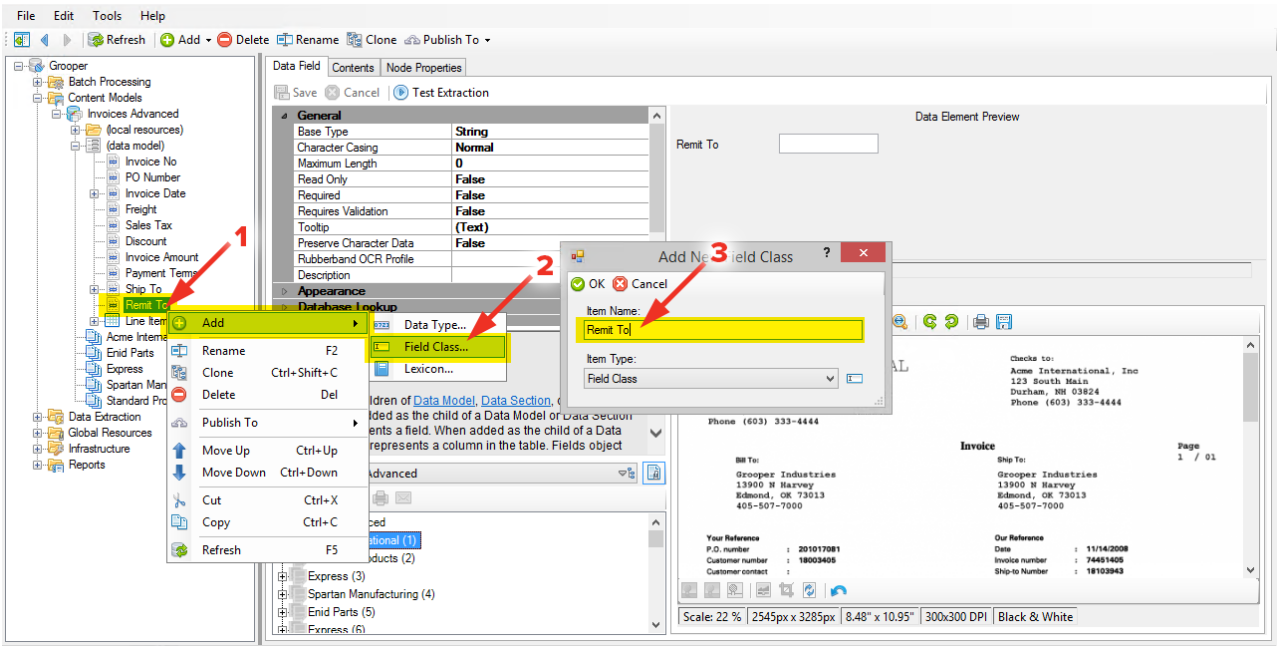


THE NEXT FIELD CLASS – BUILDING A FEATURE EXTRACTOR WITH FUZZYREGEX

The Field Class for the Ship To Data Field used a simple pattern in a Data Type to find specific sets of words, but return them as a single output. That same idea will be carried on for the next Field Class, but the Feature Extractor will now compensate for OCR errors via a method in **Grooper** called FuzzyRegEx.

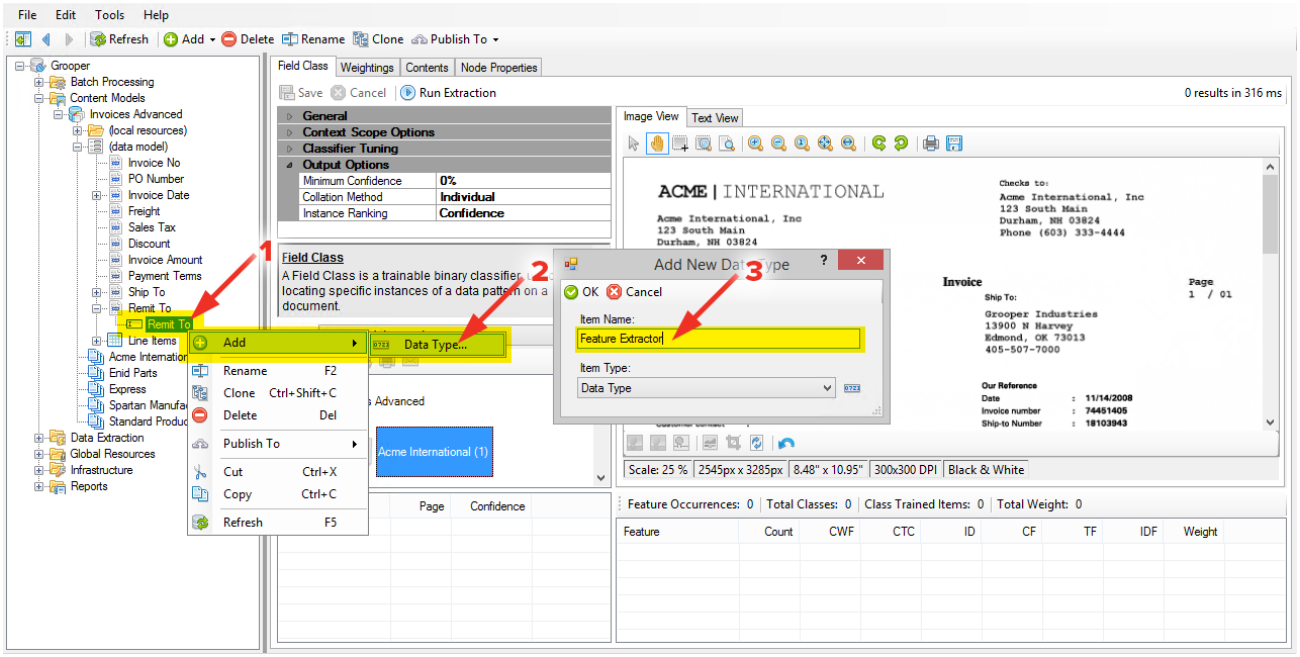
STEP 1 – ADDING A FIELD CLASS TO THE REMIT TO DATA FIELD

Right-click the Remit To Data Field and Add > Field Class... Name it Remit To.



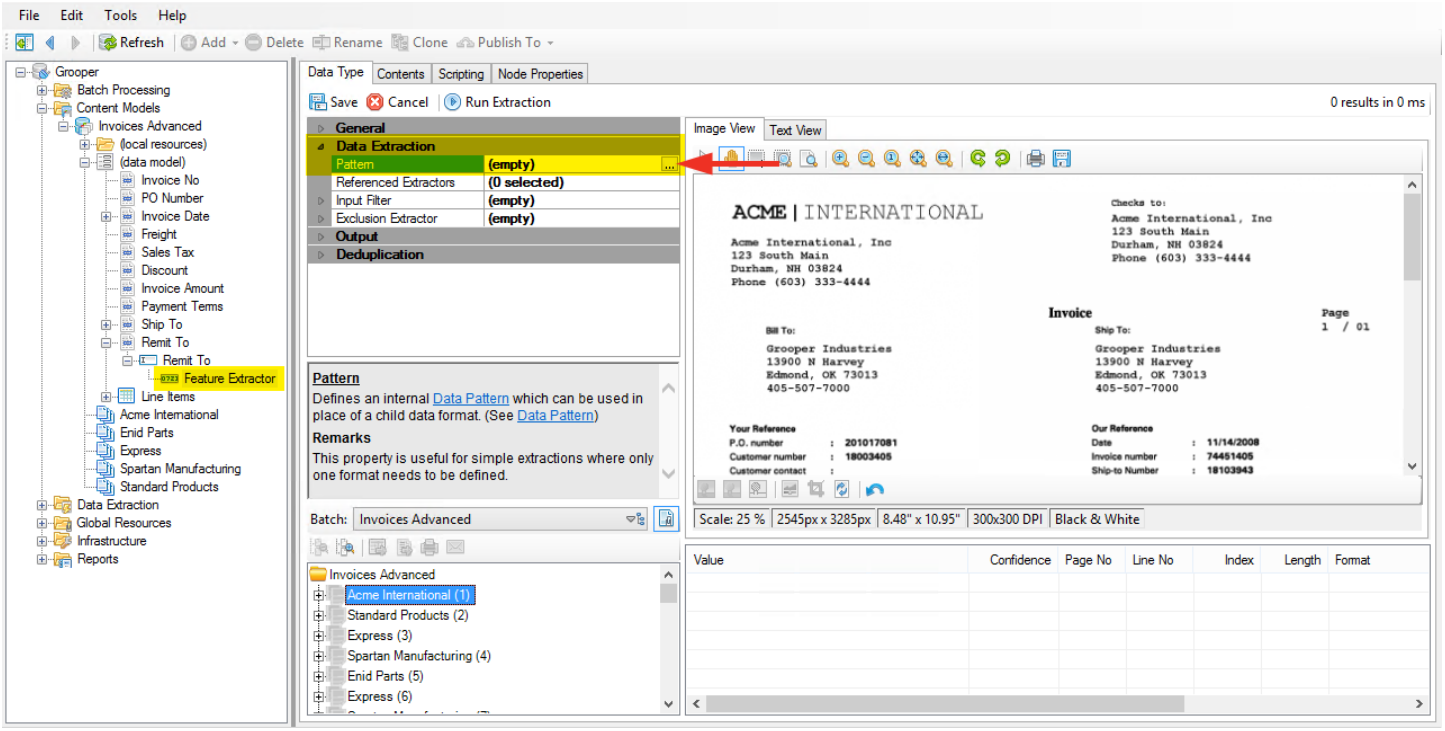
STEP 2 – ADDING A DATA TYPE TO THE SHIP TO FIELD CLASS

Right-click the Remit To Field Class and Add > Data Type... Name it Feature Extractor.



STEP 3 – ADDING AN INTERNAL PATTERN TO THE FEATURE EXTRACTOR DATA TYPE

Like the previous Feature Extractor Data Type that was built, this Feature Extractor Data Type will also use an internal pattern. Click the Pattern property within the Data Extraction section and then click the ellipsis button to bring up the pattern editor.



STEP 4 – WRITING THE PATTERN / FORCING A SPECIFIC OUTPUT

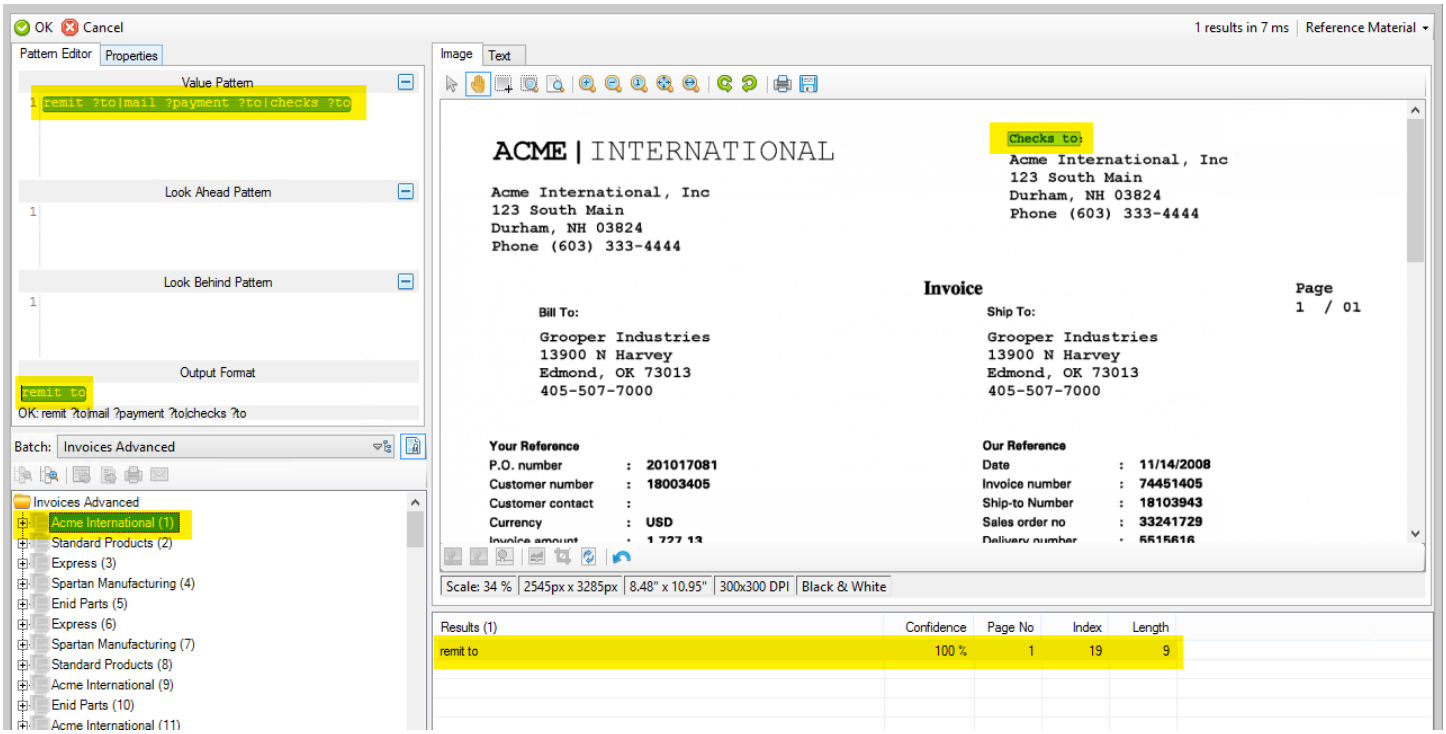
In the Value Pattern area using the following pattern:

```
remit ?to|mail ?payment ?to|checks ?to
```

This pattern is using vertical pipes again, and also making the spaces optional by using the 0-1 quantifier **?**, therefore looking for **`remit to`**, **`mail payment to`**, and **`checks to`**, with optional spaces.
In the Output Format area using the following pattern:

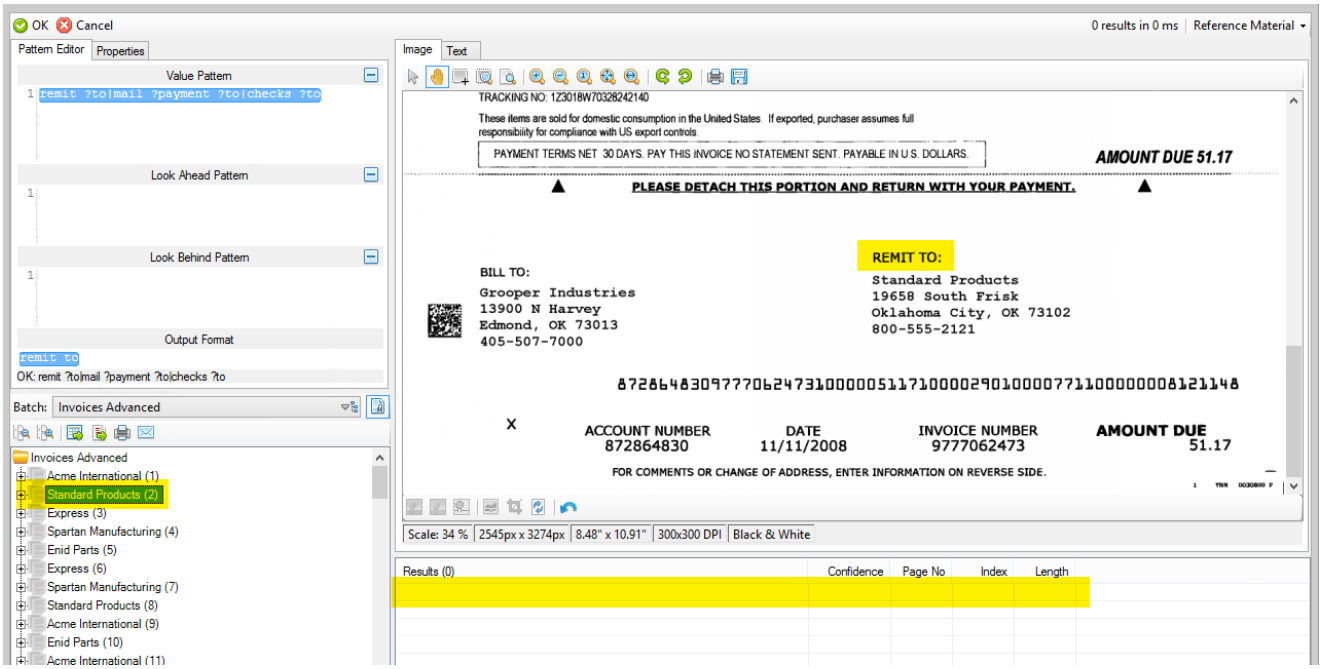
```
remit to
```

This will force all returned values to be **`remit to`**, instead of the values returned by the RegEx pattern. So notice on the Acme International (1) document that **`Checks to`** is found, but returned as **`remit to`**.



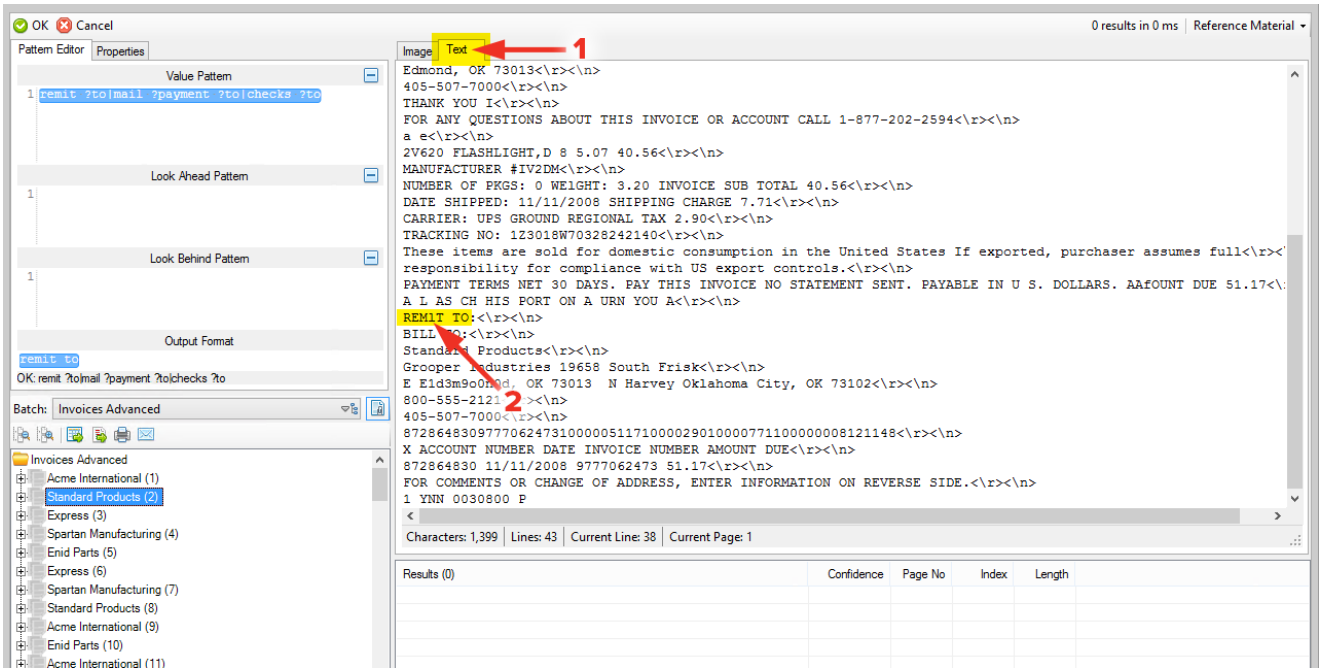
STEP 5 – DISCOVERING AN OCR ERROR

Click on the Standard Products (2) document in the Batch Viewer and notice our pattern is not returning any results. Scroll the image down in the Page Viewer and notice that **REMIT TO** is on this page, so the RegEx pattern used should find it, right?



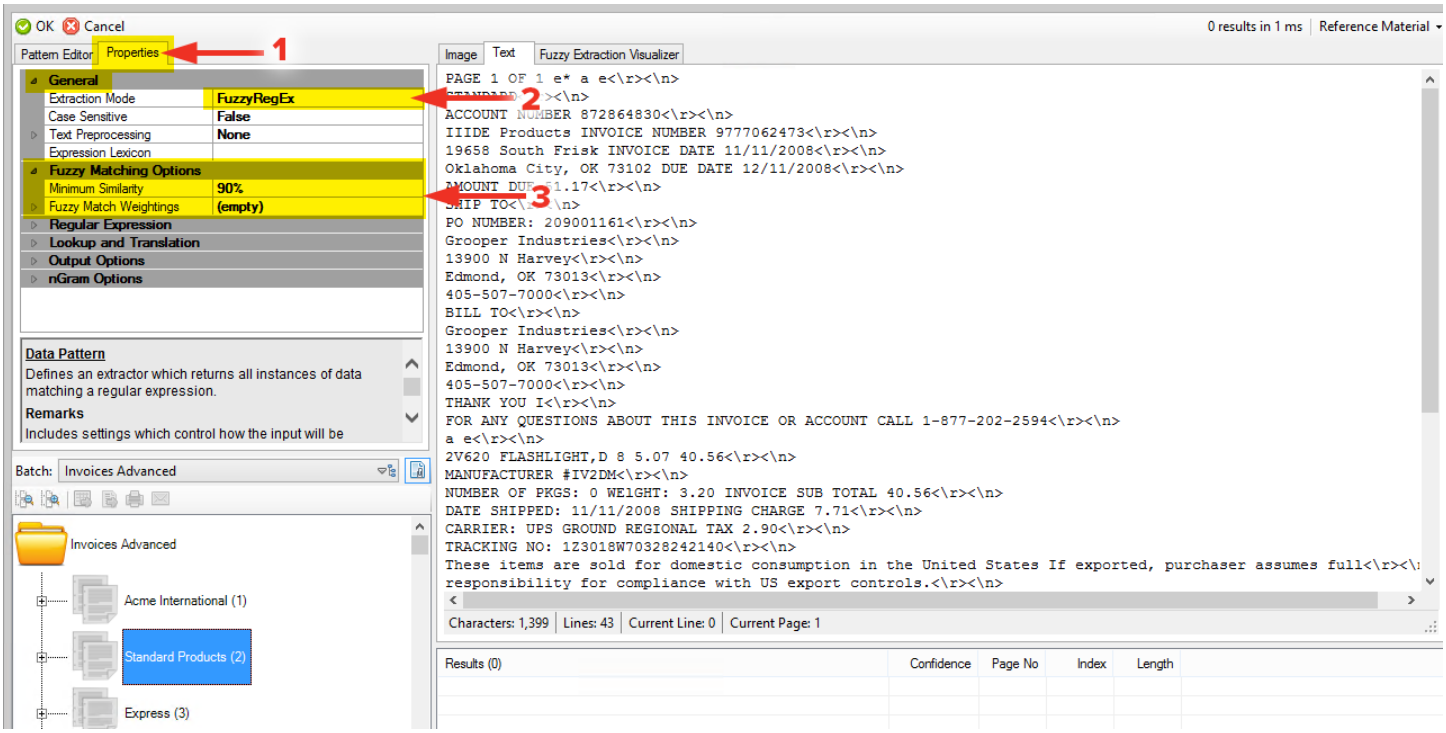
STEP 6 – DIAGNOSING THE PROBLEM

The first thing to understand about OCR is that it is imperfect, and it will never return results with 100% accuracy. **(1)** Click on the Text tab and scroll down to **(2)** see that the capital **I** in **REMIT TO** was mis-OCRed as a lower case **l**. While complicating the RegEx pattern to accommodate for this is possible, there's a more elegant solution available in **Grooper**.



STEP 7 – ENABLING FUZZYREGEX

(1) Click on the Properties tab and in the General section, **(2)** change the Extraction Mode from RegEx to FuzzyRegEx. Notice when doing this a new section of options become available: Fuzzy Matching Options.



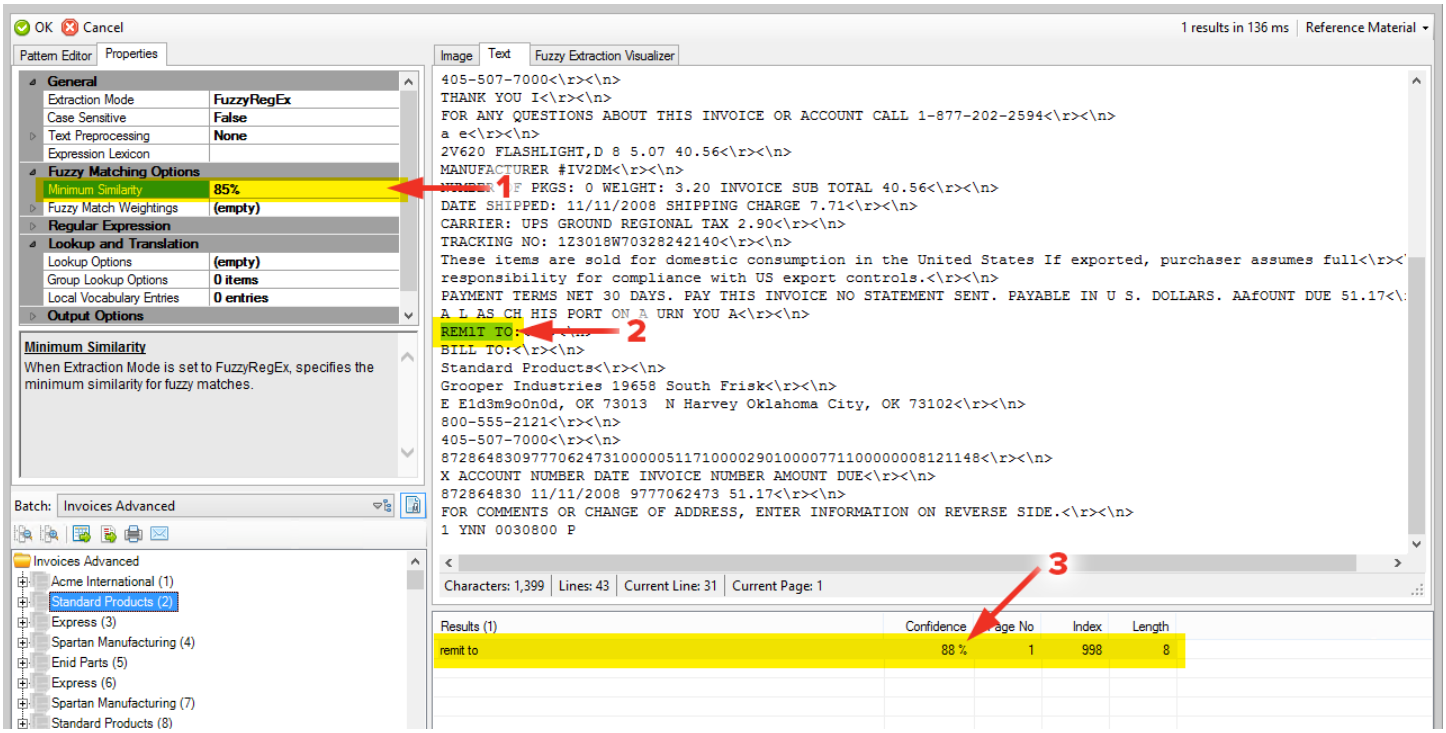
The first property to understand with FuzzyRegEx is **(3)** the Minimum Similarity. This is a threshold that is set to disallow translations of our words that fall below this set amount. At 90% we still won't get a match on this document.

STEP 8 – ADJUSTING THE MINIMUM SIMILARITY TO FIND A MATCH

(1) Change the Minimum Similarity to 85% and observe what happens. **(2)** The mis-OCRed **`REMlT TO`** will be found and translated to the specific string we told our RegEx pattern to find, which is **`remit to`** (keep in mind the Case Sensitive property is False by default.) **(3)** It's also worth noting that our Output Format is forcing the returned value to be **`remit to`**, which means any result will be returned as the string defined by the Output Format.

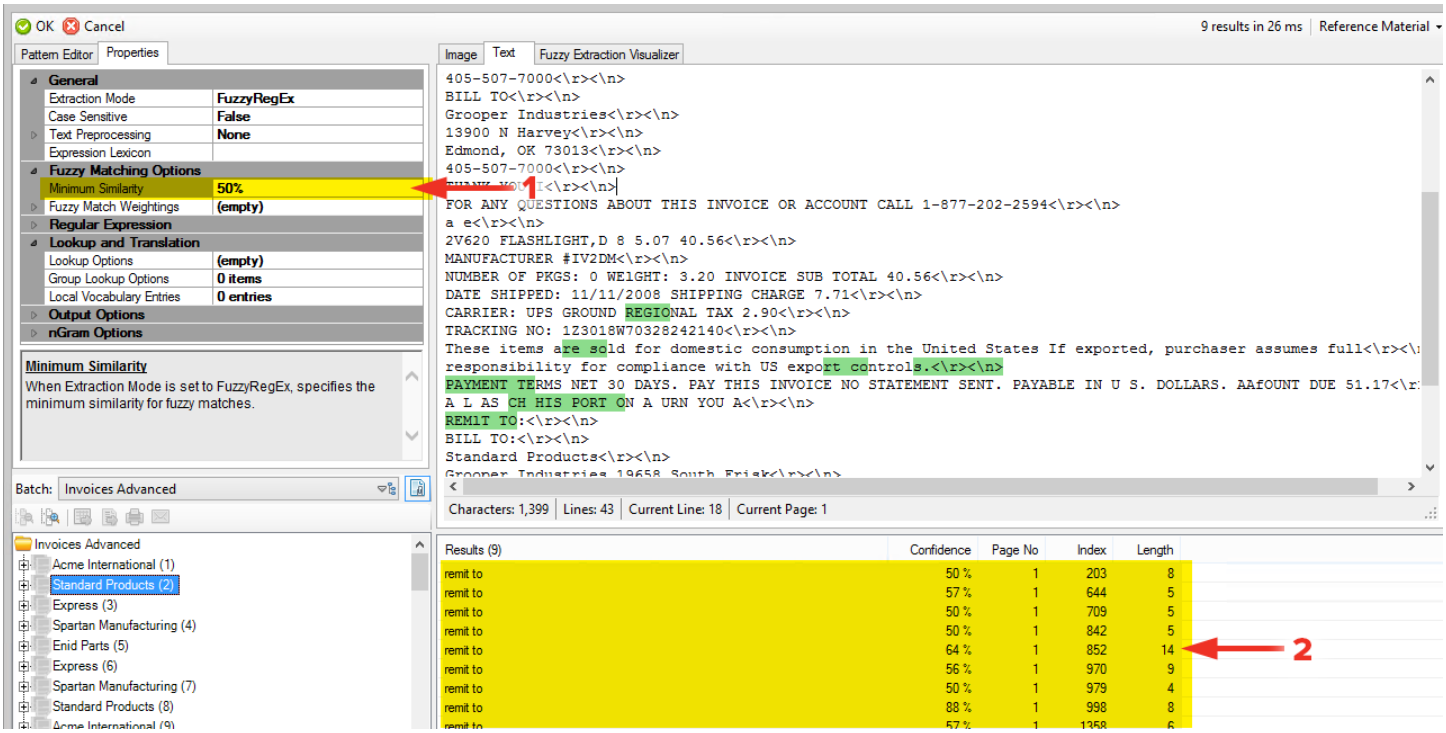
Notice also some of the information being displayed in the Results List View, mainly the Confidence of 88%. Dropping the Minimum Similarity from the 90% threshold to this now 85% allowed this match and translation to occur.

So why is the confidence of this returned result 88%? Consider that the length of the string our RegEx pattern is looking for happens to be 8 characters long. One of the characters in this string was mis-OCRed as a lower case `l` but returned from the FuzzyRegEx as an `i`. If we consider this one missed character as a percentage of the 8 being sought, it is 12%. Therefore, logically, having translated that one character, the percentage similarity drops from 100% to 88%.



STEP 9 – FURTHER UNDERSTANDING OF FUZZYREGEX

While 85% isn't necessarily a low threshold to set, we want the Minimum Similarity to be as high as possible. The reasoning behind this is to avoid false positives. **(1)** Lower the Minimum Similarity to 50% and **(2)** notice all the garbage that starts to be considered a match based on that low similarity (set it back to 85% when you're done experimenting.)

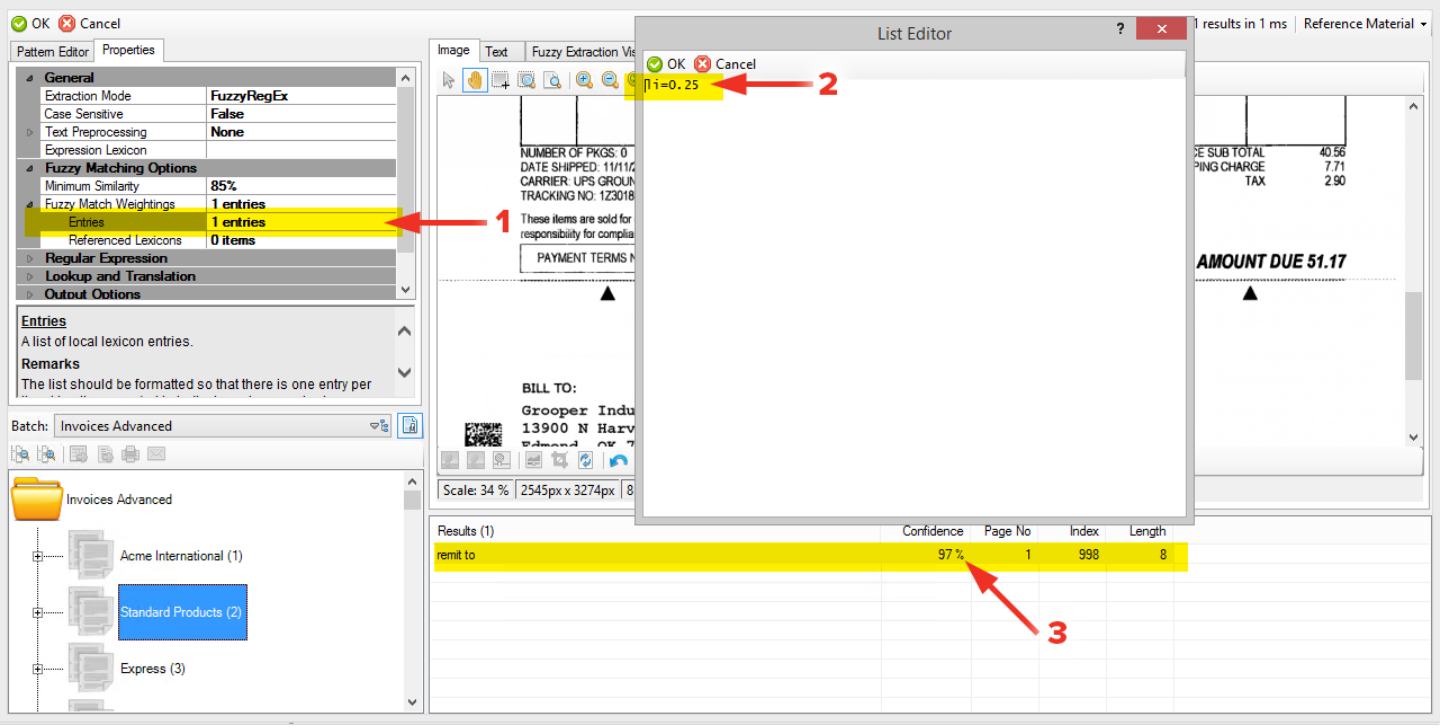


STEP 10 – FUZZY MATCH WEIGHTINGS

To keep confidence in desired results high, and as a result, keep the Minimum Similarity as high as possible to avoid false positives, **Grooper** has an excellent solution called Fuzzy Match Weightings. **(1)** Expand Fuzzy Match Weightings to reveal its properties: Entries and Referenced Lexicons. We'll focus on Entries for now. Click on the Entries property then click the ellipsis button to open a List Editor window. **(2)** Type the following in the List Editor, then click OK:

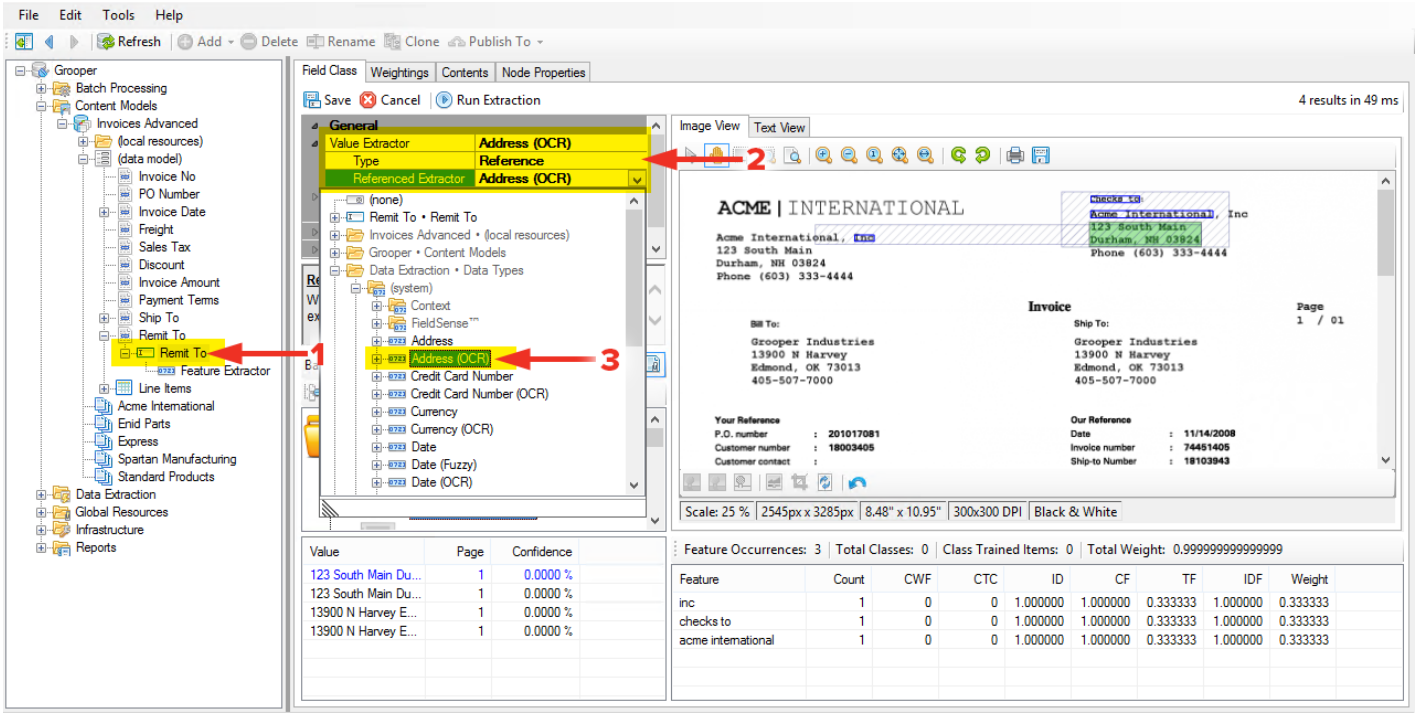
```
li=0.25
```

This Entries list uses a very specific syntax. The first character in this string is the incorrect character (keep in mind the syntax of this entries list is case sensitive.) The second is what you want the incorrect character to be translated to. Finally, the equals sign followed by the decimal number is saying what percentage of a whole cost this specific translation will be. So, in the **`remit to`** pattern the cost of the translation for this one character has been 12% (putting the confidence at 88%), but we've now defined the translation of the **`l`** to the **`i`** to be 25% of 12%, so now the cost will now only be 3% (therefore, **(3)** the confidence of this result will now be 97%.)



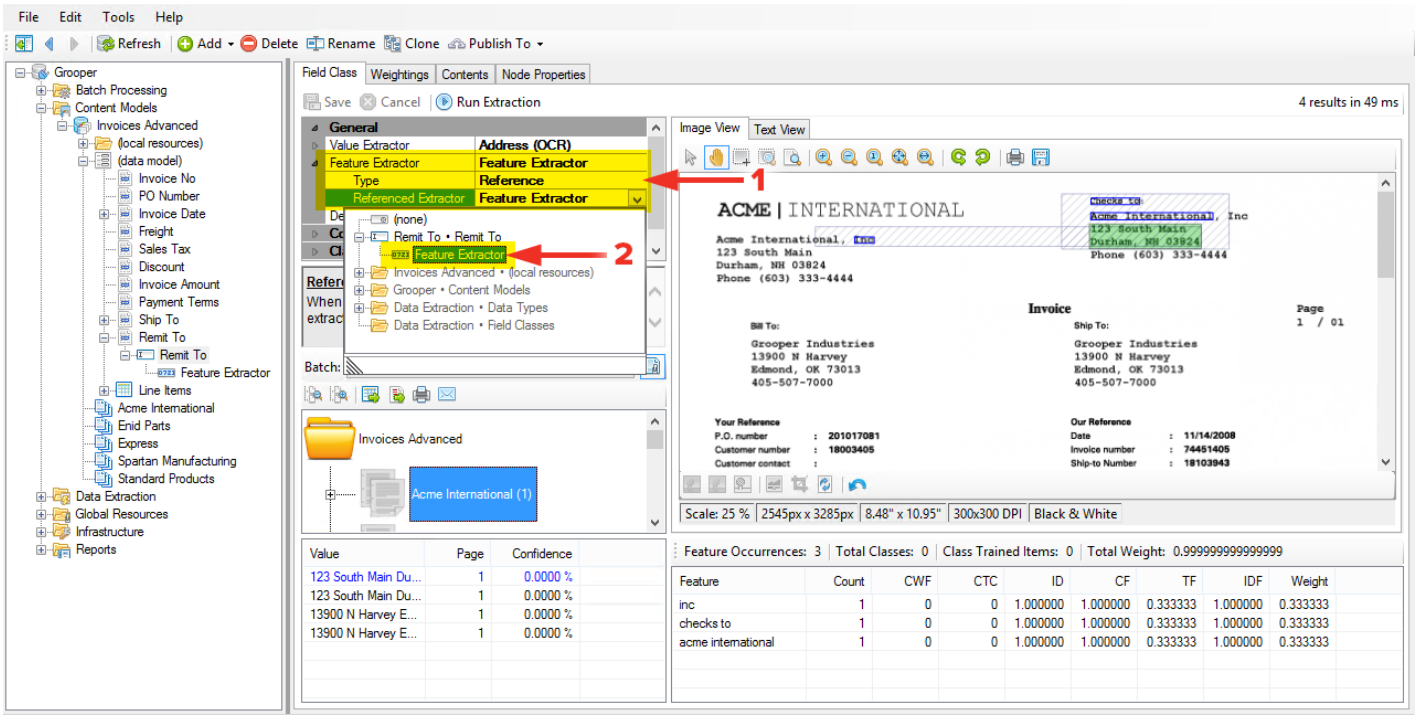
STEP 11 – SETTING THE VALUE EXTRACTOR

(1) Click back on the Remit To Field Class. **(2)** Set the Value Extractor type to Reference **(3)** and the Referenced Extractor to the Data Extraction • Data Types > (system) > Address (OCR) Data Type.



STEP 12 – SETTING THE FEATURE EXTRACTOR

(1) Set the Feature Extractor Type to Reference and **(2)** set the Referenced Extractor to the Feature Extractor Data Type that is a child object of the Remit To Field Class.

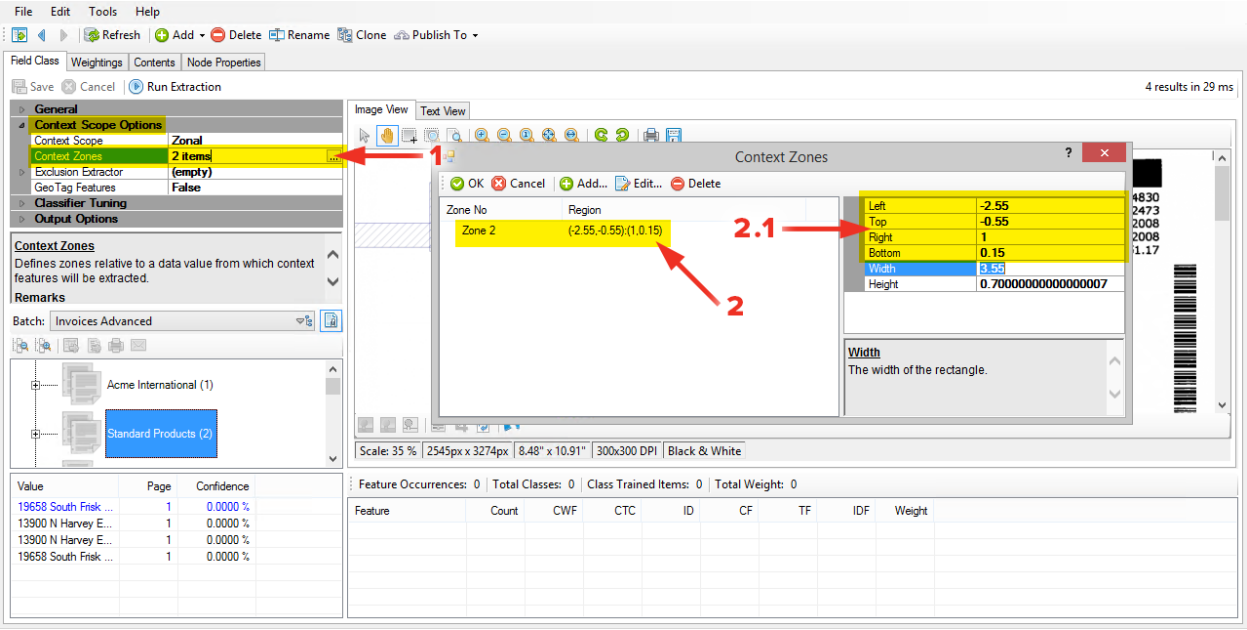


STEP 13 – ADJUSTING THE CONTEXT ZONES

In the Context Scope Options section, **(1)** select the Context Zones property and click the ellipsis button to bring up the Context Zones window. Delete Zone 1 and **(2)** set the Zone 2 dimensions to the following:

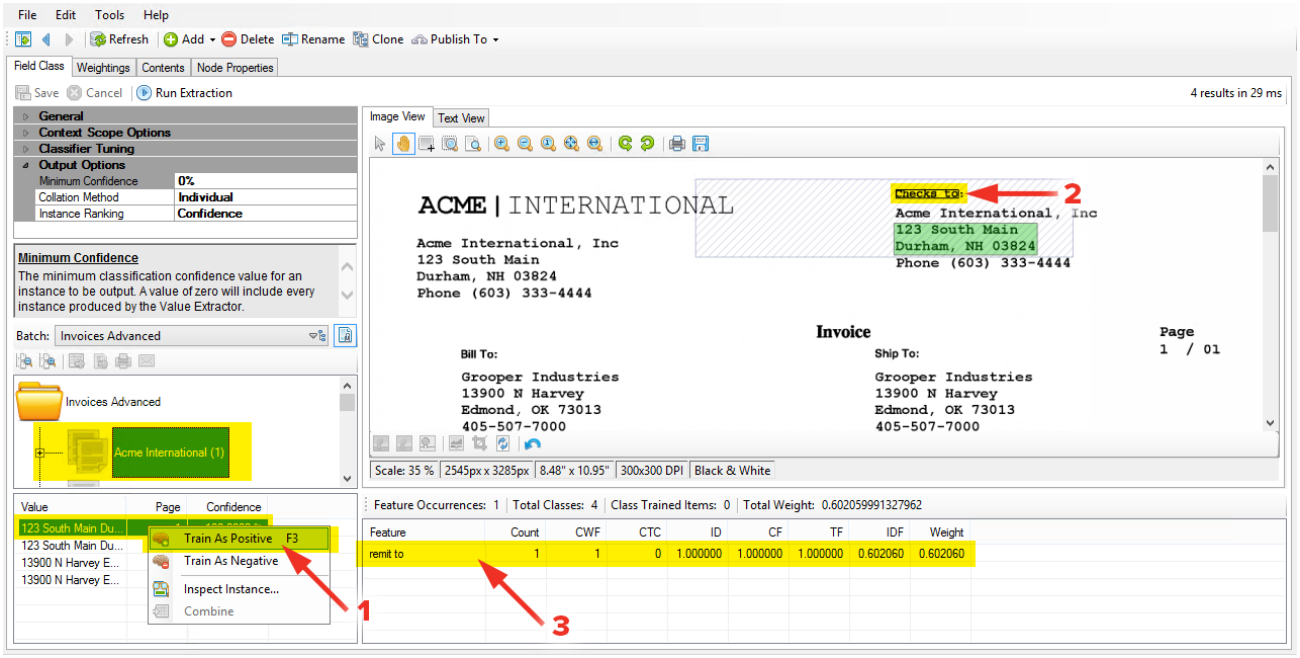
Zone 2 – Left: **-2.55** Top: **-0.55** Right: **1** Bottom: **0.15**

Click OK to close the Context Zones properties window, then Save and Run Extraction on the Field Class against the Acme International (1) document.



