Chapter 23.

November 2008

Welcome to the November 2008 edition of IBM InfoSphere Information Server Developer's Notebook. This month we answer the question;

Part II of: How do I make use of the QualityStage component to Information Server, more specifically, How do I create and use QualityStage custom rule sets?

Excellent question! While data cleansing and best record survivorship are the most commonly discussed and/or used functions within QualityStage, this component to Information Server does so much more. We regularly use QualityStage custom rule sets to parse free form text and output standardized data, allowing us to enrich data we previously owned.

QualityStage custom rule sets are easy to learn with just a small amount of practice. Using a phased approach, we previously discussed the Investigate stage of QualityStage in the prior edition of IISDN (and also over viewed QualityStage as a whole), in this edition move on to the Standardize stage and creation and testing of custom rule sets.

Software versions

All of these solutions were *developed and tested* on (IBM) InfoSphere Information Server (IIS) version 8.1, using the Microsoft Windows XP/SP2 platform to support IIS client programs, and a RedHat Linux Advanced Server 4 (RHEL 4) FixPak U6 32 bit SMP server (Linux kernel version 2.6.9-67.EL-smp) to support the IIS server side components.

IBM InfoSphere Information Server allows for a single, consistent, and accurate view of data across the full width of the corporate enterprise, be it relational or non-relational, staged or live data. As a reminder, the IBM InfoSphere Information Server product contains the following major components;

WebSphere Business Glossary Anywhere™, WebSphere Information Analyzer™, WebSphere Information Services Director™, WebSphere DataStage™, WebSphere QualityStage™, WebSphere Metadata Server and Metabridges™, WebSphere Metadata Workbench™, WebSphere Federation Server™, Classic Federation™, Event Publisher™, Replication Server™, Rational Data Architect™, DataMirror Transformation Server™, and others.

Obviously, IBM InfoSphere Information Server is a large and capable product, addressing many strategic needs across the enterprise, and supporting different roles and responsibilities.

23.1 Terms and core concepts

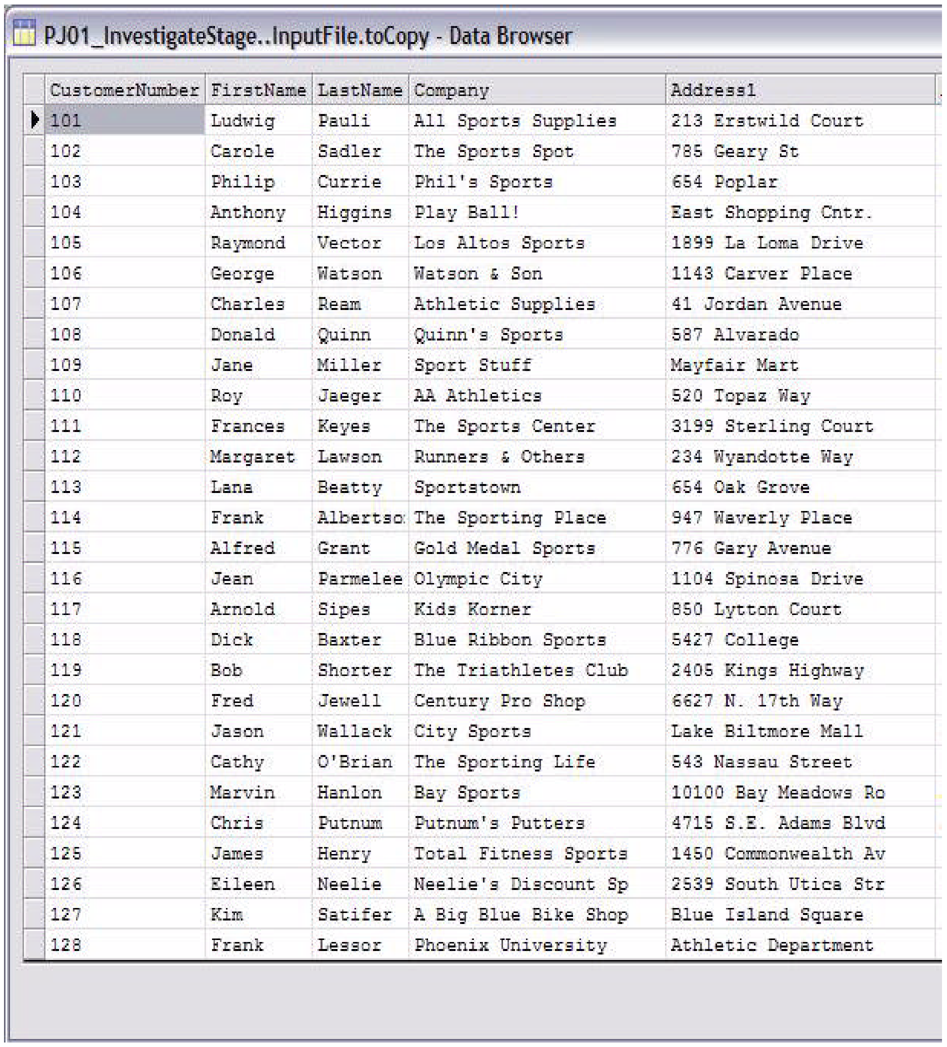
As mentioned above, this edition of (IBM) InfoSphere Information Server Developer’s Notebook (IISDN) is a continuation from the prior month. The prior month (the October/2008 edition), gave focus to the Investigate stage of the QualityStage component to Information Server, and also offered an overview of QualityStage as a whole. The Investigate stage is an optional and supporting area of functionality to this month’s topic, the Standardize stage.