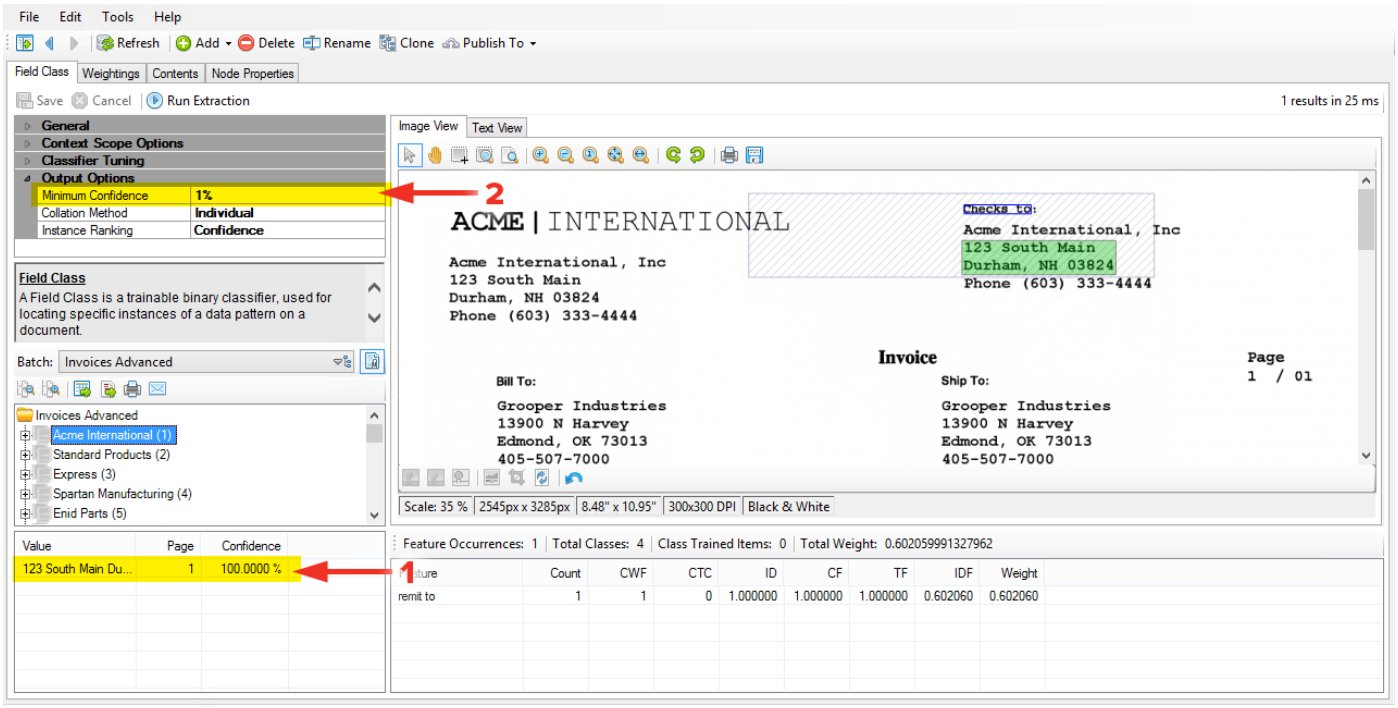
STEP 14 – TRAINING A FEATURE INSTANCE

Four values should be returned, but only one of them will capture a feature. **(1)** Select the value that returns a feature, and **(2)** notice that Checks To in the image has the blue box drawn around it, **(3)** but the Feature Occurrence that's returned is remit to. Train this instance positively.



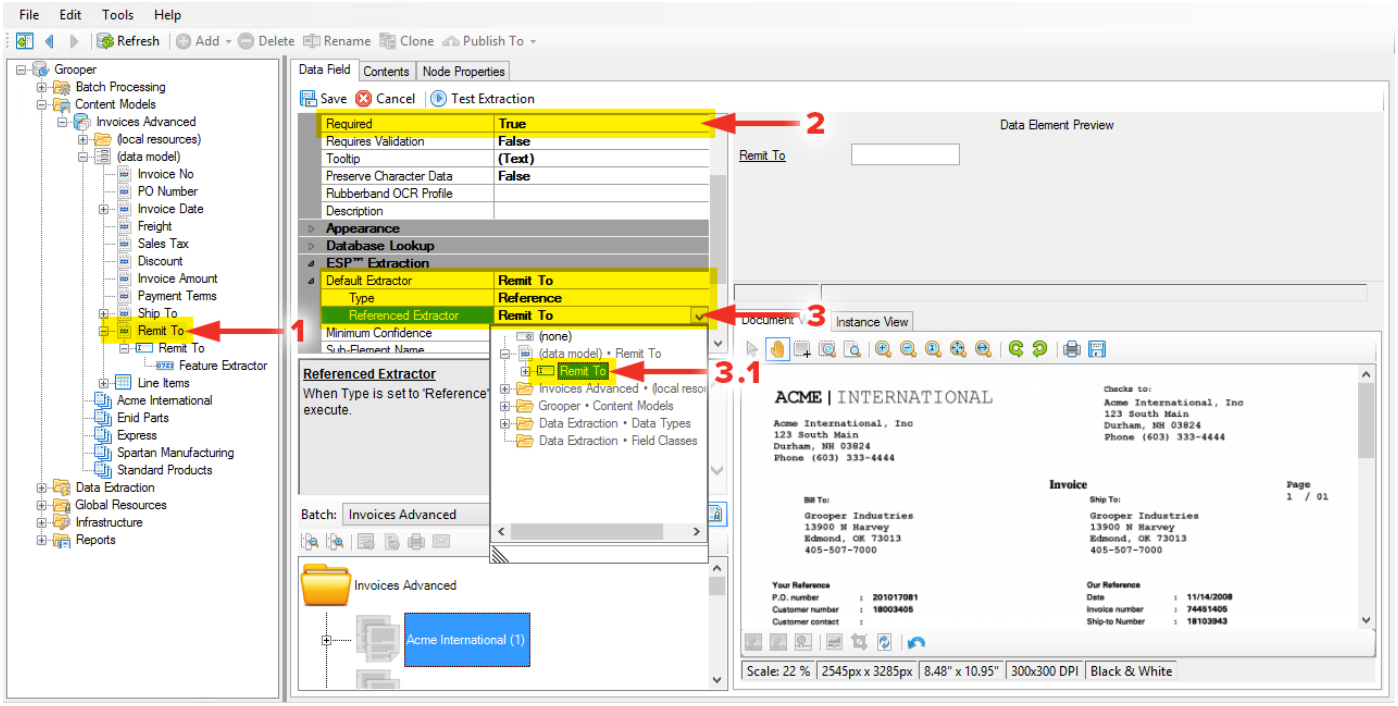
STEP 15 – SETTING MINIMUM CONFIDENCE

Because there is only one feature trained, and **(1)** it is being returned at 100% confidence, **(2)** the Minimum Confidence can be set to anything above 0% and all unwanted results will be trimmed.



STEP 16 – SETTING PROPERTIES FOR THE REMIT TO DATA FIELD

(1) Select the Remit To Data Field and **(2)** set its Required property to True. **(3)** Set the Default Extractor Type to Reference and set the Referenced Extractor to the Remit To Field Class.



THE REMAINING FIELD CLASSES – LEXICONS, CHEAT CODES, AND ARRAYS

There are a few fields left whose extraction is best suited by a Field Class. Having built out a few to gain an understanding of how they function, it's now a good time to introduce a new concept of creating a Feature Extractor that can work for many, if not all, future Field Classes.

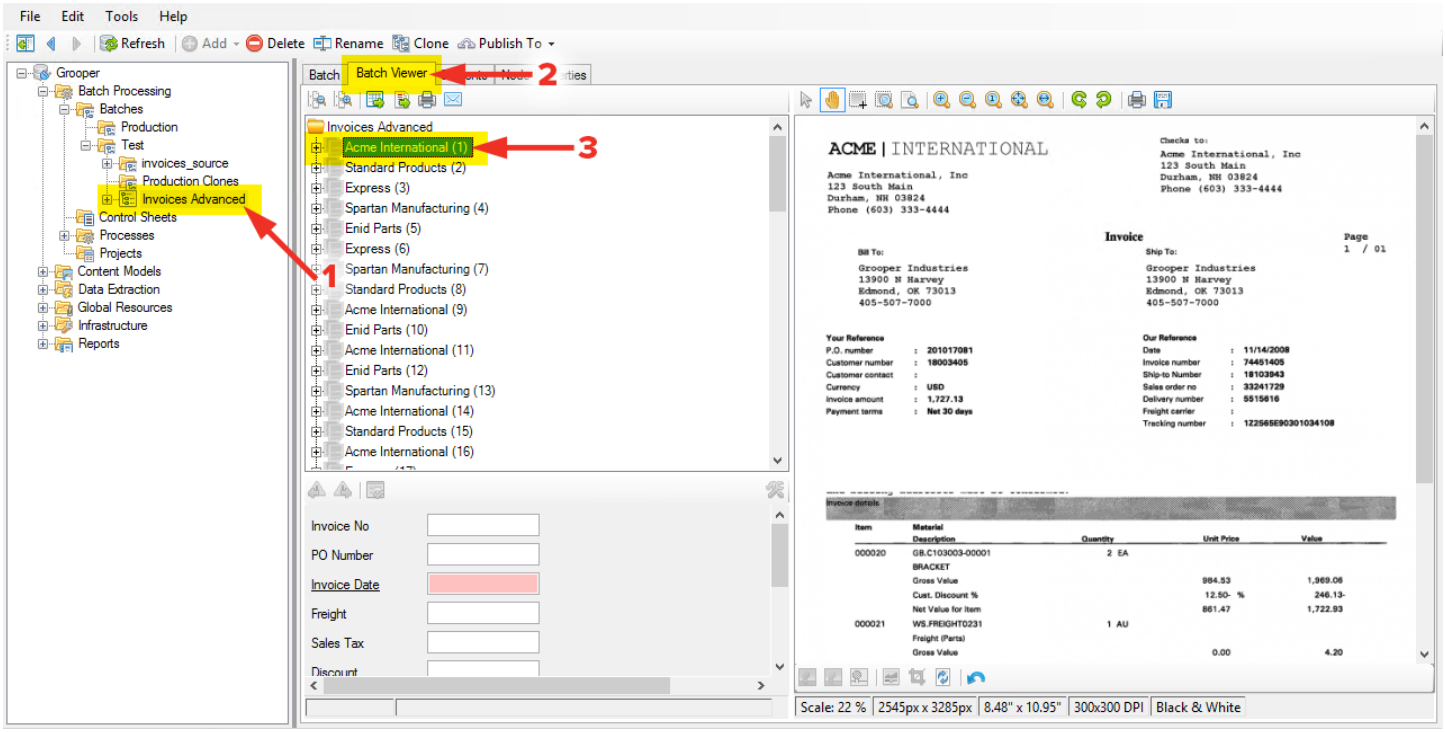
A Data Type will be built that will use several new concepts including OrderedArrays, a Lexicon with translations, and a very simple but highly effective RegEx pattern we lovingly referred to as the Cheat Code.

BUILDING A FEATURE LIST LEXICON

If you take a moment to look through the 5 main DocTypes and analyze the information on them, you can start to understand that basically every bit of relevant information on these documents has a label near it that defines what that bit of information is. If one were to take a moment and make a list of these labels he/she would have a master source of all the relevant Features that a Field Class could be trained against.

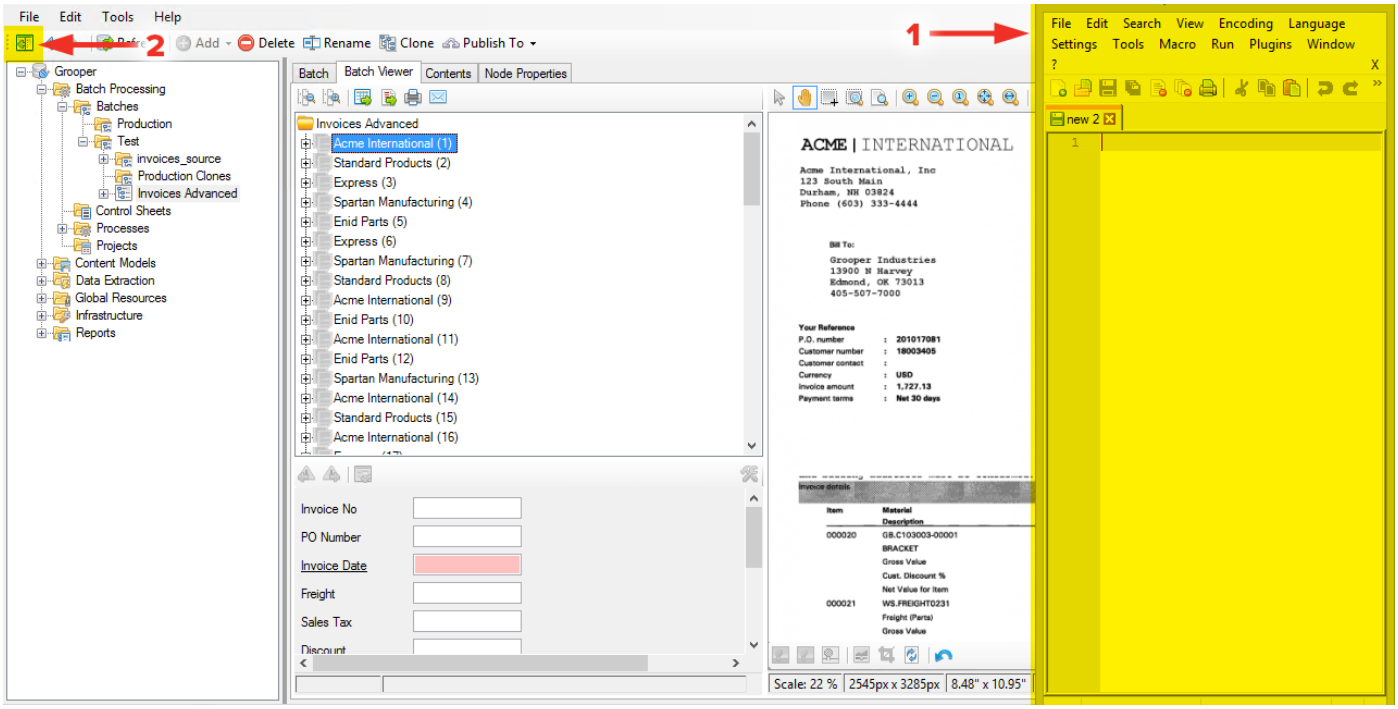
STEP 1 – VIEWING THE DOCUMENTS IN GROOPER

(1) Navigate to Grooper > Batch Processing > Batches > Test and select the Invoices Advanced batch.
(2) Click on the Batch Viewer tab and **(3)** select a document from the Batch Viewer.



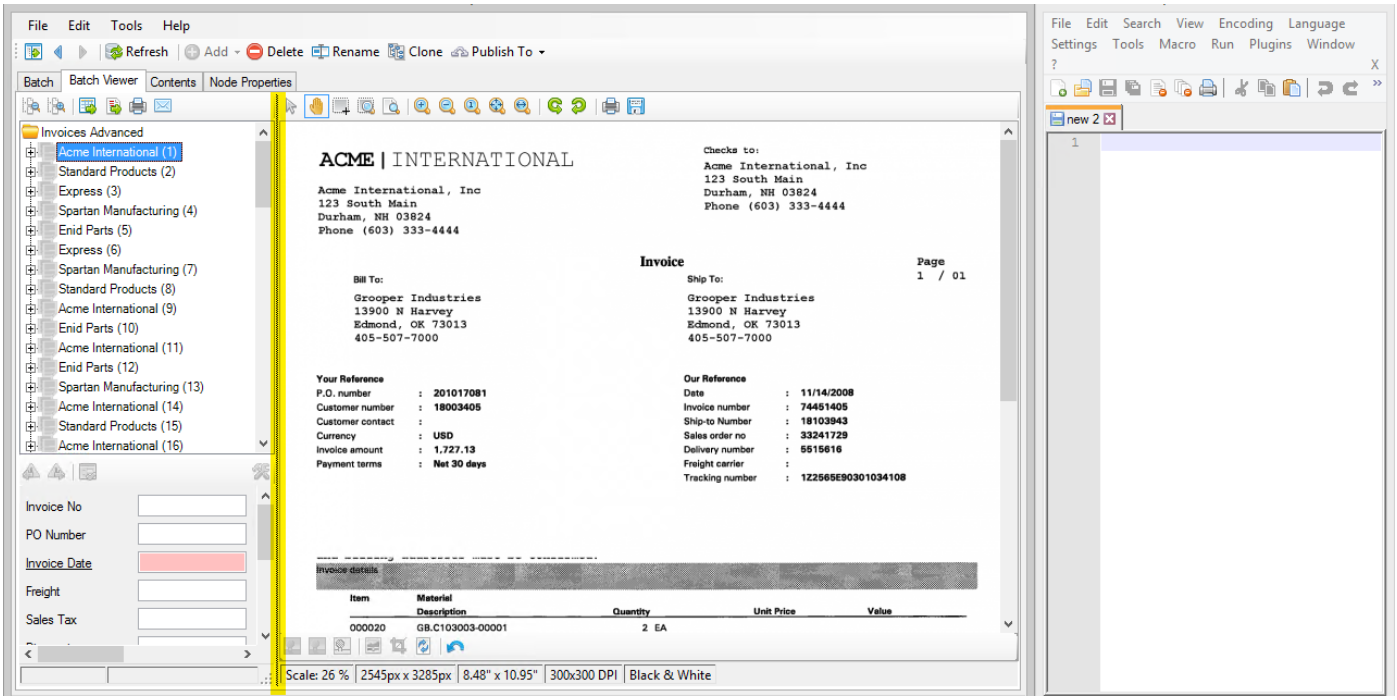
STEP 2 – OPEN A NOTEPAD APPLICATION AND HIDE TREE PANEL

In Windows, **(1)** open a Notepad program and anchor it to the side of the screen. You'll be writing multiple lines of short strings of information, so feel free to make it narrow, as you'll be sharing screen real estate with **Grooper**. In **Grooper**, **(2)** click the Hide Tree Panel button. Clicking the same button will bring it back as well.



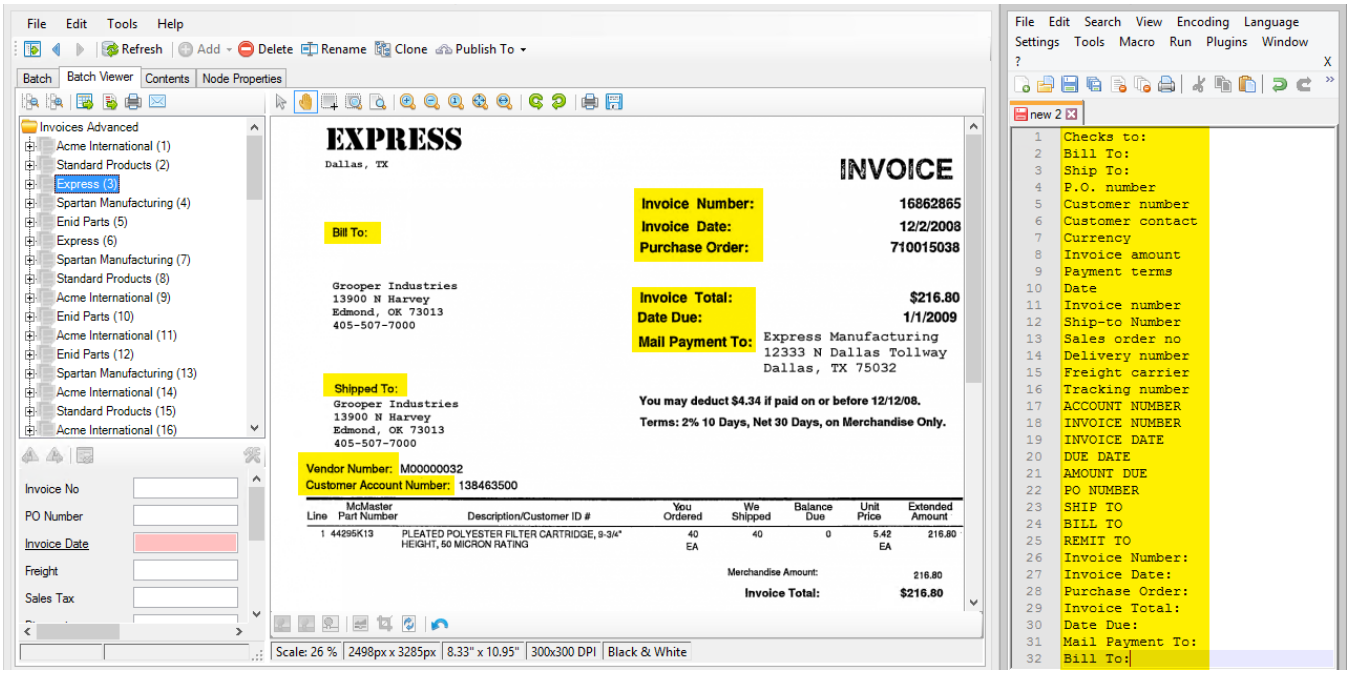
STEP 3 – ADJUSTING THE UI

Anchor **Grooper** to the other side of the screen, then move the panels around within the **Grooper** UI so that the Page Viewer allows easy reading of the documents.



STEP 4 – CREATING A LIST OF LABELS

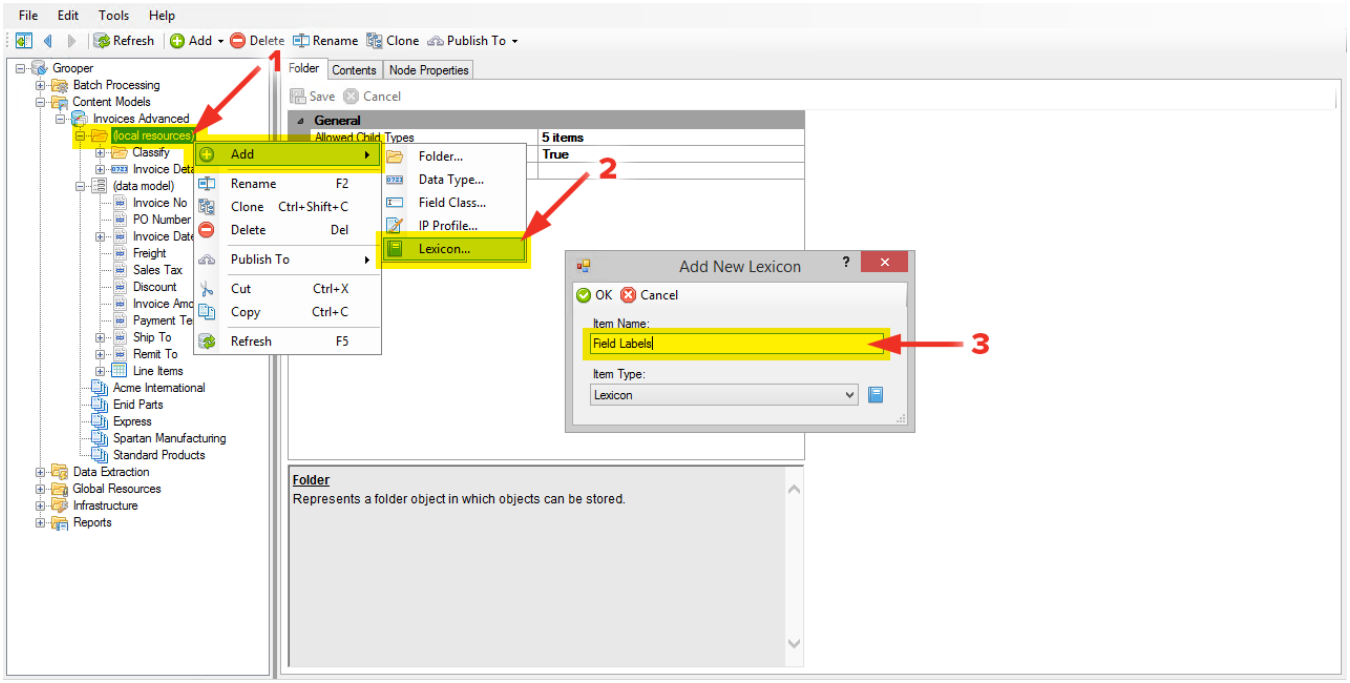
Look over the documents and begin to type into the Notepad application all the labels that define the fields of information within the documents. Don't worry about duplicate entries or what order they're in. Be sure to spell exactly as they appear on the documents, including spacing and special characters. Considering that **Grooper** can be configured for case sensitivity, it's best practice to also type the correct casing.



There are only 5 main DocTypes to go through, and when you get to the Enid Parts (5), don't bother listing the labels that are stacked on top of one another like Order Number and Purchase Order Number and so forth. While you read them as a single unit, they're on separate lines and aren't returned together. This will be clear later when we capture them with arrays.

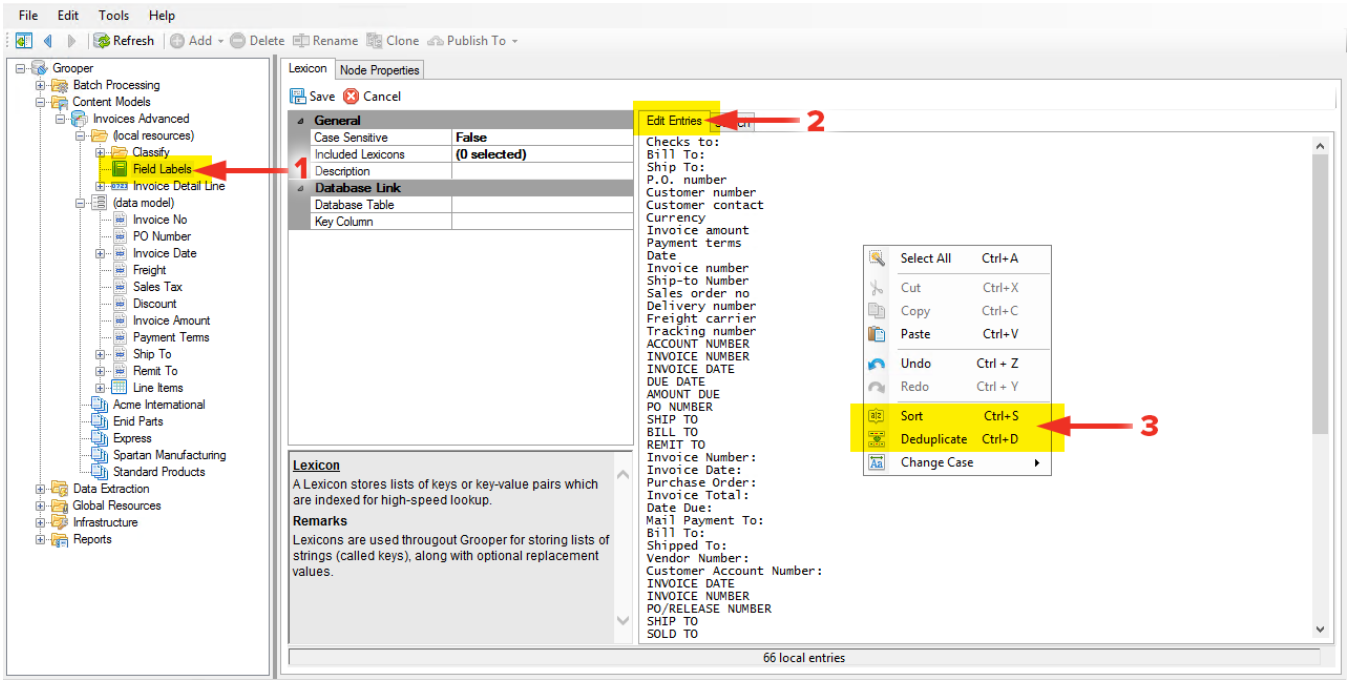
STEP 5 – CREATING A GROOPER LEXICON

When you've gotten through all 5 DocTypes, copy the list to the clipboard. Back in **Grooper**, **(1)** navigate to Grooper > Content Models > Invoices Advanced and **(2)** right-click on the (local resources) folder node and Add > Lexicon... **(3)** Name it Field Labels.



STEP 6 – CREATE LEXICON ENTRIES AND SORT

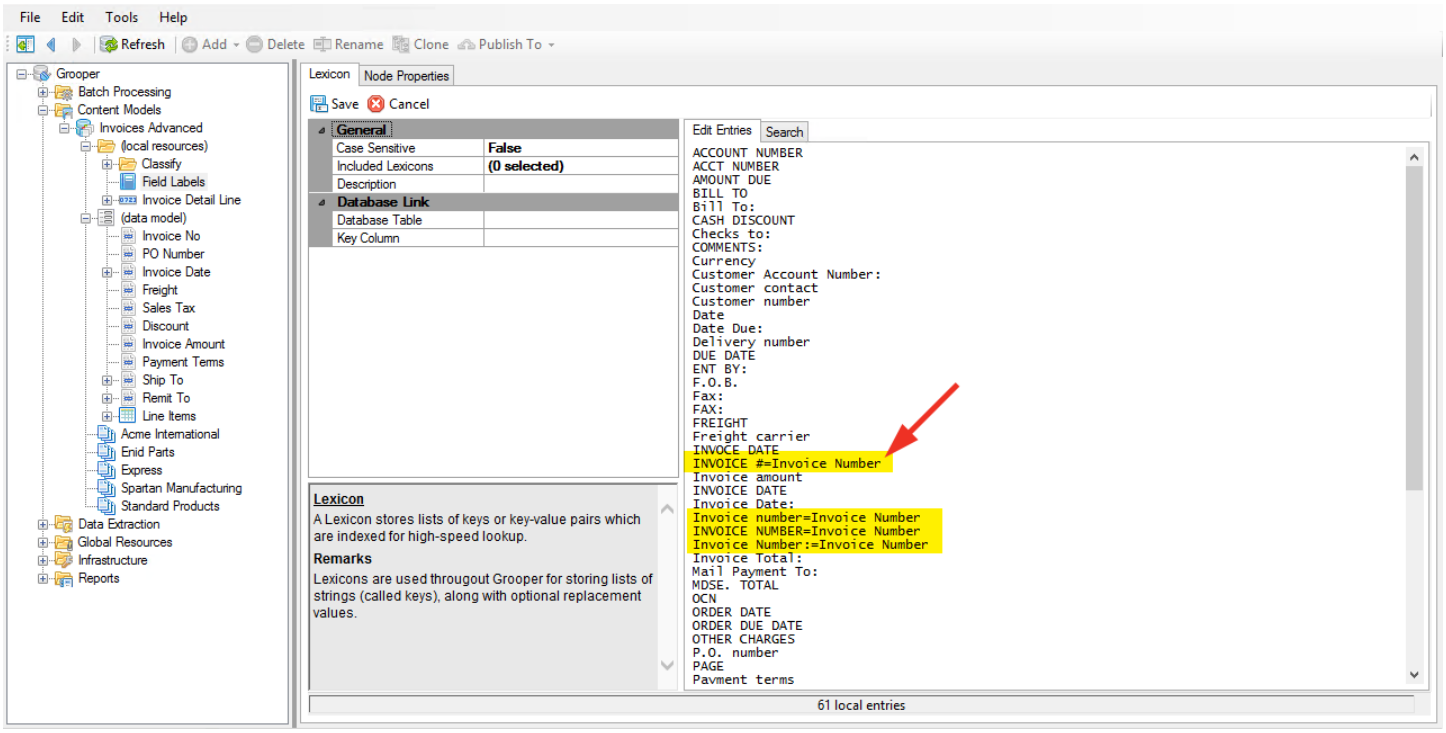
(1) With the Field Labels Lexicon selected, **(2)** paste the contents of the clipboard into the Edit Entries area. This should be all the things you typed in the Notepad program. With entries in the list, **(3)** right-click and use the object commands for Deduplicate as well as Sort. This will eliminate duplicate entries and alphabetize the list.



STEP 7 – ADDING TRANSLATIONS

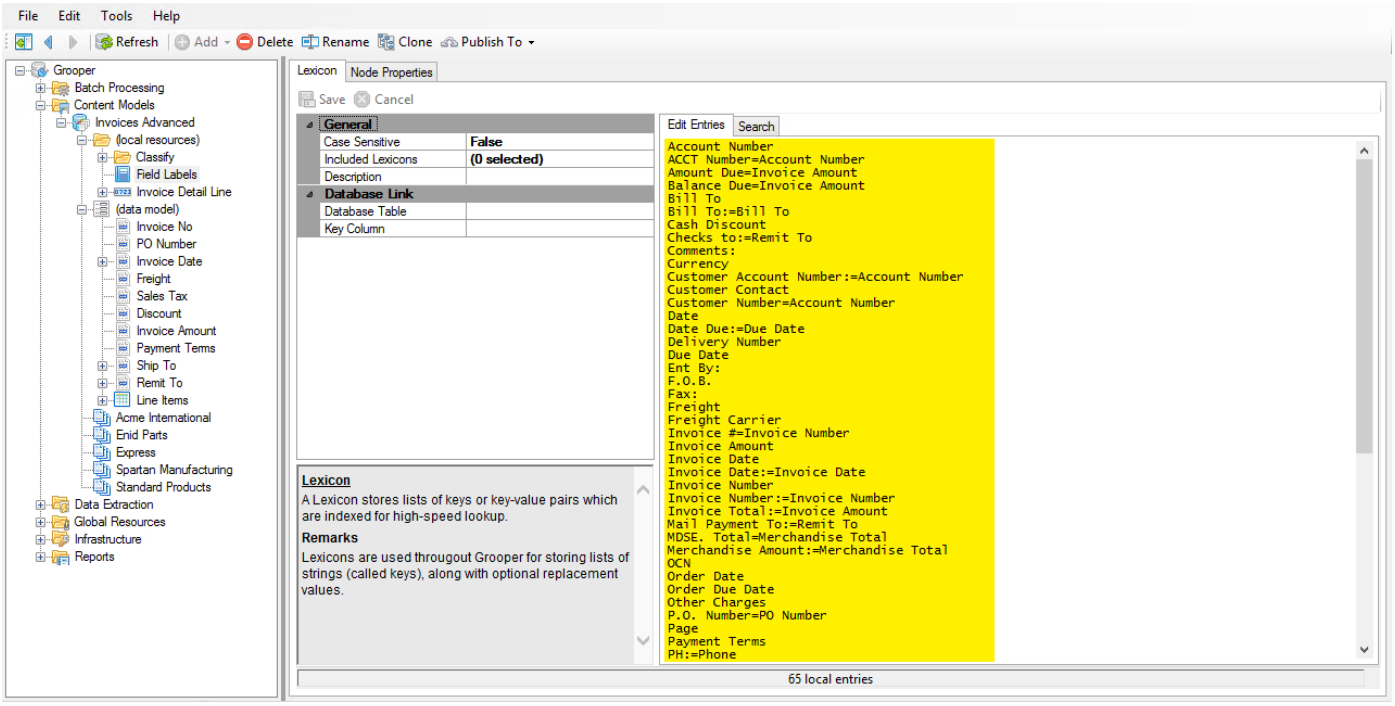
Notice that there are entries for labels from the different doc types that are called different things. For example invoice number is listed as Invoice number, INVOICE NUMBER, Invoice Number:, and INVOICE #. Lexicons can allow multiple entries to equate to the same output, similar in a fashion to using an Output Format in a Data Type or Data Format RegEx pattern. For the multiple entries of invoice number, put an equals sign after them and type how you'd like those values translated. For example:

INVOICE #=Invoice Number



STEP 8 – GIVING YOU THE LIST

While it was important to understand the process of typing out the labels and so forth, the list has also been provided below. This list includes all translations that will be leveraged by the Data Model moving forward. Providing the list will also make sure what you have is in sync with what is expected from the remainder of this document. To save space on this document, I've put the list into a few columns of a table. Just copy the contents from each column and paste them in succession in your Lexicon. Overwrite what you already have in the Lexicon as well.



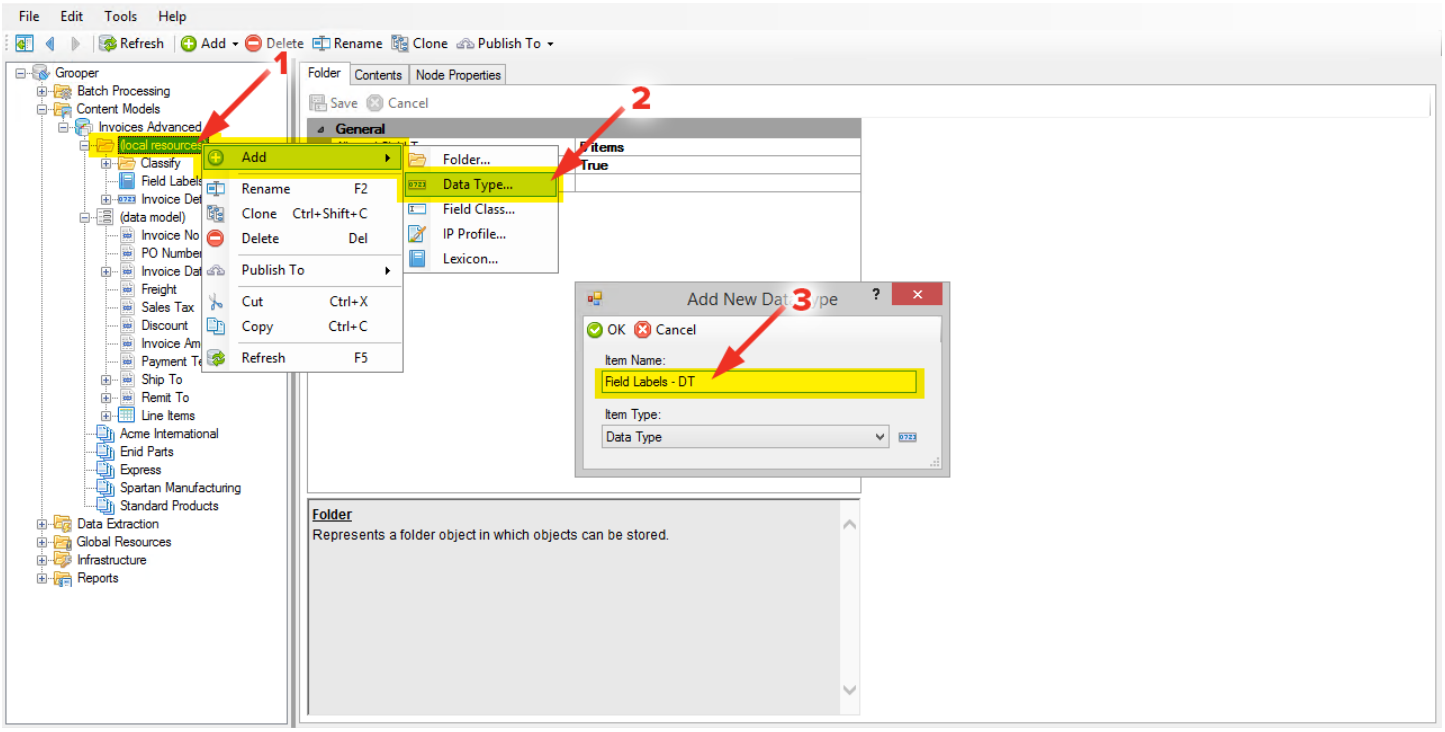
Account Number
ACCT Number=Account Number
Amount Due=Invoice Amount
Balance Due=Invoice Amount
Bill To
Bill To:=Bill To
Cash Discount
Checks to:=Remit To
Comments:
Currency
Customer Account Number:=Account Number
Customer Contact
Customer Number=Account Number
Date
Date Due:=Due Date
Delivery Number
Due Date
Ent By:
F.O.B.
Fax:
Freight
Freight Carrier
Invoice #=Invoice Number
Invoice Amount
Invoice Date
Invoice Date:=Invoice Date
Invoice Number
Invoice Number:=Invoice Number
Invoice Total:=Invoice Amount
Mail Payment To:=Remit To
MDSE. Total=Merchandise Total
Merchandise Amount:=Merchandise Total
OCN
Order Date
Order Due Date
Other Charges
P.O. Number=PO Number
Page
Payment Terms
PH:=Phone
Phone:=Phone
PO Number
PO NUMBER:=PO Number
PO/Release Number=PO Number
PO / Release Number=PO Number
Purchase Order:=PO Number
Remit To:=Remit To
Sales Order No
Sales Tax
Ship=Ship To
Ship Date
Ship To
Ship To:=Ship To
Ship Via
Shipped To:=Ship To
Shipped Via:
Ship-To Number
Sold To
Taken By:
Terms
Total Due=Invoice Amount
Total Packages:
Total Weight:
Tracking Number
Vendor Number:

CREATE THE DATA TYPE THAT WILL LEVERAGE THE FIELD LABELS LEXICON

The Field Labels Lexicon by itself doesn't accomplish anything. It needs to be leveraged by an extractor for its purpose to become apparent.

STEP 1 – CREATE NEW DATA TYPE IN LOCAL RESOURCES

(1) Right-click the (local resources) folder node and **(2)** Add > Data Type... **(3)** Name it Field Labels – DT.

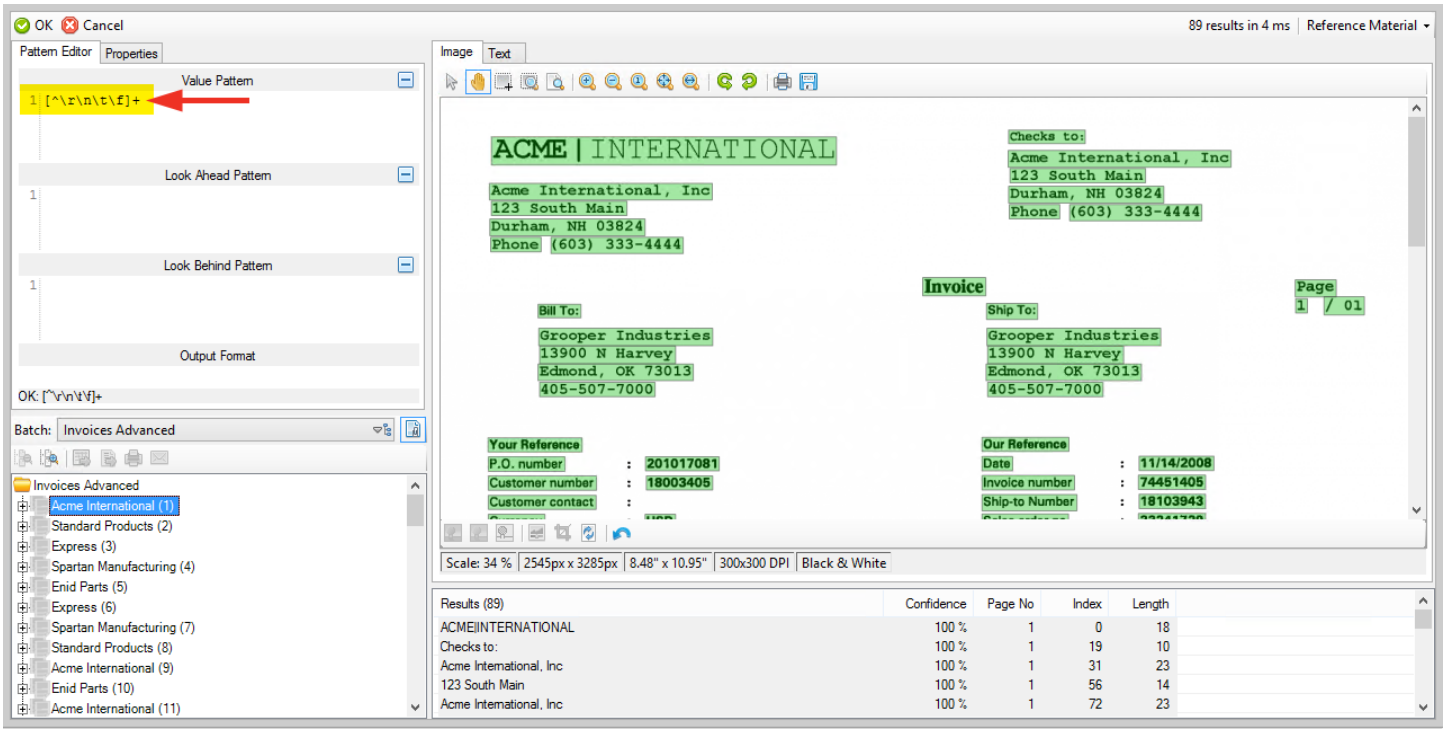


STEP 2 – CHEAT CODE REGEX

With the Data Type made, click on the ellipsis button for the Pattern property to pull up a Pattern Editor. In the Properties tab, in the General section, within Text Preprocessing, be sure Tab Marking is set to Enabled. Use the following pattern:

```
[^\r\n\t\f]+
```

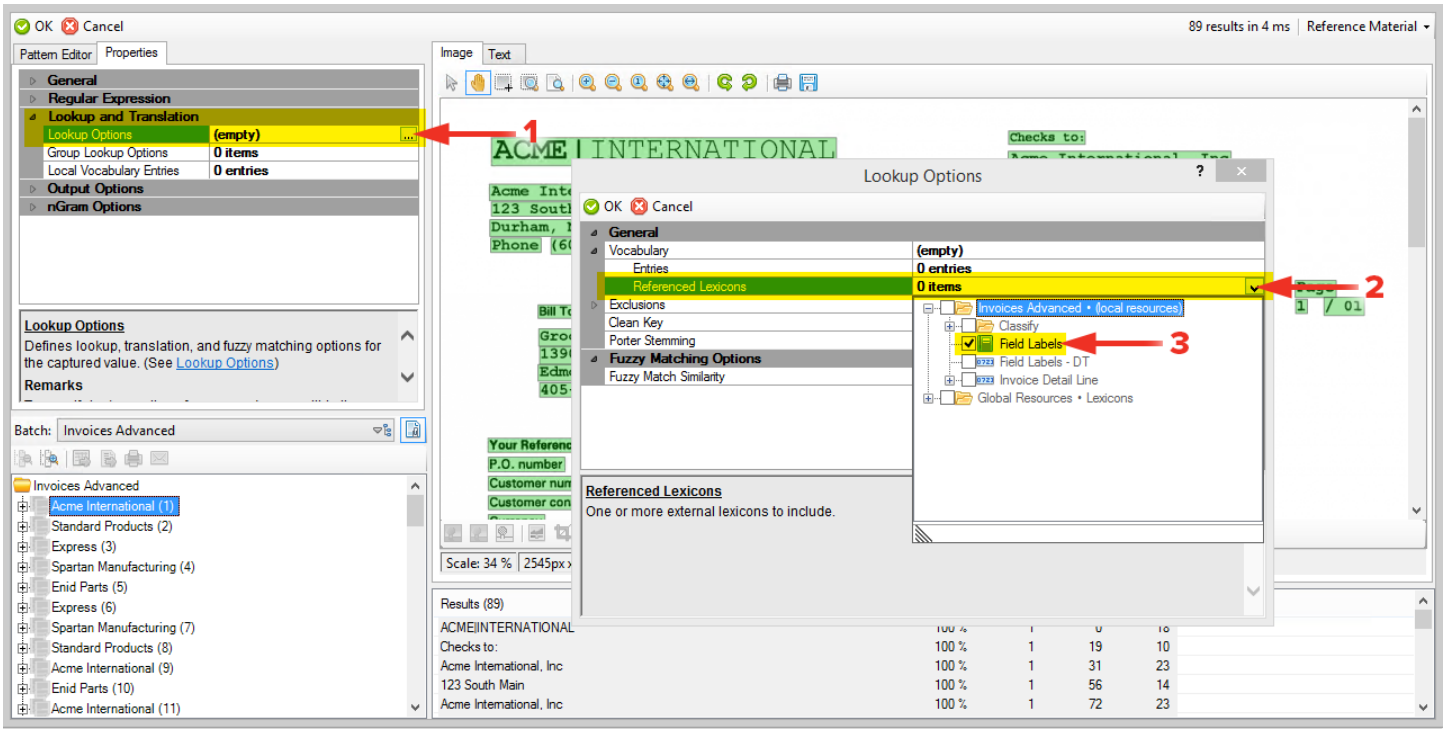
This very simple pattern is lovingly referred to as the Cheat Code, because it very effectively grabs every chunk of logical data on a document. The character set **[ ]** starting with the caret **^** determines not, then following it is carriage returns **\r**, new line feeds **\n**, tabs **\t**, and form feeds **\f** (these are the invisible characters at the end of documents that tell a document to continue to the next page.) So NOT any of those things, and the one or more quantifier **+** says as many of those in a row as you can get until you get to one of the things listed to not get.

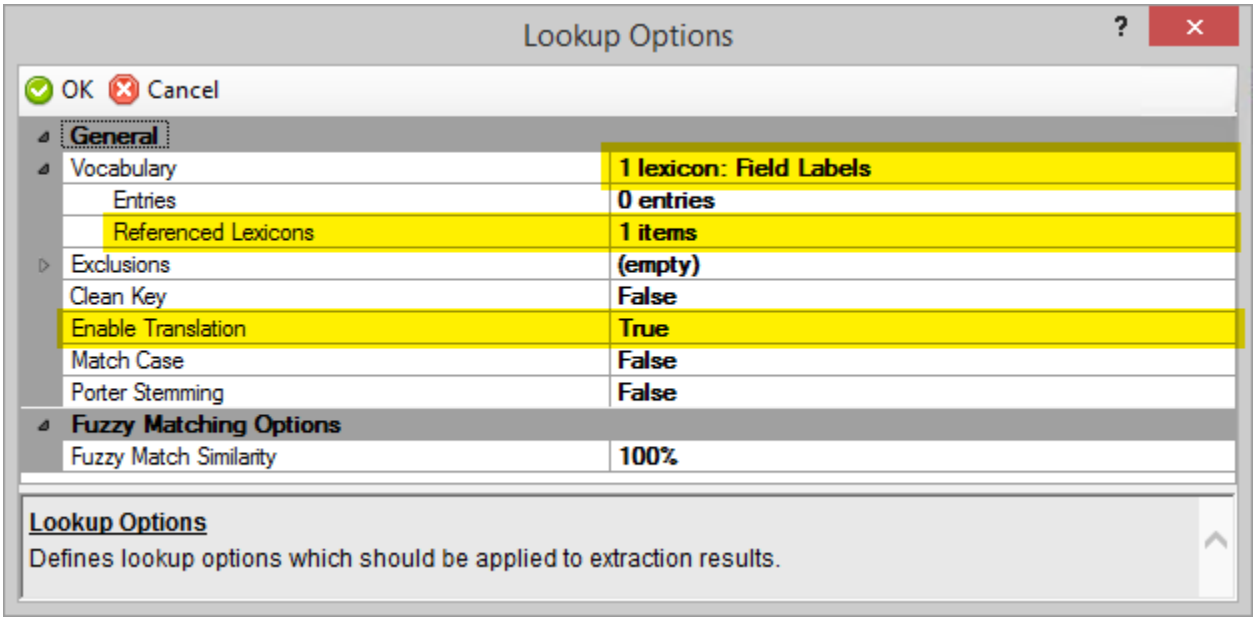


STEP 3 – LOOKUP AND TRANSLATION

Now that our RegEx pattern is grabbing every logical chunk of data, let's filter that out by comparing it against the Field Labels Lexicon that was just made. Click on the Properties tab and expand the Lookup and Translation section. **(1)** Select the Lookup Options property and click the ellipsis button to bring up the Lookup Options window. In this window expand the Vocabulary section and **(2)** select the Referenced Lexicons property. Click the drop-down and within that menu **(3)** put a check mark in the box for the Field Labels Lexicon within Invoices Advanced • (local resources). Finally, set the Enable Translation property to True.

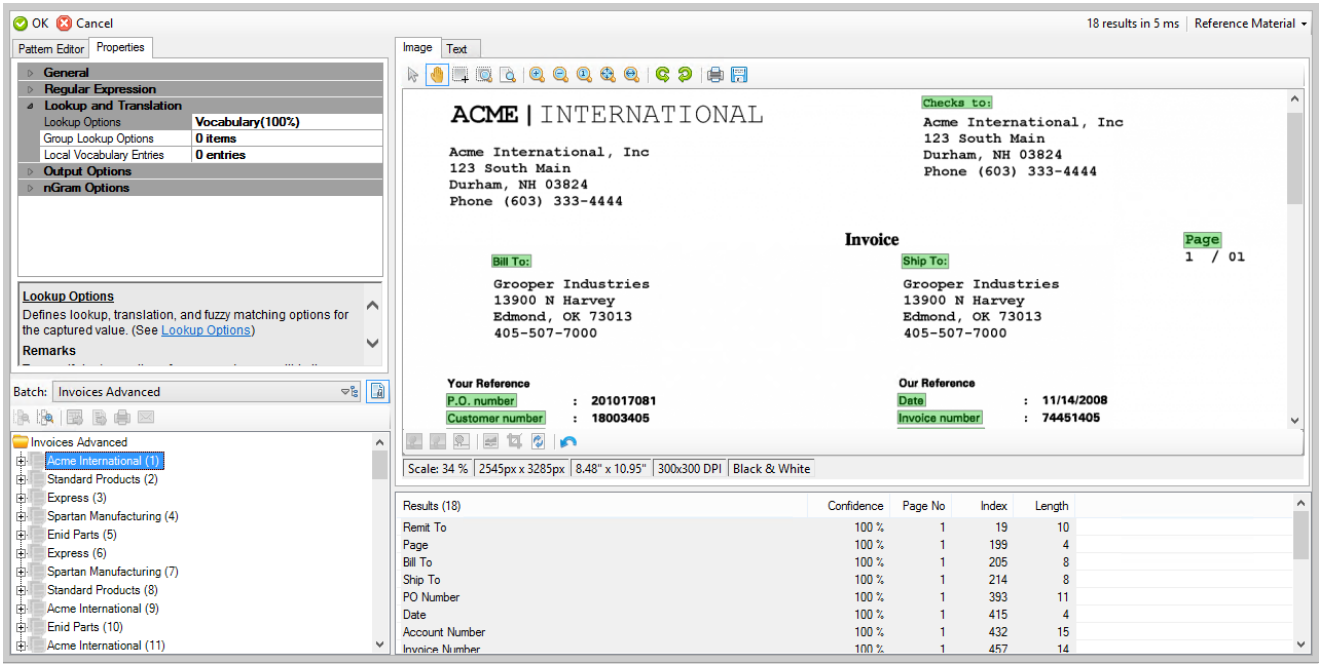
This will limit the Cheat Code RegEx pattern to only return results that are listed within the Lexicon, as well as allow the translations that were built into the Lexicon to function.





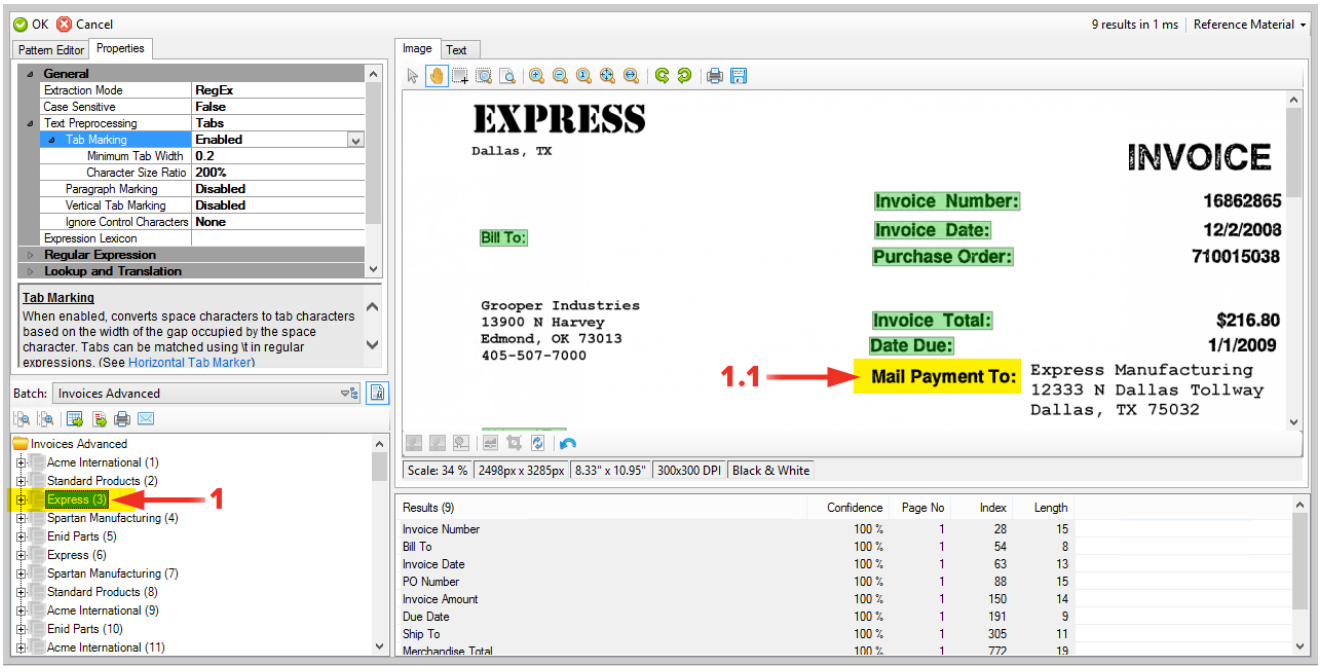
STEP 4 – VIEWING RESULTS

With the Acme International (1) document selected in the Batch Viewer, notice now that the results being returned are only those listed within the Lexicon. You can also see that translation is working in that **`Checks to:`** is being returned as **`Remit To`**, and **`P.O. Number`** is being returned as **`PO Number`**. Feel free to click through more documents within the Batch Viewer to see those results as well.



STEP 5 – PROBLEM WITH MAIL PAYMENT TO:

In browsing the other documents you'll soon come to a problem. **`(1)`** On Express (3), **`Mail Payment To:`** is a desired feature, and listed in the Field Labels Lexicon, but it's not being returned. This is happening because the default settings of Tab Marking are preventing the space between the strings **`Mail Payment To:`** and **`Express Manufacturing`** from being read as a tab.



STEP 6 – ADJUSTING TAB MARKING SETTINGS

(1) Click the Text tab and **(2)** take note of the string:

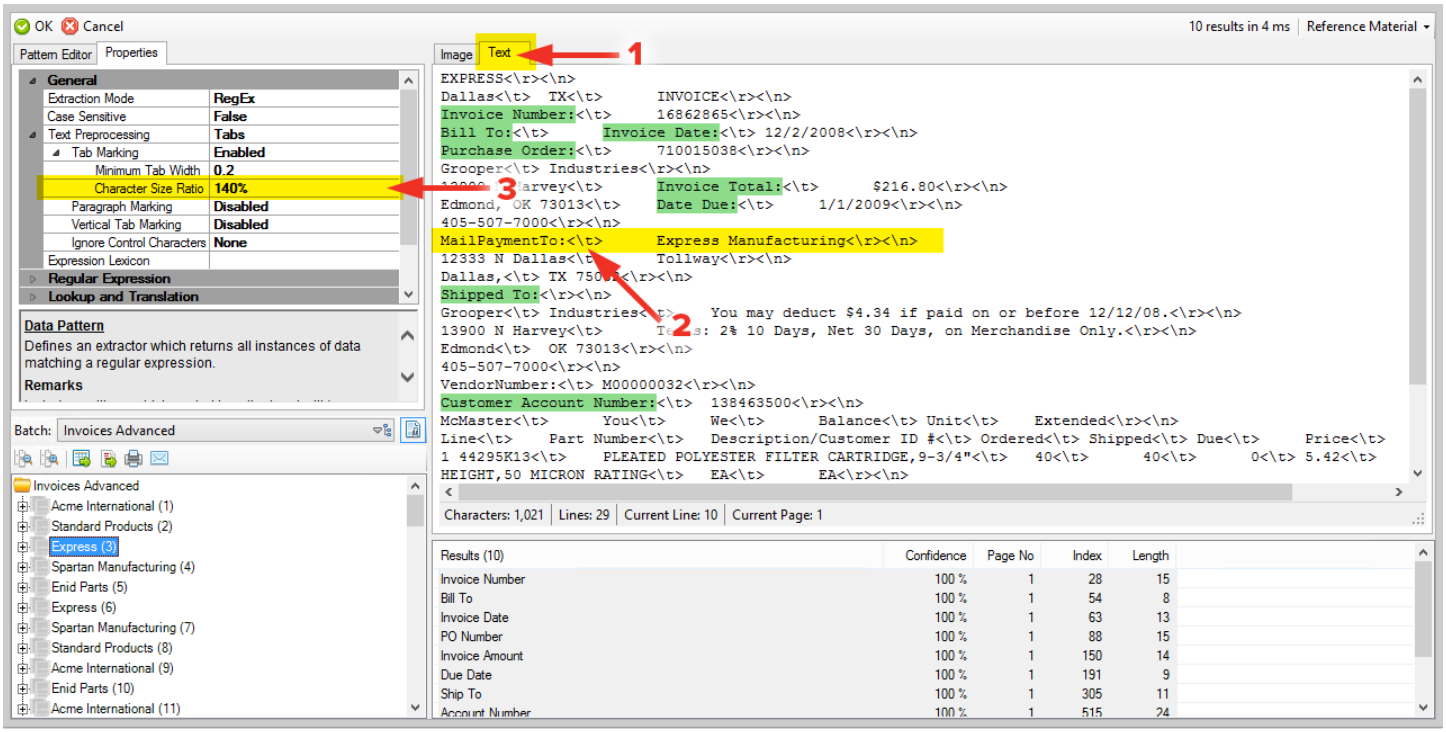
`MailPaymentTo: Express Manufacturing<\r><\n>`.

So the Cheat Code RegEx pattern would successfully return **`MailPaymentTo: Express Manufacturing`** as a value, but this is not an entry in the Field Labels Lexicon. Because the Cheat Code pattern looks for NOT tabs, carriage returns, new line feeds, and form feeds, **(3)** we need to adjust the Tab Marking settings so it will insert a tab character after **`MailPaymentTo:`**.

Character Size Ratio is the property to adjust to allow this to happen. The tooltip for this reads:

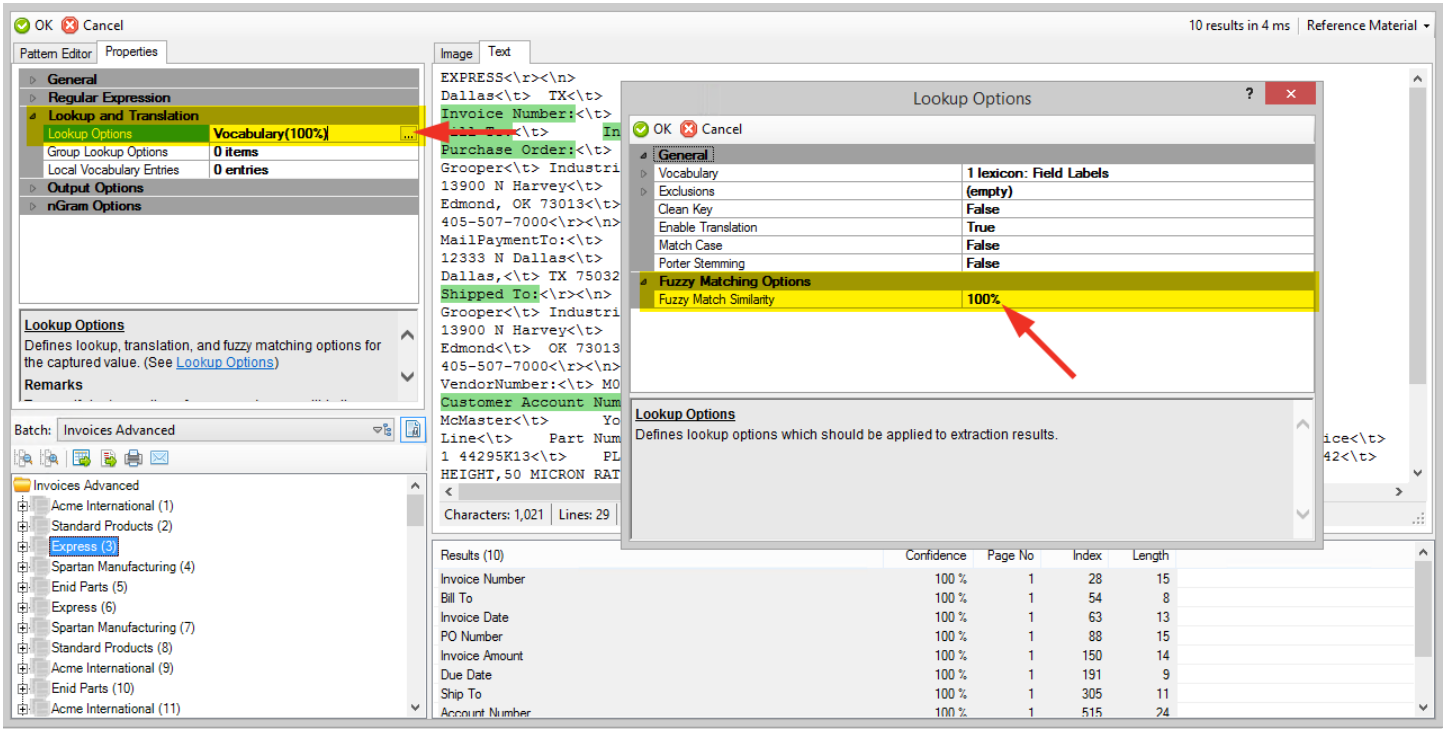
Space characters which have a horizontal width greater than the height of the previous character times this ratio will be converted to tab characters.

Currently the size of the space after **`MailPaymentTo:`** is not being read as a tab with the Character Size Ratio at 200%, but decreasing this to 140% will allow the space to be a tab.



STEP 7 – FUZZY MATCHING THE LIST

With the tab character successfully inserted, there's one more problem. The Field Labels Lexicon lists **`Mail Payment To:`** not **`MailPaymentTo:`**. The lack of spaces between the words here is a result of an OCR error, and needs to be compensated for. Back in the Lookup and Translation section, select the Lookup Options property again, and click the ellipsis button to get the Lookup Options window again. Notice in the Fuzzy Matching Options section that the property Fuzzy Match Similarity is at 100%, basically meaning Fuzzy Matching is off. This needs to be lowered to allow OCR errors to be translated properly.

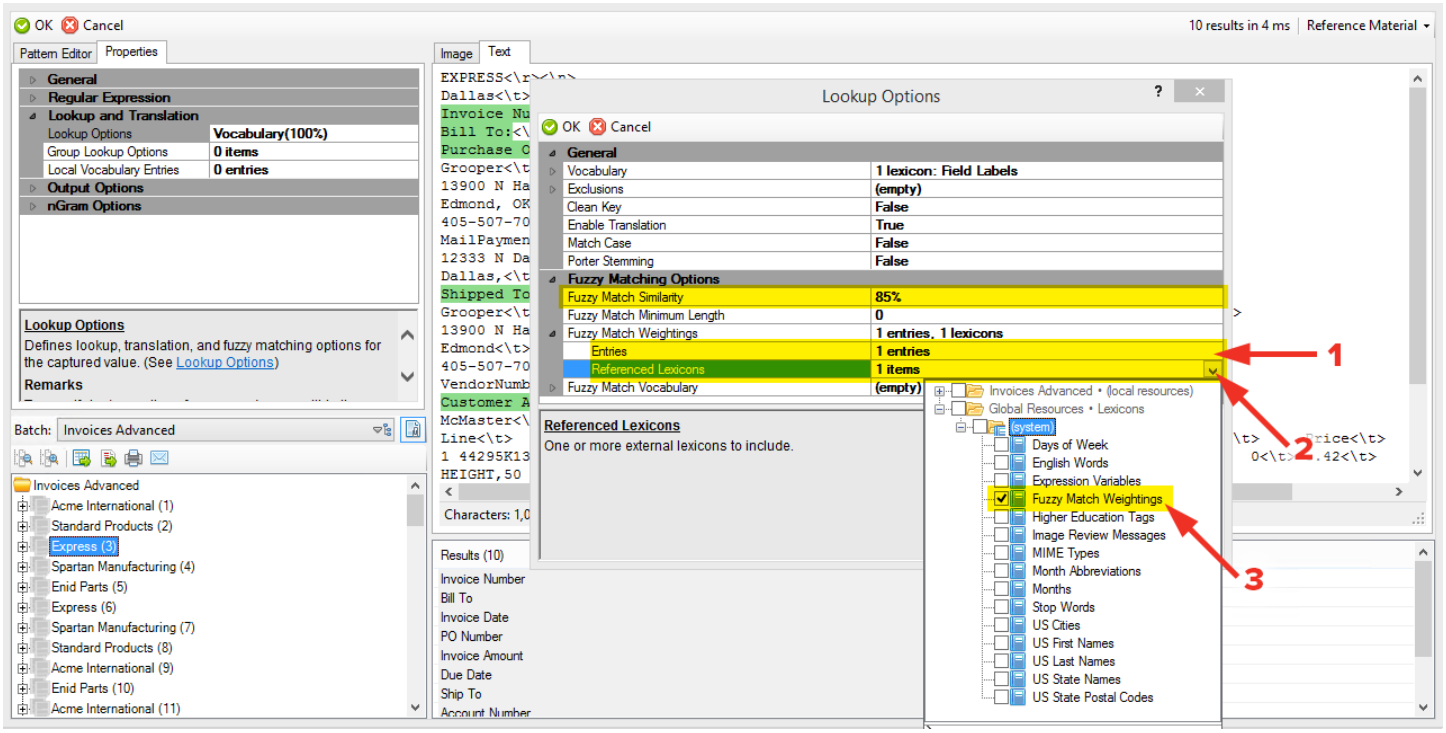


STEP 8 – LOWERING FUZZY MATCH SIMILARITY

(1) Set the Fuzzy Match Similarity property to 85% and new properties will be exposed. **(2)** Add the following to the Entries list from within the Fuzzy Match Weightings properties area:

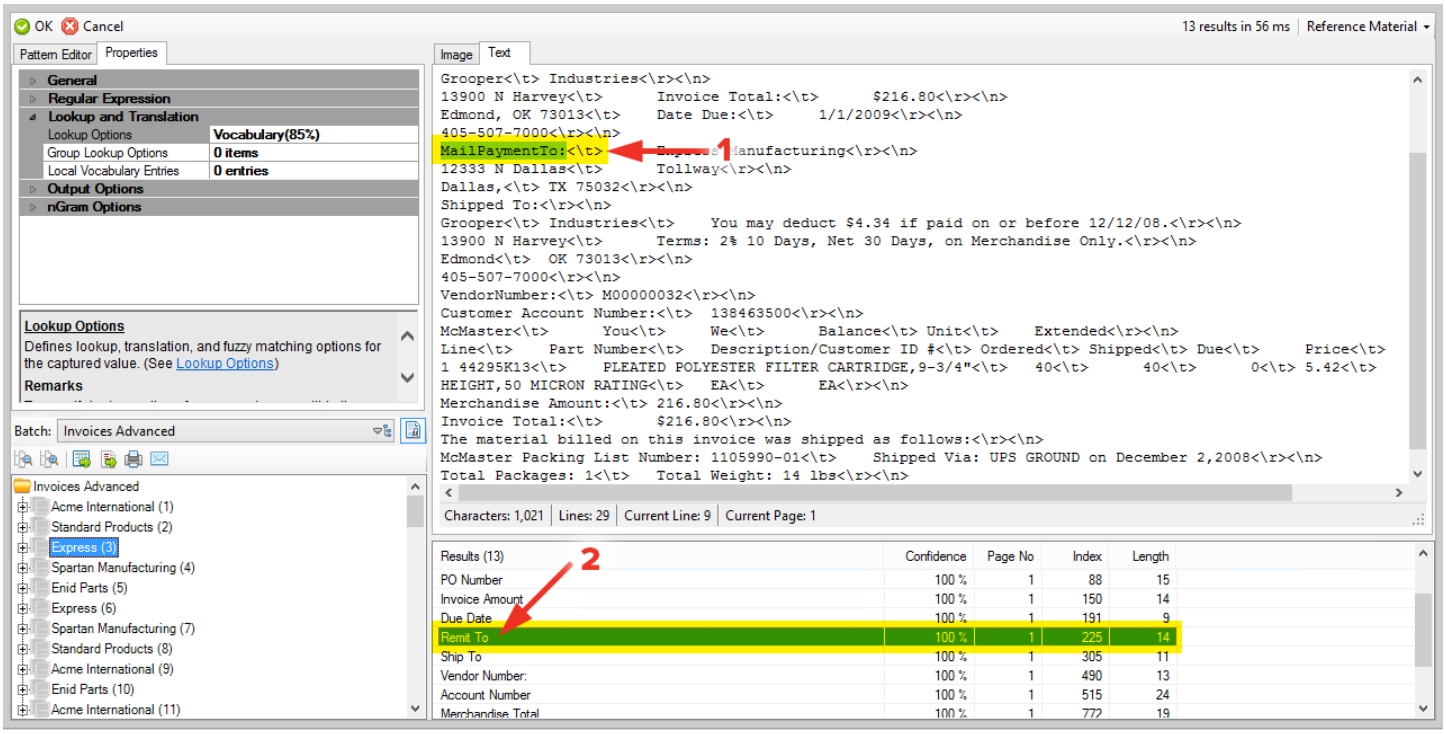
```
i/=0.25
```

Also, **(3)** click the drop-down for Referenced Lexicons and check the Fuzzy Match Weightings Lexicon in the Global Resources • Lexicons > (system) area. This is a system Lexicon with several entries for commonly missed OCR characters and reduced weightings to allow common errors to not be such a heavy penalty on Fuzzy Matching.



STEP 9 – VIEWING SUCCESSFUL RESULTS

(1) With these properties adjusted, **`MailPaymentTo:`** will now be found, **(2)** and because of a translation in the Field Lables Lexicon it will be returned as **`Remit To`**.



The power of, and reason for this extractor being built the way it is (specifically leveraging the Lexicon that utilizes translation), is the ease with which it can be updated. Say you're a company processing a lot of invoices for a plethora of different vendors, at any point, if you add a new vendor and they label something like Invoice Number differently than what you have in the Lexicon (perhaps something like Inv.Numb.), you can simply add an entry to the Lexicon and translate it to what you've established as your normalized version, and you're all set. So for this example:

`Inv.Numb.=Invoice Number`

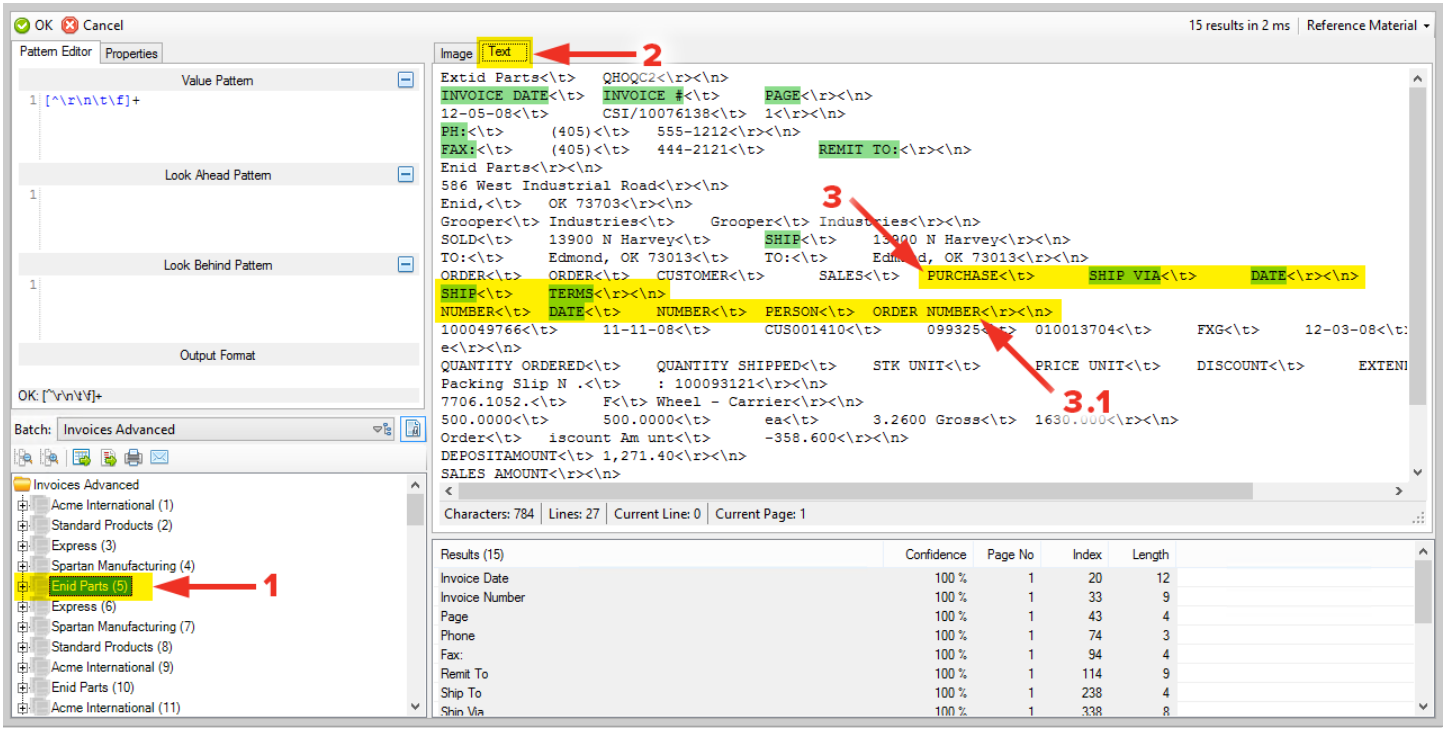
This will become abundantly clear when we train a Field Class to leverage this, as the feature it will train against will always be the normalized translation you've established. As a result, you'll only need to train the Field Class once because the feature will always be returned as what you have it translated to. There may be a need to adjust Context Zones, but you can handle that on a case by case basis, and it'll be very infrequent.

TACKLING MULTI-LINE FIELD LABELS WITH ORDERED ARRAYS

The field labels that have been sought so far have been easy to work with, if for no other reason than that they've been on a single line. When field labels are on multiple lines we logically read them as one unit, but OCR synthesis doesn't understand the relationship of vertically oriented labels, so it does what it always does and puts lines of text in continuous, logical lines. As a result, writing simple RegEx patterns to consistently capture these labels can be a challenge. Thankfully, once again, **Grooper** has an elegant solution to this problem.

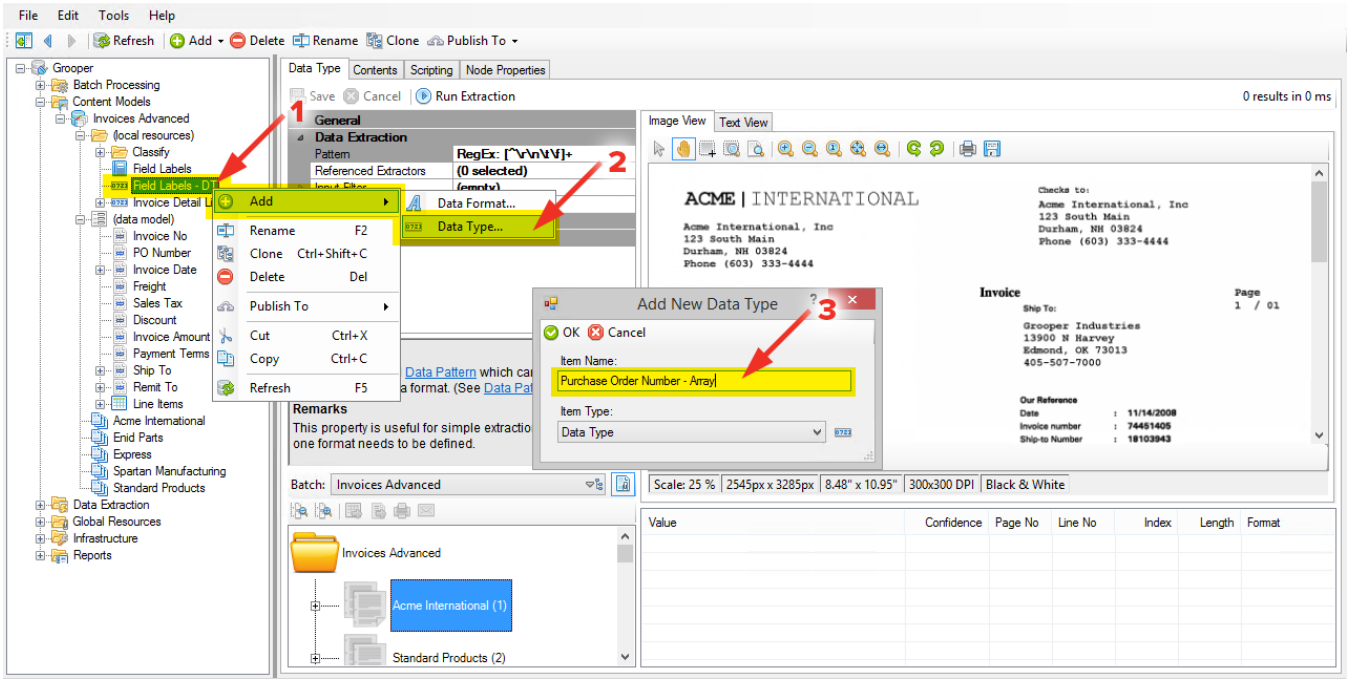
STEP 1 – OBSERVING THE MULT-ILINE PROBLEM

With the Field Labels – DT Data Type still selected, **(1)** select Enid Parts (5) from the Batch Viewer. There are several field labels here, but the problem is they are on different lines. While this makes visual sense, and you can read it fine, **(2)** if you look at the Text tab you'll notice **(3)** there's quite a lot between **PURCHASE** and **ORDER NUMBER** (we'll focus on this feature as it is a field in our Data Model we wish to populate.) This document is structured enough to allow a somewhat complex RegEx pattern to work, but there's a simpler solution to return **PURCHASE** and **ORDER NUMBER** as a single result.



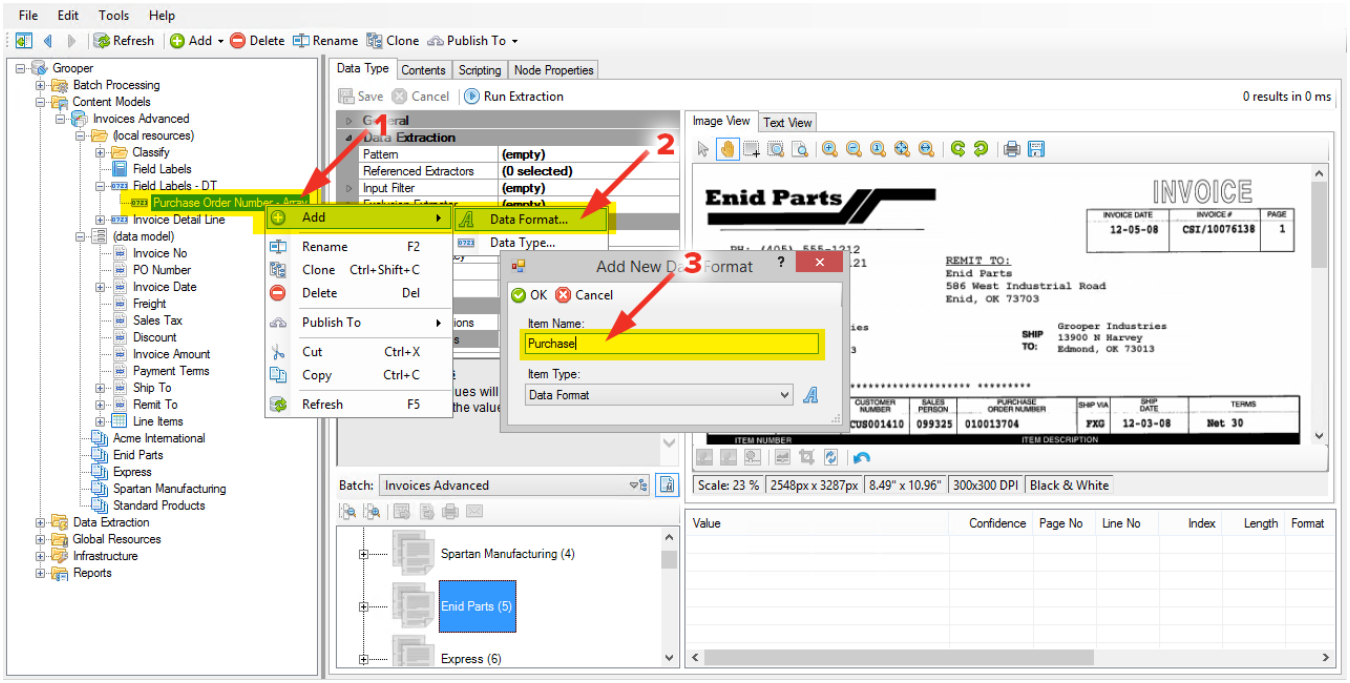
STEP 2 – ADDING CHILD DATA TYPE TO FIELD LABELS – DT

Up to this point we've seen child Data Formats to Data Types, and understand that they return their results to their parent Data Type. However, A Data Type has yet to be added as a child to another Data Type, but it will work the same way by returning its result to its parent object. **(1)** Right-click on the Field Labels – DT Data Type and **(2)** Add > Data Type... and **(3)** name it Purchase Order Number – Array.



STEP 3 – ADDING CHILD DATA FORMATS

(1) Right-click on the Purchase Order Number – Array Data Type **(2)** and Add > Data Format... **(3)** Name it Purchase. Add another Data Format and name it Order Number.



STEP 4 – SIMPLE REGEX PATTERN – PURCHASE TO PO

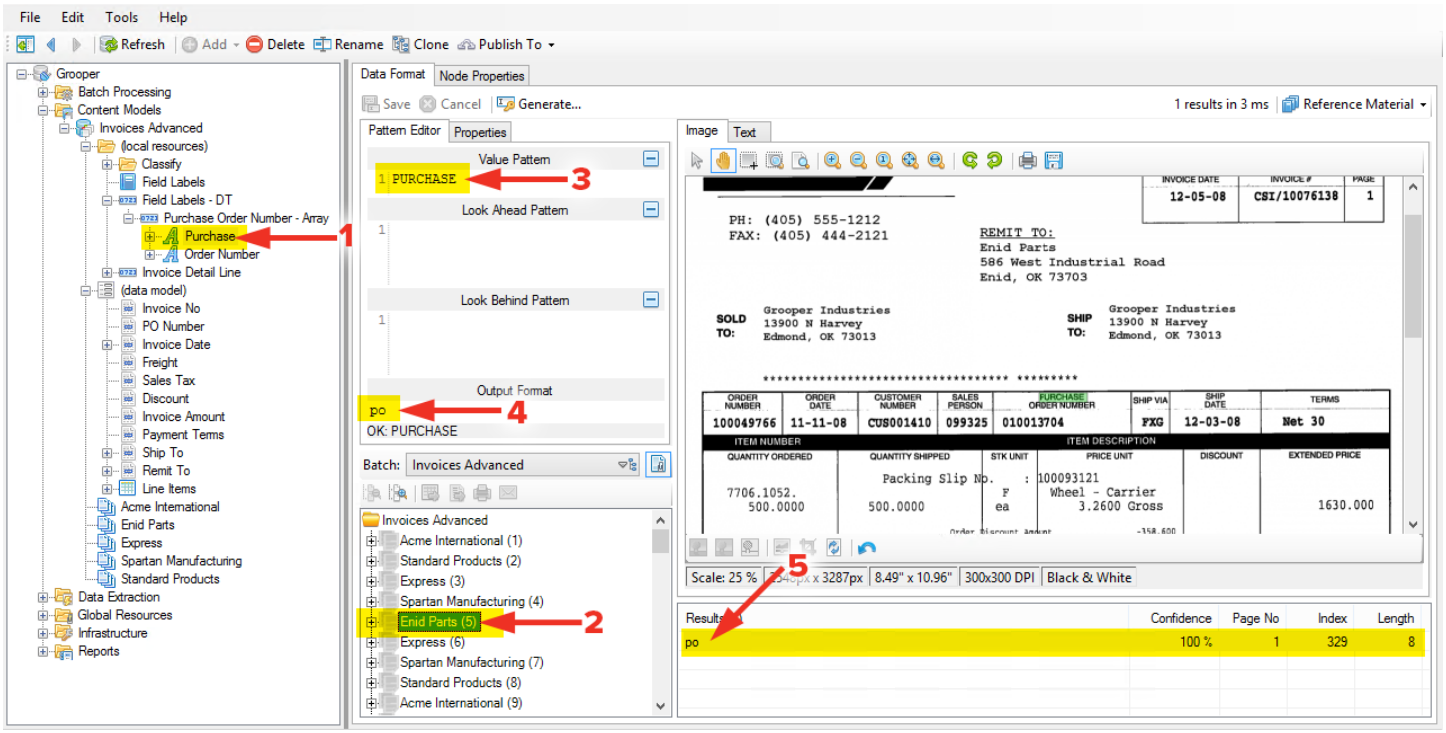
(1) With the Purchase Data Format selected, **(2)** and the Enid Parts (5) document in the Batch Viewer, **(3)** type the following in the Value Pattern:

PURCHASE

(4) In the Output Format type the following:

po

(5) This will find **PURCHASE** in the Enid Parts (5) document, but return it as **po**.



STEP 5 – SIMPLE REGEX PATTERN – ORDER NUMBER TO NUMBER

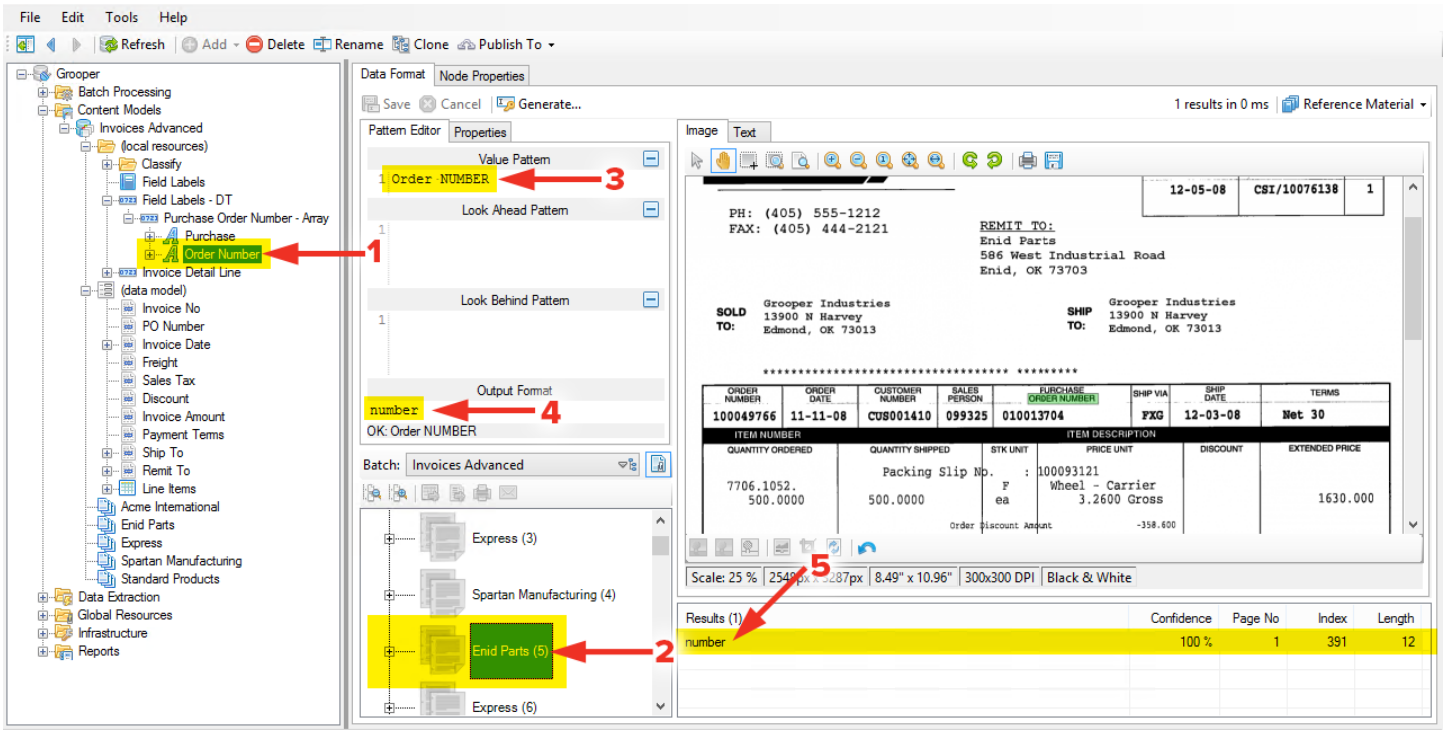
(1) Select the Order Number Data Format, **(2)** and the Enid Parts (5) document in the Batch Viewer, **(3)** type the following in the Value Pattern:

```
ORDER NUMBER
```

(4) In the output format type the following:

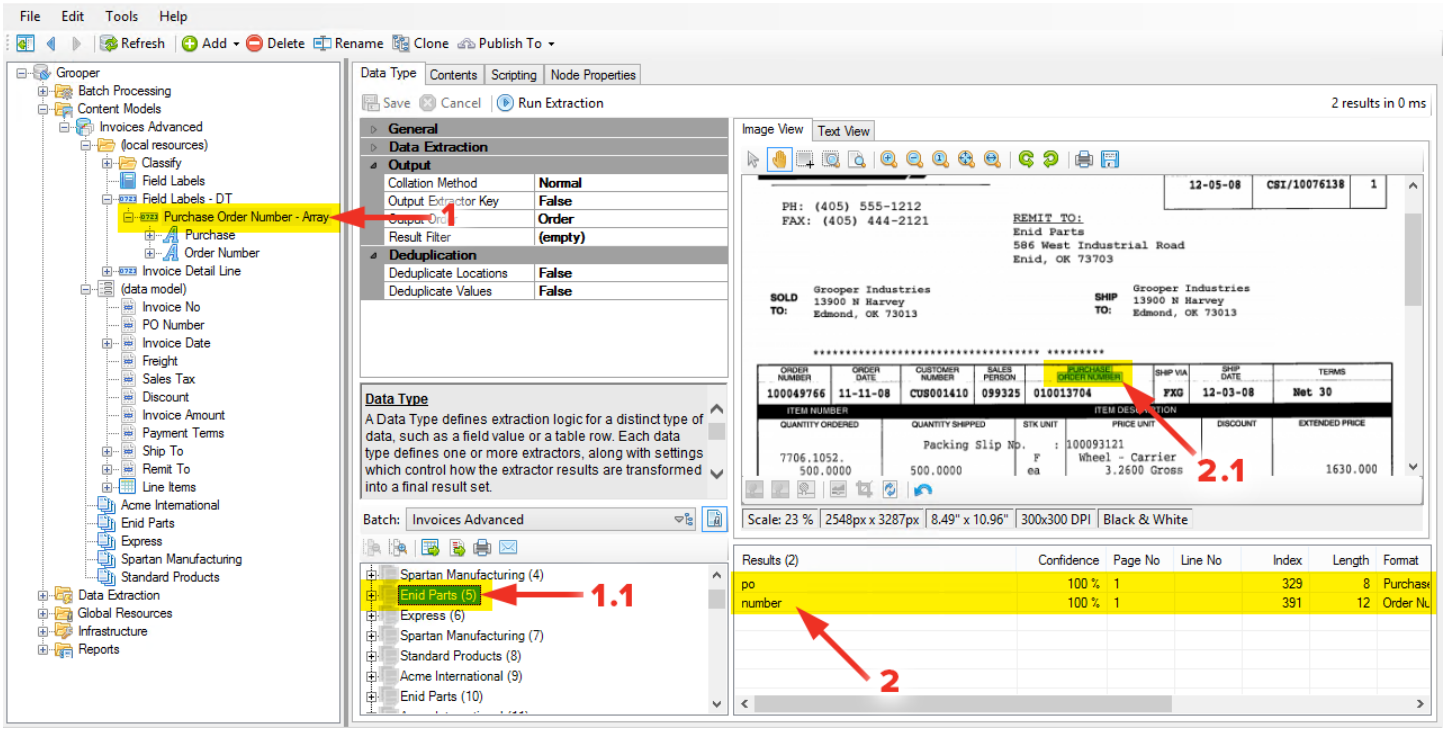
```
number
```

(5) This will find **ORDER NUMBER** in the Enid Parts (5) document, but return it as **number**.



STEP 6 – SEEING RESULTS ON THE PARENT DATA TYPE

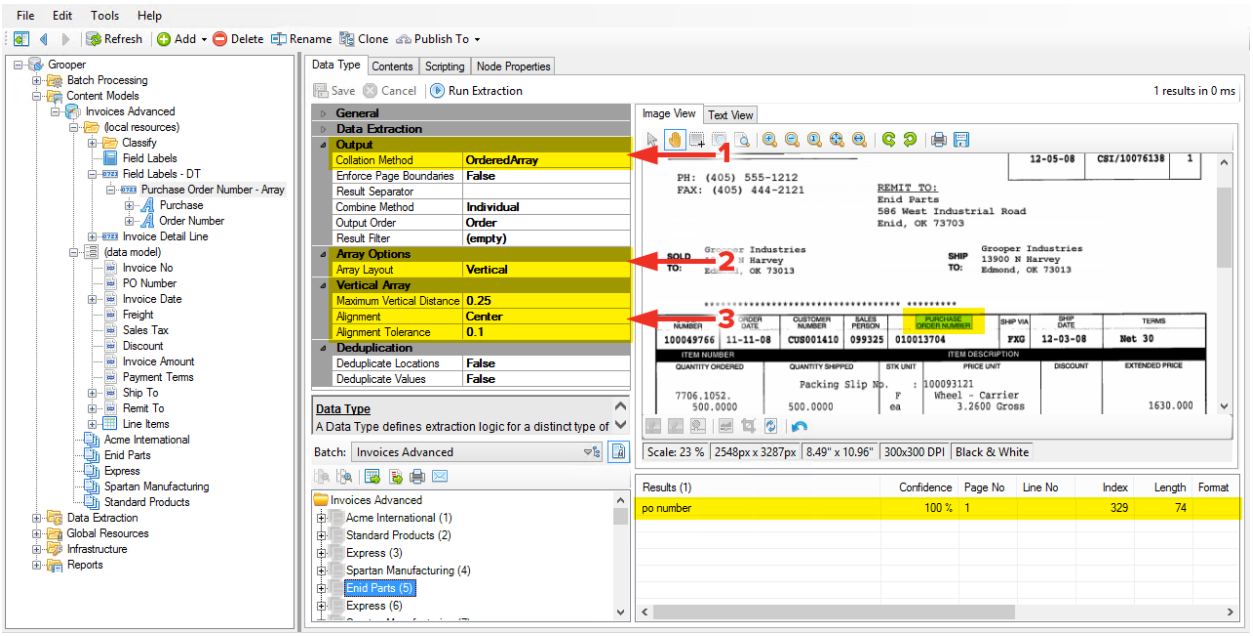
(1) Select the Purchase Order Number – Array Data Type and **(2)** notice the results from the child Data Formats being returned. We don't want separate results, so we'll setup this Data Type to leverage a Collation Method called an Ordered Array.