The Standardize stage to QualityStage is a high function area within Information Server; imagine being able to search a free form piece of text, and look for occurrences of the phrase “expiration date” within (n) words of the keyword “contract”. (In essence, look for contracts which are about to expire. Cool stuff.) The Standardize stage does this and hundreds more high function operations.

While QualityStage includes many pre-built rule sets (Standardization Rules) to operate on data elements like address and postal codes, company names, and so on, the real power is learning how to create your own custom rule sets; to operate on whatever piece of data you encounter. Your personally created custom rule sets use the very same technology as the rule sets that come with QualityStage. In fact, you can view the (source code) to the pre-built rule sets as a means to further understand the advanced capabilities of QualityStage.

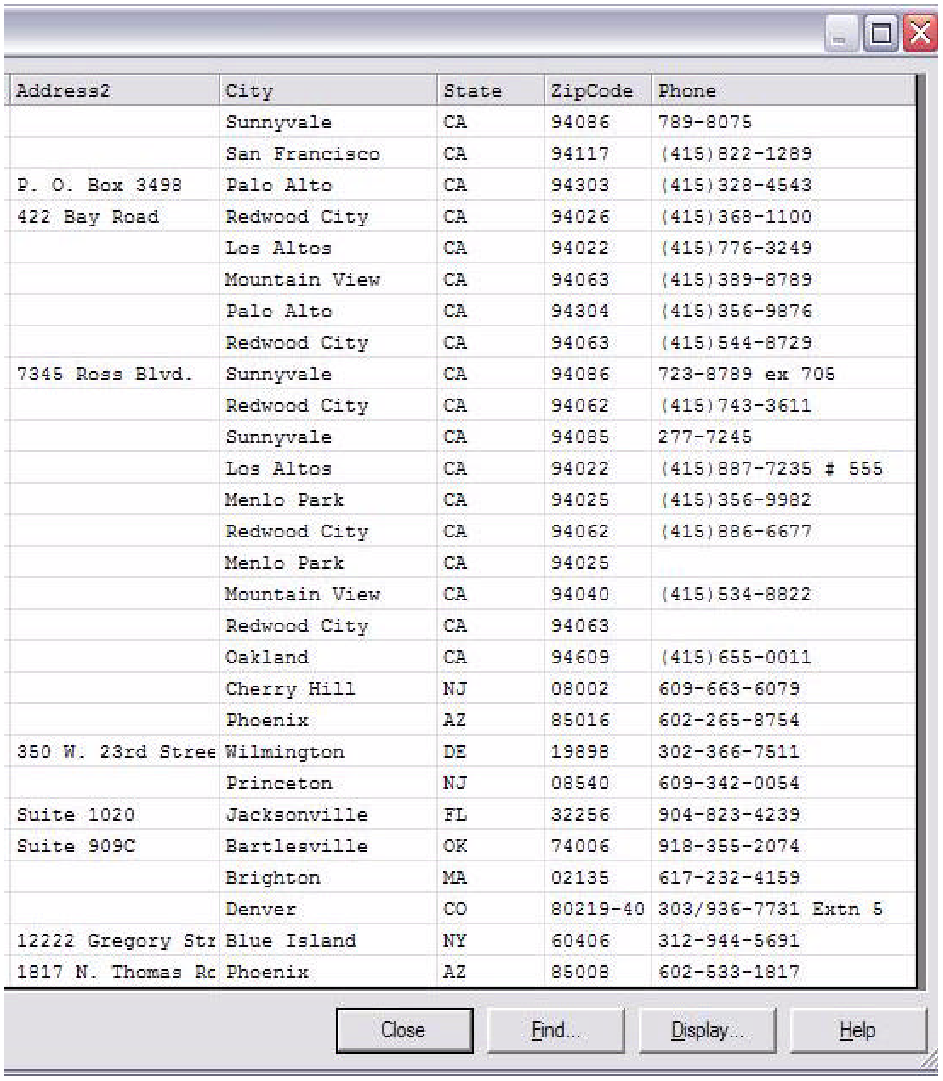
In this edition of IISDN we build two QualityStage custom rules. Comments include;

– We use the same input sample data set from the previous edition of IISDN. This sample data set is displayed in Figure23-1 and Figure23-2.