STEP 7 – SETTING UP THE ORDERED ARRAY

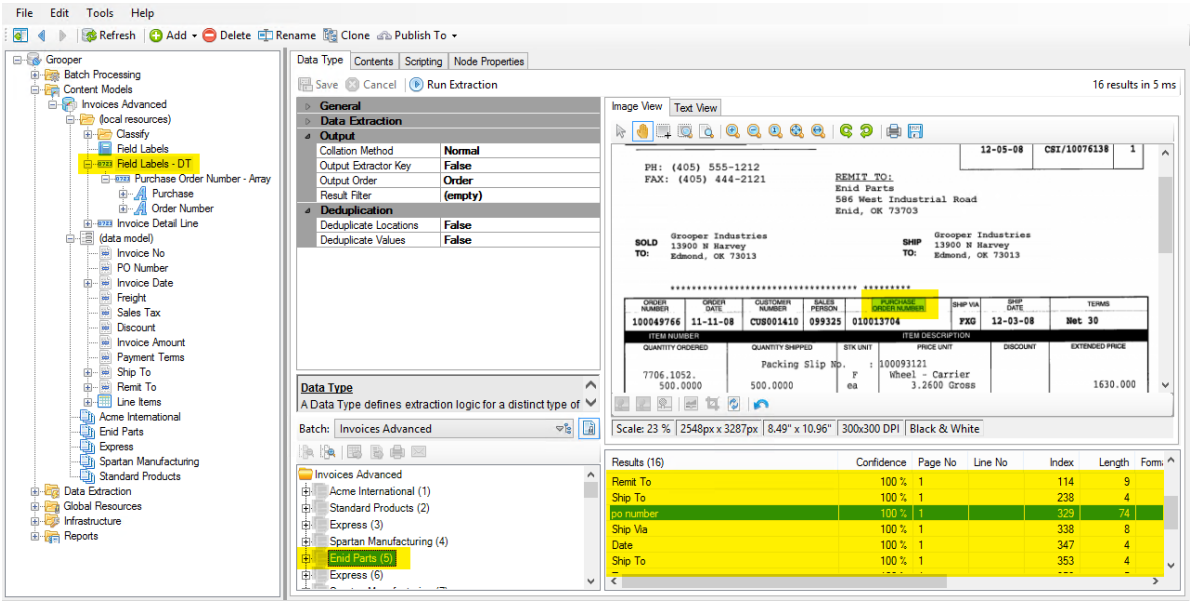
(1) In the Output section set the Collation Method to OrderedArray. **(2)** In the Array Options section set the Array Layout to Vertical. **(3)** In the Vertical Array section, set the Maximum Vertical Distance to 0.25, Alignment to Center, and Alignment Tolerance to 0.1. Save and Run Extraction.

This will take the results of the child Data Formats and combine them into one result if they are found in a vertical layout, within a quarter of an inch of one another, aligned in their center and a tolerance of what is considered centered by 0.1 inches. These are very specific conditions that prevent the possibility of false results being returned by the very simple RegEx patterns that were written within the child Data Formats.



STEP 8 – VIEWING RESULTS ON THE FIELD LABELS – DT PARENT DATA TYPE

Select the Field Labels – DT Data Type and click through some documents in the Batch Viewer. Notice on the End Parts (5) document that **PURCHASE ORDER NUMBER** is being found, and returned as **po number**.

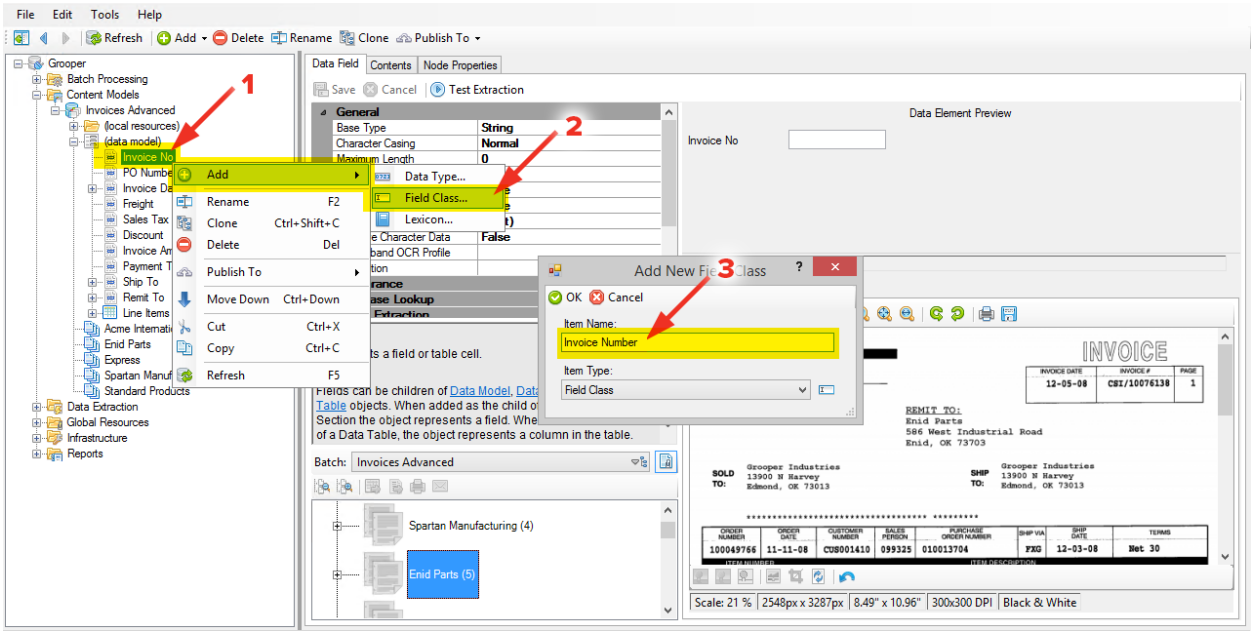


THE NEXT FIELD CLASS – USING THE NEWLY BUILT FIELD LABELS – DT DATA TYPE

All the work that just went into making the Field Lables – DT Data Type will make the creation of the last few Field Classes very straight forward. There is now one Feature Extractor to work with that will consistently return desired results from this set of documents, accounting for OCR errors as well as translating disparate values to reduce the amount of training that is required for the features within each Field Class.

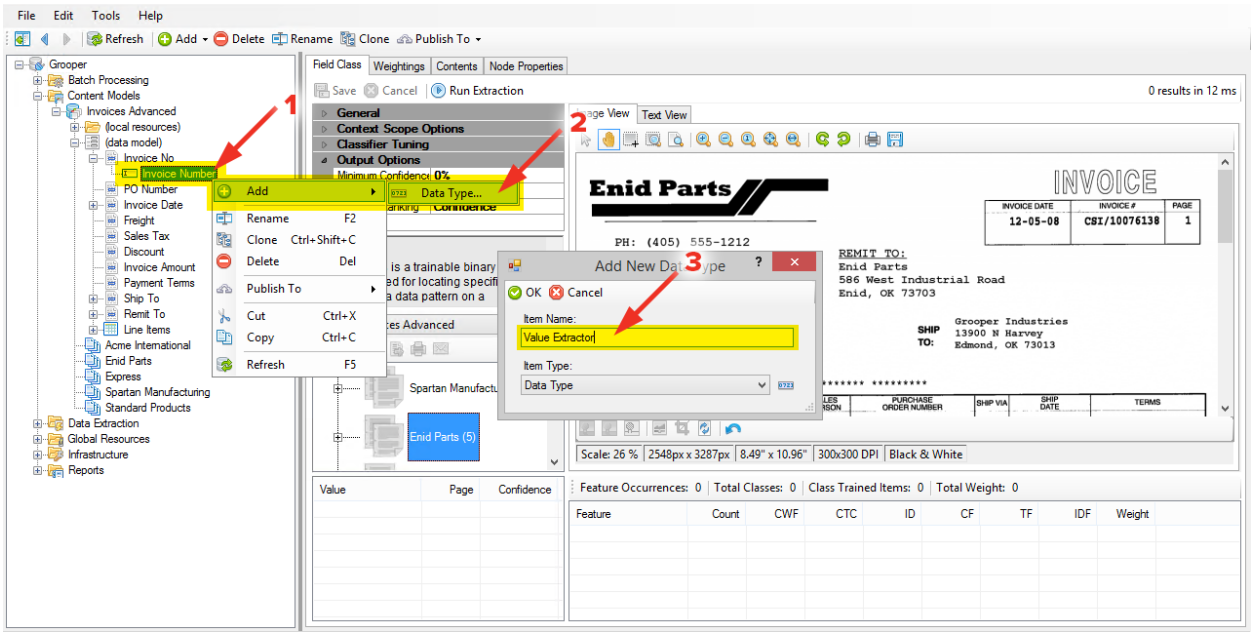
STEP 1 – ADDING A FIELD CLASS TO INVOICE NO

(1) Select the Invoice No Data Field and **(2)** right-click Add > Field Class... **(3)** Name it Invoice Number.



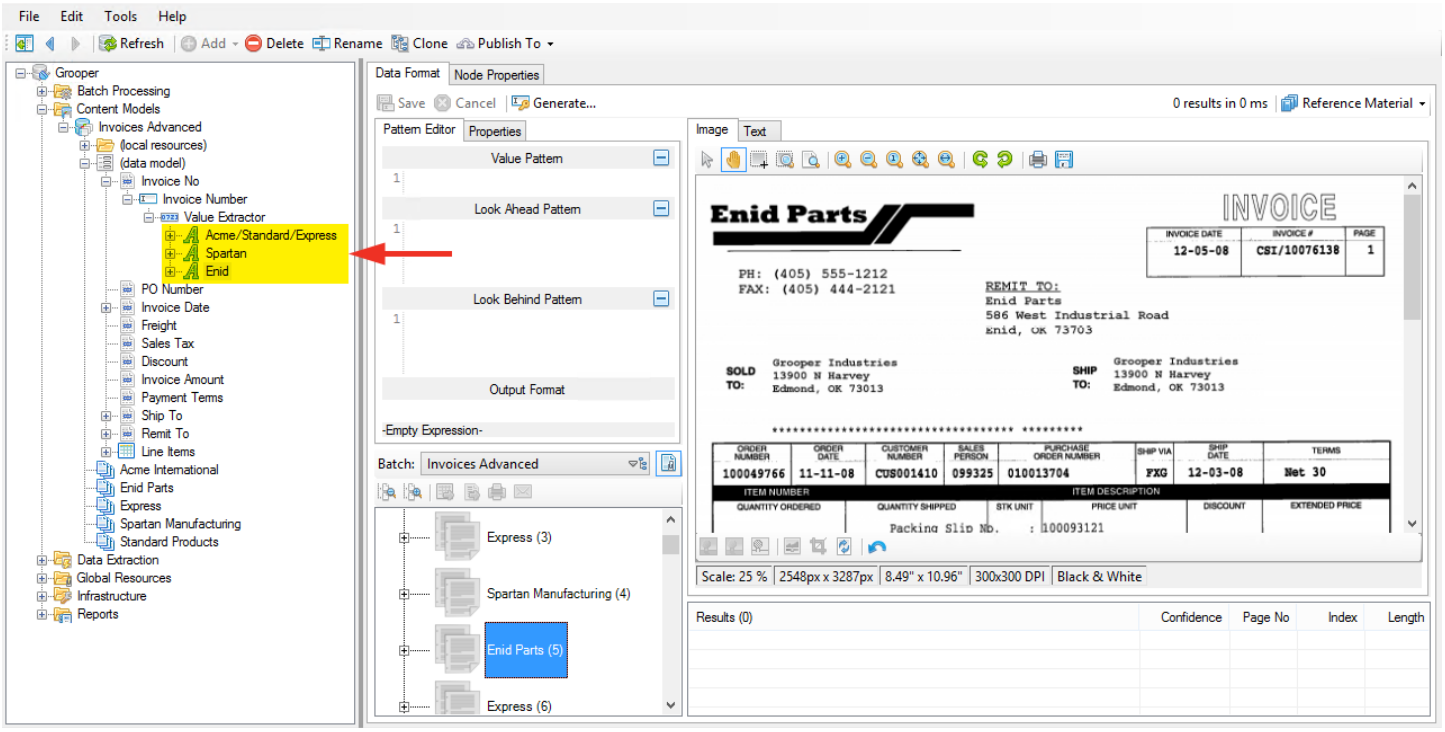
STEP 2 – ADDING A DATA TYPE

(1) Select the Invoice Number Field Class, **(2)** right-click Add > Data Type... **(3)** Name it Value Extractor.



STEP 3 – ADDING DATA FORMATS

Add three Data Formats to the Value Extractor Data Type. Name them Acme/Standard/Express, Spartan, and Enid.

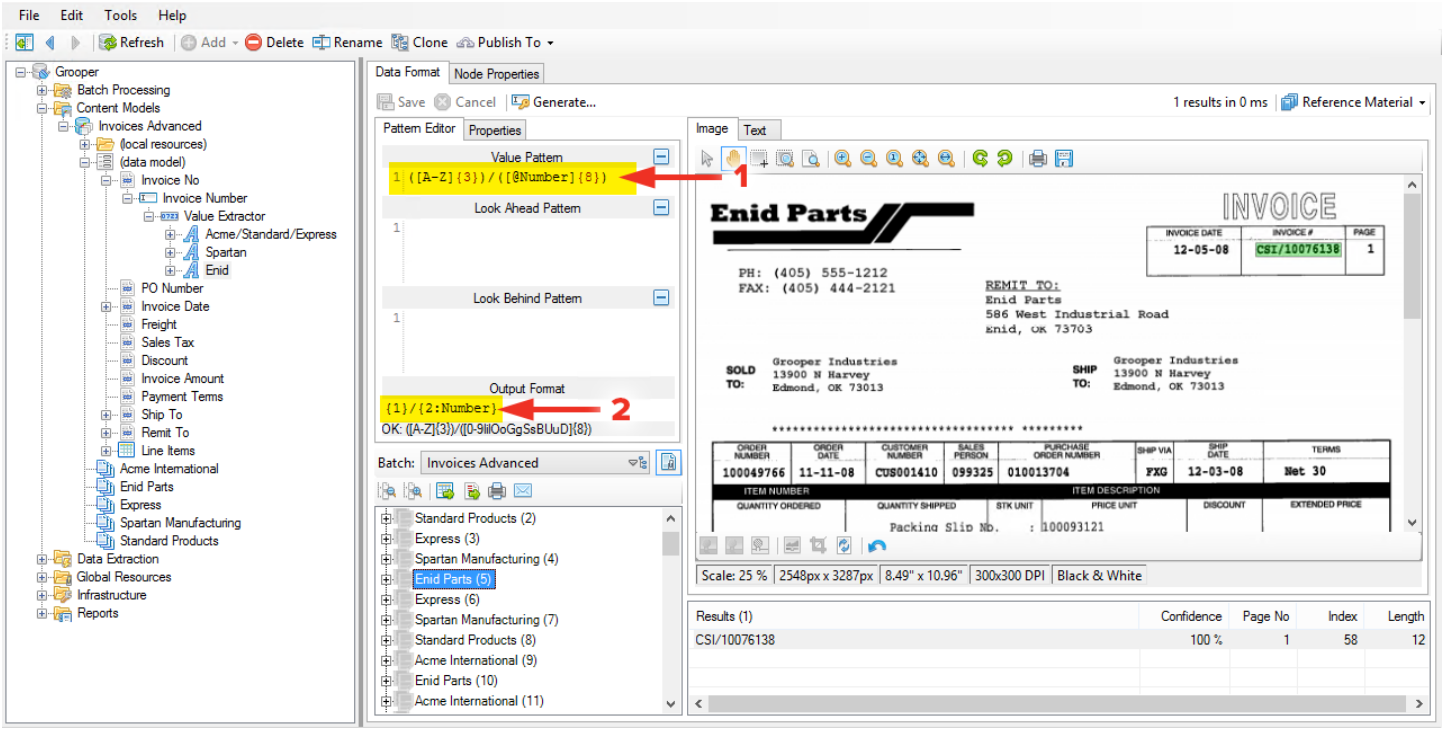


STEP 4 – VALUE PATTERNS AND OUTPUT FORMATS

(1) Use the Information below to populate the Pattern Editor of the newly created Data Formats.

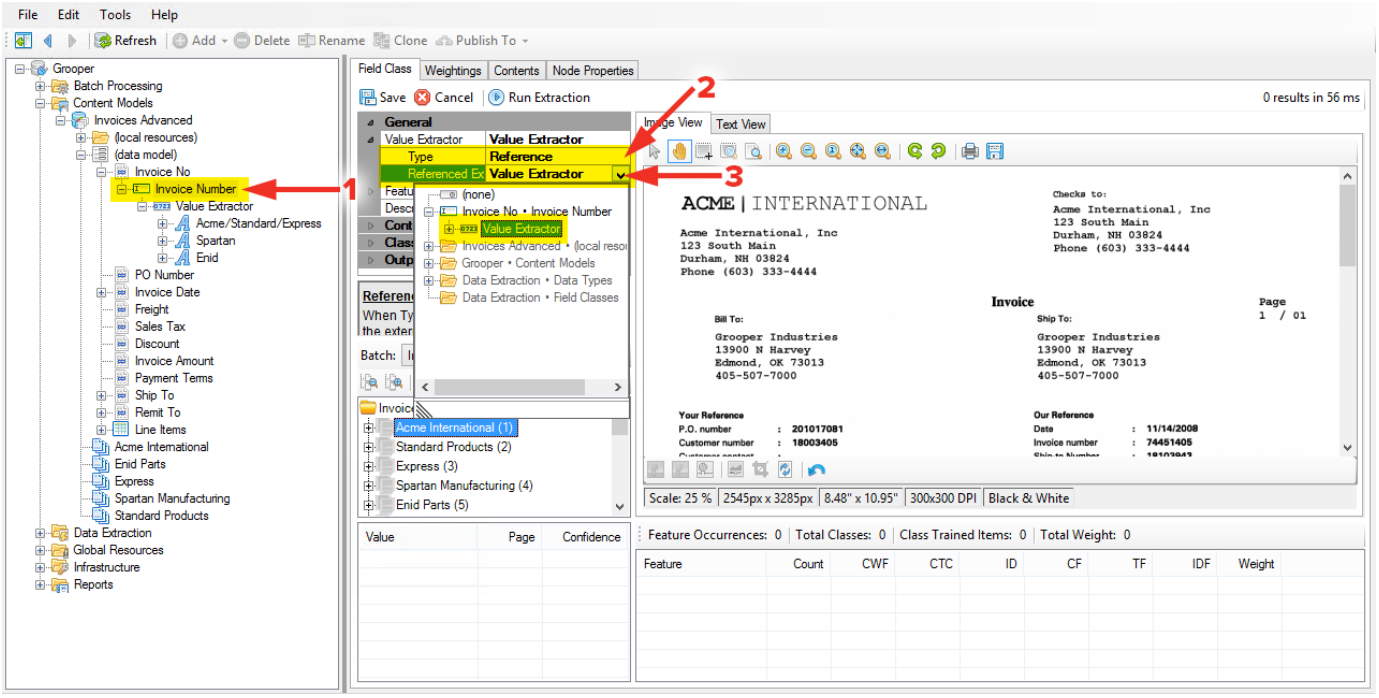
(2) Take note of the Output Formats for Enid and Spartan. Notice the syntax with a number, followed by a colon, and the string Number. This leverages the @Number expression variable and forces it to return the known number variants that its RegEx pattern is compensating for. Without this, the @Number expression variable would capture something like an **l**, but still return it as an **l**, as opposed to the digit it's commonly mis-OCRed for, a **1**.

	Value Pattern	Output Format
Acme/Standard/Express	`[0-9]{6,16}`	
Spartan	`([A-Z]{2})([@Number]{2})-([@Number]{6})`	`{1}{2:Number}-{3}`
Enid	`([A-Z]{3})/([@Number]{8})`	`{1}/{2:Number}`



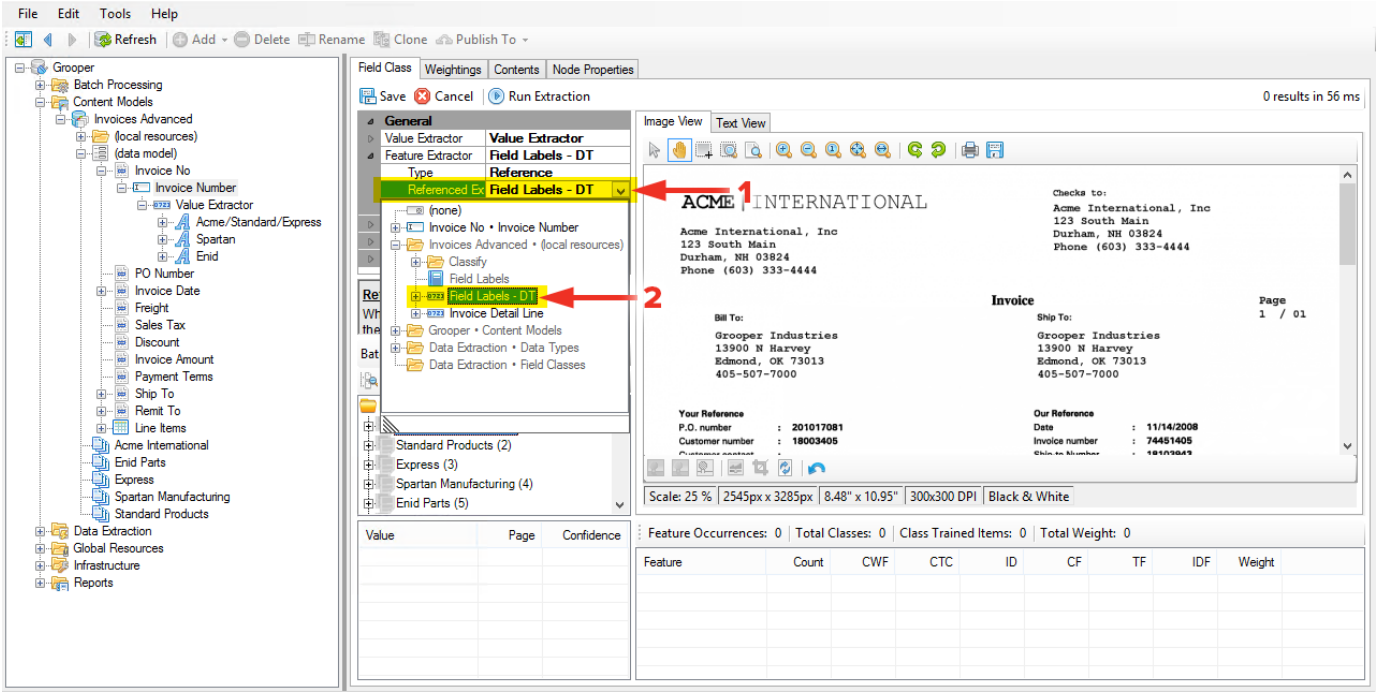
STEP 5 – SET THE VALUE EXTRACTOR

(1) Select the Invoice Number Field Class and set the **(2)** Value Extractor Type to Reference and **(3)** set the Referenced Extractor to the Value Extractor Data Type that is a child of the Invoice Number Field Class.



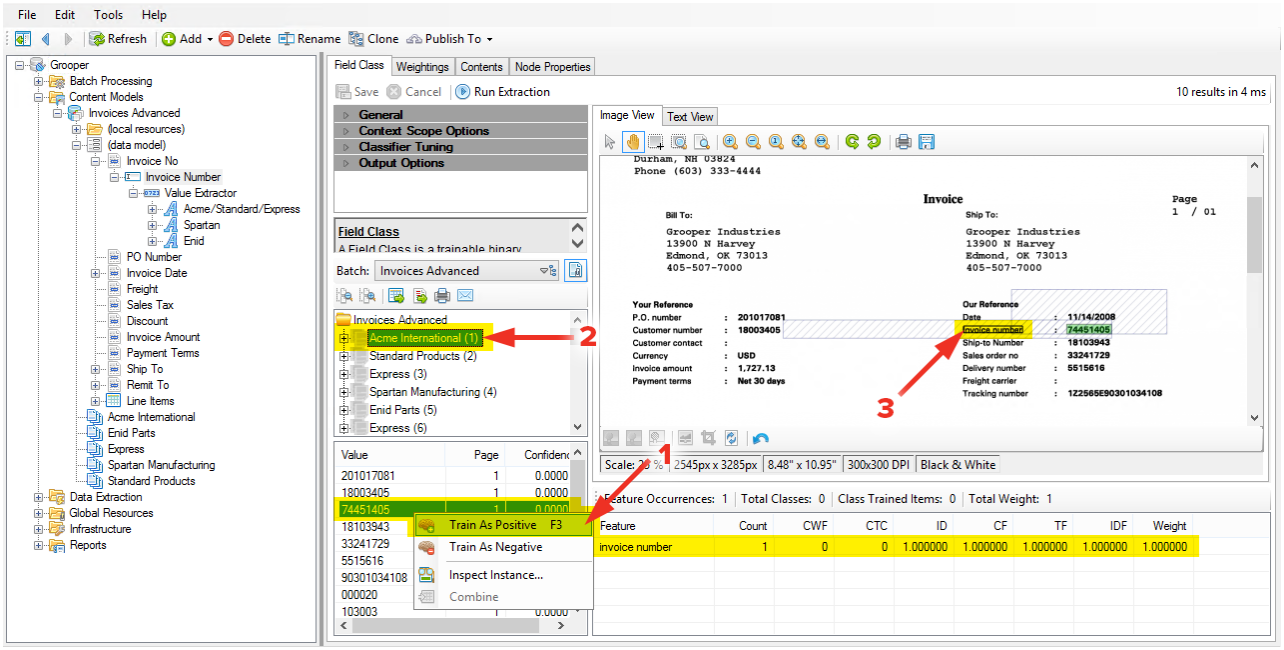
STEP 6 – SET THE FEATURE EXTRACTOR

(1) Expand the Feature Extractor properties and change the Referenced Extractor from the nGrams 1-3 Data Type, **(2)** to the Field Labels – DT Data Type within the Invoices Advanced • (local resoureces). Save and Run Extraction with the Acme International (1) document selected in the Batch Viewer.



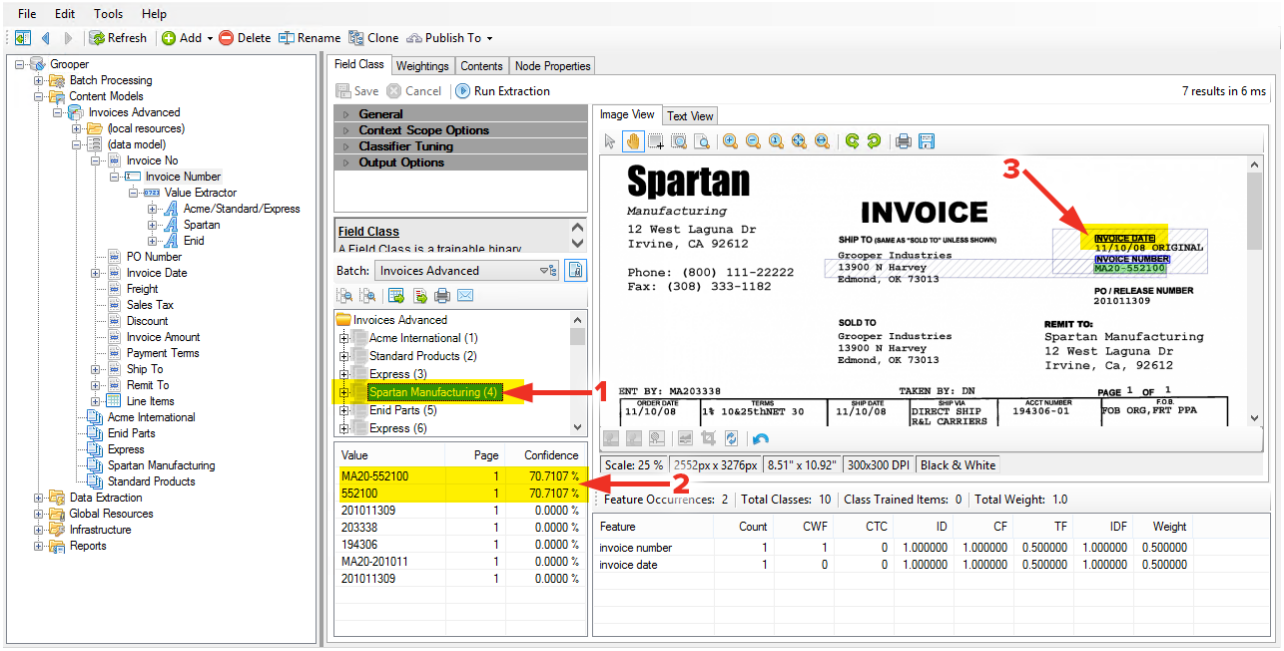
STEP 7 – TRAIN A FEATURE

(1) On the Acme International (1) document, **(2)** select the 74451405 value in the Value List View and right-click Train as Positive. **(3)** This will add the feature invoice number to the Positive Weightings, as well as negatively weight all other features captured within the Context Zones of the other values that were listed.



STEP 8 – FEATURE DETECTION PROBLEM – BAD CONTEXT ZONES

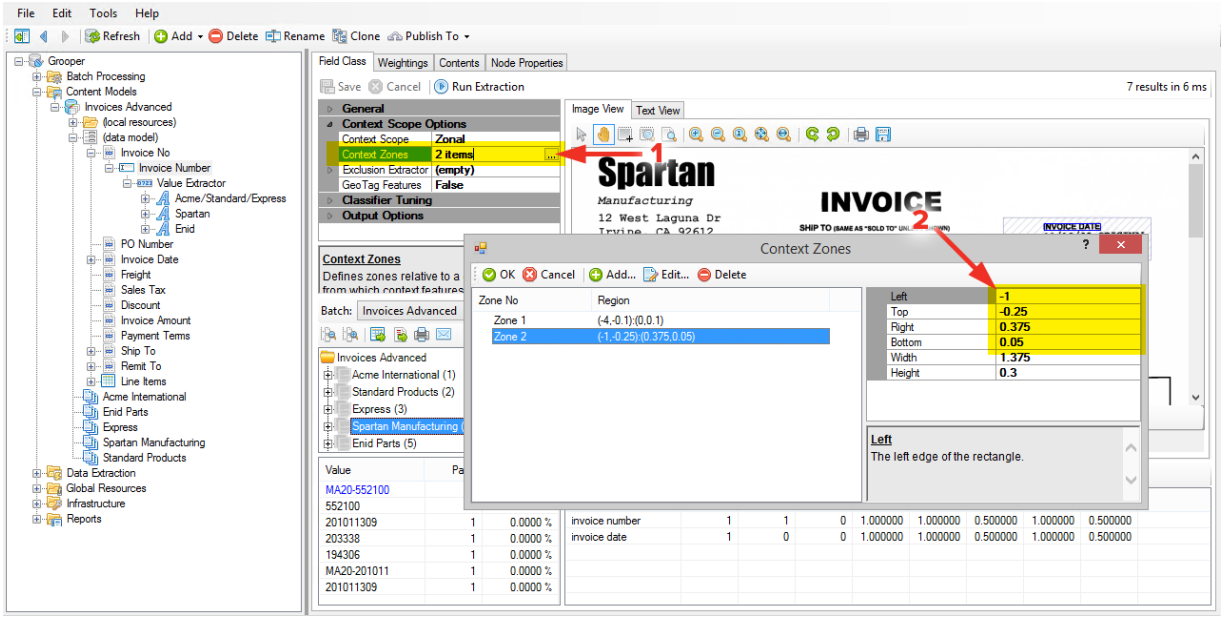
Due to translation in the Lexicon that the Field Labels – DT Data Type is using, all variations of Invoice Number will be returned the same, so, this one feature being trained should return 100% on all DocTypes. However, **(1)** select Spartan Manufacturing (4) and **(2)** notice the correct value in the Value List View being returned at ~70%. **(3)** Notice in the Page Viewer that the Zone 2 Context Zone is too tall, and consequently grabbing the invoice date feature.



STEP 9 – ADJUSTING THE CONTEXT ZONES

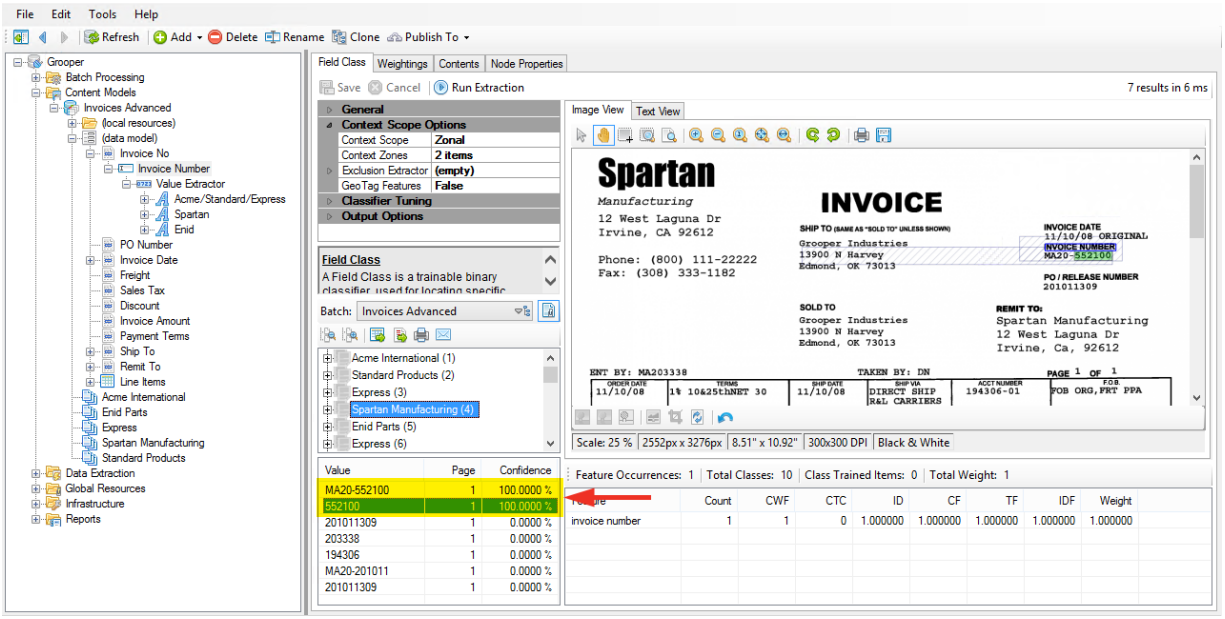
(1) Expand the Context Scope Options, select the Context Zones property, and click the ellipsis button to bring up the Context Zones property window. Feel free to click Zone 2 and manually edit it, but **(2)** the proper values to allow this zone to function properly have been provided as well as the values for Zone 1. After closing the property window, Save and Run Extraction.

Zone 1 – Left: **-4** Top: **-0.02** Right: **0.35** Bottom: **0.1**
Zone 2 – Left: **-1** Top: **-0.25** Right: **0.375** Bottom: **0.05**



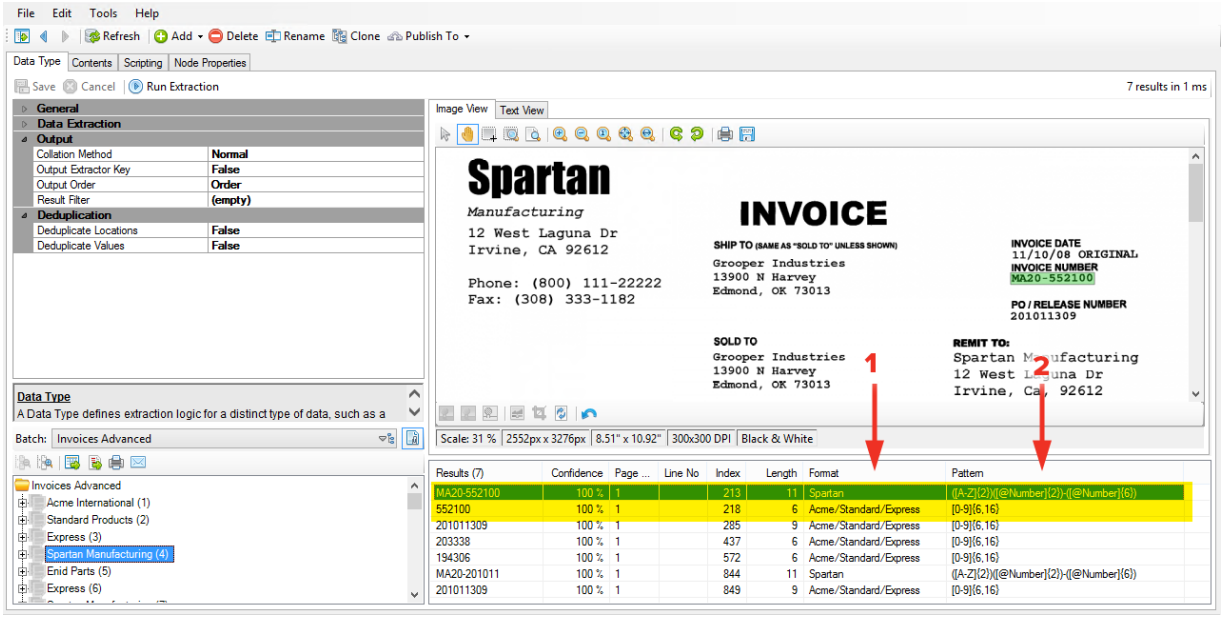
STEP 10 – DUPLICATE ENTRIES

Notice for Spartan Manufacturing (4) there are two values listed at 100% confidence, but the values themselves are not the same. Look more closely and you'll notice that the second value is the integer portion of the first value after the hyphen. This is because there are overlapping results being returned by the child Data Formats of the Value Extractor Data Type.



STEP 11 – ANALYZING THE VALUE EXTRACTOR DATA TYPE FOR DUPLICATE VALUES

Select the Value Extractor Data Type and observe the results being returned in the Results List View. Take specific heed of **(1)** the Format **(2)** and Pattern columns. The MA20-552100 result is being returned by the Spartan format, and the 552100 result is being returned by the Acme/Standard/Express format. The latter's format is generic enough to capture this six-digit value.

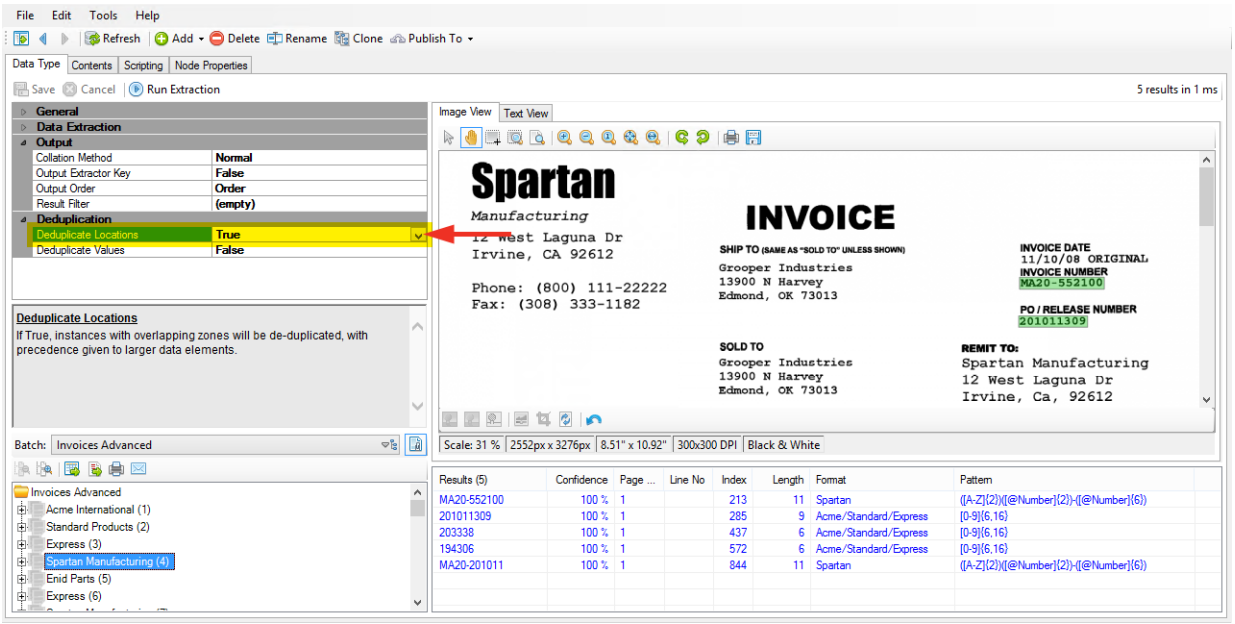


STEP 12 – DEDUPLICATE LOCATIONS

Data Types have a property to compensate for this problem called Deduplicate Locations. Its tooltip reads:

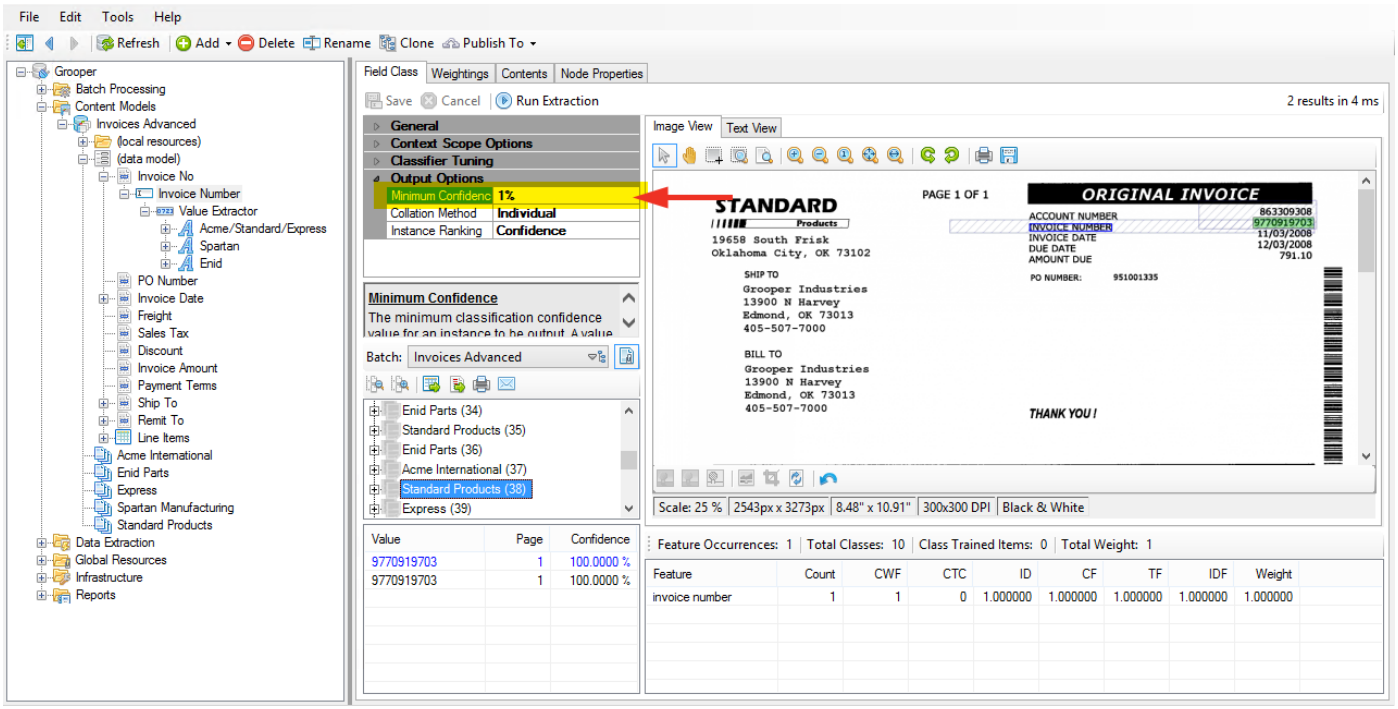
If True, instances with overlapping zones will be de-duplicated, with precedence given to larger data elements.

So, the zones for the MA20-552100 and 552100 are definitely overlapping, and because the former pattern is the longer one and the desired result, setting this property to True is the solution to this problem.



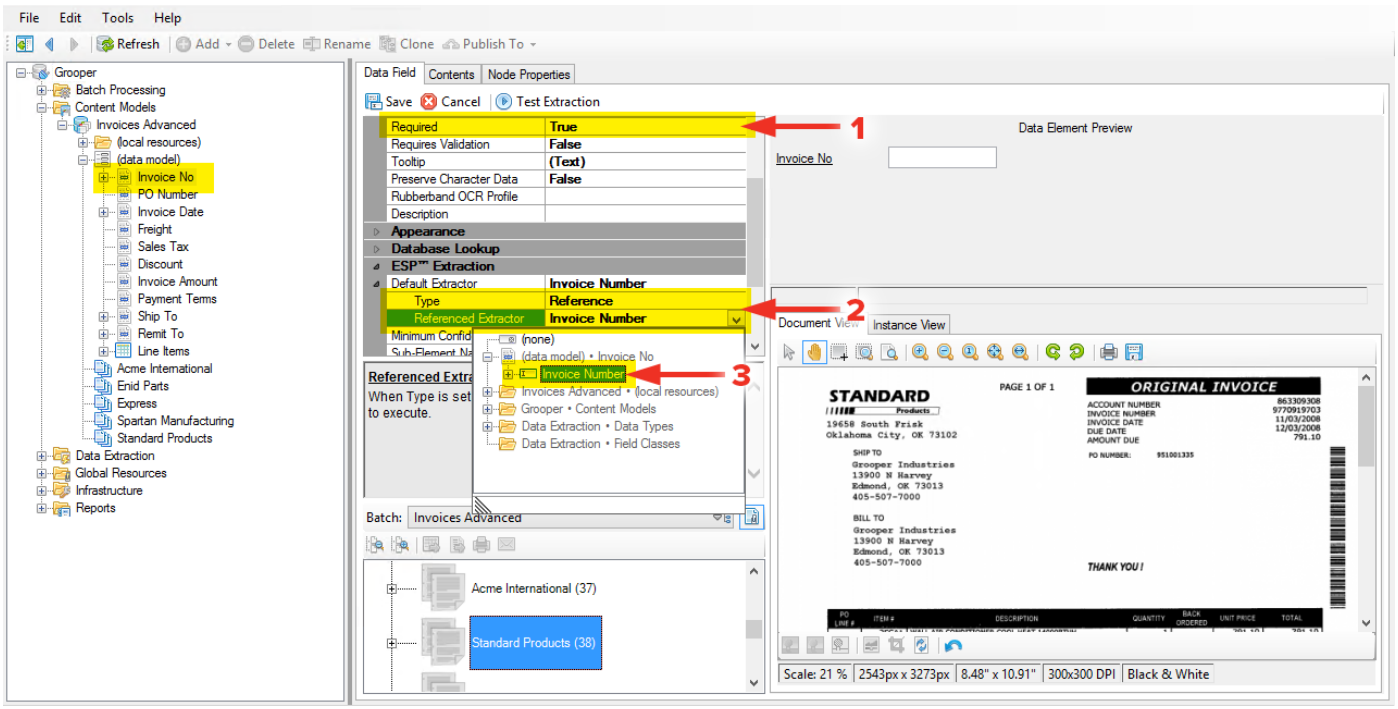
STEP 13 – SETTING MINIMUM CONFIDENCE

With the adjustment made to the Value Extractor Data Type and saved, select the Invoice Number Field class. Because there is only one feature trained, and it is being returned at 100% confidence, the Minimum Confidence can be set to anything above 0% and all unwanted results will be trimmed.



STEP 14 – SETTING PROPERTIES FOR THE INVOICE NO DATA FIELD

Select the Invoice No Data Field and **(1)** set its Required property to True. **(2)** Set the Default Extractor Type to Reference and **(3)** set the Referenced Extractor to the Invoice Number Field Class.

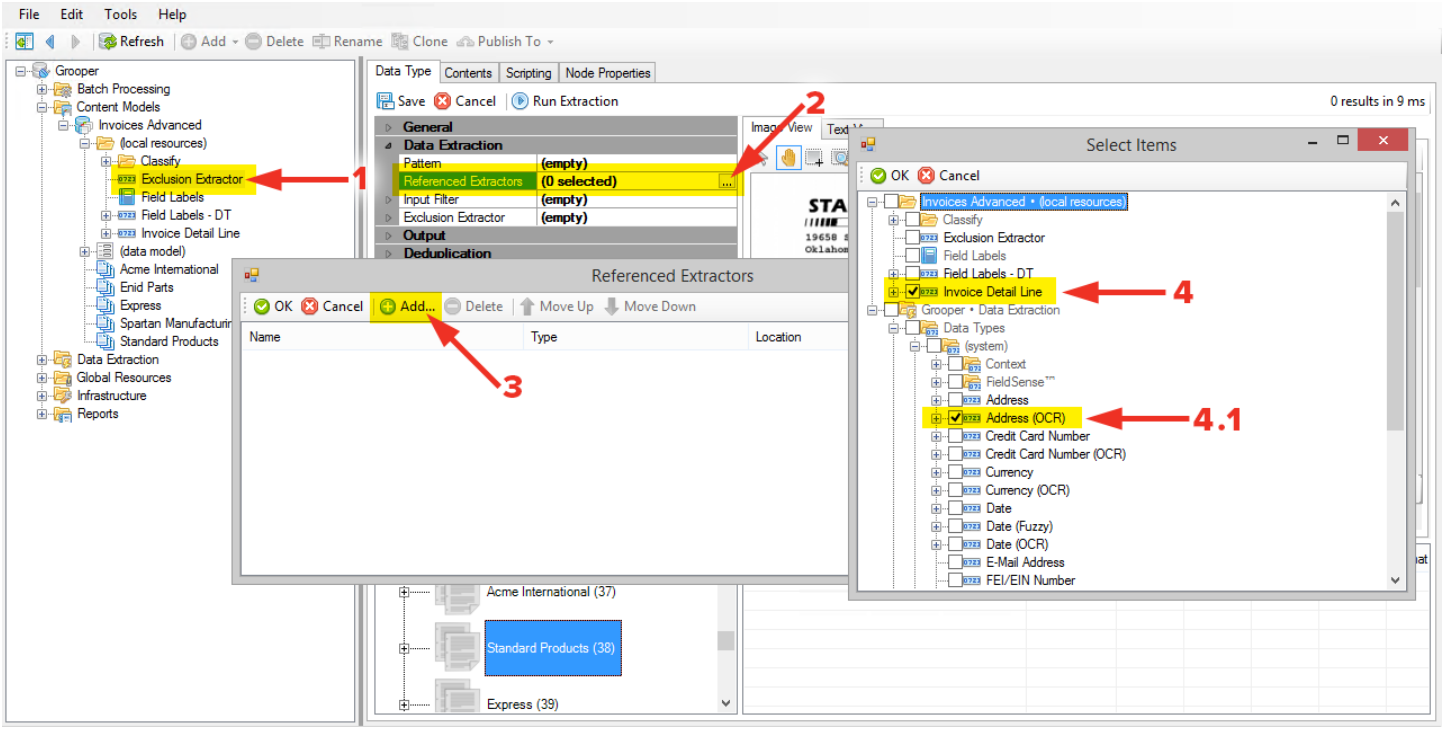


THE NEXT FIELD CLASS – VALUE EXTRACTOR LEVERAGING AN EXCLUSION EXTRACTOR

A Field Class will now be made for the PO Number Data Field. The Value Extractor for this Field Class will be built locally to the PO Number, but it will use a very generic RegEx pattern that will return far more results than is necessary. While this isn't inherently a problem with a Field Class, it's a good opportunity to introduce a new property of Data Types. Data Types can reference other Data Types in multiple ways, and one of those ways is as an exclusion.