CustomerNumber	FirstName	LastName	Company	Address1
101	Ludwig	Pauli	All Sports Supplies	213 Erstwild Court
102	Carole	Sadler	The Sports Spot	785 Geary St
103	Philip	Currie	Phil's Sports	654 Poplar
104	Anthony	Higgins	Play Ball!	East Shopping Cntr.
105	Raymond	Vector	Los Altos Sports	1899 La Loma Drive
106	George	Watson	Watson & Son	1143 Carver Place
107	Charles	Ream	Athletic Supplies	41 Jordan Avenue
108	Donald	Quinn	Quinn's Sports	587 Alvarado
109	Jane	Miller	Sport Stuff	Mayfair Mart
110	Roy	Jaeger	AA Athletics	520 Topaz Way
111	Frances	Keyes	The Sports Center	3199 Sterling Court
112	Margaret	Lawson	Runners & Others	234 Wyandotte Way
113	Lana	Beatty	Sportstown	654 Oak Grove
114	Frank	Albertso	The Sporting Place	947 Waverly Place
115	Alfred	Grant	Gold Medal Sports	776 Gary Avenue
116	Jean	Parmelee	Olympic City	1104 Spinosa Drive
117	Arnold	Sipes	Kids Korner	850 Lytton Court
118	Dick	Baxter	Blue Ribbon Sports	5427 College
119	Bob	Shorter	The Triathletes Club	2405 Kings Highway
120	Fred	Jewell	Century Pro Shop	6627 N. 17th Way
121	Jason	Wallack	City Sports	Lake Biltmore Mall
122	Cathy	O'Brian	The Sporting Life	543 Nassau Street
123	Marvin	Hanlon	Bay Sports	10100 Bay Meadows Ro
124	Chris	Putnum	Putnum's Putters	4715 S.E. Adams Blvd
125	James	Henry	Total Fitness Sports	1450 Commonwealth Av
126	Eileen	Neelie	Neelie's Discount Sp	2539 South Utica Str
127	Kim	Satifer	A Big Blue Bike Shop	Blue Island Square
128	Frank	Lessor	Phoenix University	Athletic Department

Figure 23-1 Sample data, part 1 of 2.



Address2	City	State	ZipCode	Phone
	Sunnyvale	CA	94086	789-8075
	San Francisco	CA	94117	(415)822-1289
P. O. Box 3498	Palo Alto	CA	94303	(415)328-4543
422 Bay Road	Redwood City	CA	94026	(415)368-1100
	Los Altos	CA	94022	(415)776-3249
	Mountain View	CA	94063	(415)389-8789
	Palo Alto	CA	94304	(415)356-9876
	Redwood City	CA	94063	(415)544-8729
7345 Ross Blvd.	Sunnyvale	CA	94086	723-8789 ex 705
	Redwood City	CA	94062	(415)743-3611
	Sunnyvale	CA	94085	277-7245
	Los Altos	CA	94022	(415)887-7235 # 555
	Menlo Park	CA	94025	(415)356-9982
	Redwood City	CA	94062	(415)886-6677
	Menlo Park	CA	94025	
	Mountain View	CA	94040	(415)534-8822
	Redwood City	CA	94063	
	Oakland	CA	94609	(415)655-0011
	Cherry Hill	NJ	08002	609-663-6079
	Phoenix	AZ	85016	602-265-8754
350 W. 23rd Stree	Wilmington	DE	19898	302-366-7511
	Princeton	NJ	08540	609-342-0054
Suite 1020	Jacksonville	FL	32256	904-823-4239
Suite 909C	Bartlesville	OK	74006	918-355-2074
	Brighton	MA	02135	617-232-4159
	Denver	CO	80219-40	303/936-7731 Extn 5
12222 Gregory Str	Blue Island	NY	60406	312-944-5691
1817 N. Thomas Rc	Phoenix	AZ	85008	602-533-1817

Figure 23-2 Sample data, part 2 of 2.

– The first custom rule set we will create within QualityStage will take the Company (Name) column from Figure23-1, and remove common words from the leading portion of this text. For example,

The Sport's Center

A Big Blue Bicycle Shop

become,

Sport's Center, The

Big Blue Bicycle Shop, A

This is a very handy routine to have, allowing you to sort and aggregate data more accurately and effectively.

– The second rule set we will create is measurably more complicated, and will take the PhoneNumber column, and seek to standardize that.

– By means of overview, our activities will include first creating a custom rule set, then creating a DataStage/QualityStage Parallel Job that makes use of that custom rule in the form of a Standardize stage.

23.2 Create the first custom rule set, Company Name

Creating and testing a new QualityStage custom rule set is done inside the DataStage/QualityStage graphical Designer program. Complete the following steps;

1. Create a sample data set similar to that as displayed in Figure23-1 and Figure23-2. It is important that you have the same or similar conditions within your data as outlined above.
2. Log on to the DataStage/QualityStage graphical Designer program, and create the new (and also initially empty) custom rule set.

   Log on to the DataStage/QualityStage graphical Designer program.

   Just as one makes a new Parallel Job, so too do you make a new custom rule set. From the Menu Bar, select,

   File -> New -> Data Quality -> Rule Set.

   The new rule name must be composed entirely of uppercase letters; the user interface does not enforce this rule, but its there none the less. (We called our rule, RS01_MYCOMPANY.) You may save the rule anywhere in the Repository view that you wish.

   Click OK when you are done.

   This action will produce the entries as displayed in Figure23-3; while several new entries are present, and represent the many component pieces of this newly created custom rule set, the only entity you interact with directly is the entry with the icon named "SET". ("SET" is also displayed in red.)

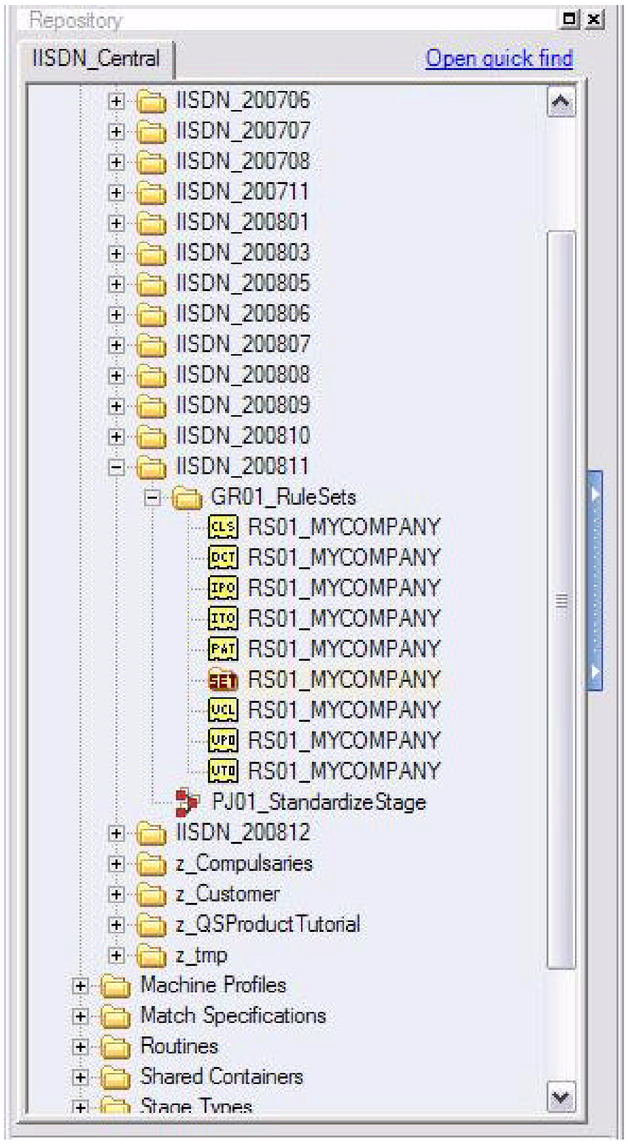


Figure 23-3 Repository view displaying new custom rule set, RS01_MYCOMPANY.

3. Double-click the newly created custom rule entry with the "SET" icon. This action produces the dialog box as displayed in Figure 23-4.