STEP 1 – BUILDING THE EXCLUSION EXTRACTOR

(1) Create a Data Type in the (local resources) folder and name it Exclusion Extractor. In the Data Extraction section **(2)** select the Referenced Extractors property and click the ellipsis button to bring up the Referenced Extractors window. **(3)** Click the Add button in the Referenced Extractors window to open the select Items window. **(4)** Put check boxes on the Invoice Detail Line Data Type in the Invoices Advanced • (local resources) area, and the Address (OCR) Data Type within the Grooper • Data Extraction > Data Types > (system) area. Save and Run Extraction and notice without adding a pattern to this Data Type, because it is referencing other already built Data Types, it will combine their results and return them.

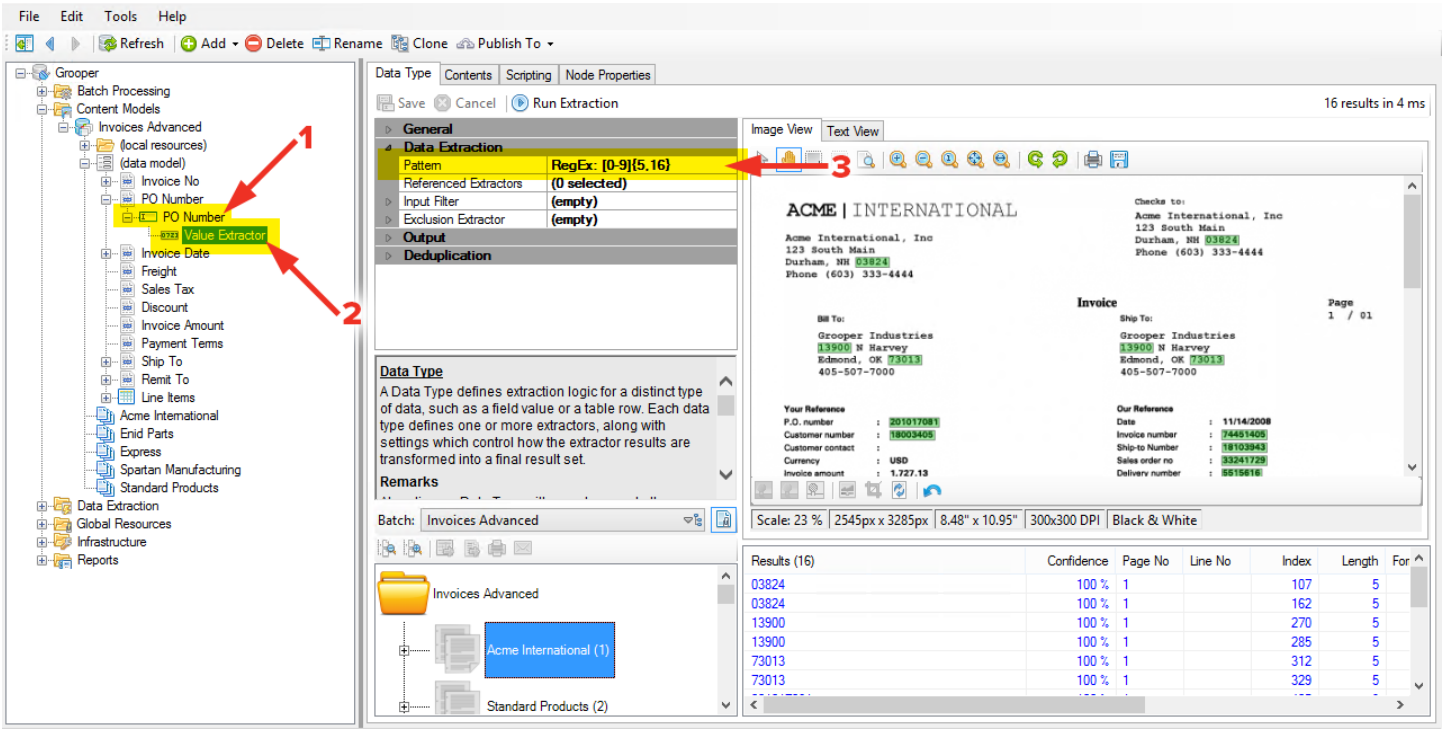


STEP 2 – ADDING PO NUMBER AND VALUE EXTRACTOR FIELD CLASS AND DATA TYPE

(1) Add a Field Class named PO Number as a child object to the PO Number Data Field. **(2)** Also, add a Data Type named Value Extractor as a child object to the PO Number Field Class. **(3)** Add the following as the Pattern for this Data Type:

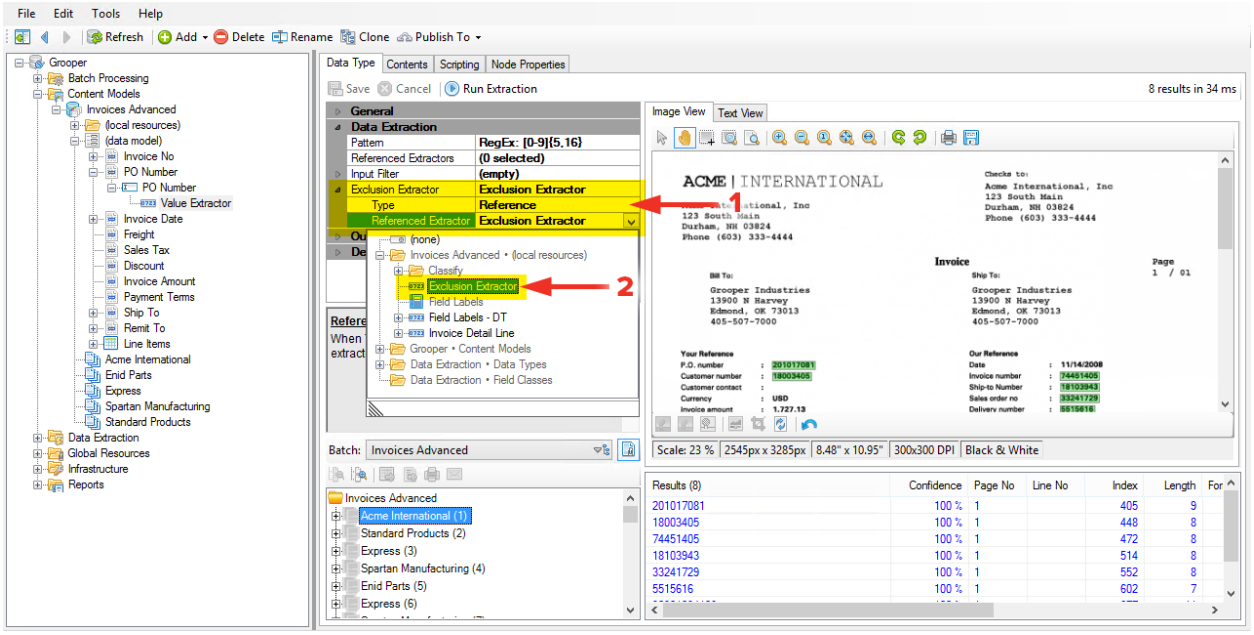
```
[0-9]{5,16}
```

Save and Run Extraction and take note of the results returned.



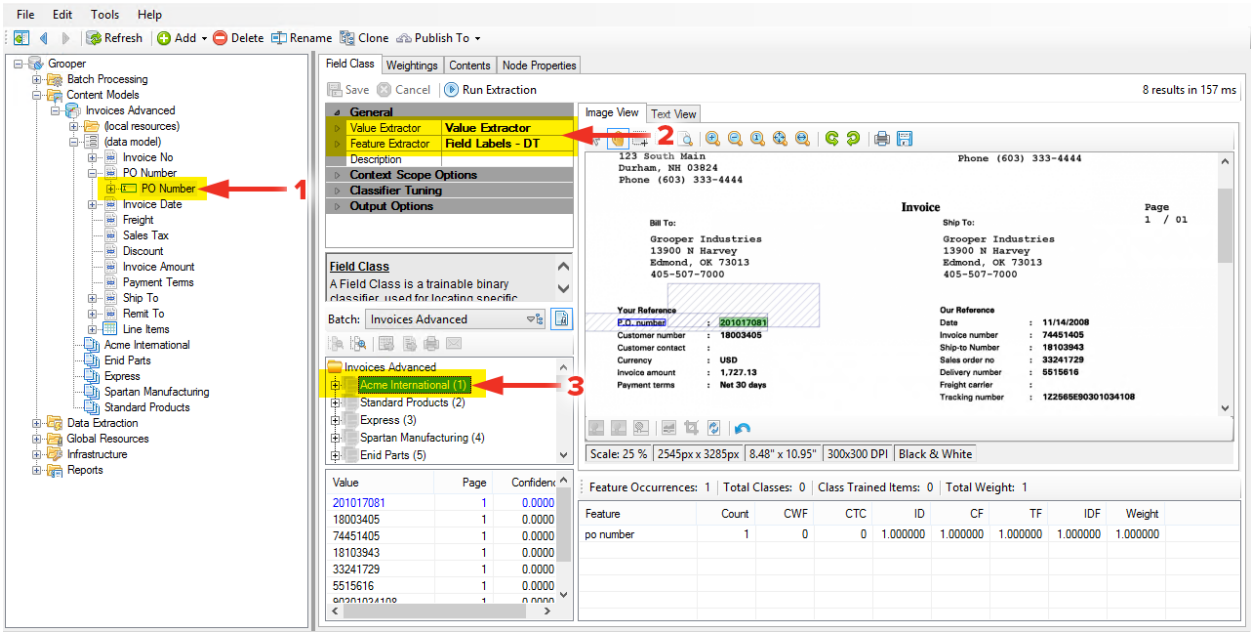
STEP 3 – SETTING THE EXCLUSION EXTRACTOR PROPERTY

This very generic pattern is getting far more results than needed. The Exclusion Extractor Data Type that was just built references Data Types of areas within this document set where PO Numbers will never appear: in addresses, and in line item details of the invoices. **(1)** Set the Exclusion Extractor Type to Reference, **(2)** and the Referenced Extractor to the Exclusion Extractor Data Type within the Invoices Advanced • (local resources) area. Save and Run Extraction and notice now that a bulk of unwanted results have been eliminated.



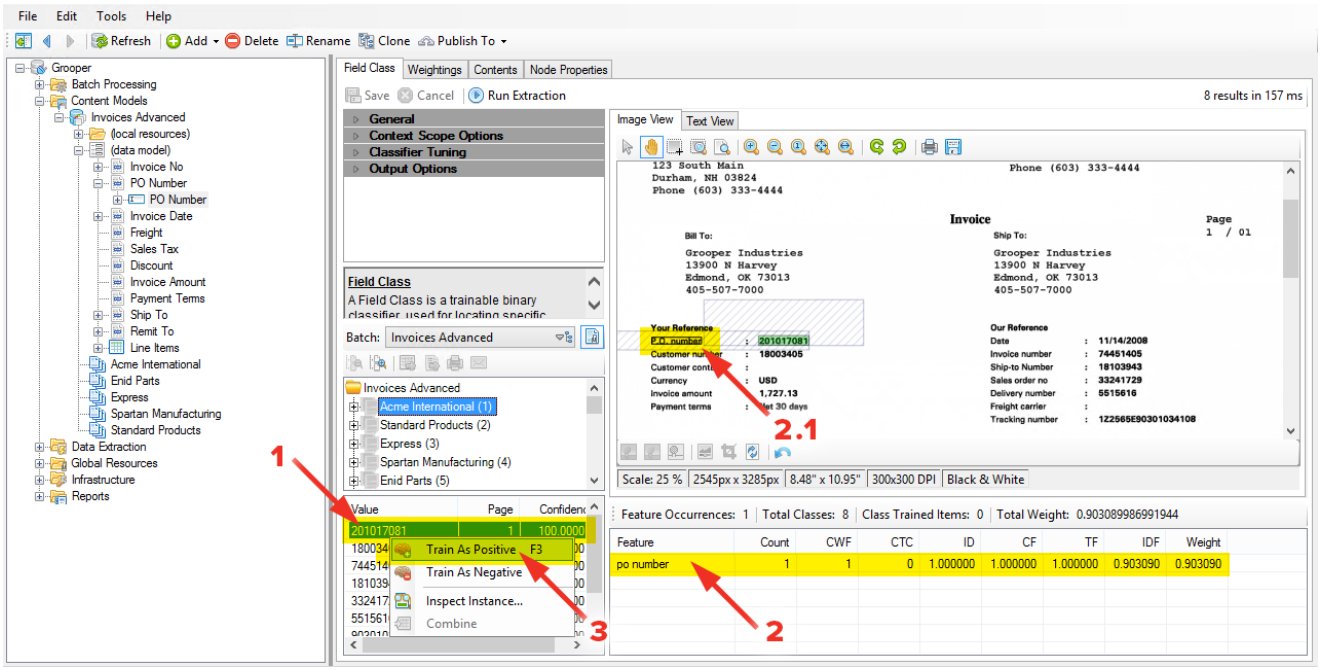
STEP 4 – SETTING THE FIELD CLASS EXTRACTORS

(1) Select the PO Number Field Class, **(2)** set the Value Extractor Type to Reference, and change the Referenced Extractor to the Value Extractor Data Type that is a child to the PO Number Field Class. Change the Referenced Extractor of the Feature Extractor section to the Field Labels – DT Data Type within the Invoices Advanced • (local resources) area. **(3)** Save and Run Extraction on the Acme International (1) document.



STEP 5 – TRAINING THE PO NUMBER FEATURE

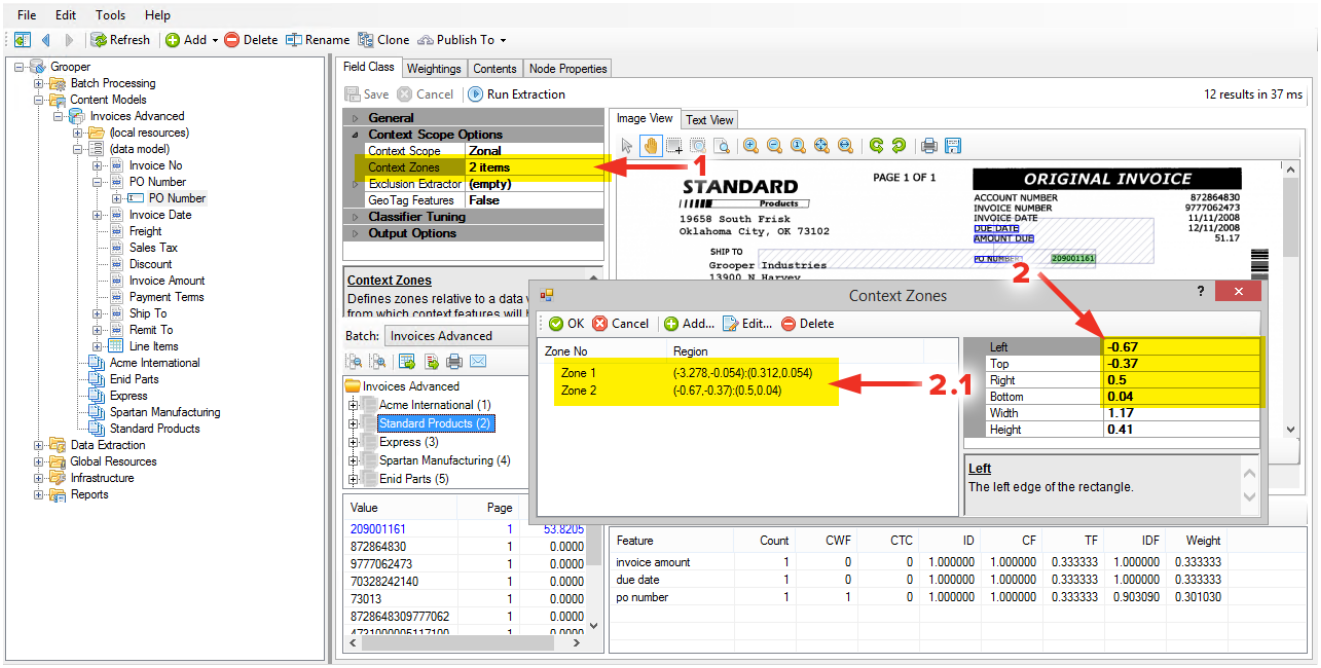
(1) Select the 201017081 value in the Value List View and **(2)** notice again that translation is occurring for the feature from P.O. Number to po number. **(3)** Train this positively.



STEP 6 – ADJUSTING CONTEXT ZONES

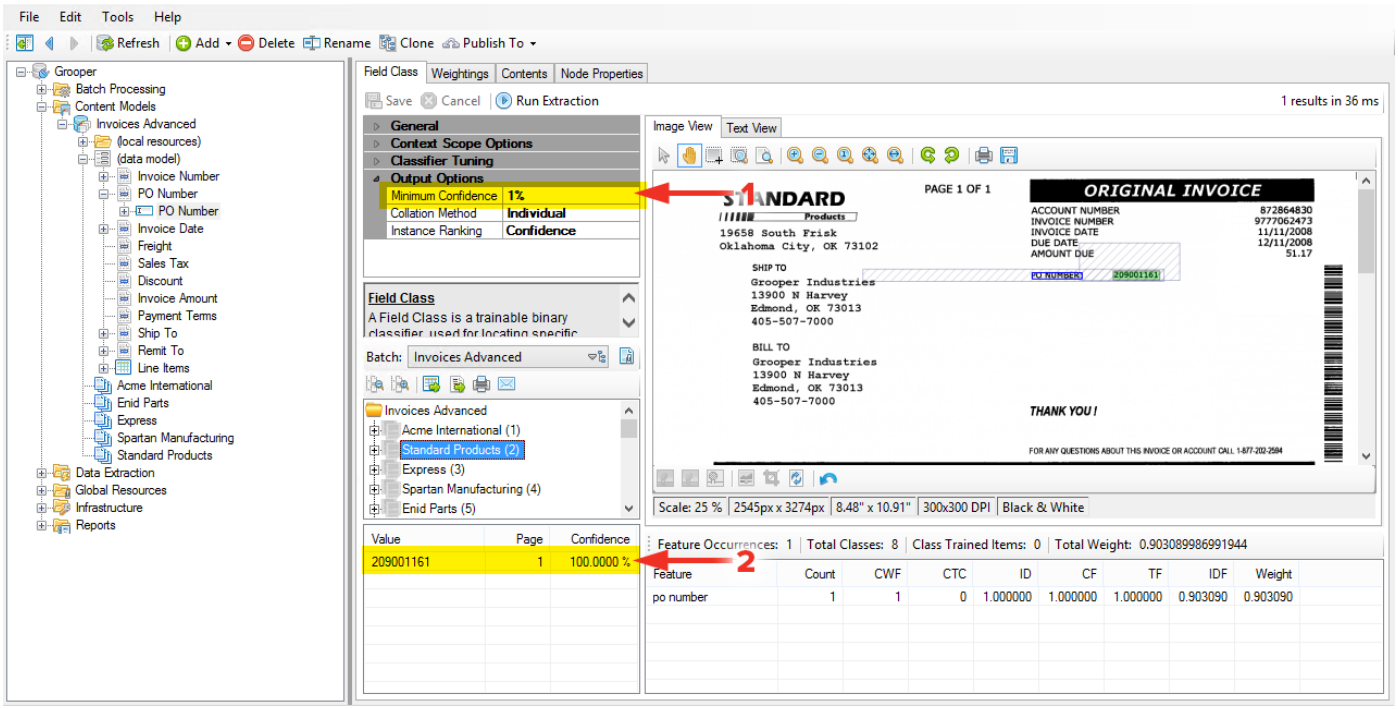
Checking extraction results against the Standard Products (2) document should reveal that **(1)** the Context Zones for this Field Class need to be adjusted due to Zone 2 picking up the unwanted features Due Date and Amount Due. **(2)** Use the following values for the Context Zones:

Zone 1 – Left: **-3.278** Top: **-0.054** Right: **0.312** Bottom: **0.054**
Zone 2 – Left: **-0.67** Top: **-0.37** Right: **0.5** Bottom: **0.04**



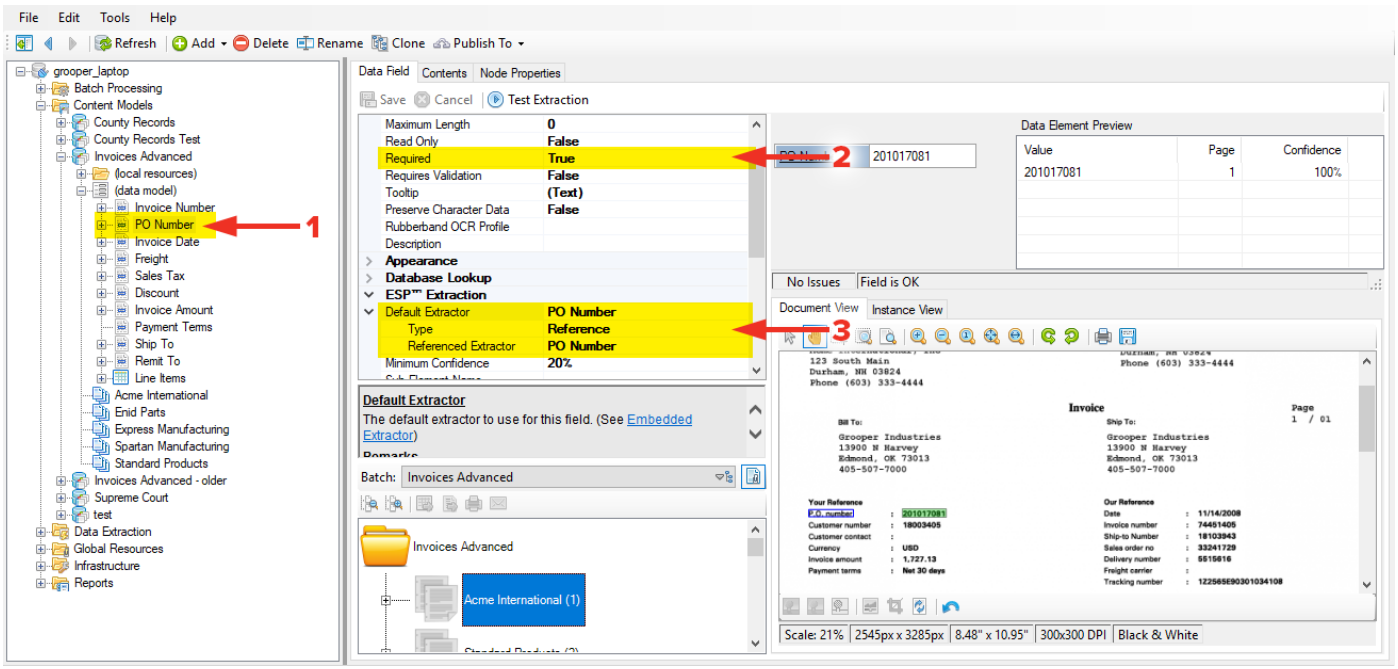
STEP 7 – SETTING MINIMUM CONFIDENCE

Because there is only one feature trained, and it is being returned at 100% confidence, **(1)** the Minimum Confidence can be set to anything above 0% **(2)** and all unwanted results will be trimmed.



STEP 8 – SETTING PROPERTIES FOR THE INVOICE NO DATA FIELD

(1) Select the PO Number Data Field and **(2)** set its Required property to True. **(3)** Set the Default Extractor Type to Reference and set the Referenced Extractor to the PO Number Field Class that is a child object of the PO Number Data Field.

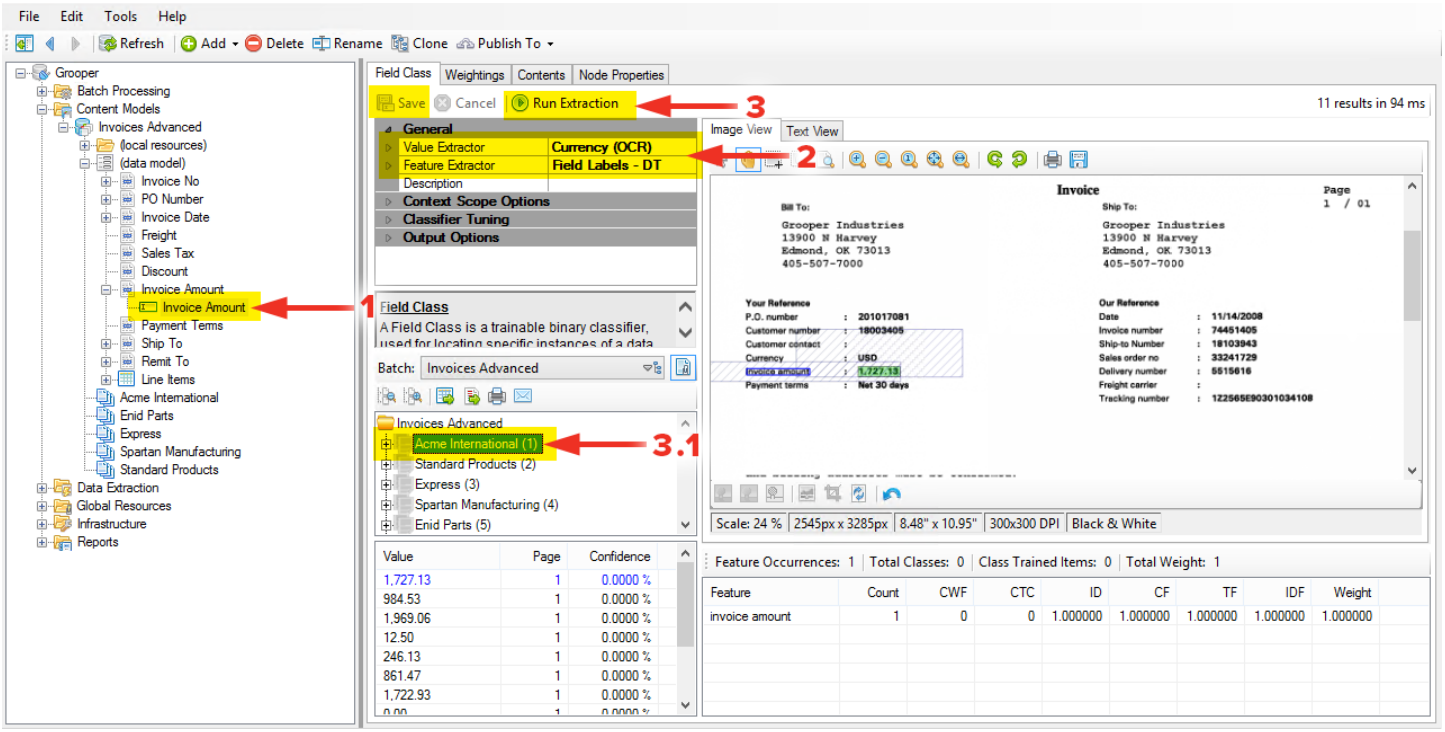


THE LAST FIELD CLASS

Finishing off the usage of the Field Class for this Data Model will end with the Invoice Amount. This Field Class will not introduce any new concepts, but is needed to complete extraction. Sometimes it is best, however, to realize you don't have to re-invent the wheel every time and just use tools that are available and achieve quick results.

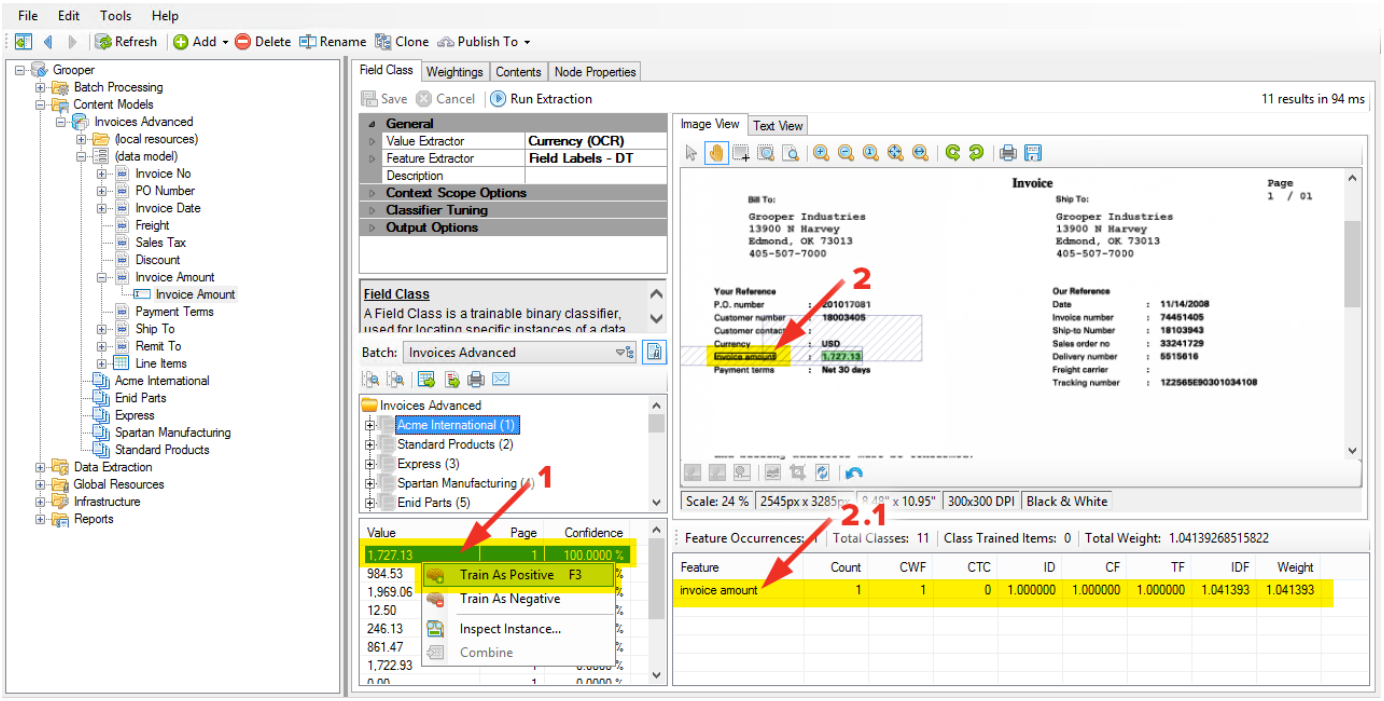
STEP 1 – ADDING THE INVOICE AMOUNT FIELD CLASS AND CHOOSE EXTRACTORS

(1) Add a Field Class as a child object to the Invoice Amount Data Field and name it Invoice Amount.
(2) Set the Value Extractor Type to Reference and the Referenced Extractor to the Currency (OCR) Data Type within the Data Extraction • Data Types > (system) area. Leave the Feature Extractor Type as Reference but change the Referenced Extractor to the Field Labels – DT Data Type within the Invoices Advanced • (local resources) area. **(3)** Save and Run Extraction on the Acme International (1) document.



STEP 2 – TRAIN THE INVOICE AMOUNT FEATURE

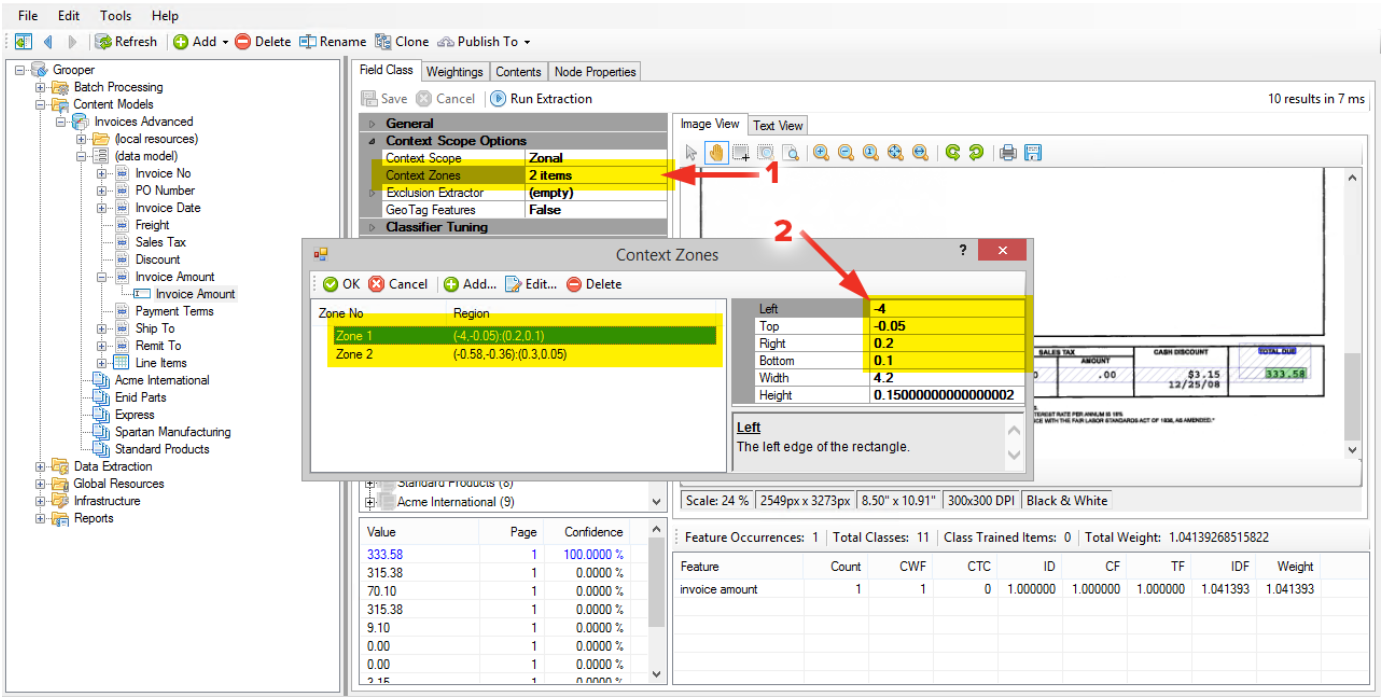
(1) Select the 1,727.13 value from the Values List View and **(2)** train its feature invoice amount.



STEP 3 – ADJUST CONTEXT ZONES

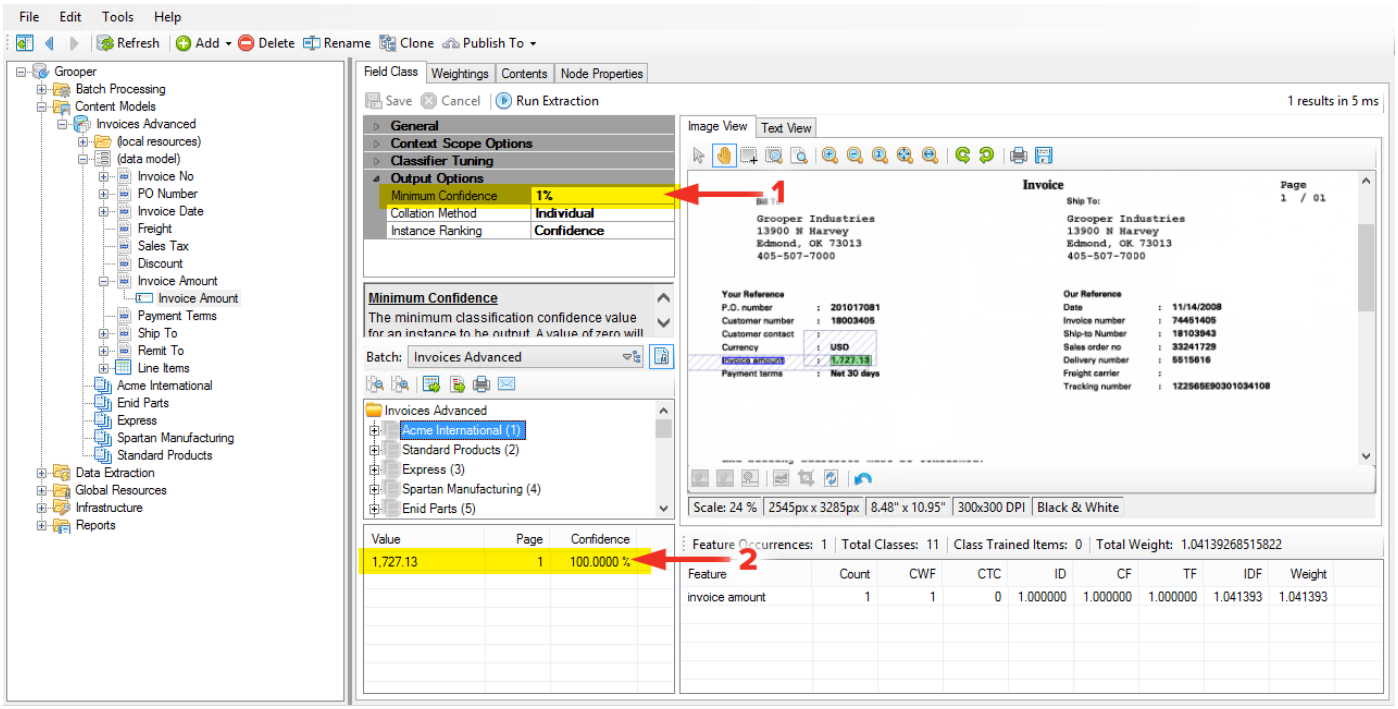
To prevent currency values that are near the trained feature from getting any confidence, **(1)** the Context Zones must be adjusted. **(2)** Use the following values for the Context Zones:

Zone 1 – Left: **-4** Top: **-0.05** Right: **0.2** Bottom: **0.1**
Zone 2 – Left: **-0.58** Top: **-0.36** Right: **0.3** Bottom: **0.05**



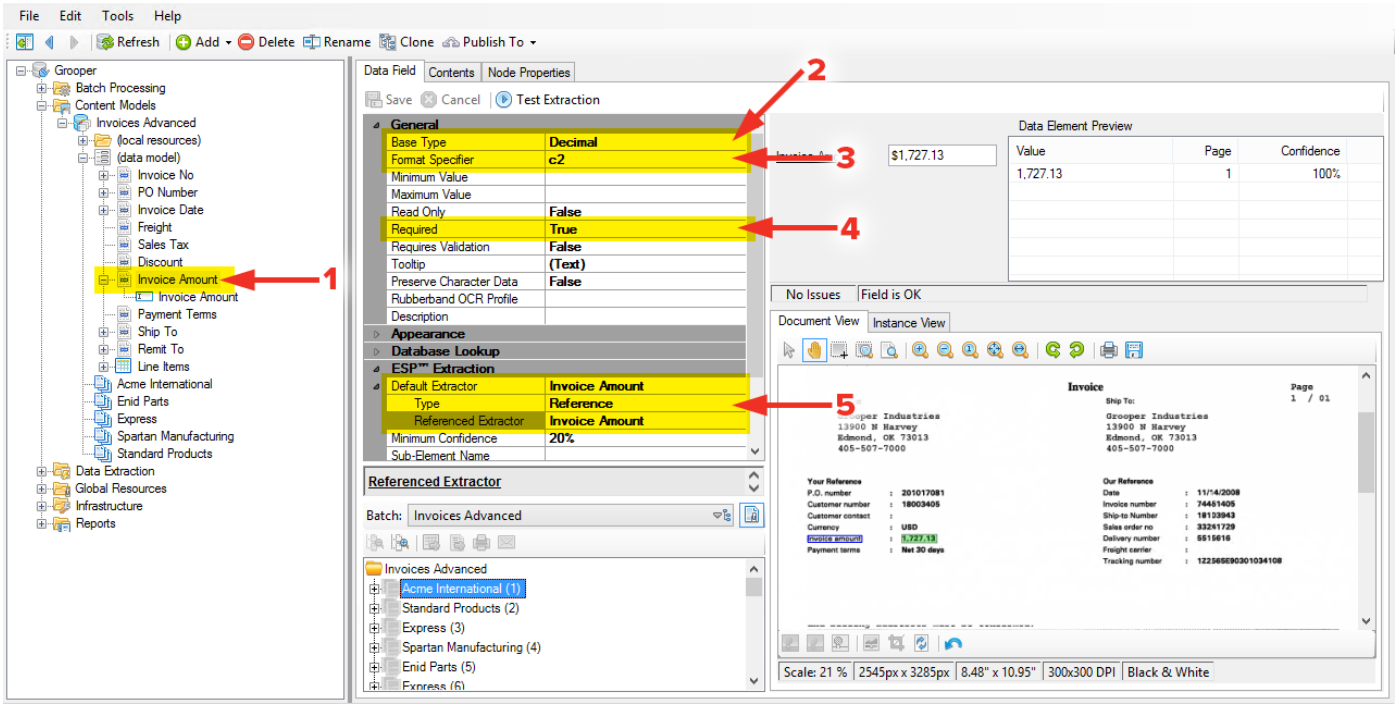
STEP 4 – SETTING MINIMUM CONFIDENCE

Because there is only one feature trained, and it is being returned at 100% confidence, **(1)** the Minimum Confidence can be set to anything **(2)** above 0% and all unwanted results will be trimmed.



STEP 5 – SETTING PROPERTIES FOR THE INVOICE NO DATA FIELD

(1) Select the Invoice Amount Data Field, **(2)** set the Base Type to Decimal, and **(3)** the Format Specifier to c2. **(4)** Set its Required property to True. **(5)** Finally, Set the Default Extractor Type to Reference and set the Referenced Extractor to the Invoice Amount Field Class that is a child object of the Invoice Amount Data Field.



THE REMAINING DATA FIELDS

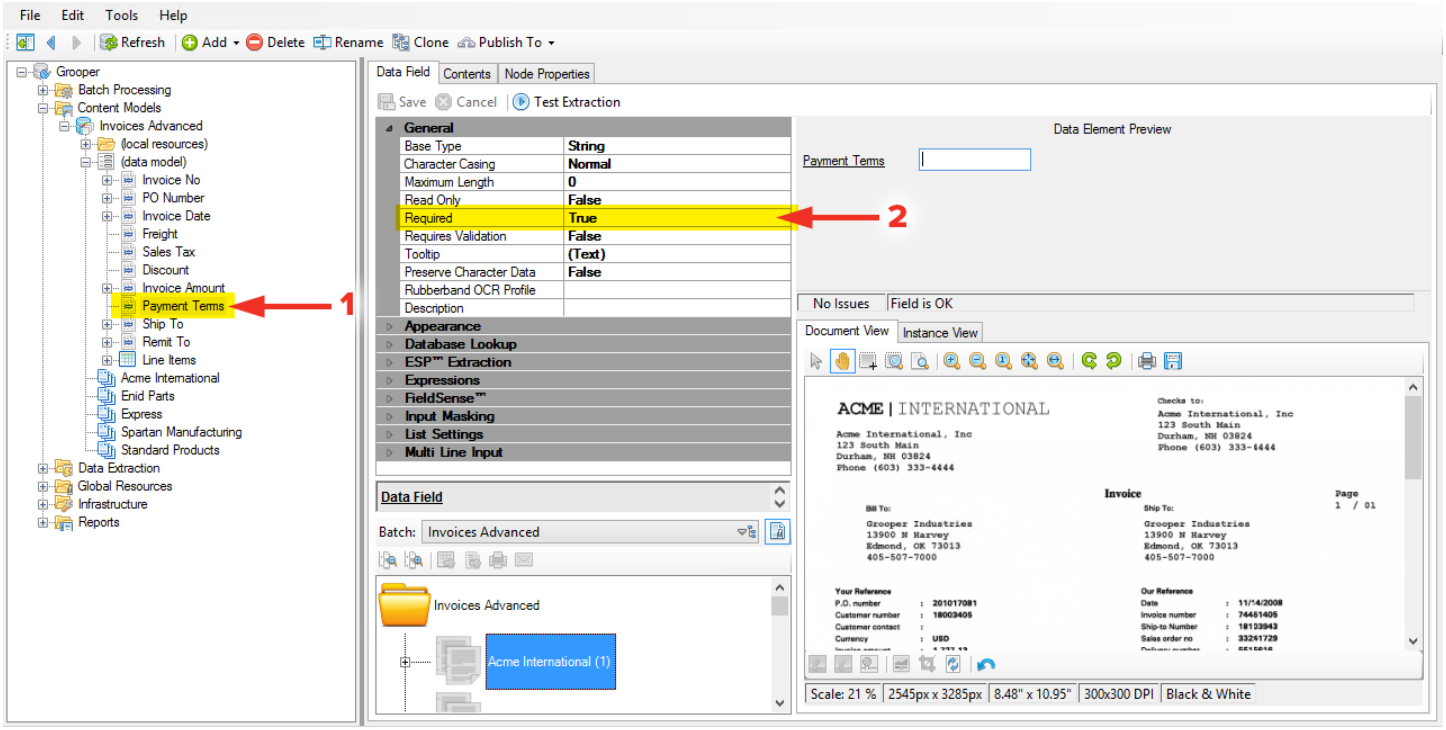
The last few Data Fields of this Data Model will not leverage a Field Class, but Data Types instead. The last Data Field that will be required is Payment Terms; the rest are not always present on the documents and don't require returned results. Because they are dollar amounts they can, however, be returned as $0.00, instead of being left blank. Finally, the concept of per-DocType overrides called Data Element Profiles will be introduced.

THE FINAL REQUIRED FIELD – PAYMENT TERMS

This information is present on all documents and desired as an extracted result. As such, it will be required, but the last of its type. Its setup will be very simple, and will not even reference an extractor.

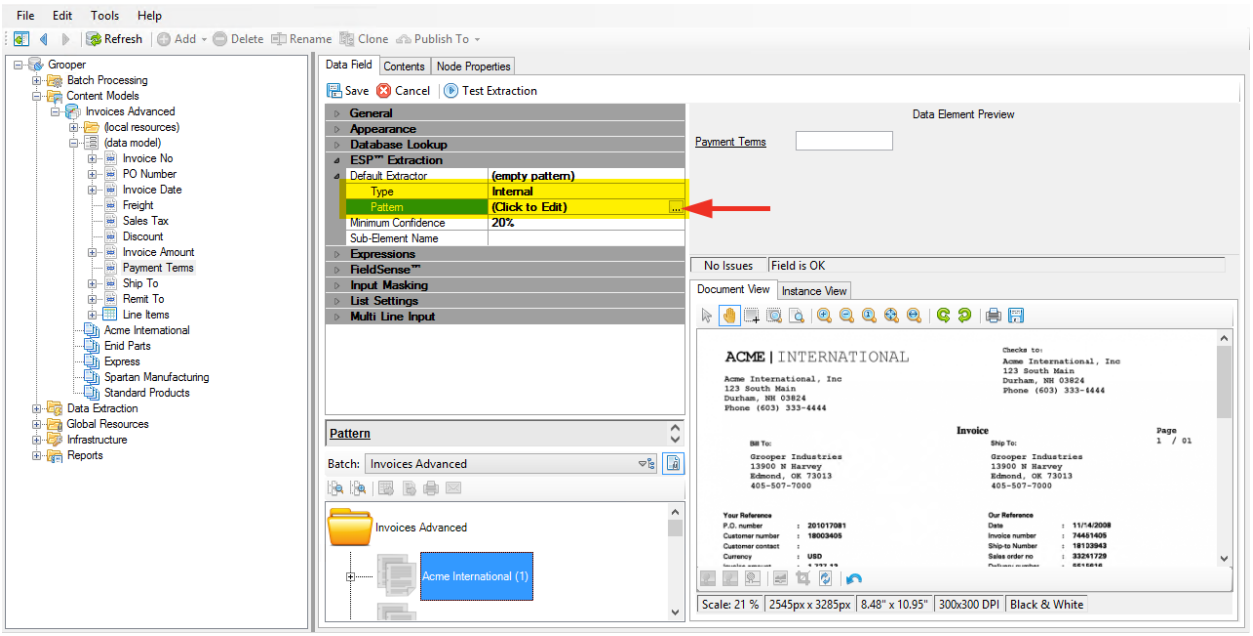
STEP 1 – SET REQUIRED PROPERTY

(1) Select the Payment Terms Data Field and, as mentioned, **(2)** set its Required Property to True.



STEP 2 – INTERNAL PATTERN ON DATA FIELD

To this point all Data Fields have referenced an extractor, but patterns can also be written directly to Data Fields. In the ESP Extraction section, set the Default Extractor Type to Internal. Click the ellipsis button on the Pattern property to bring up a Pattern Editor window.