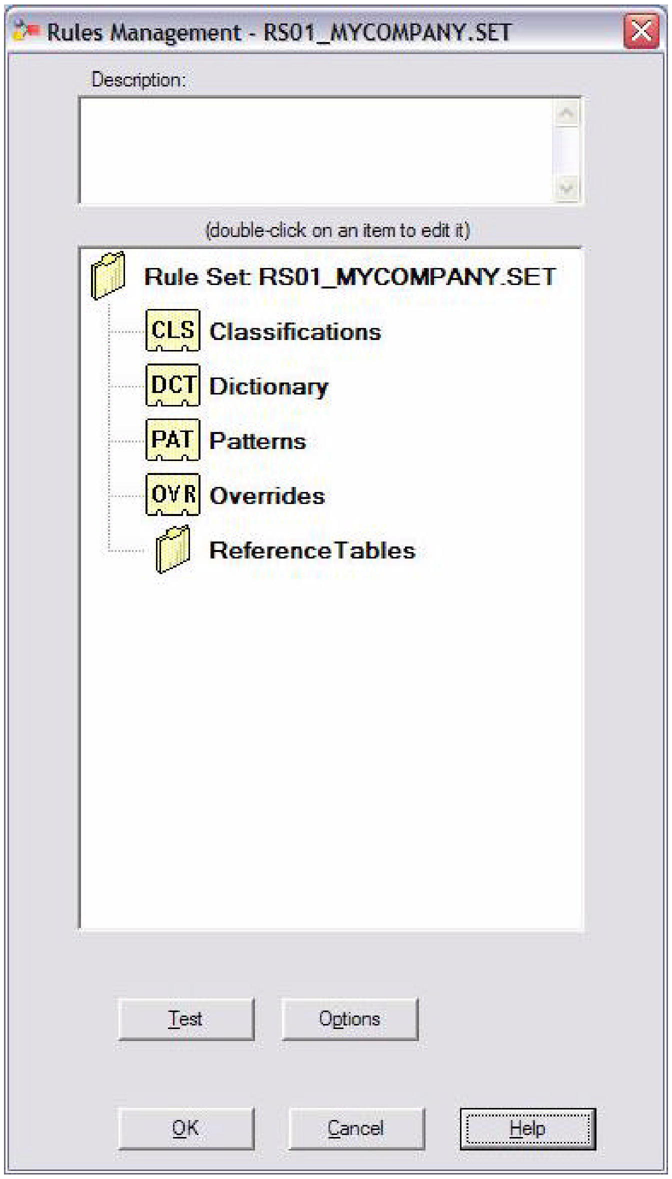


Figure 23-4 Rule Set configuration and testing dialog box.

Comments related to Figure23-4 include;

– The dialog box displayed in Figure 23-4 is your gateway to everything you need to do to this rule set; program it, test it, etcetera.

This dialog box also lists the 5 primary components to a rule set. A newly created rule set is non-functional; it doesn't know what you want to output or how to process the input it receives.

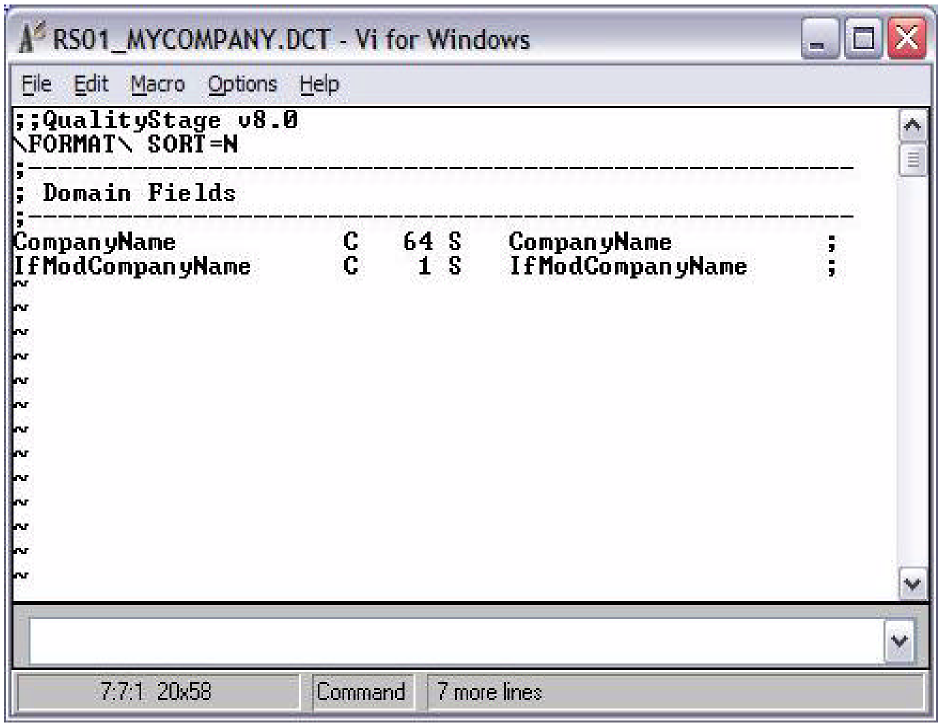
Note: Right now we are in overview/discussion mode. Don't worry about completing these steps at this time.

While all 5 components to a rule set are always present, you only need to edit 2 components to get started. Further comments include;

- The Dictionary File-

  This is one of the 5 components to a rule set that you must edit in order for the rule set to function.

  The Dictionary File specifies the output record format (columns) that this rule set produces. Figure23-5 displays the Dictionary File we will create for our first rule set. A code review of the Dictionary File contents displayed in Figure23-5 follows.



```
;;QualityStage v8.0
\FORMAT\ SORT=N
;-------------------------------------------------------------
; Domain Fields
;-------------------------------------------------------------
CompanyName              C    64 S    CompanyName         ;
IfModCompanyName         C     1 S    IfModCompanyName    ;
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Figure 23-5 Rule set Dictionary File for the first rule set we will create.

A code review for the Dictionary File displayed in Figure23-5 includes;

Lines 1-2, are mandatory in their location, format, everything. These lines are required "as is"; don't edit or move lines 1 and 2. These lines will be present in a newly created rule set Dictionary File.

Lines 3-5, are comment lines. Comments start with a semicolon and continue until the end of line marker. These lines are present in a newly created rule set Dictionary File.

Lines 6-7, specify that this Dictionary File, this rule set, outputs two columns.

In the case of Line 6, we call to output a column entitled, CompanyName, of type CHAR(64).

In the case of Line 7, we call to output a column entitled, IfModCompanyName, of type CHAR(1).

We don't know what goes in these two columns yet, we are merely specifying that these two columns exist as our output record format.

Note: The Dictionary File entries in Figure23-5 specify that we output two columns. If we wished to output 10 or 20 columns, then we would have 10 or 20 lines where Lines 6 and 7 currently are found.

Generally there are three functional types of columns that a Dictionary File, from an associated rule set, output or contains. Loosely these are referred to;

- Business intelligence columns-

  Like those displayed above, the basic output of a rule set.
- Reporting columns-

  Generally return unhandled or exception data.
- Matching columns-

  Return the NYSIIS, or SOUNDEX/Reverse-SOUNDEX values of business intelligence columns; QualityStage having these libraried capabilities built in to it.

  The topics of NYSIIS and SOUNDEX are beyond the scope of this document, and are not discussed further here.

What a given single line entry for an output column may contain, including output data type, etcetera, is another (detailed) topic not discussed further in this document.

- The Classification File-

  This is *not* one of the 5 components to a rule set that you must edit in order for the rule set to function.

  We list this component second, because our example rule will make use of this rule set component, and this component is small; an easy topic to discuss before moving on to the next component which is large, and far reaching.

  The Classification File specifies the *user-defined pattern classes*. There are *simple pattern classes* (10 or so), that are pre-built into all rule sets. User-defined pattern classes let you extend and enhance the simple pattern classes you already have.

Note: So what then is a pattern class anyway?

In effect, your rule set receives an input string that you will wish to process, cleanse, standardize, whatever verb you wish to apply. The rule set will take this input string and break it into smaller manageable pieces called tokens (also sometimes called words, no difference, just a synonym for token).

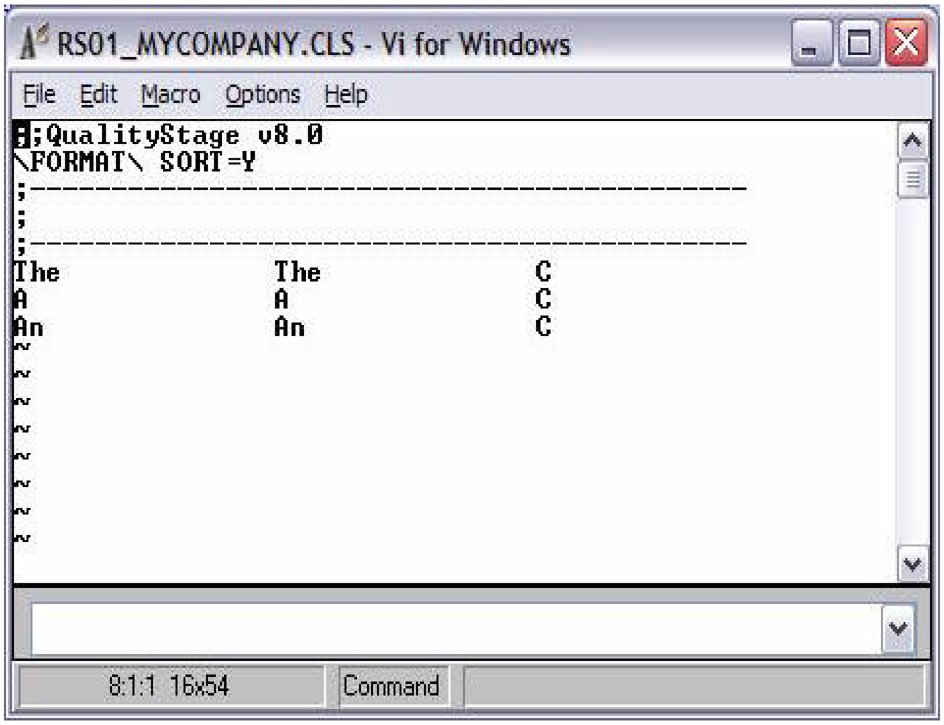
How the rule set breaks the larger input string into tokens is determined by something called a SEPLIST, (not yet shown or discussed here). Also, the rule set will offer to strip whatever characters you want out of this larger input string, (also not yet shown or discussed here).

To aid you in your processing, the rule set will identify these individual tokens as being of given type (a given pattern class). For example, the rule set might say, "I just gave you 4 tokens, 1 number followed by three words".

This categorization of what type of token you just received allows you to start coarsely; for example, "I got a number", and then proceed with something more specific like, "Yes, but is it the correct number?".

Again, there are built in pattern classes (simple pattern classes), and then those you create (user-defined pattern classes). And the Classification File is where you create the user-defined pattern classes.

Figure23-6 displays the Classification File we will create for our first rule set. A code review of the Classification File contents displayed in Figure23-6 follows.


RS01_MYCOMPANY.CLS - Vi for Windows
File Edit Macro Options Help
8:1:1 16x54
Command