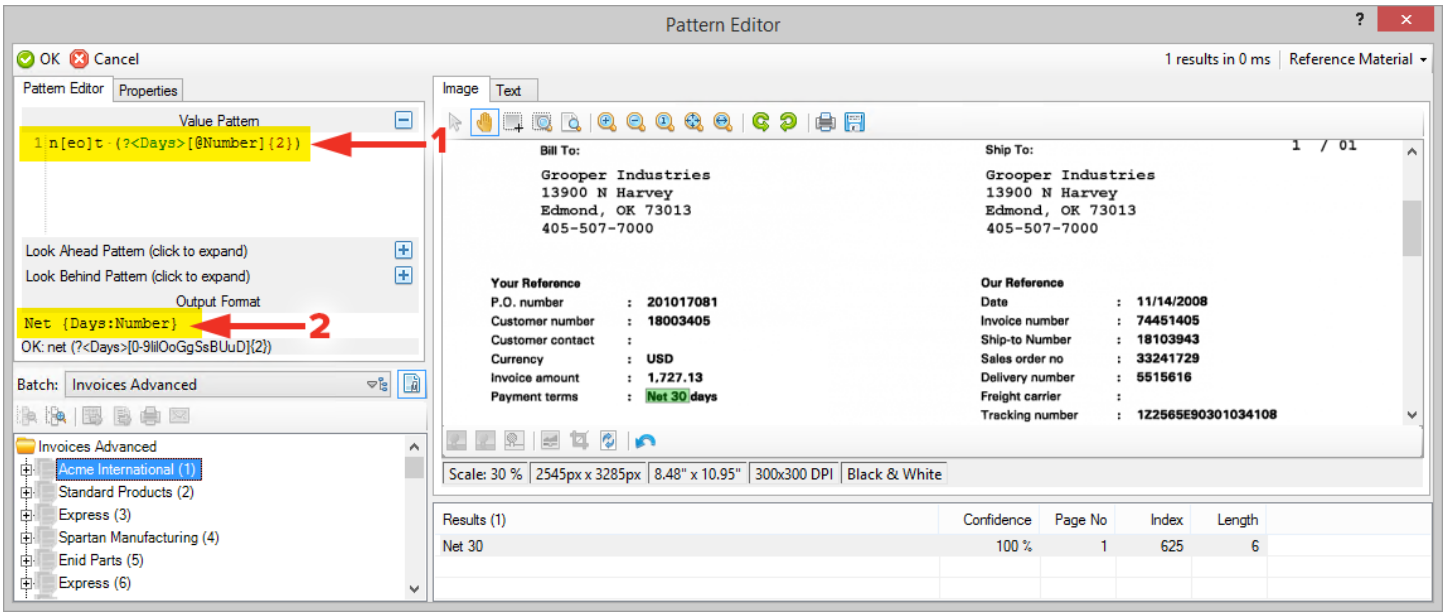
STEP 3 – VALUE PATTERN AND OUTPUT FORMAT

(1) For the Value Pattern use the following:

```
n[eo]t (?<Days>[@Number]{2})
```

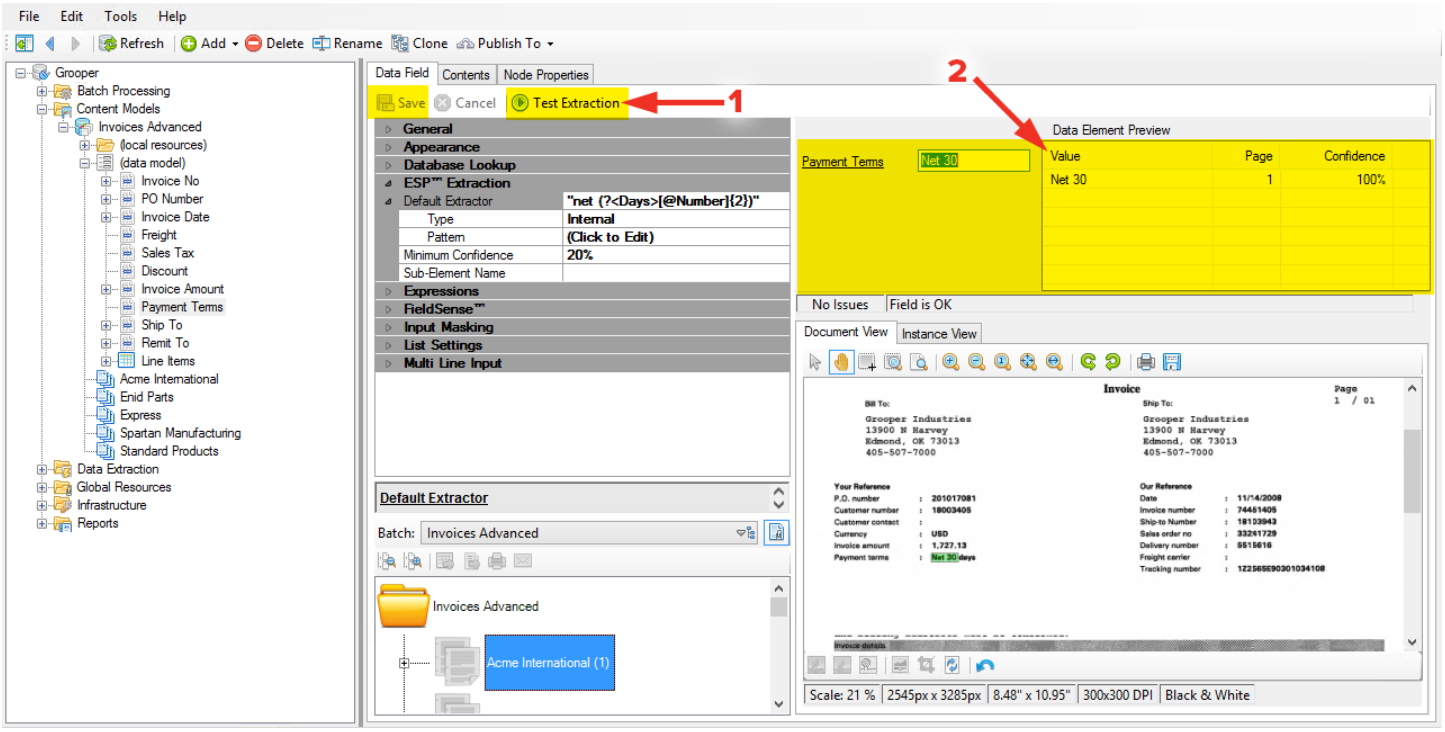
(2) For the Output Format use the following:

```
Net {Days:Number}
```



STEP 4 – SAVE AND RUN EXTRACTION

Click OK to close the Pattern Editor window, then **(1)** Save and Test Extraction to **(2)** see results.

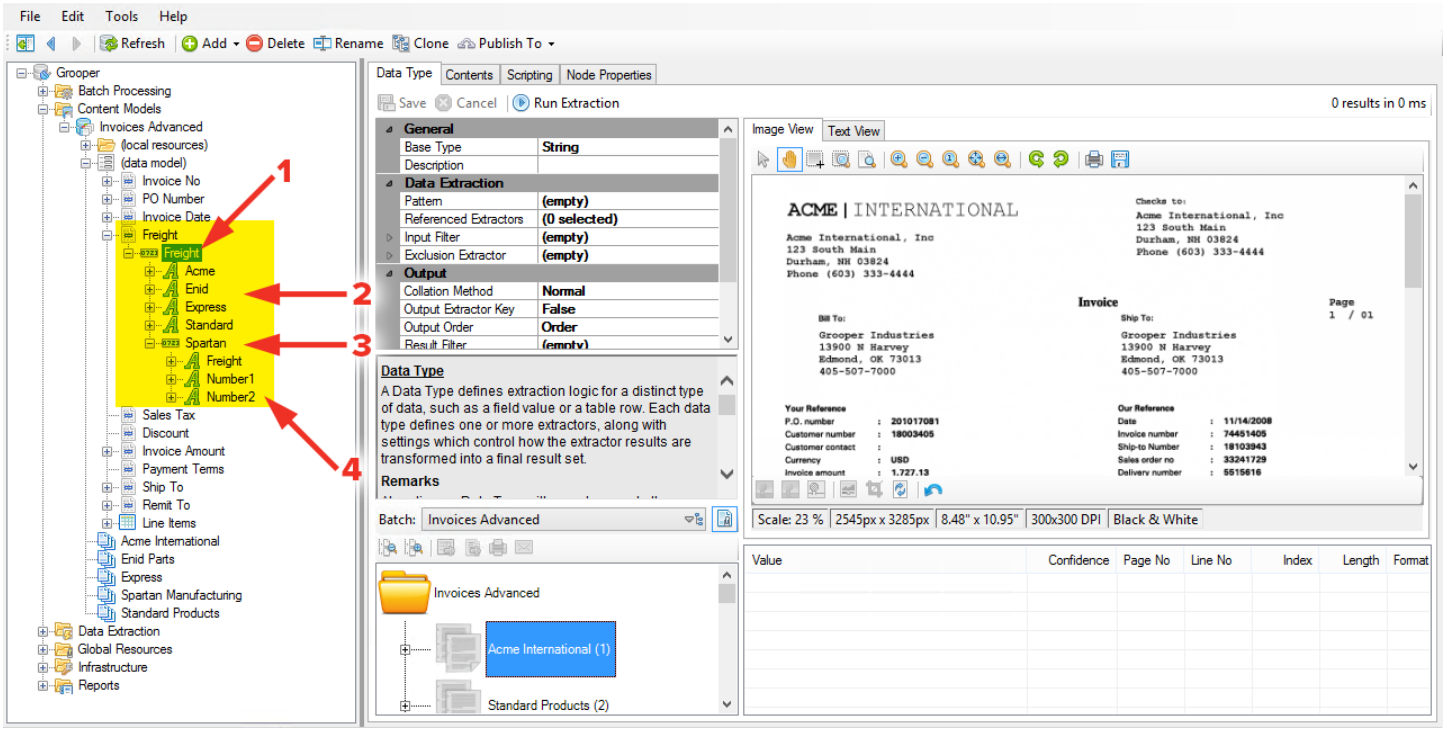


SETTING UP THE FREIGHT DATA FIELD

The extractor built for this field is somewhat complex, but is all based on concepts that have been seen already, except for an Ordered Array that returns a summed result.

STEP 1 – ADDING ALL OF THE PARTS

(1) Add a Data Type named Freight as a child object to the Freight Data Field. **(2)** Add four child Data Formats and name them Acme, Enid, Express, and Standard respectively. **(3)** Add a Data Type named Spartan as a child object to the Freight Data Type. **(4)** Add three Data Formats to this Data Type and name them Freight, Number1, and Number2.



STEP 2 – SETTING UP THE ACME DATA FORMAT

(1) Make sure tab marking is enabled, and for the Value Pattern enter the following:

```
(?<Freight>[@Number.]{3,12})
```

(2) For the Look Ahead Pattern enter the following:

```
WS.FREIGHT[0o]231[^\n]+\n
[^\n]+\n
[^\t]+\t
[^\t]+\t
```

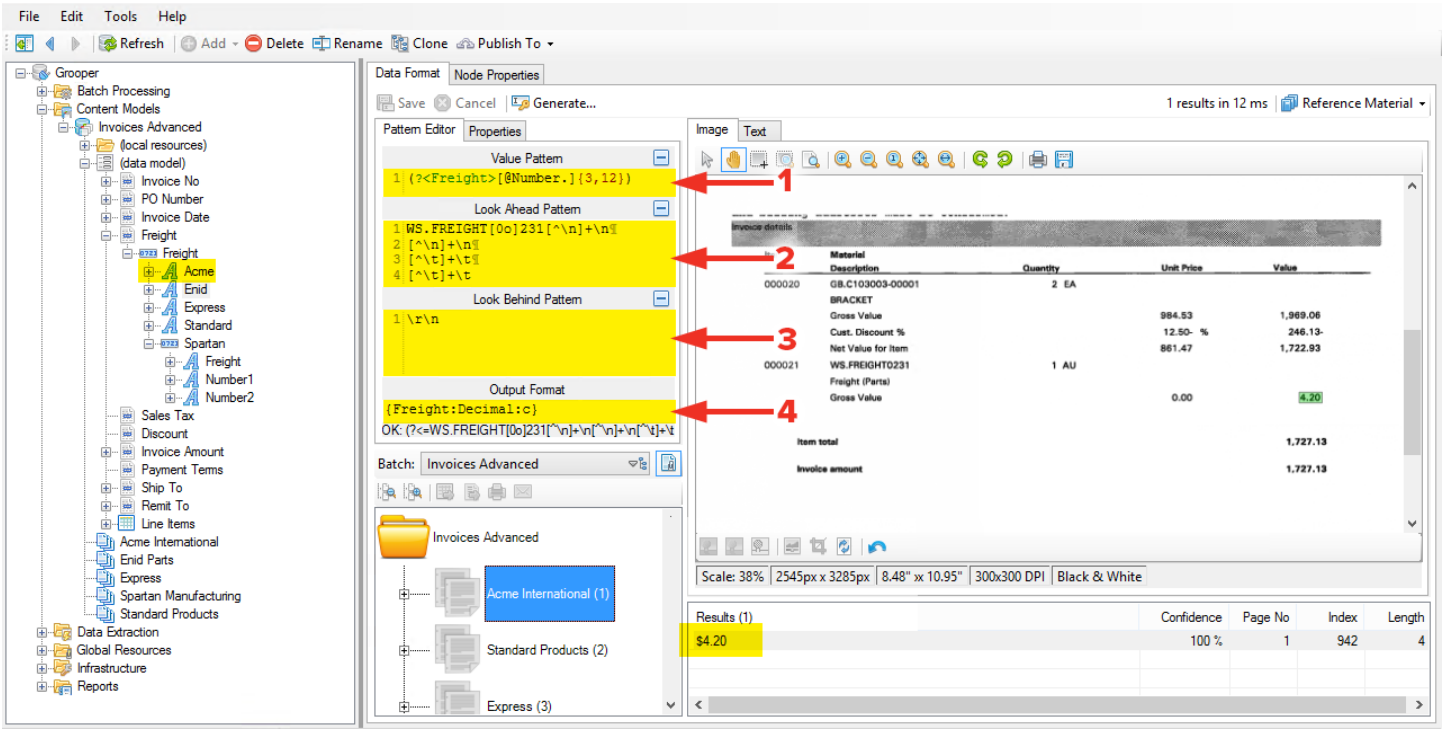
(3) For the Look Behind Pattern enter the following:

```
\r\n
```

(4) For the Output Format enter the following:

```
{Freight:Decimal:c}
```

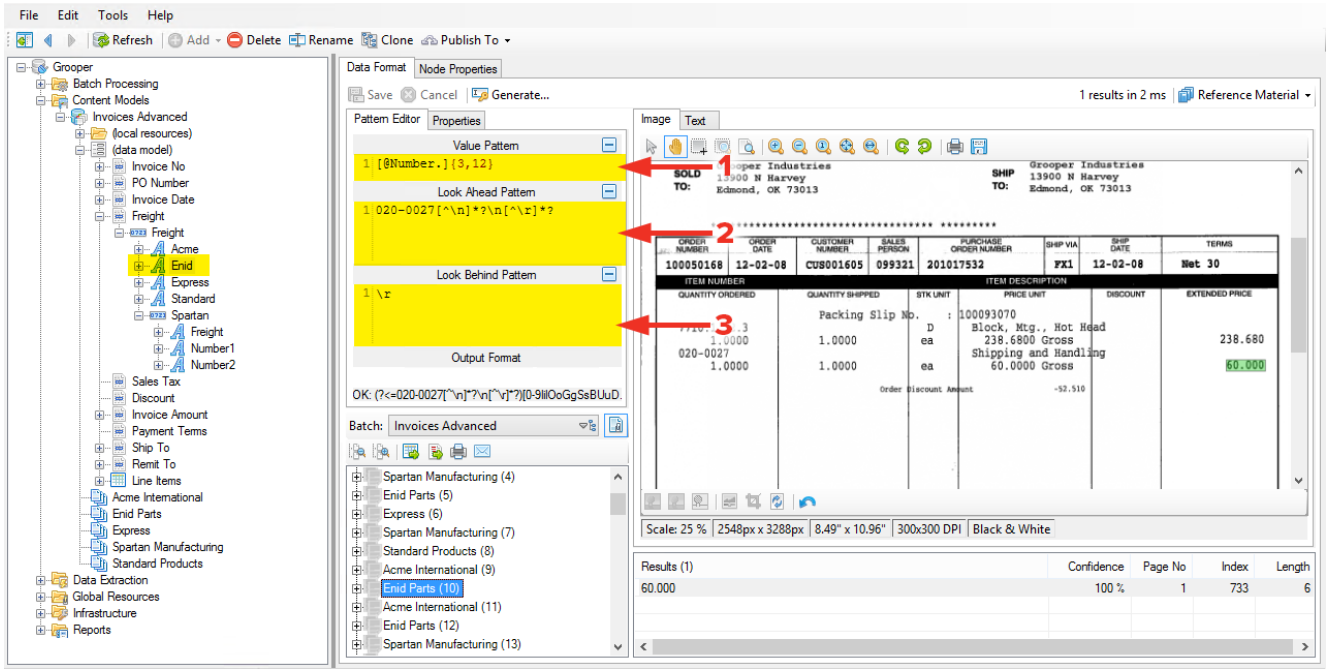
This Output Format will not only return the named group Freight, and force the results from the @Number Expression Variable to return decimal results, but will return it as a currency amount due to the format specifier c. Finally, enable Tab Marking from the Properties tab within the Text Preprocessing properties.



STEP 3 – SETTING UP THE ENID DATA FORMAT

(1) For the Enid Data Format Value Pattern, enter the following:
(2) For the Look Ahead Pattern enter the following:
(3) For the Look Behind Pattern enter the following:
Use Enid Parts (10) to see a result.

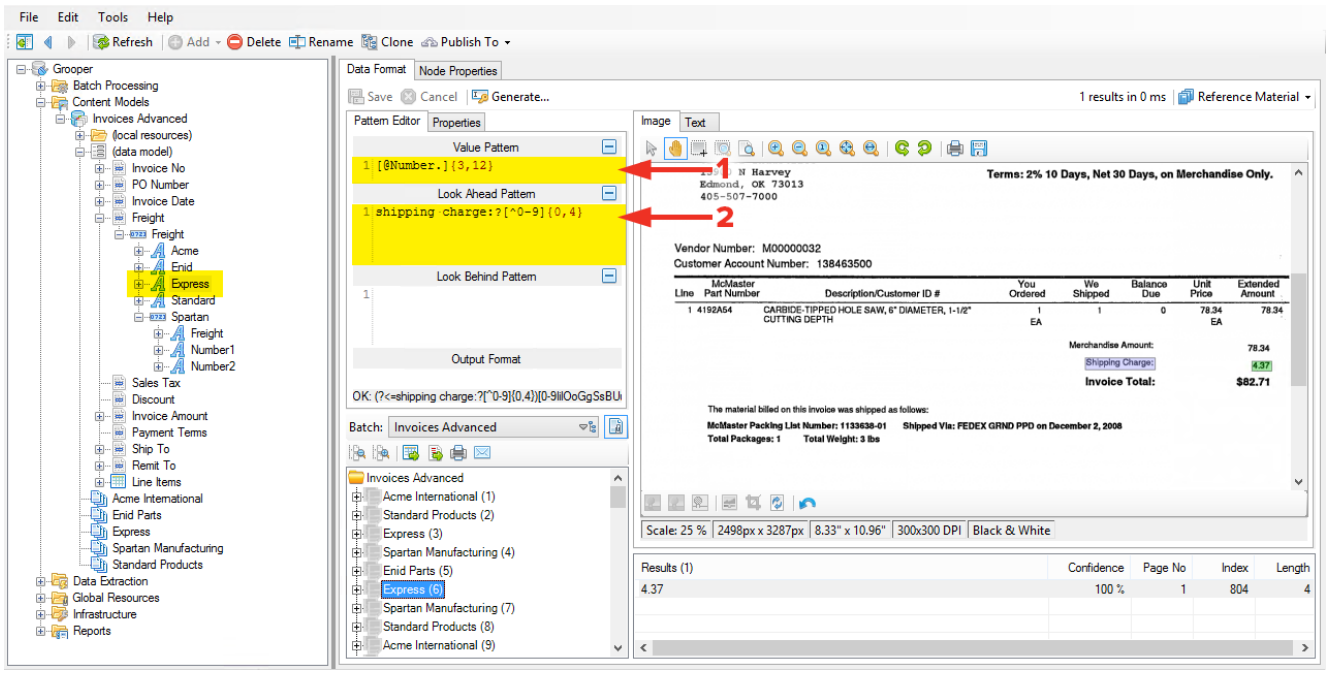
```
[@Number.]{3,12}
020-0027[^\n]*?\n[^\r]*?
\r
```



STEP 4 – SETTING UP THE EXPRESS DATA FORMAT

(1) For the Express Data Format Value Pattern, enter:
(2) For the Look Ahead Pattern enter the following:
Use Express Manufacturing (6) to get a result.

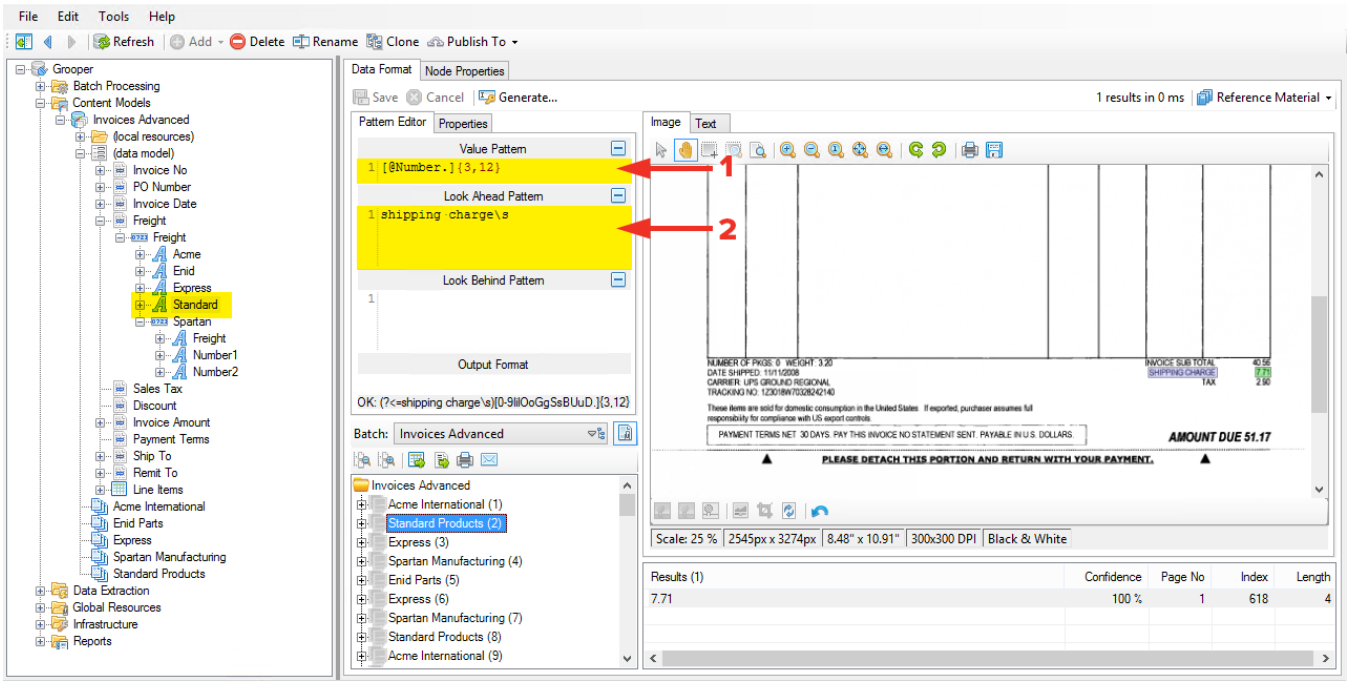
```
[@Number.]{3,12}
shipping charge:?[^0-9]{0,4}
```



STEP 5 – SETTING UP THE STANDARD DATA FORMAT

(1) For the Standard Data Format Value Pattern, enter the following: `[@Number.]{3,12}`

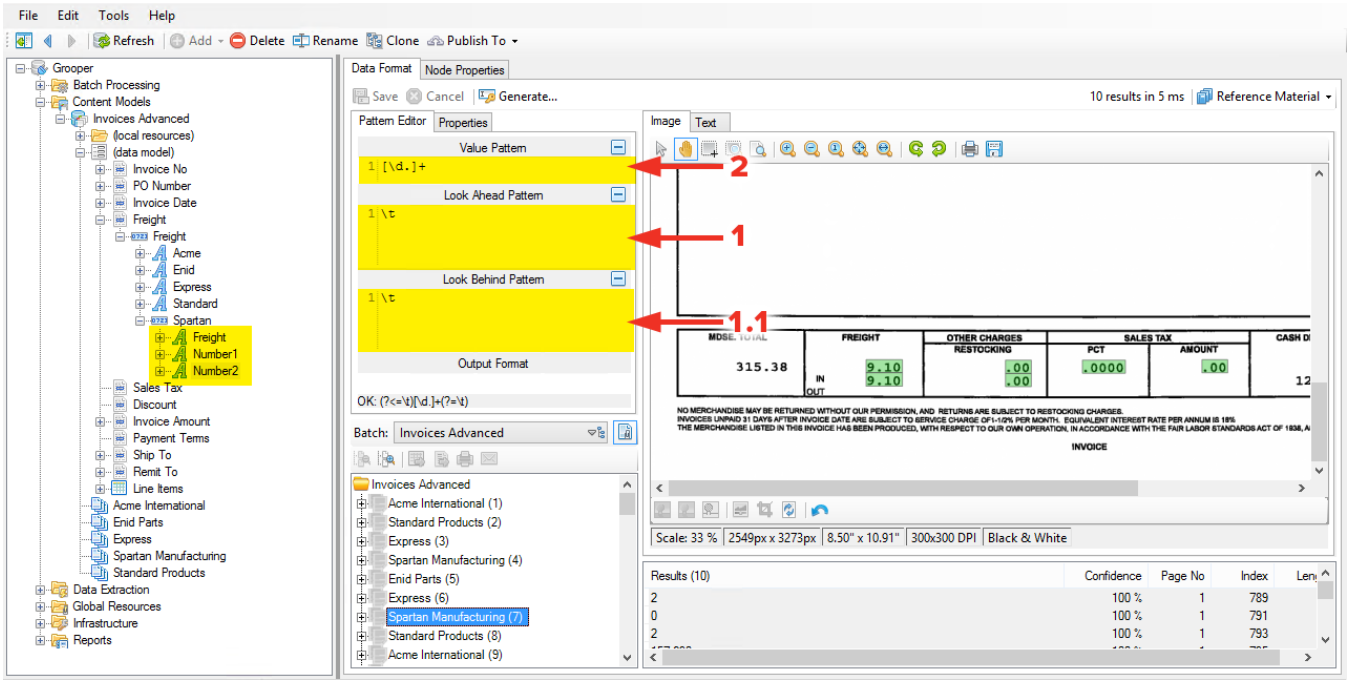
(2) For the Look Ahead Pattern enter the following: `shipping charge\s`



STEP 6 – SETTING UP THE SPARTAN DATA FORMATS

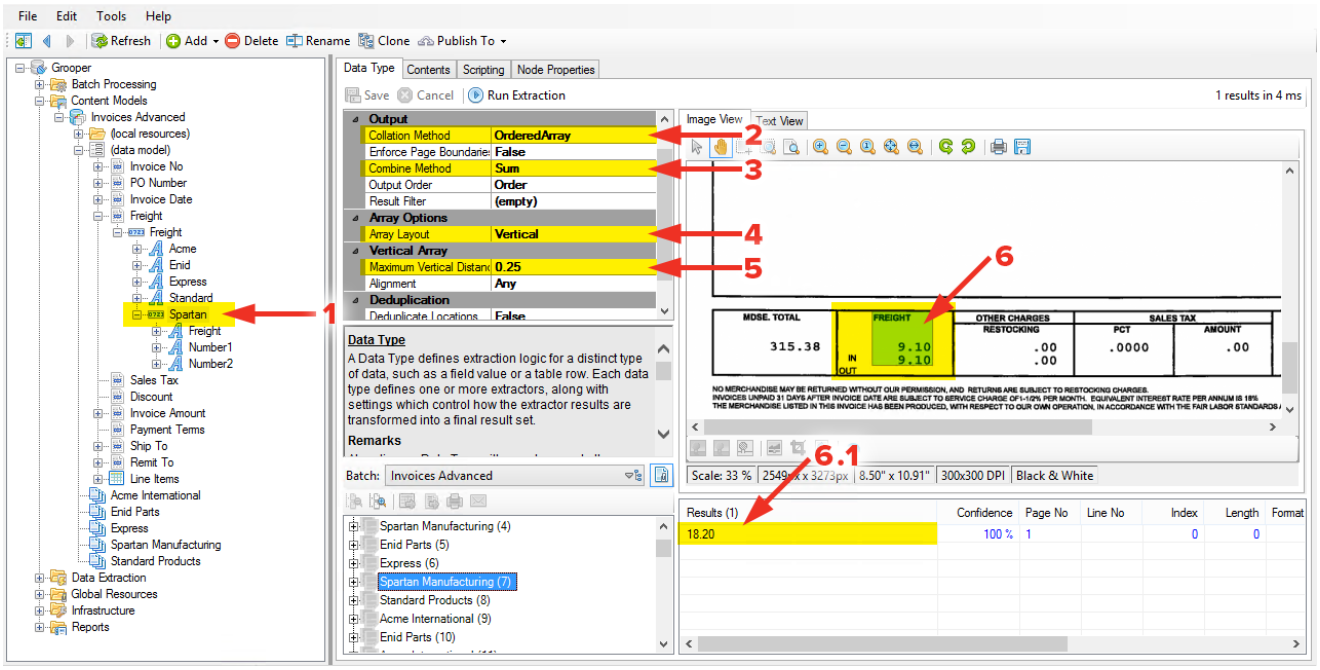
(1) For the Freight Data Format enable Tab Marking and use a tab for the Look Ahead Pattern and the Look Behind Pattern; and for the Value Pattern simply use the string: `freight`.

(2) The Number1 and Number2 Data Formats are the same, so enable Tab Marking for them, use a tab for the Look Ahead Pattern and Look Behind Pattern, and for the Value Pattern use: `[\d.]+`



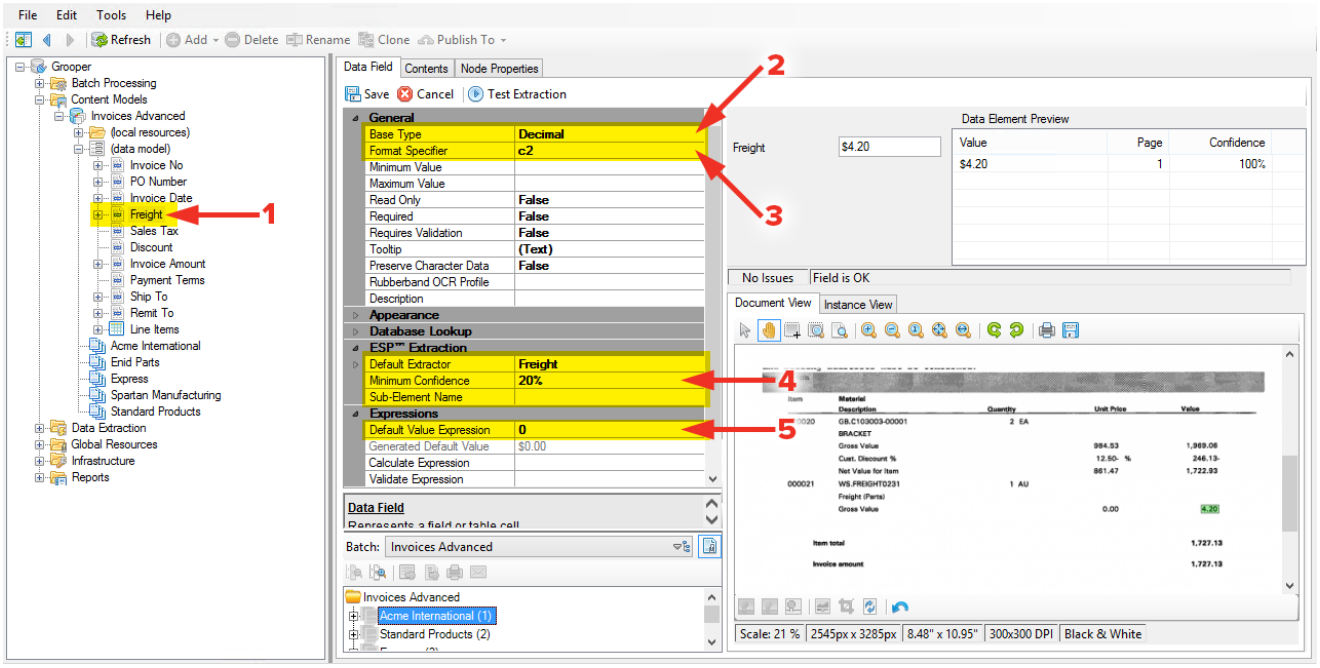
STEP 7 – OUTPUT OPTIONS FOR THE SPARTAN DATA TYPE

(1) Select the Spartan Data Type and **(2)** change the Collation Method to OrderedArray. **(3)** Set the Combine Method as Sum and **(4)** change the Array Layout to Vertical only. **(5)** Finally, set a Maximum Vertical Distance of 0.25. **(6)** Notice the result of the two 9.10 values being added up to 18.20.



STEP 8 – SETTING THE PROPERTIES OF THE FREIGHT DATA FIELD

(1) Select the Freight Data Field, **(2)** set its Base Type to Decimal, and **(3)** its Format Specifier to c2. **(4)** Set the Default Extractor Type to Reference and the Referenced Extractor to the Freight Data Type that is the child of the Freight Data Field. (5) Finally, in the Expressions section, set the Default Value Expression to 0. Since this value isn't required, when a result isn't returned, it will at least present $0.00.

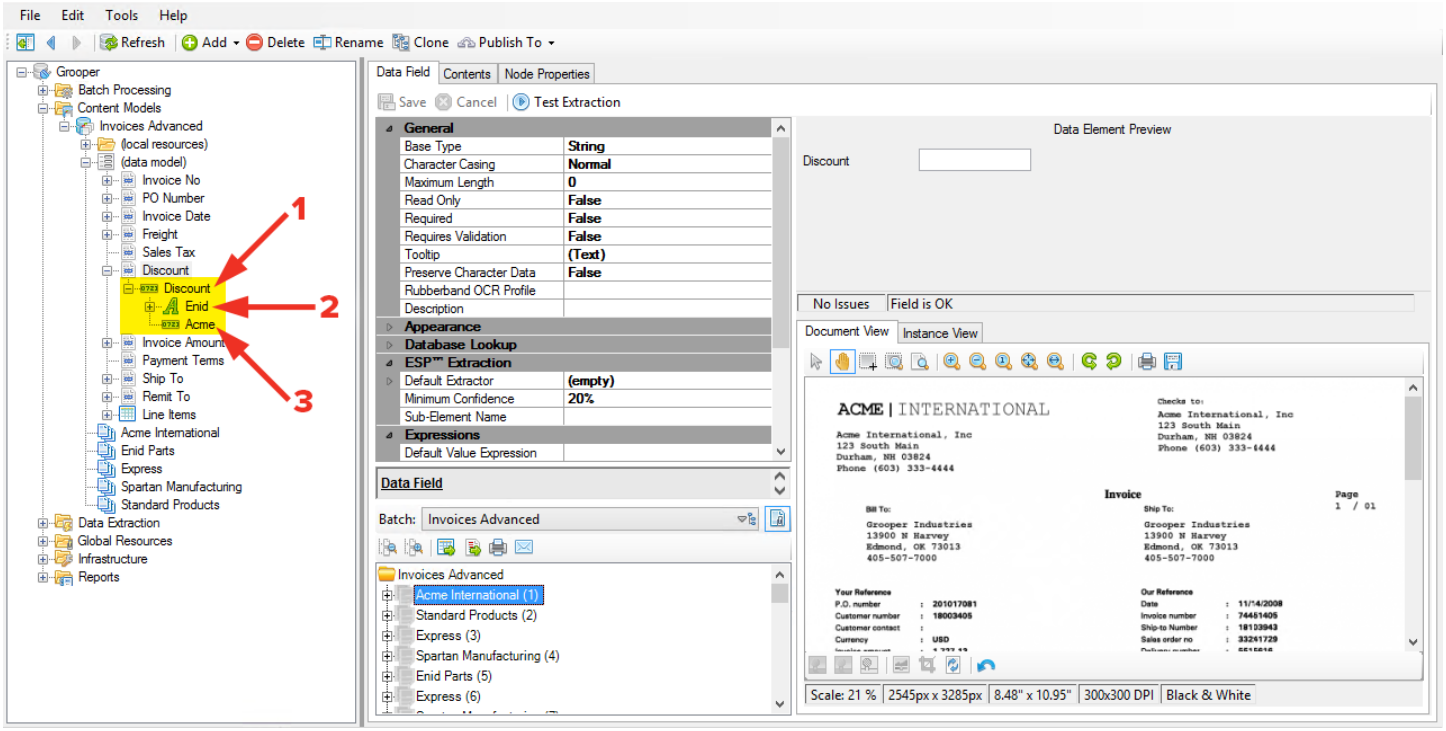


SETTING UP THE DISCOUNT DATA FIELD

This will be another Data Type setup with a combination Data Format/Data Type sibling situation, but a bit simpler than the previous example. The Enid Parts and Acme International documents are the only ones that have discounts, so this extractor will focus on them.

STEP 1 – ADDING ALL OF THE PARTS

(1) Add a Data Type as a child object to the Discount Data Field and name it Discount. **(2)** Add a Data Format to this Data Type and name it Enid. **(3)** Also, as a child object to the Discount Data Type, add another Data Type and name it Acme.



STEP 2 – SETTING UP THE ENID DATA FORMAT

(1) Select the Enid Data Format, and for the Value Pattern use:

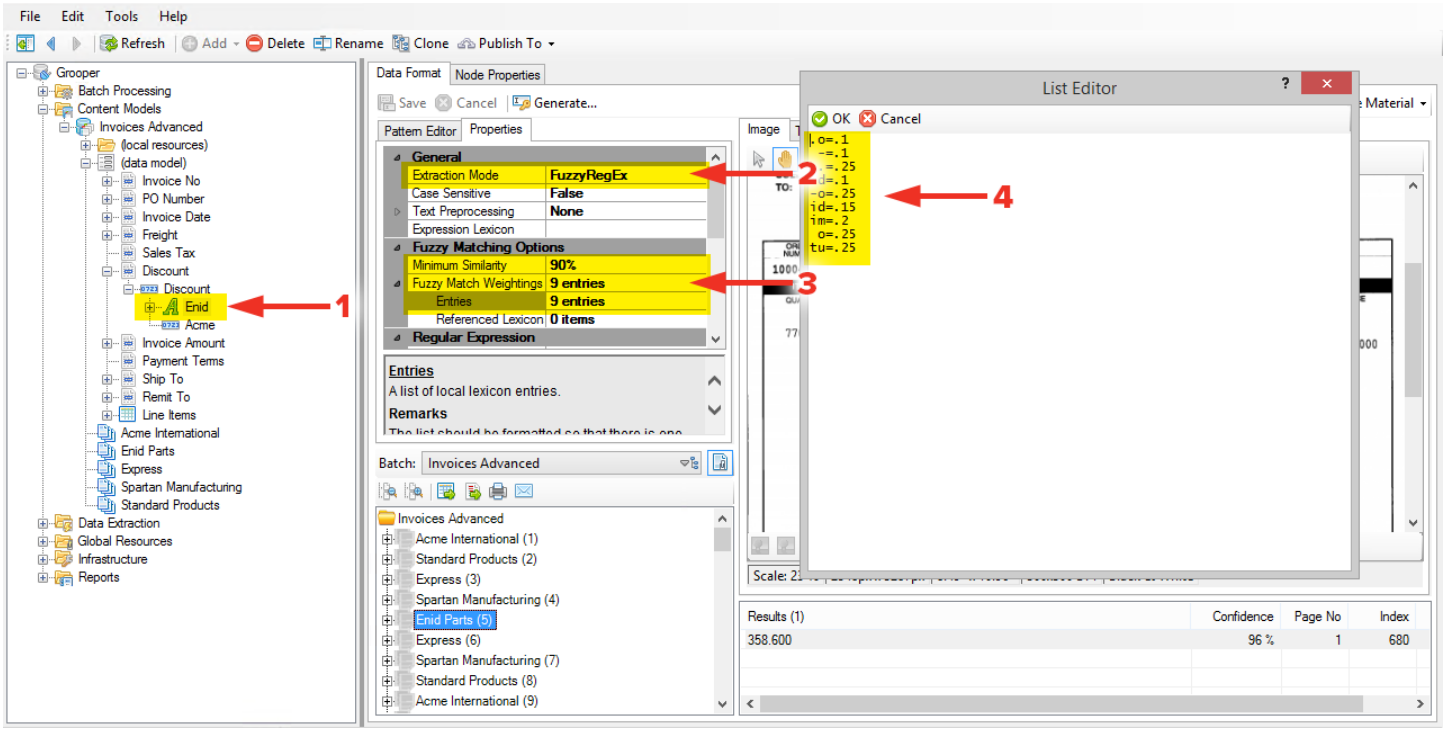
```
(?<Discount>\d{1,6}\.\d{2,3})
```

For the Look Ahead Pattern use:

```
order discount amount –
```

(2) On the Properties tab, set the Extraction Mode to FuzzyRegEx **(3)** and leave the Minimum Similarity at 90%. Select the Entries property and use the ellipsis button to bring up the List Editor window, and within it, **(4)** enter the following (the spaces at the beginning of the first three entries won't property copy from a PDF):

```
 .=.25
 -=.1
 o=.25
.o=.1
-o=.25
ld=.1
id=.15
im=.2
tu=.25
```



The string in the Look Ahead Pattern falls in a bad place on the Enid documents due to the lines of the form. These lines throw off OCR pretty badly, and cause a lot of bad characters to be read. The settings used above help compensate for this issue.

STEP 3 – SETTING UP THE ACME DATA TYPE

Select the Acme Data Type and for the Value Pattern use the following:

```
[@Number.,]{3,12}
```

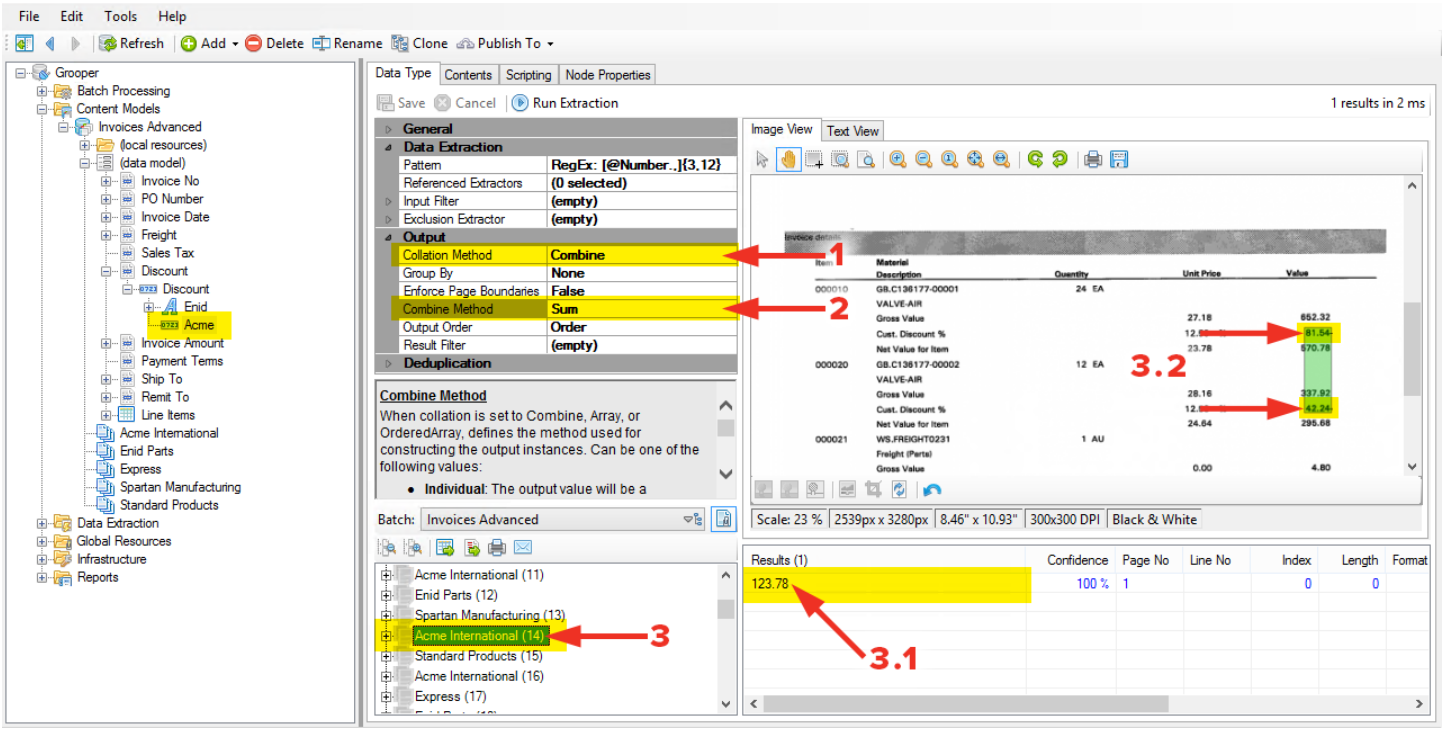
For the Look Ahead Pattern use the following:

```
cust\.\sdiscount\s%(\t|\r\n)[^\t]+\t
```

For the Output Format use the following:

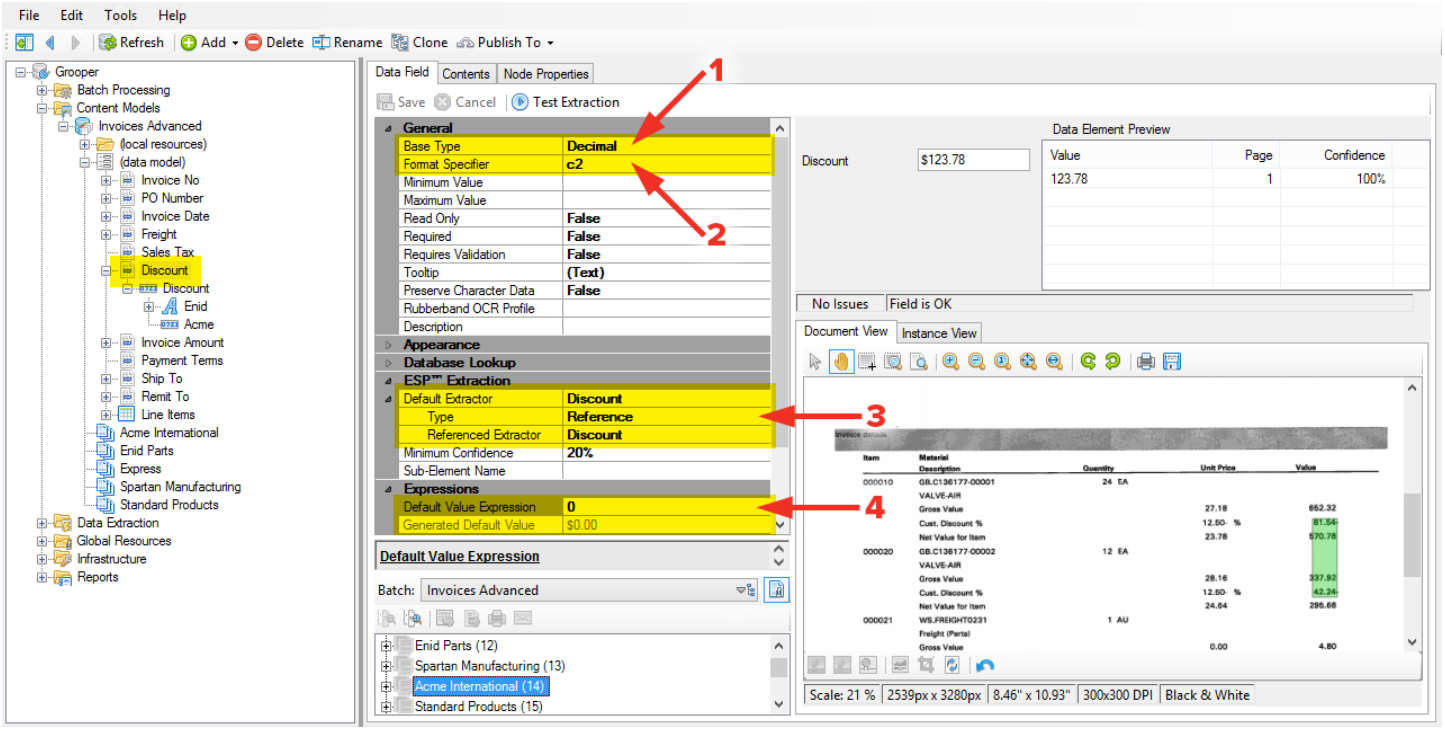
```
{0:Number}
```

Make sure Tab Marking is enabled in the Text Preprocessing area of the General section on the Properties tab.
(1) In the Output section set the Collation Method to Combine **(2)** and the Combine Method to Sum.
(3) Notice on Acme International (14) that this setup will find two discount amounts and add them together.



STEP 4 – SETTING UP THE PROPERTIES FOR THE DISCOUNT DATA FIELD

(1) Select the Discount Data Field and set the Base Type to Decimal **(2)** and the Format Specifier to c2. **(3)** Set the Default Extractor Type to Reference and set the Referenced Extractor to the Discount Data Type that is a child of the Discount Data Field. **(4)** Set the Default Value Expression to 0.