```
;QualityStage v8.0
\FORMAT\ SORT=Y
;-----------------------------------------------
;
;-----------------------------------------------
The                 The                 C
A                   A                   C
An                  An                  C
~
~
~
~
~
~
~
~
```

Figure 23-6   Rule set Classification File for the first rule set we will create.

A code review for the Dictionary File displayed in Figure23-6 includes;

Lines 1-2, are mandatory in their location, format, everything. These lines are required "as is"; don't edit or move lines 1 and 2. These lines will be present in a newly created rule set Dictionary File.

Lines 3-5, are comment lines. Comments start with a semicolon and continue until the end of line marker. These lines are present in a newly created rule set Dictionary File.

Lines 6-8, specify that this Classification File, this rule set, will identify the words, "The", "A", and "An" by the user-defined pattern class "C". If we wished to add a fourth word to the pattern class of "C", we'd add a fourth line also with a third column of "C".

The design intent, what we are trying to accomplish with this rule set, is to receive Company Names and remove common words (like "The", "A", and "An") from their prefix, and then append these common words on the end of the Company Name. For example,

The Sport's Shop

becomes,

Sport’s Shop, The

A simple rule certainly, but a useful one none the less.

Currently we can specify as many as 26 user-defined pattern classes, from the letters A-Z, and you may have thousands of recognized words in this file. (We rarely use more than 10 or 15 user-defined pattern classes. If you found a valid use case for 26+ user defined pattern classes, you might be best served to split this (need) into multiple Classification Files, and as a result, multiple rule sets; always seeking to keep things simple.

Note: There is an optional fourth column to the Classification File that we are not displaying or discussing at this time, a *threshold weight* column. For example, given an entry similar to,

Mississippi Mississippi S 700

This rule set would recognize the word Mississippi by the user-defined token S, even if the inputted data were missing a brief letter “i”, “p”, or “s” somewhere. (We would likely create entries for every state name to return the user-defined pattern class “S”.)

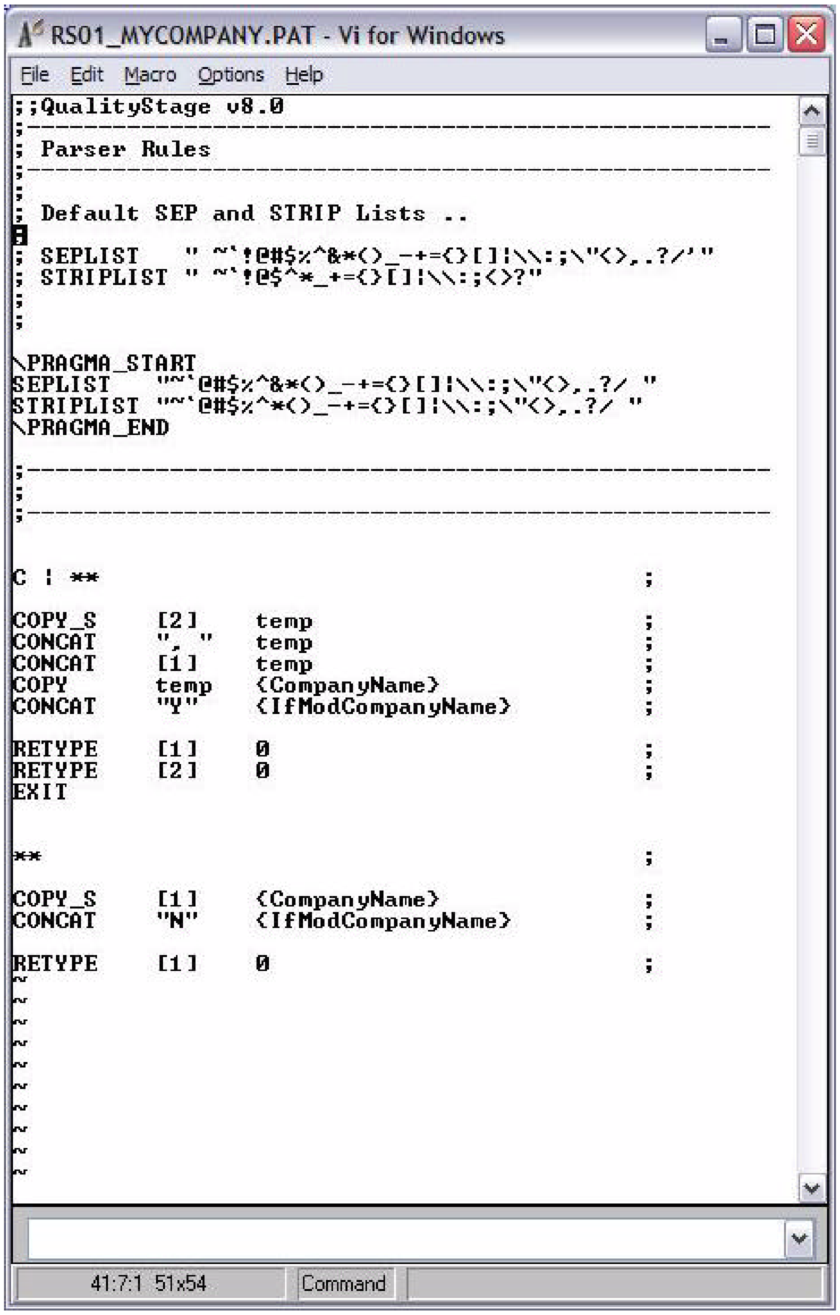
This feature is useful for near mis-spellings and such.

- The Pattern Action File-

  This is one of the 5 components to a rule set that you must edit in order for the rule set to function.

  By volume, probably 95% or more of what we do is in the Pattern Action File.

  Figure23-7 displays the Pattern Action File we will create for our first rule set. A code review of the Pattern Action File contents displayed in Figure23-7 follows.


RS01_MYCOMPANY.PAT - Vi for Windows
File Edit Macro Options Help
41:7:1 51x54
Command