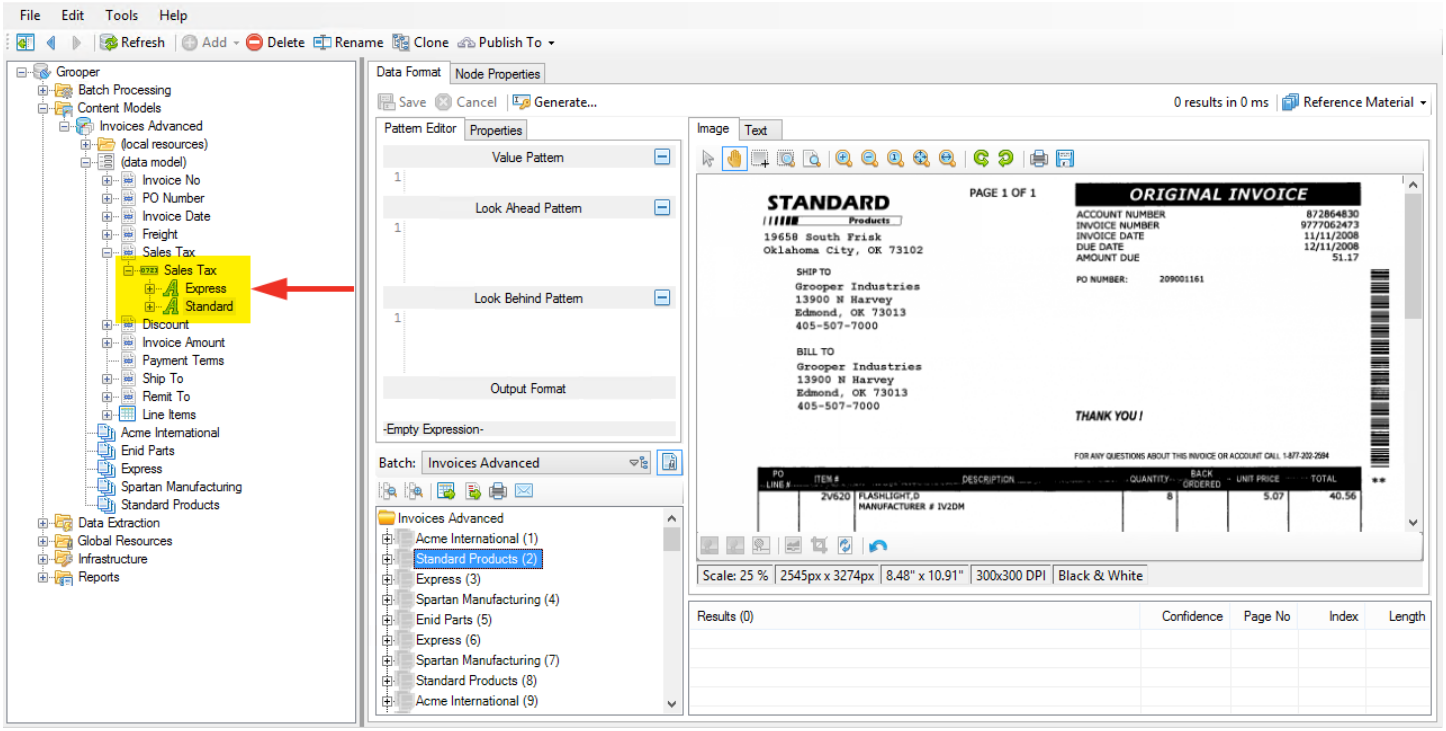


SETTING UP THE SALES TAX DATA FIELD – DATA ELEMENT PROFILE OVERRIDES

The final field of information to setup within the Data Model is Sales Tax. A Data Type will be used to extract information for the Express and Standard documents as they contain Sales Tax information. Enid Parts, however, will be best suited to use a Field Class as its extractor, and because a Data Field can only have one Default Extractor it will have to be overwritten for that specific DocType. This will be accomplished by a profile being added to the Enid Parts Data Element Profies.

STEP 1 – ADDING ALL OF THE PARTS

Add a child Data Type to the Sales Tax Data Field and name it Sales Tax. Add two Data Formats as child objects to the Sales Tax Data Type and name them Express and Standard.

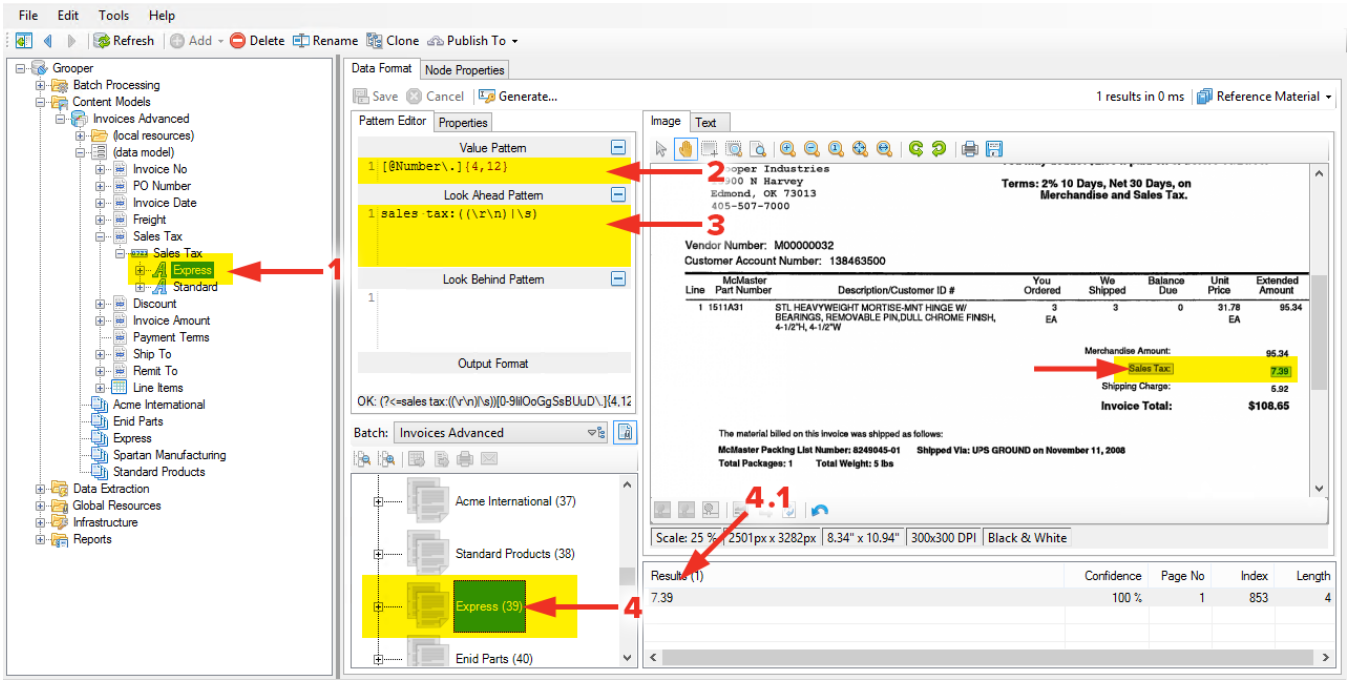


STEP 2 – SETTING UP THE EXPRESS DATA FORMAT

(1) Select the Express Data Format and **(2)** for the Value Pattern use:
(3) For the Look Ahead Pattern use the following:

```
[@Number.]{4,12}
sales tax:((\r\n)|\s)
```

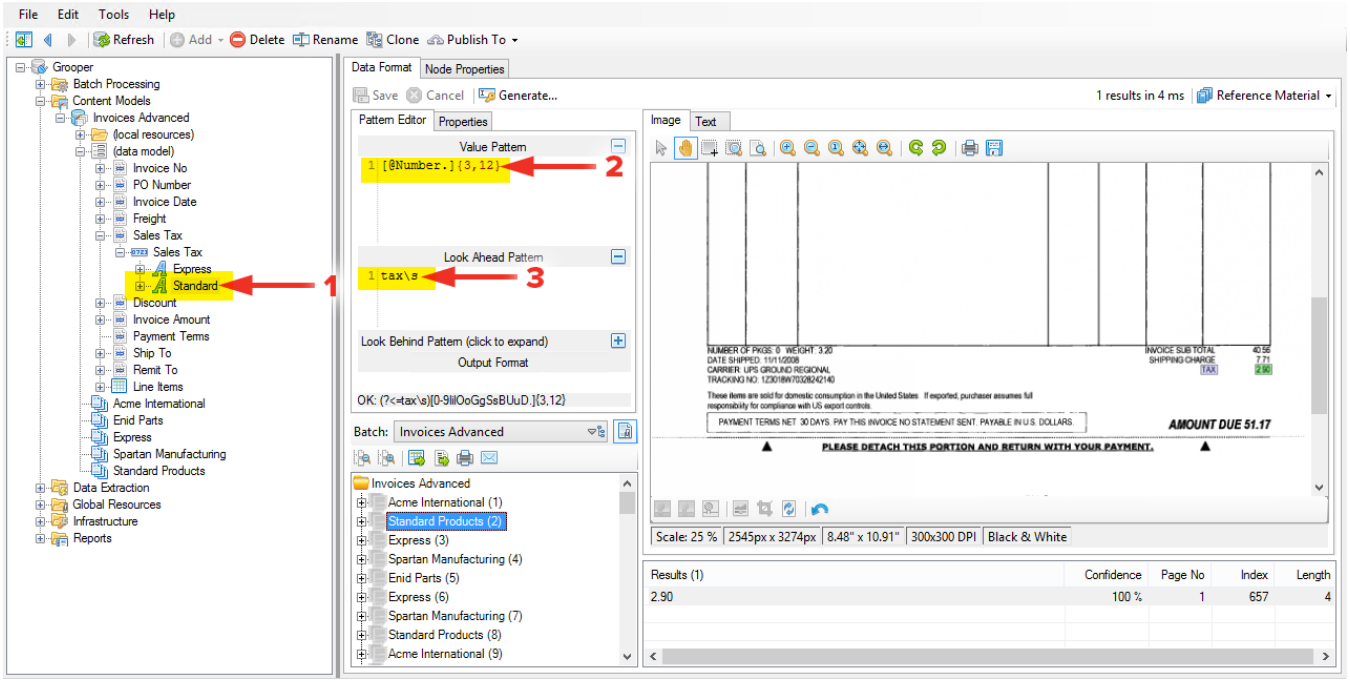
Sales Tax isn't a frequent occurrence, so don't be surprised if you don't **(4)** see a result until about Express (39).



STEP 3 – SETTING UP THE STANDARD DATA FORMAT

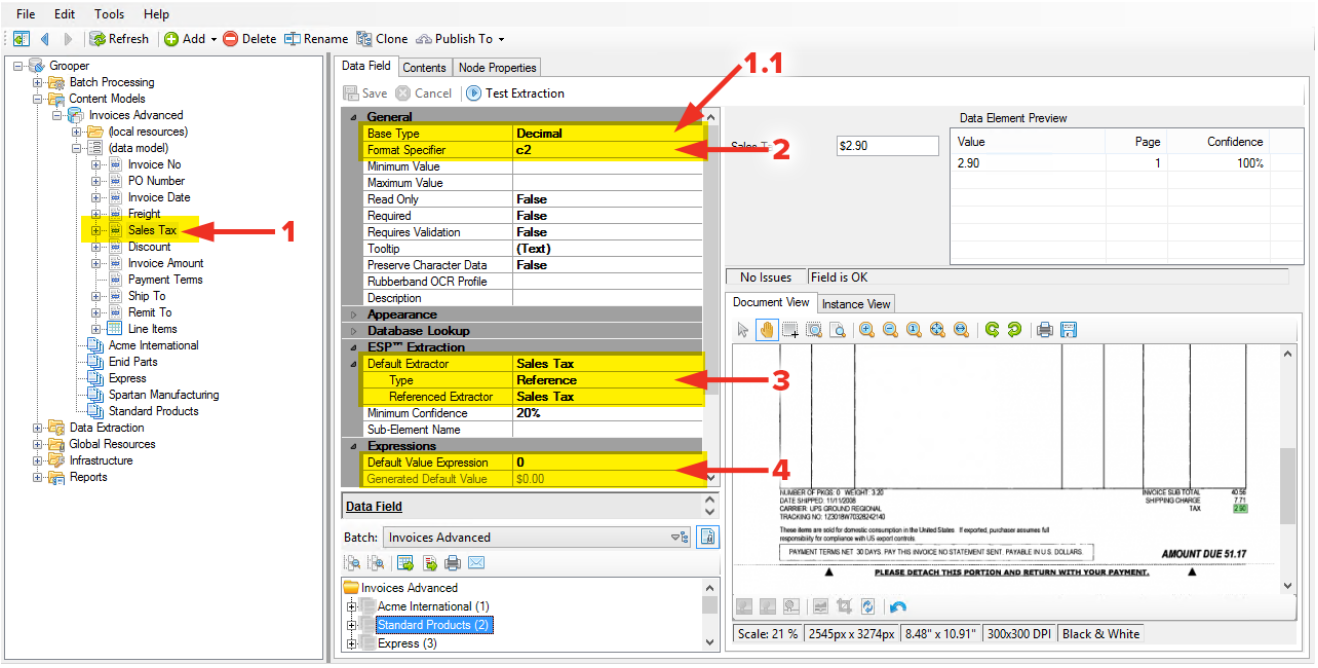
(1) Select the Standard Data Format and **(2)** for the Value Pattern use:
(3) For the Look Ahead Pattern use the following:

```
[@Number.]{3,12}
tax\s
```



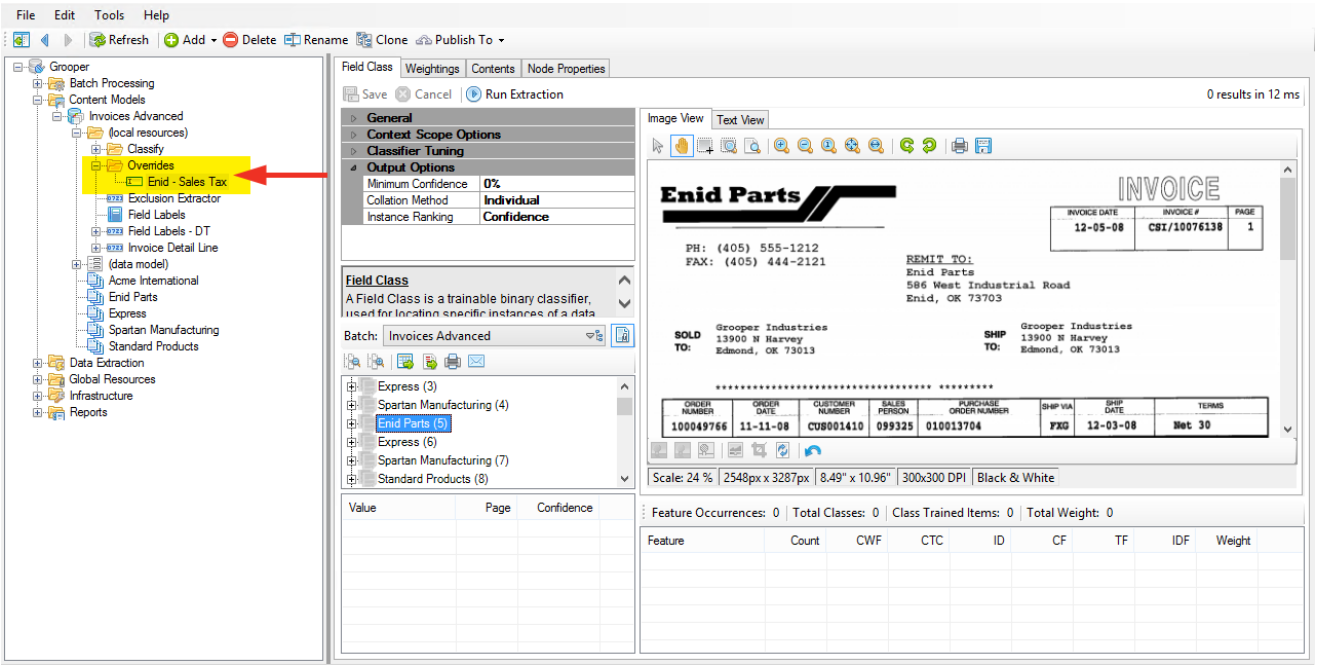
STEP 4 – SETTING UP THE PROPERTIES OF THE SALES TAX DATA FIELD

(1) Select the Sales Tax Data Field and set the Base Type to Decimal **(2)** and the Format Specifier to c2. **(3)** Set the Default Extractor Type to Reference and set the Referenced Extractor to the Sales Tax Data Type that is a child of the Sales Tax Data Field. **(4)** Set the Default Value Expression to 0.



STEP 5 – CREATING AN ALTERNATE EXTRACTOR FOR ENID PARTS DOCTYPES

As mentioned, the Enid Parts DocTypes have Sales Tax we want to capture, but the simplest mechanism for reliably capturing it is a Field Class. (Okay, that's a bit of a fib, a RegEx pattern would be very simple too {see if you can't figure out how...}, but this is a great opportunity to introduce this concept so bear with me...) Create a folder in the (local resources) folder and name it Overrides. Then, create a Field Class named Enid – Sales Tax within it.

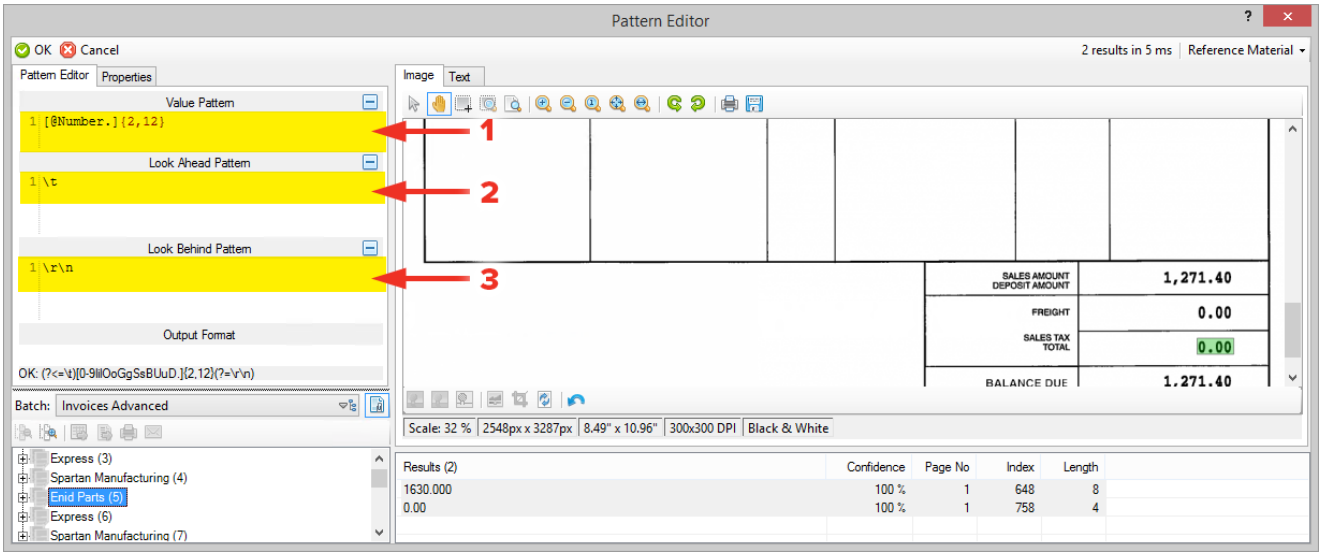


STEP 6 – SETTING THE EXTRACTORS FOR THE ENID – SALES TAX FIELD CLASS

With the Enid – Sales Tax Field Class selected, set the Value Extractor Type to Internal, then select the Pattern property and click the ellipsis button to bring up the Pattern Editor window.

(1) For the Value Pattern use the following: `[@Number.]{2,12}`
(2) For the Look Ahead Pattern use the following: `\t`
(3) For the Look Behind Pattern use the following: `\r\n`

Make sure Tab Marking is enabled in the Text Preprocessing area of the General section on the Properties tab. The Feature Extractor can be left to point to nGrams 1-3.

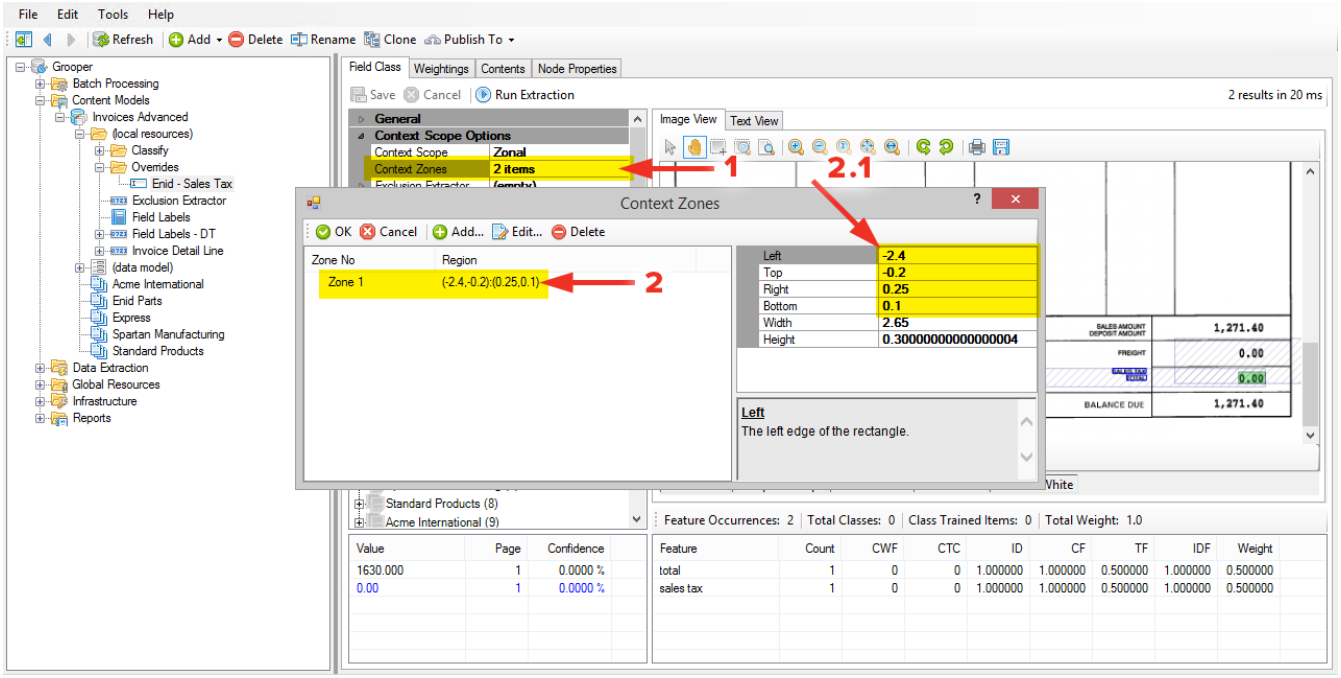


STEP 7 – EDITING THE CONTEXT ZONES

(1) Bring up the Context Zones options and delete Zone 2. **(2)** Use the following settings for Zone 1.

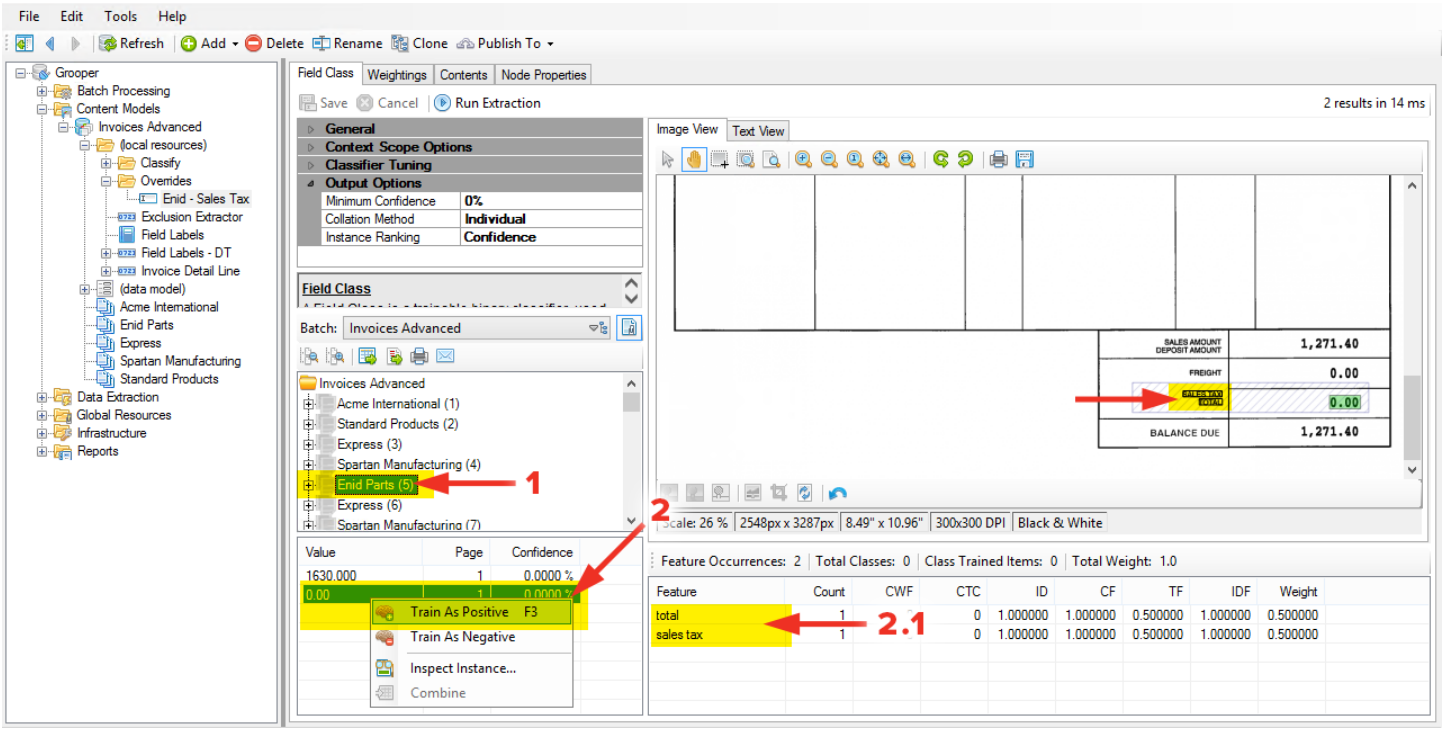
Zone 1 – Left: **-2.4** Top: **-0.2** Right: **0.25** Bottom: **0.1**

Save and run Extraction on the Enid Parts (5) document.



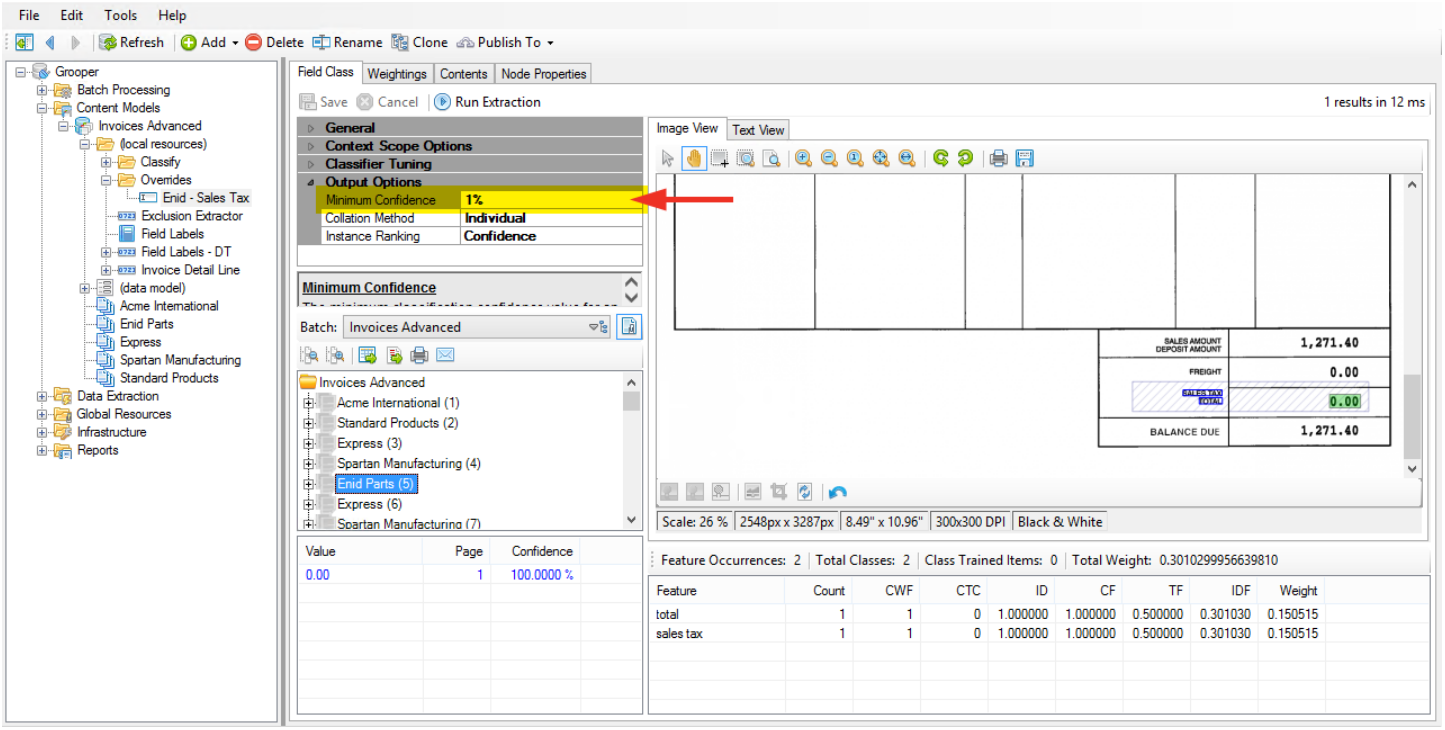
STEP 8 – TRAINING SALES TAX AND TOTAL FEATURES

(1) On Enid Parts (5), **(2)** select the 0.00 value that highlights the sales tax and total features. Train it positively.



STEP 9 – MINIMUM CONFIDENCE

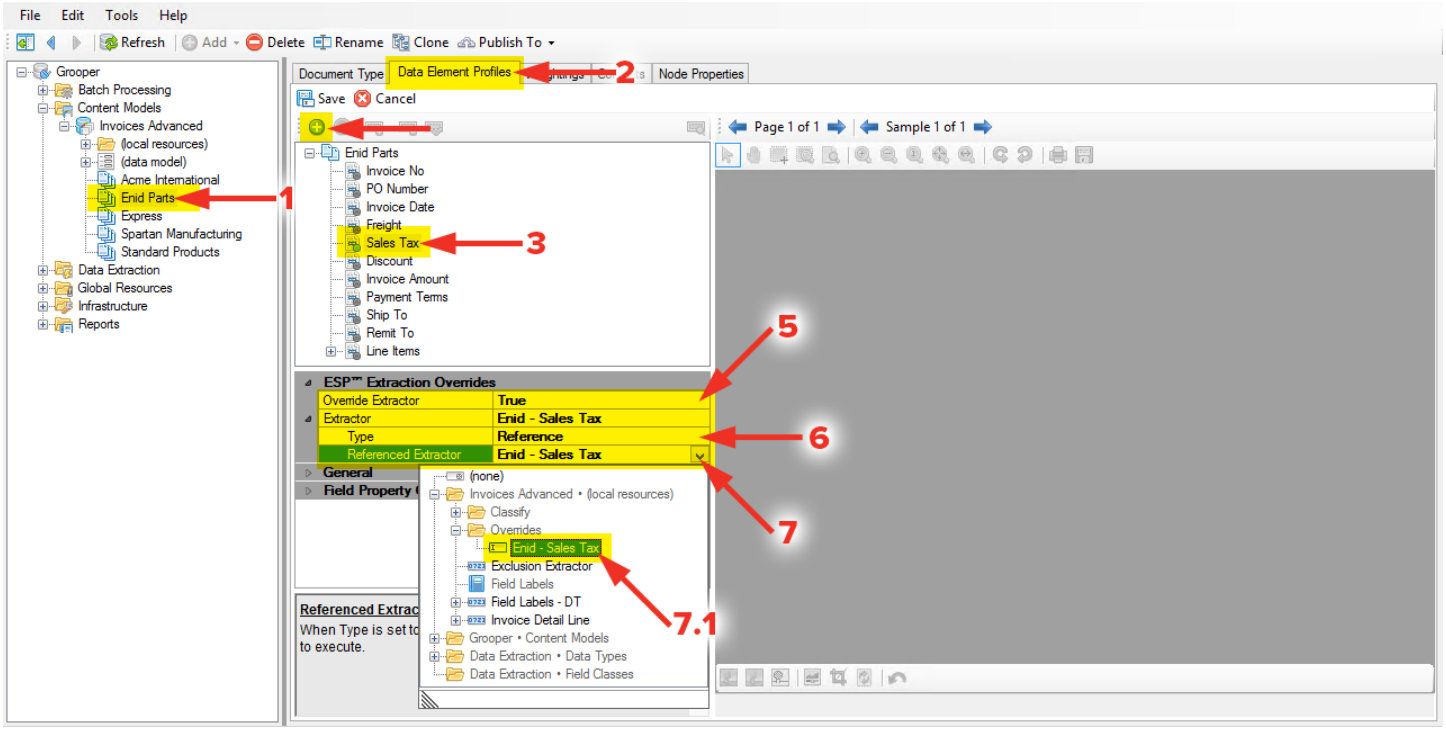
No other features need to be trained, so with a confidence of 100%, set the Minimum Confidence to anything above zero to eliminate unwanted values being returned.



STEP 10 – DATA ELEMENT PROFILES

With this new extractor made, it needs to be applied to the Enid Parts DocType. **(1)** Selet the Enid Parts document and **(2)** click on the Data Element Profiles tab. **(3)** Select the Sales Tax Data Field and **(4)** click the plus button. In the ESP Extraction Overrides section **(5)** set Override Extractor to True. **(6)** Set the Extractor Type to Reference, and **(7)** set the Referenced Extractor to the Enid – Sales Tax Field Class within the Invoices Advanced • (local resources) > Overrides area. Save the changes.

When is extraction is run now, this profile will override the Default Extractor for the Sales Tax Data Field and use this specified extractor instead.



FINAL DATA MODEL ADJUSTMENT AND REVIEW

There are a few minor tweaks to be made to the appearance of the Data Model. It is also best to look at and review, in once place, the data that is being returned by all the extractors that were just setup.

DATA MODEL AND DATA FIELD APPEARANCE SETTINGS

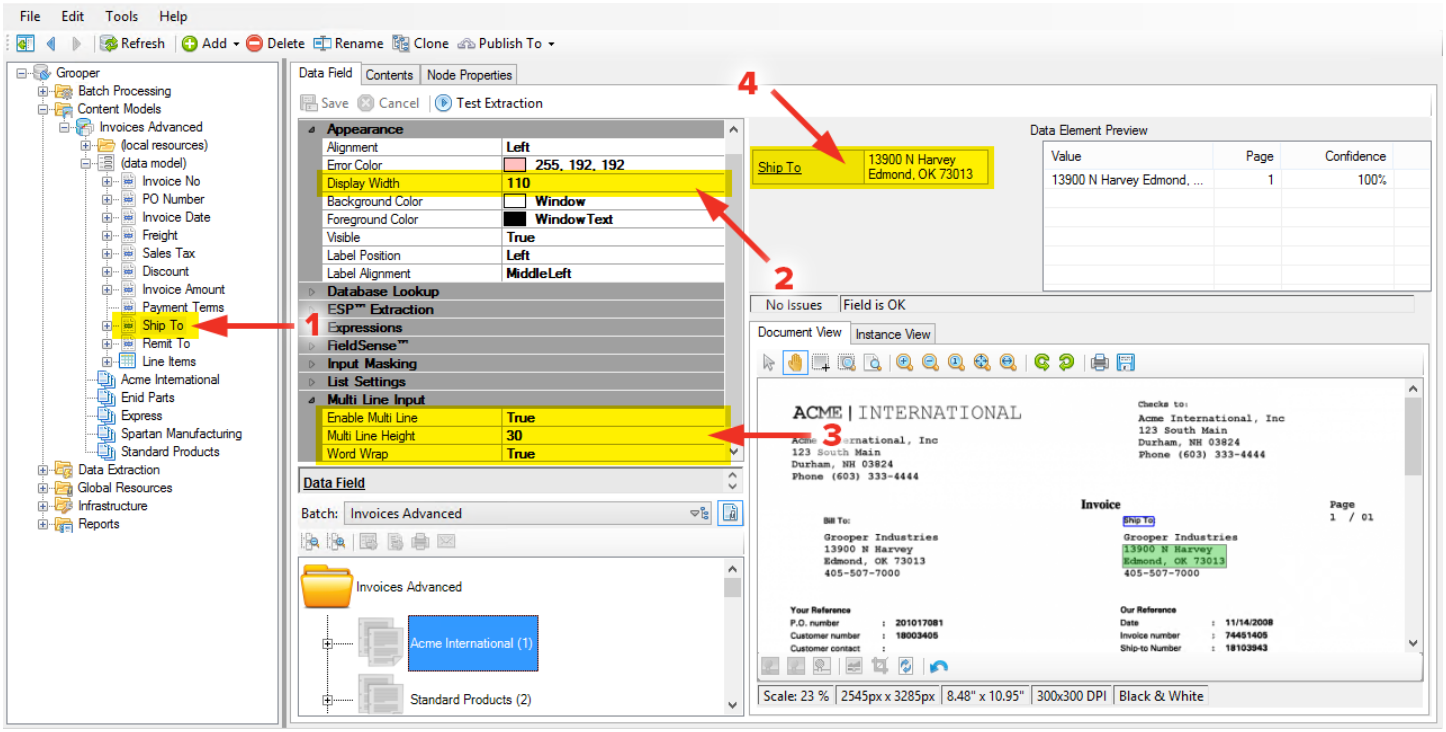
While not critical to get the data you want, appearance settings play a functional role in the attractiveness in the way that captured data is given back to you when reviewing said data. Not all the individual appearance properties need to be covered in this case, but a select few will be.

ADDRESS DISPLAY

(1) Select the Ship To Data Field and Test Extraction. Notice, in the Data Element Preview, that the width of the field is cutting off the way the information is being display. Also, an address isn't written on a single line.

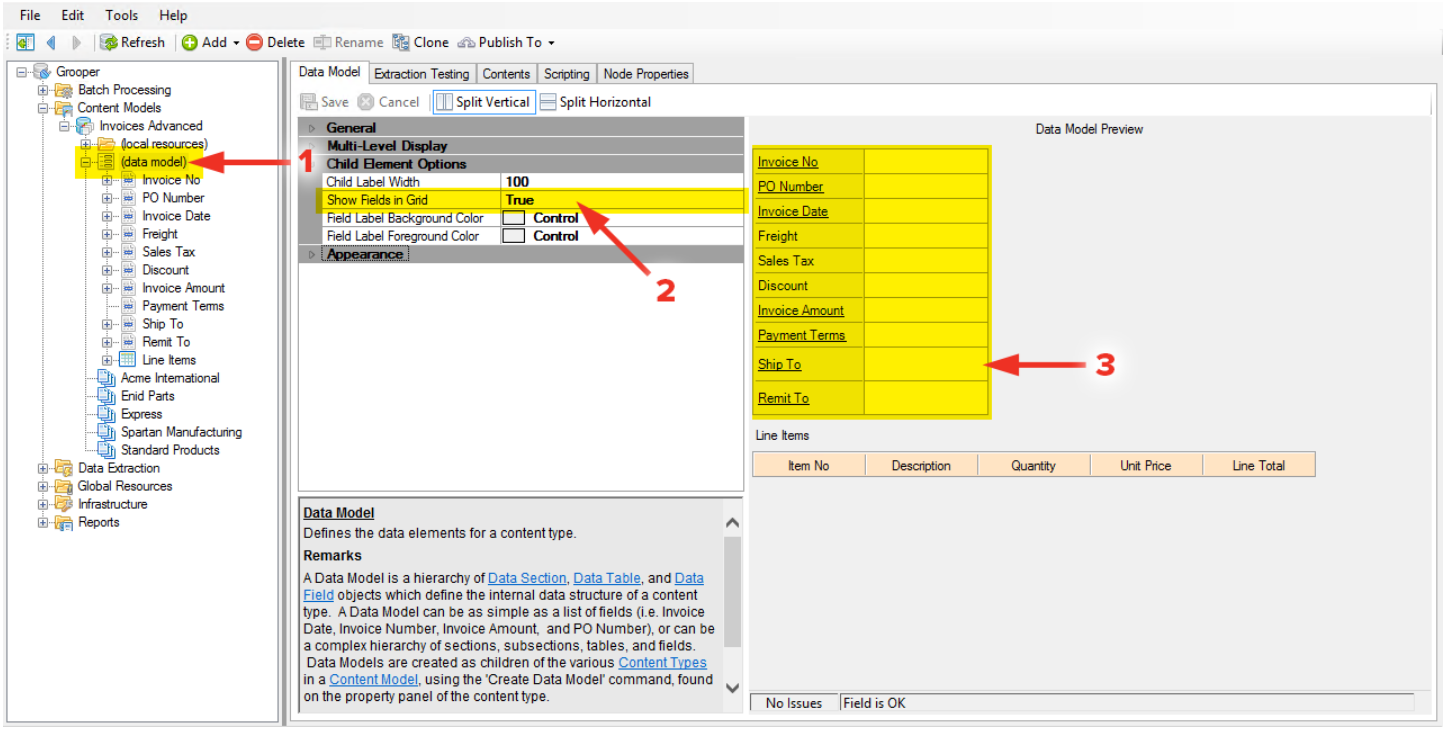
(2) In the Appearance section change the Display Width to 110. **(3)** In the Multi Line Input section, change Enable Multi Line to True, Multi Line Height to 30, and Word Wrap to True. Save and Test Extraction and **(4)** notice the difference.

Apply these same settings to the Remit To field.



DATA MODOEL – SHOW FIELDS IN A GRID

(1) Select the (data model) and in the Child Element Options section **(2)** set Show Fields in a Grid to True. **(3)** Notice now all the fields displayed in a table like structure. An interesting tid-bit for this feature is that the longest Display Width of a given field will determine the width for all fields when in a grid like this. Considering that, changing the Remit To Display Width to 110 just now, and using this feature, puts the Display Width for all fields to 110.

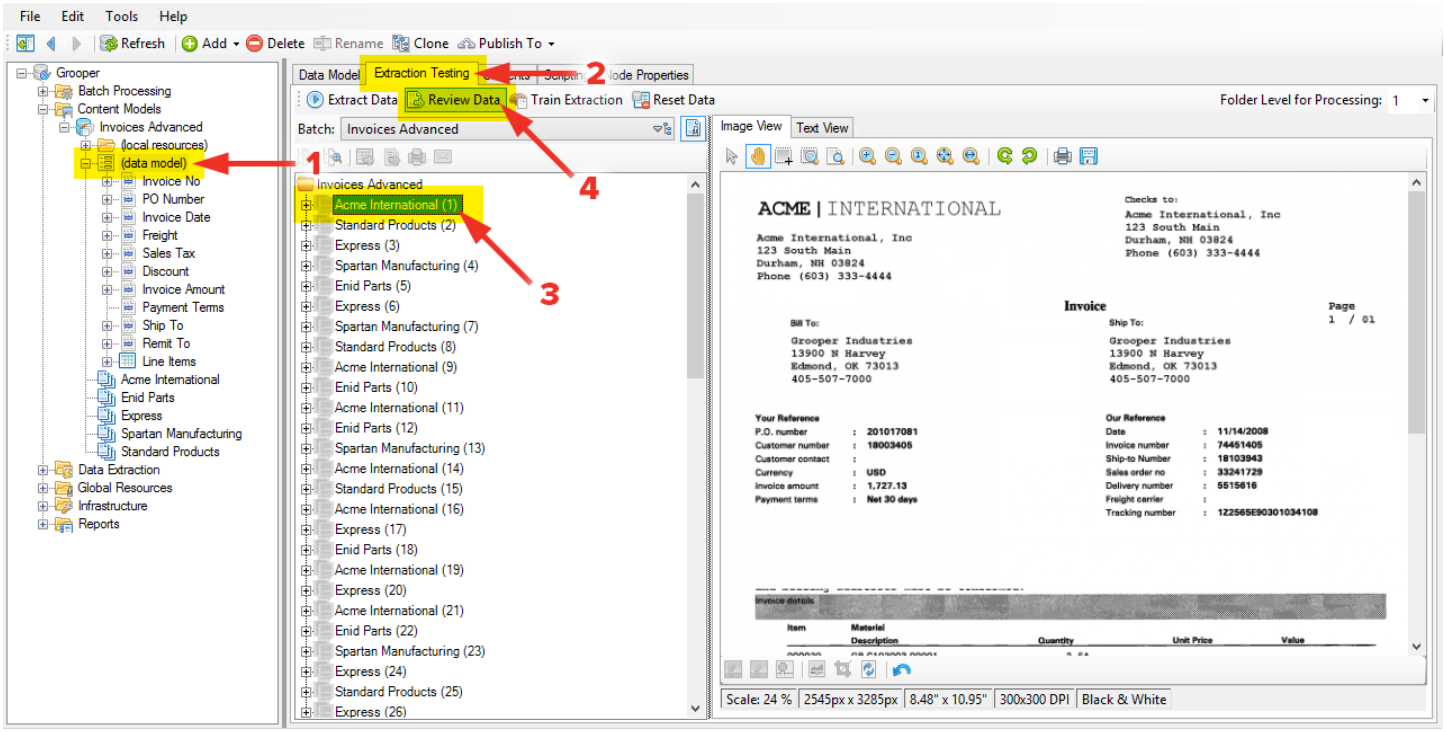


EXTRACTION TESTING

Any amount of extraction testing that has occurred as of yet has been done either while developing pattern recognition, or on a per field basis. With the Data Model fully built out, it is best to review the data as a whole. Data Models allow for this review in a module that can resemble the Data Review Attended Module.

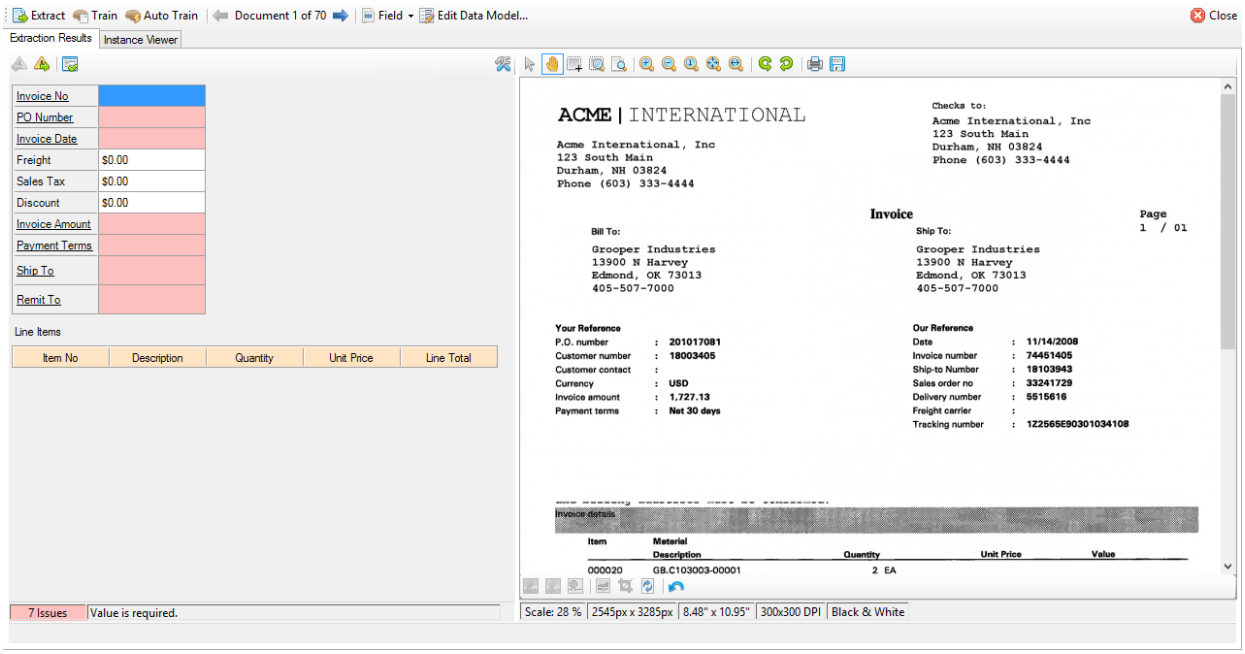
REVIEW DATA

(1) Select the (data model) and **(2)** click the Extraction Testing tab. From here, **(3)** select a document in the Batch Viewer and **(4)** click the Review Data button.



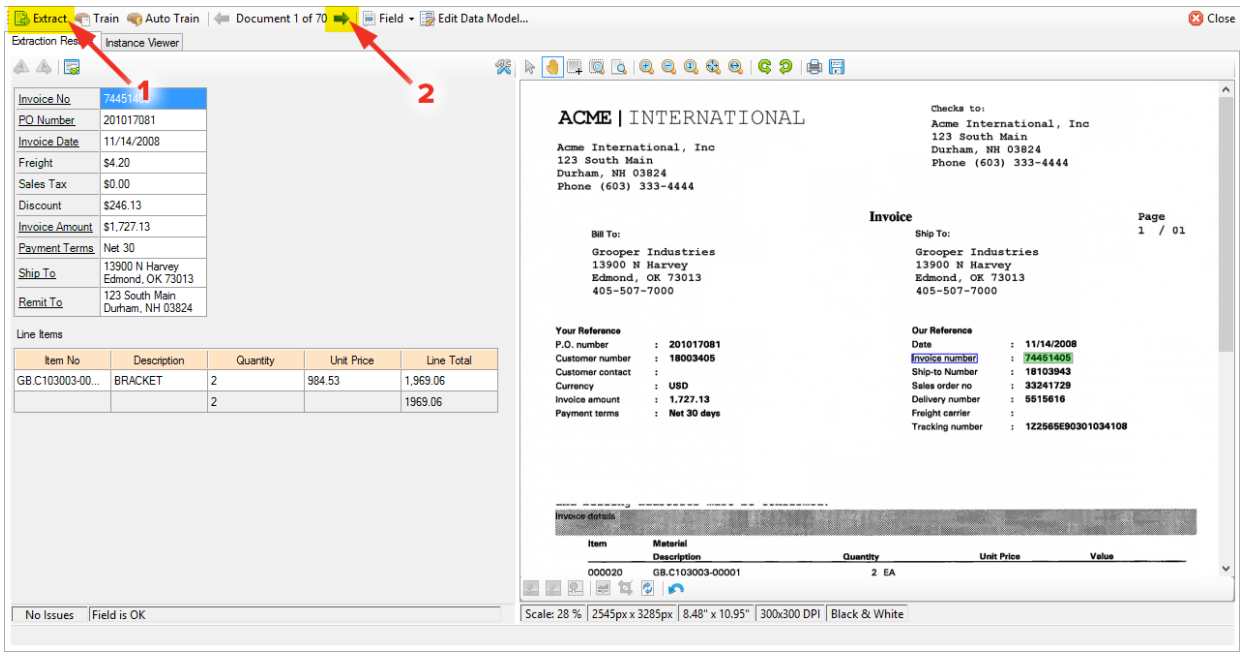
REQUIRED FIELDS AND DEFAULT VALUES

In the Grooper Data Extraction Test Utility that appears, there are a few things to take note of. The main things to notice are that all Data Fields that have been set to required are empty and reddish. This is to let you know there are validation errors. This is also made apparent by the 7 Issues – Value is required. at the bottom. There are also 3 fields that are not returning errors, and instead have default values of $0.00. Given that those fields were not set to required, and that they were given values for their Default Value Expression, this should seem logical.



EXTRACT DATA

(1) Click the Extract button at the top left, and due to all the work that was put into the Data Model, the fields should be populated with information. **(2)** Feel free to click the advanced document arrow to review subsequent documents.



PHASE 5 – DELIVER

Having built all the tools required to take our documents through the first four phases, it is now time to send the extracted information to a place it can be useful.

BUILDING A BATCH PROCESS

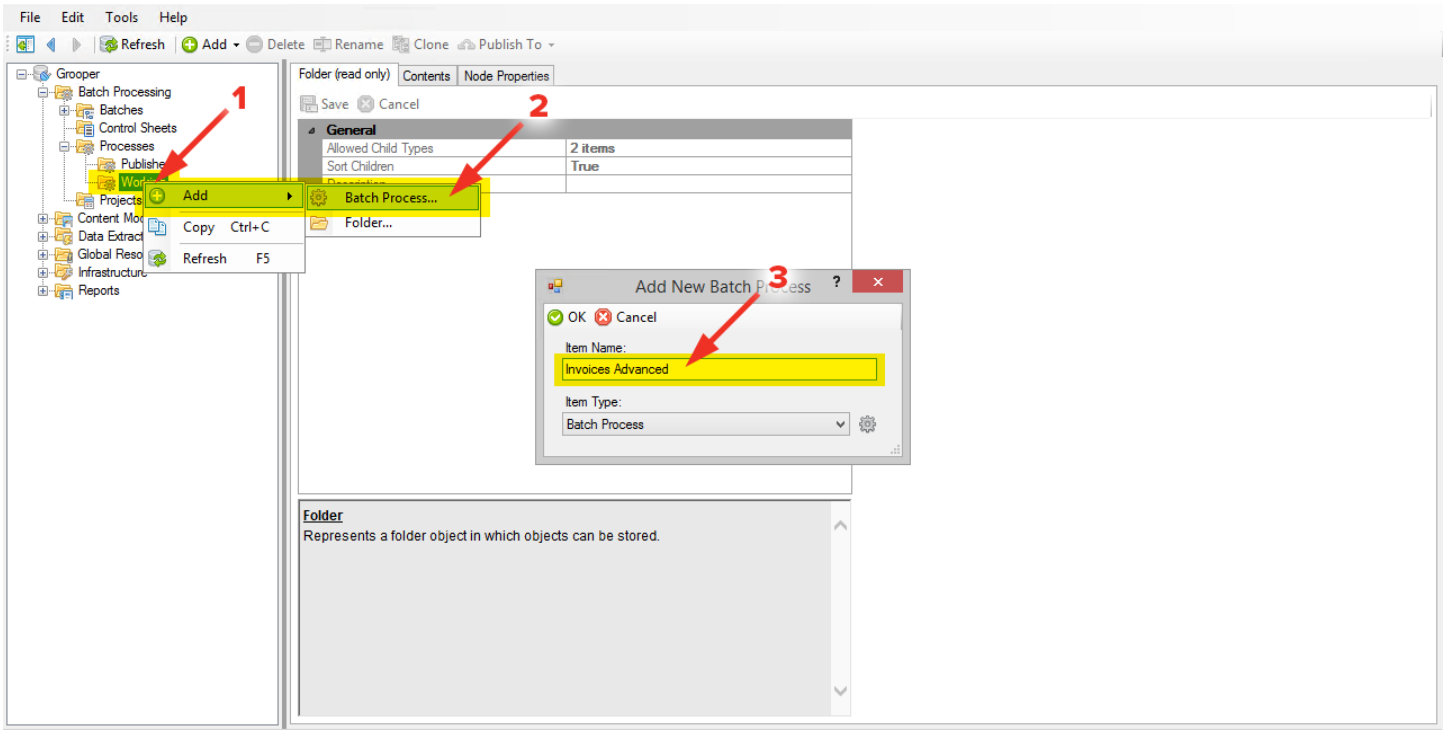
In the previous document, the Batch Process was built out as the tools were being learned about, yet in this document, a Batch Process hasn't even been mentioned. Delivery of the extracted information will make the most sense as a final step of a process.

CREATING THE STEPS OF A PROCESS

Before making the step for delivery, let's build out the logical sequence of events that will procedurally take our documents through each phase.

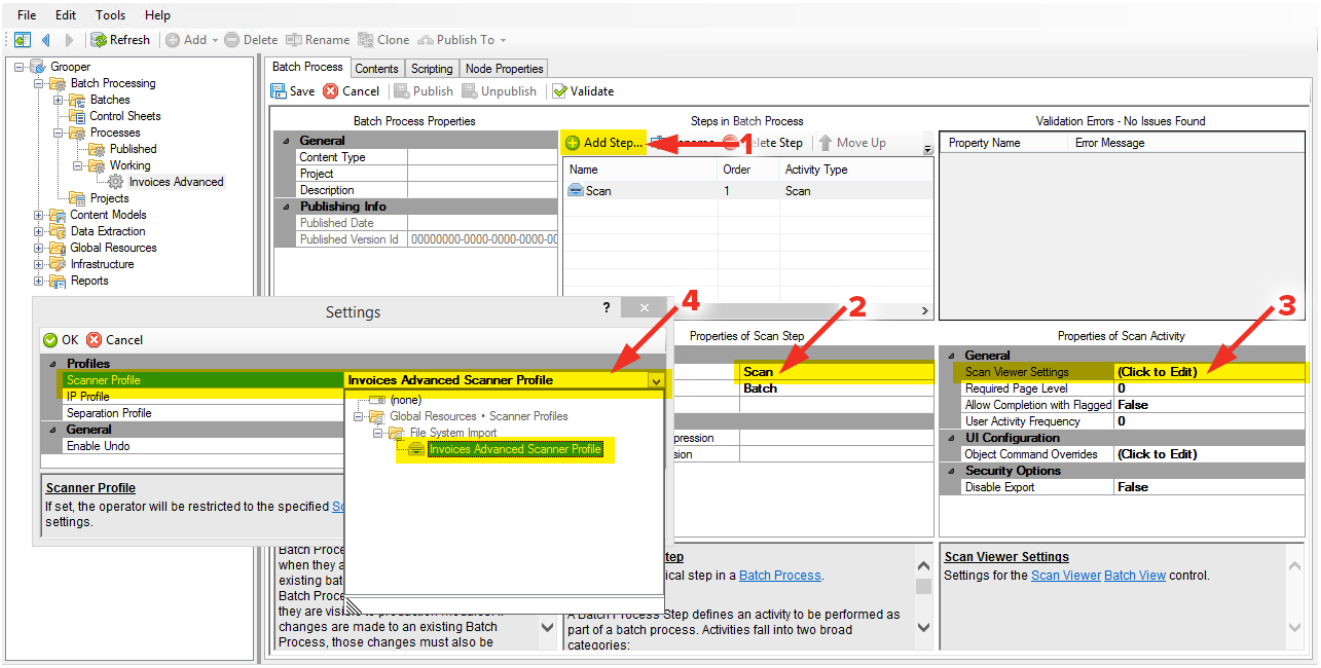
STEP 1 – CREATING A BATCH PROCESS

(1) Navigate to Grooper > Batch Processing > Processes and select the Working node. **(2)** Right click and Add > Batch Process... and **(3)** name it Invoices Advanced. From here, when adding steps, think through the sequence of events about the Phase in which they occur, and it will make the most sense.



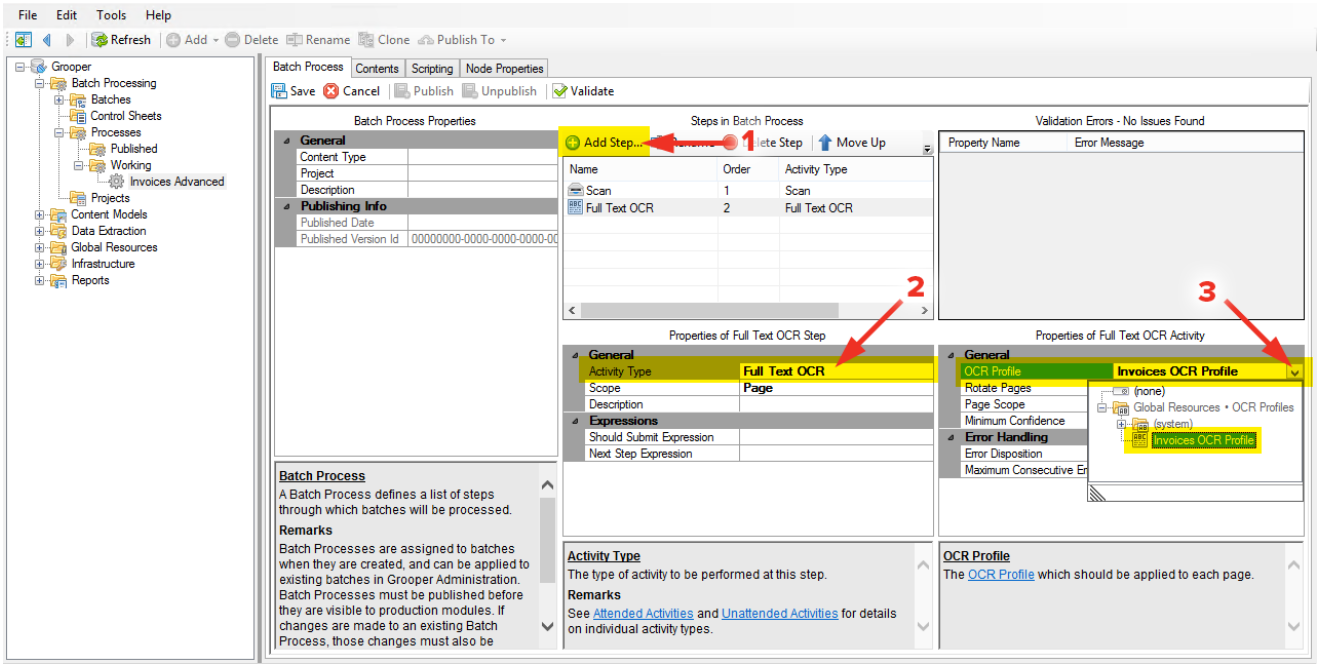
STEP 2 – PHASE 1 – ACQUIRE • SCAN

To acquire the documents for our Batch Process we setup a Scanner Profile. **(1)** Click Add Step... and **(2)** set the Activity Type to Scan. After this **(3)** select the Scan Viewer Settings property and click the ellipsis button. In the Settings window that appears **(4)** select the Scanner Profile property and set it to the Invoices Advanced Scanner Profile within the Global Resources • Scanner Profiles > File System Import area.



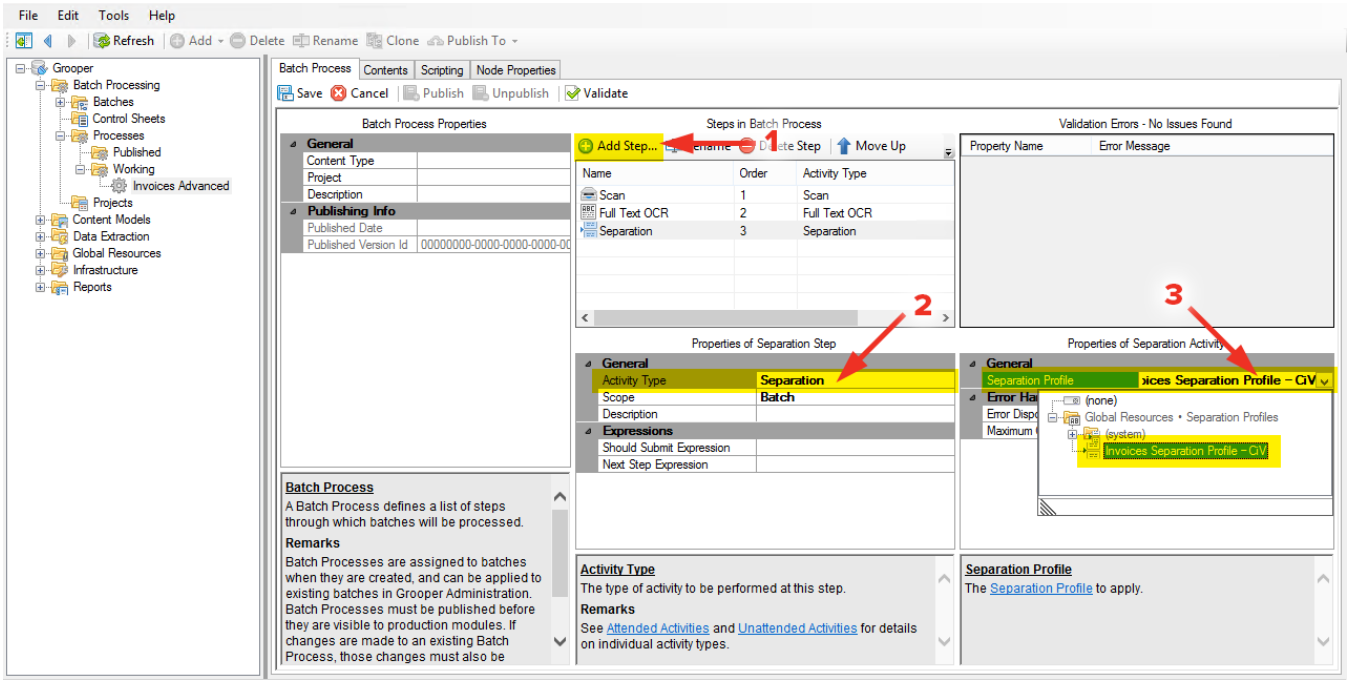
STEP 3 – PHASE 2 – CONDITION • OCR

The next step in the Batch Process consists of conditioning the documents. **(1)** Add another Step and **(2)** set its Activity Type to Full Text OCR. **(3)** Set the OCR Profile to the Invoices OCR Profile within the Global Resources • OCR Profiles area.



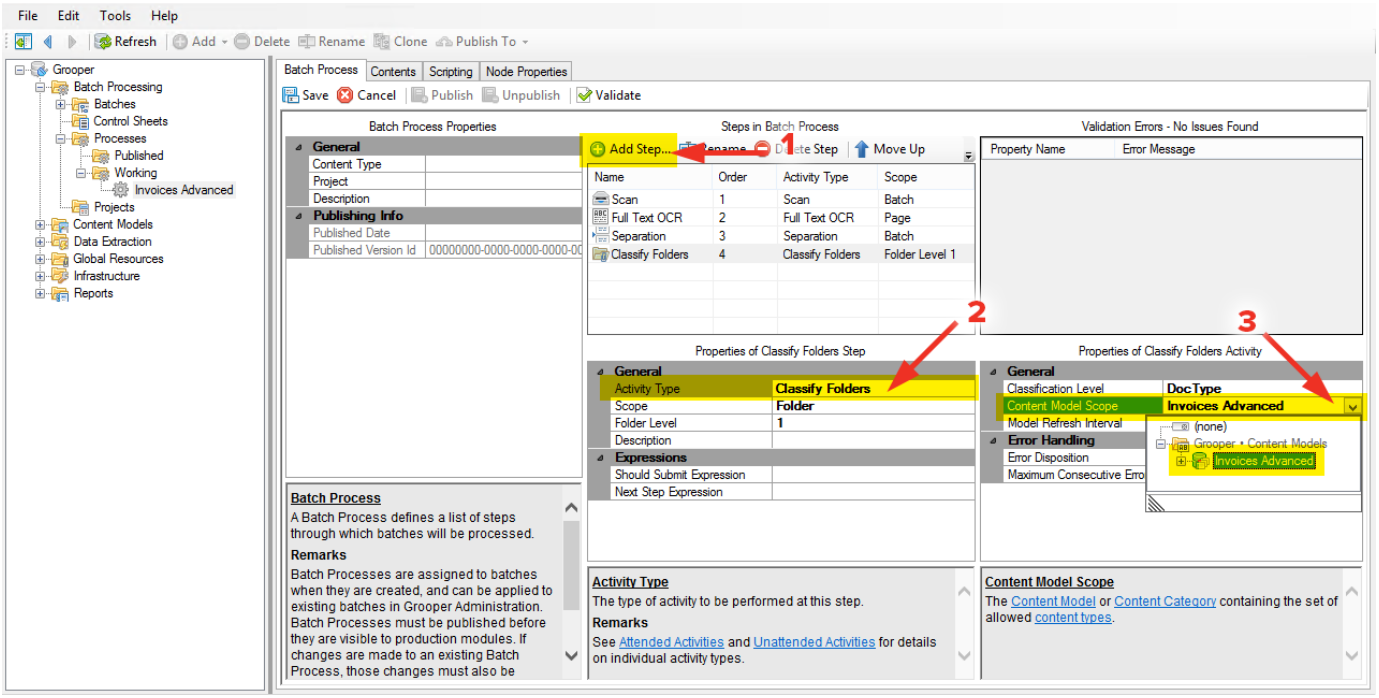
STEP 4 – PHASE 3 – ORGANIZE • SEPARATION

Only after conditioning the documents can they be organized. **(1)** Add a Step and **(2)** set its Activity Type to Separation. **(3)** Set the Separation Profile to the Invoices Separation Profile – CiV in the Global Resources • Separation Profile area.



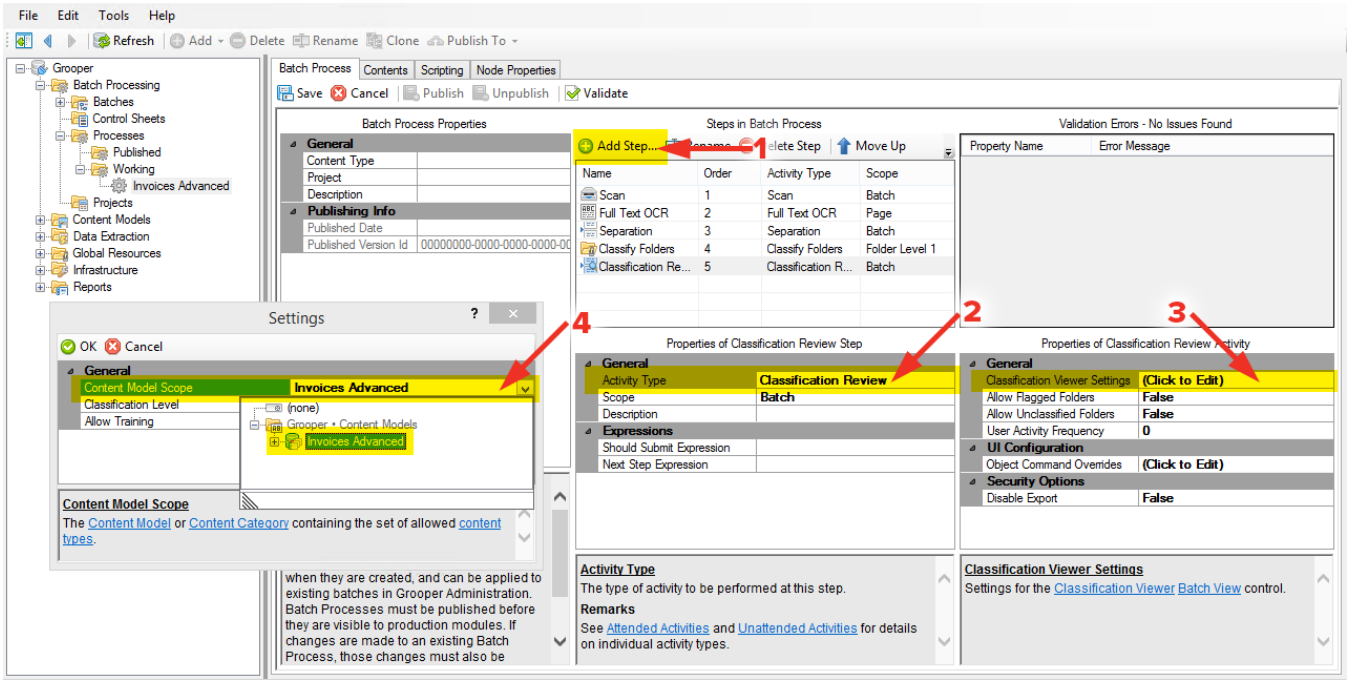
STEP 5 – PHASE 3 – ORGANIZE • CLASSIFICATION

With the pages separated in to logical documents, the folders that they exist within can now be classified. **(1)** Add another Step, **(2)** set its Activity Type to Classify Folders, and **(3)** set the Content Model Scope to the Invoices Advanced Content Model within the Grooper • Content Models area.



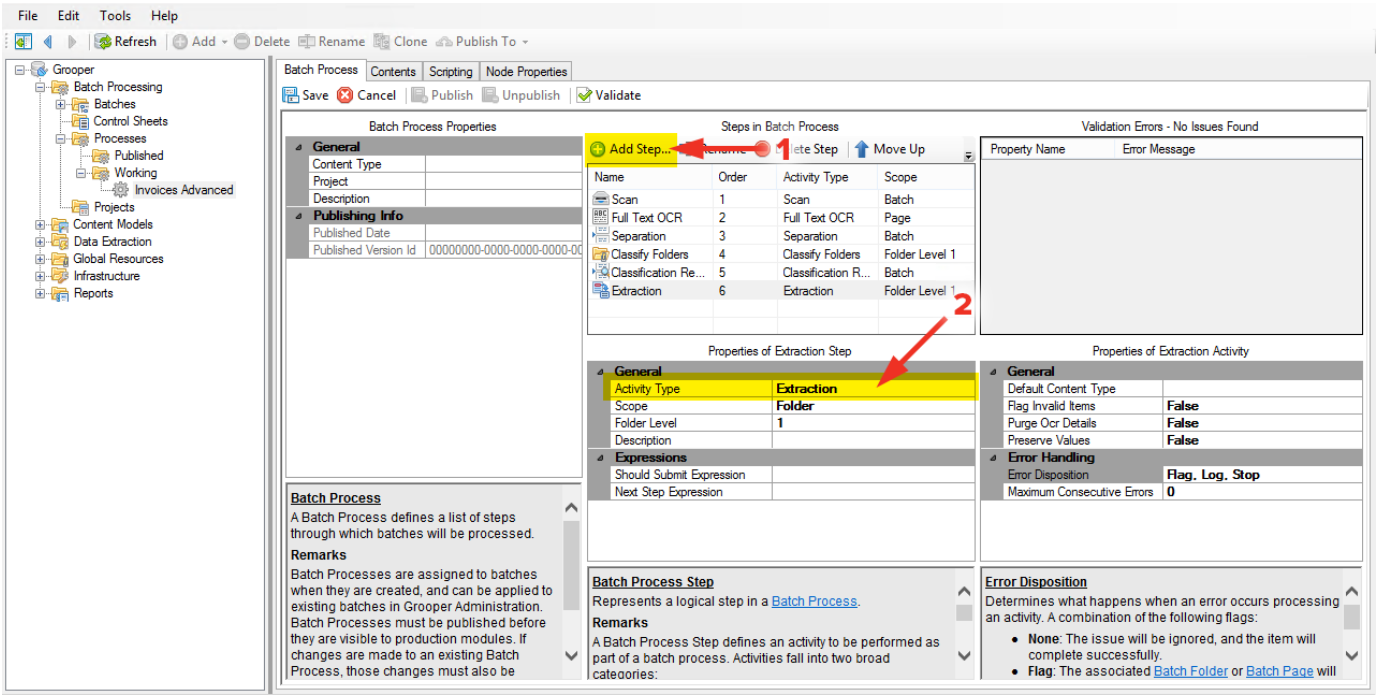
STEP 6 – PHASE 3 – ORGANIZE • CLASSIFICATION REVIEW

It is typically best practice to have a person review Classification. **(1)** Add a Step, **(2)** set its Activity Type to Classification Review, and **(3)** click the ellipsis button for the Classification Viewer Settings property. In the Settings window **v**set the Content Model Scope property to the Invoices Advanced Content Model within the Grooper • Content Models area.



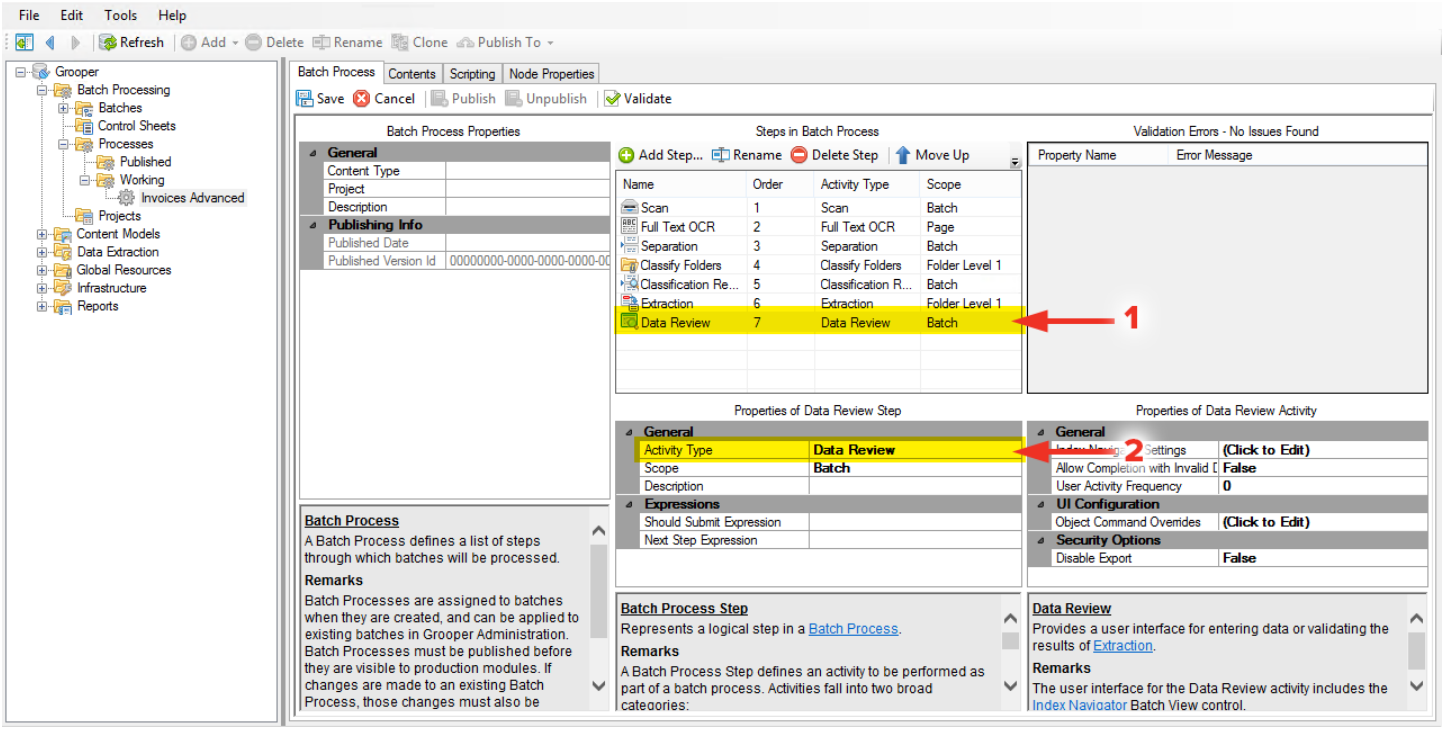
STEP 7 – PHASE 4 – EXTRACTION

As much work went into developing the Content Model to perform the extraction, it almost seems anti-climactic that it comes down to one step, but I digress. As such, add another and set its Activity Type to Extraction.



STEP 8 – PHASE 4 – EXTRACTION • DATA REVIEW

Reviewing the extracted data is important as it puts human eyes on procedurally collected information to verify it accuracy. **(1)** Add a Step and **(2)** set its Activity Type to Data Review.



STEP 9 – PHASE 5 – DELIVER • DOCUMENT EXPORT

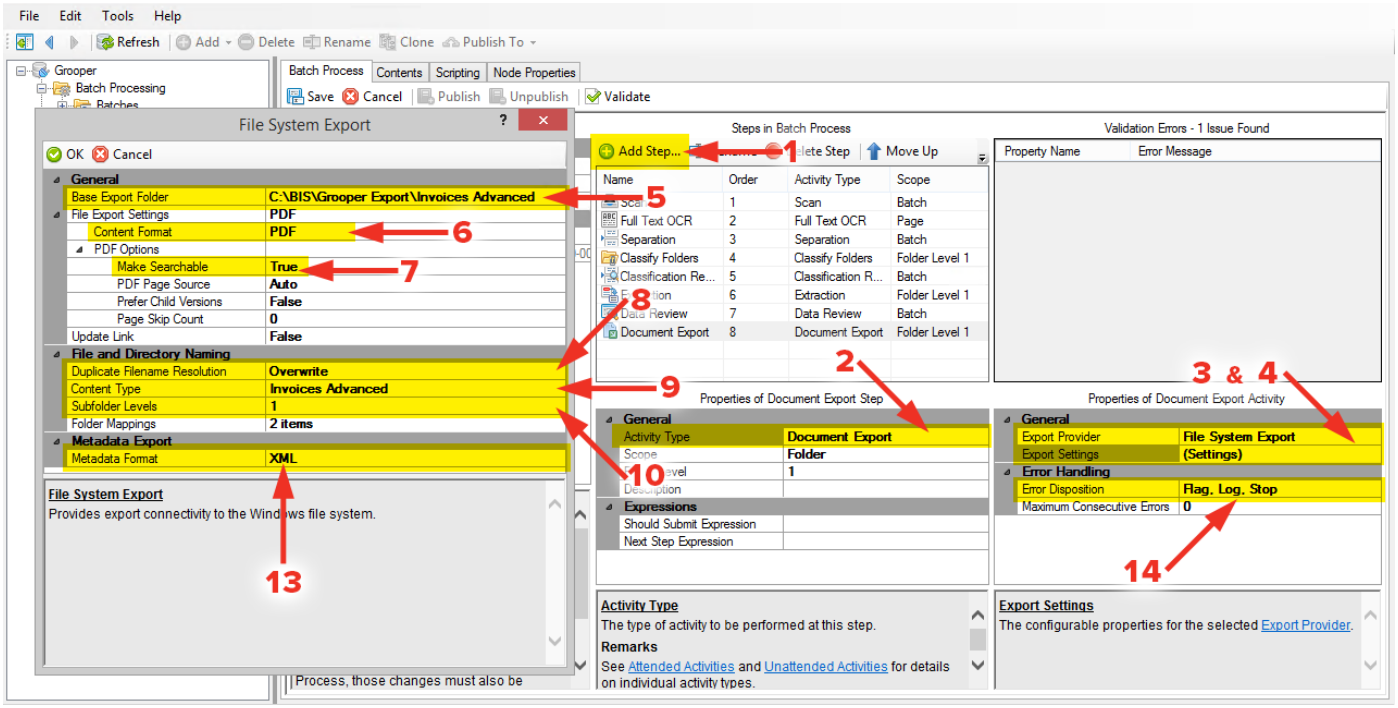
The culmination of our Batch Process is the delivery of the data. **(1)** Add a Step and **(2)** set its Activity Type to Document Export. **(3)** Set File System Export as the Export Provider and **(4)** open the settings for the Export Settings property. In the File System Export settings window **(5)** set the Base Export Folder to:

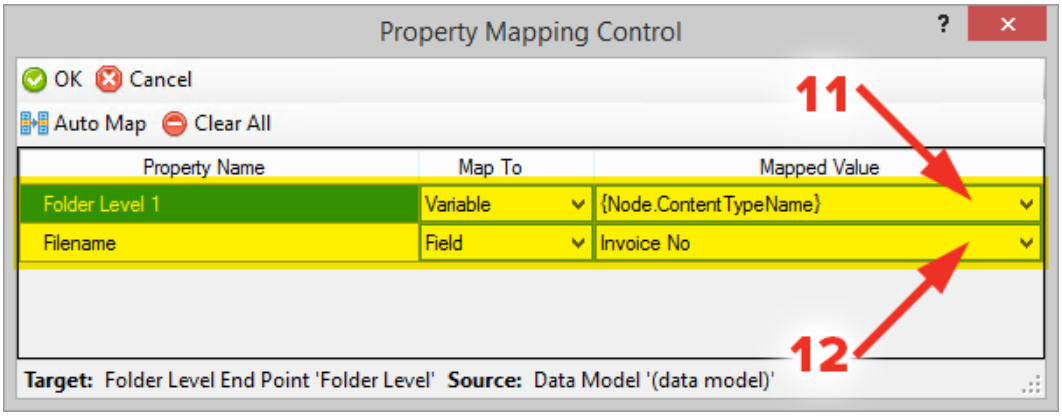
`C:\BIS\Grooper Export\Invoices Advanced`

Expand File Export Settings and **(6)** set the Content Format to PDF. Expand PDF Options and **(7)** set Make Searchable to True.

(8) Change Duplicate Filename Resolution to Overwrite. **(9)** Set the Content Type to the Invoices Advanced Content Model. **(10)** Set Subfolder Levels to 1. Open the Folder Mappings settings and **(11)** set Folder Level 1 to a Variable of {Node.ContentTypeName}. **(12)** Set Filename to a Field of Invoice No. **(13)** Change the Metadata Formal to XML.

Finally, **(14)** change the Error Disposition to Flag, Log, Stop. Its worth considering this Error Disposition for other steps such as Full Text OCR, so that if an error occurs, the Batch Process stops completely to prompt Administrator review. It is critical, however, it be set for Document Export, since the batch is disposed of immediately after export.

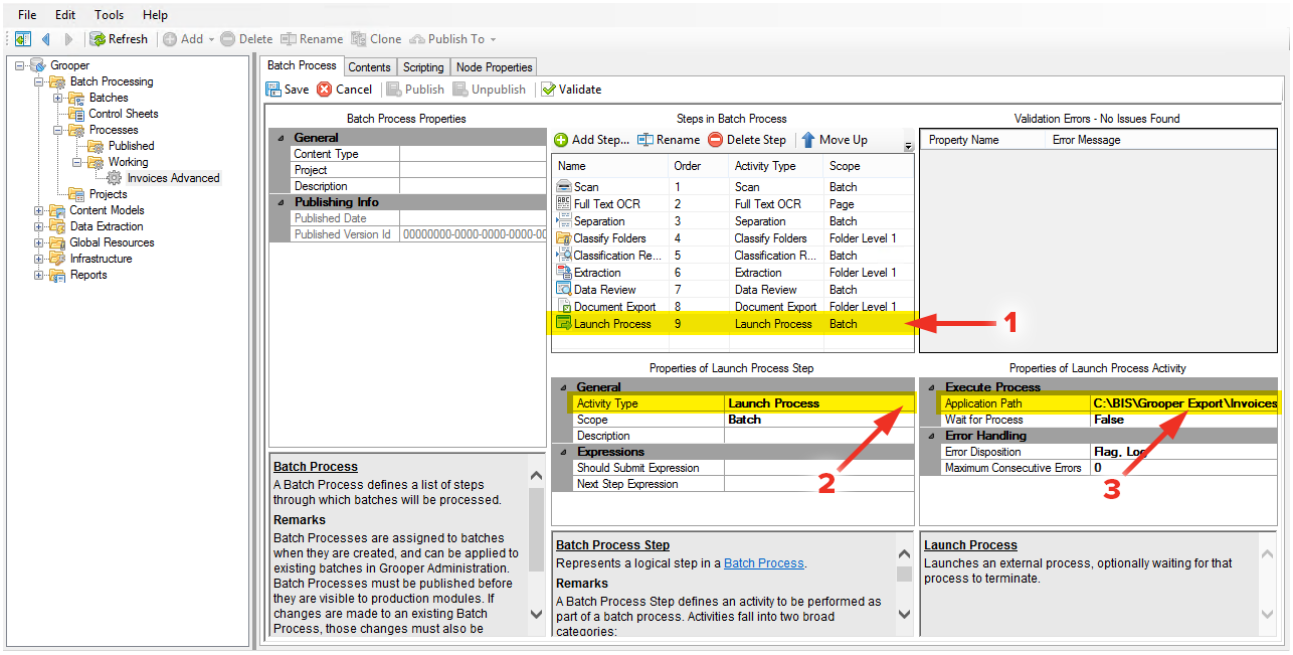




STEP 10 – PHASE 5 – DELIVER • LAUNCH PROCESS

(1) Add a Step and **(2)** set its Activity Type to Launch Process. This is a unique activity in Grooper that allows external processes to be launched. In this case, we will set it to open a Windows File Explorer that houses the now exported documents. **(3)** Set the Application Path to:

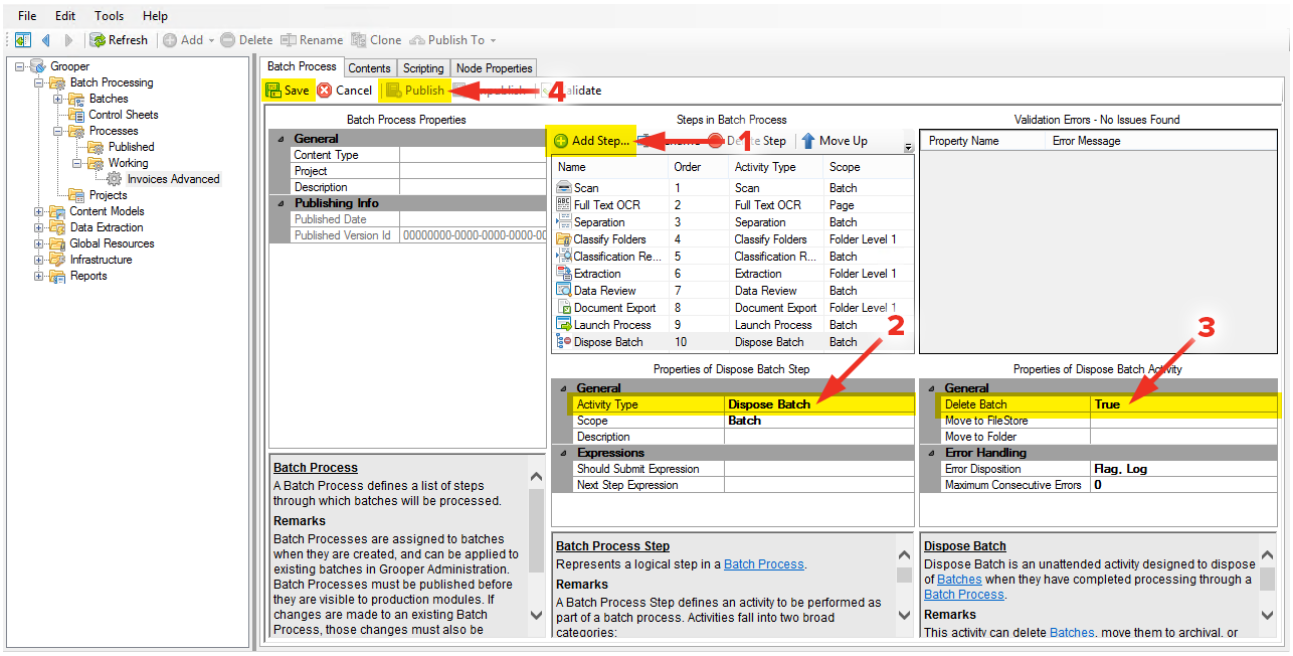
`C:\BIS\Grooper Export\Invoices Advanced`



STEP 11 – DISPOSE BATCH

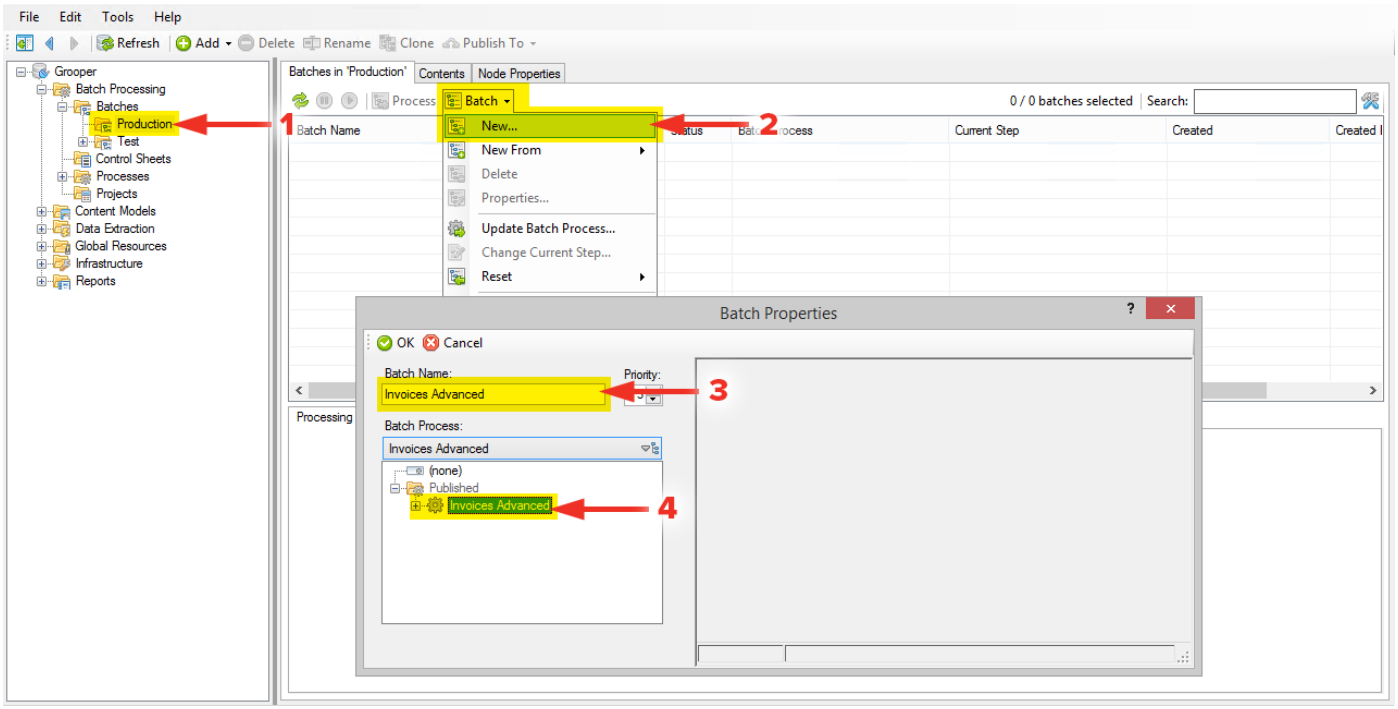
Grooper is not a destination for data, rather, a mechanism to Extract, Transform, and Load that data somewhere else. As such, it is prudent to dispose of batches once they are complete. **(1)** Add a Step and **(2)** set its Activity Type to Dispose Batch. **(3)** Set Delete Batch to True.

(4) Save and Publish this Batch Process.



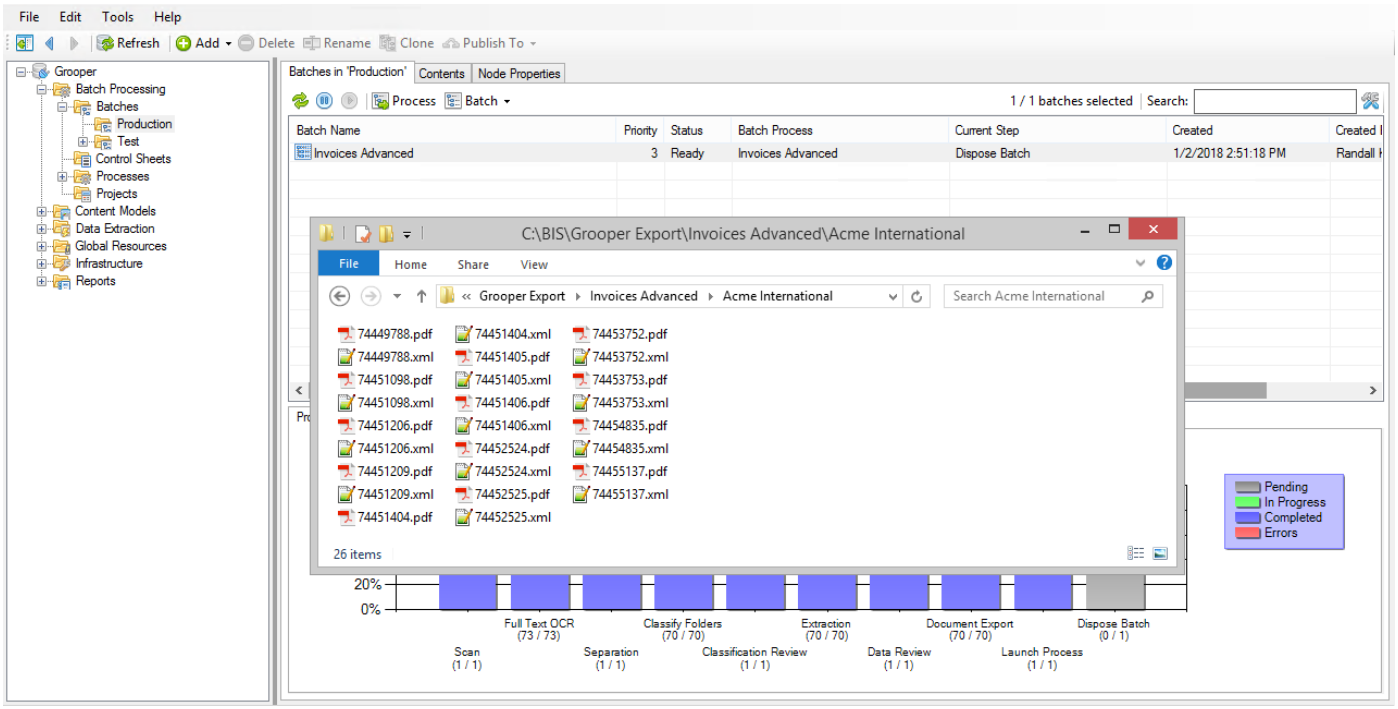
STEP 12 – LAUNCHING A PRODUCTION BATCH

(1) Navigate to Grooper > Batches and with either Production or Test selected, **(2)** click the Batch drop-down and create a New... batch. **(3)** Name it Invoices Advanced. **(4)** Select the Invoices Advanced Batch Process.



STEP 13 – VIEWING RESULTS

Either via Grooper Services you've established, or by manually processing each step, you'll get to a point where the Launch Process step will open a Windows Explorer to the export destination. Look through this export folder to view your searchable PDFs as well as the indexed XML data.



A FINAL NOTE

To begin, you can download everything that was worked on throughout the course of this document by following this link:

Grooper A.C.E. - Architect Training Vol.2 - Advanced Invoice Processing - Final Export

This zip file contains the source batch that the Scanner Profile refers to, as well as the complete Batch Process. Due to all the references it has to other profiles and the Content Model, the Batch Process object brings with it all necessary other objects to complete the process.

A lot has been covered in this volume of training. We focused on importing objects, creating an OCR Profile, Change in Value Separation, Classification based on positive extractors, Data Tables, FuzzyRegEx, Field Classes, Lexicons, Cheat Codes, Ordered Arrays, Exclusion and Inclusion extractors, and Data Element Profiles. It's quite a lot to take in all at once, and honestly this is just the beginning.

I feel quite confidant in saying that having made it through learning a bit about applying all these tools, and the navigation in and out of **Grooper** required to get to this point, that your familiarity with **Grooper**, and ways to leverage it for your business purposes, is greatly heightened.

The next steps for you will involve diving in and applying this knowledge with **Grooper** and staying in contact with the team here at **BIS** to help you grow as a **Grooper** user.

We very much look forward to your growth within this **Grooper** family and seeing you in **Grooper A.C.E. • Consultant** training.

Finally, and most importantly, congratulations for making it through **Grooper A.C.E. • Architect** training!

Grooper | ACE

Architect · Consultant · Engineer