```
;;QualityStage v8.0
;-----------------------------------------------------
; Parser Rules
;-----------------------------------------------------
;
; Default SEP and STRIP Lists ..
;
; SEPLIST   " ~`!@#$%^&*()_-+={}[]|\\:;\"<>,.?/'"
; STRIPLIST " ~`!@$^*_+={}[]|\\:;<>?"
;
;

\PRAGMA_START
SEPLIST   "~`@#$%^&*()_-+={}[]|\\:;\"<>,.?/ "
STRIPLIST "~`@#$%^*()_-+={}[]|\\:;\"<>,.?/ "
\PRAGMA_END

;-----------------------------------------------------
;
;-----------------------------------------------------


C | **                                            ;

COPY_S     [2]     temp                           ;
CONCAT     ", "    temp                           ;
CONCAT     [1]     temp                           ;
COPY       temp    {CompanyName}                  ;
CONCAT     "Y"     {IfModCompanyName}             ;

RETYPE     [1]     0                              ;
RETYPE     [2]     0                              ;
EXIT


**                                                ;

COPY_S     [1]     {CompanyName}                  ;
CONCAT     "N"     {IfModCompanyName}             ;

RETYPE     [1]     0                              ;
~
~
~
~
~
~
~
~
~
~
```

Figure 23-7 Rule set Pattern Action File for the first rule set we will create.

A code review for the Pattern Action File displayed in Figure23-7 includes;

Line 1, is mandatory in its location, format, everything. This line is required "as is"; don't edit or move line 1. This line will be present in a newly created rule set Pattern Action File.

A number of comment lines follow, as denoted by the leading semi colon character.

The block of lines delimited by the "PRAGMA" words include the SEPLIST and STRIPLIST. The default SEPLIST and STRIPLIST are entered above as comments.

The SEPLIST specifies which characters are used to divide the larger input string into tokens. Recall that these tokens are then identified as being a user-defined or simple pattern class. The STRIPLIST are those characters removed from the input string.

Note: From our example, our sample data contains the three special characters of "'" (a single right quote), "&", and "!". These characters appear in the Company Name we are processing/cleansing. We certainly don't wish to absently strip (STRIPLIST) these characters. Its okay if we strip them, but we then have the responsibility to encode and recreate them later.

We probably wish to at least consider stripping the other punctuation characters; if for no other reason, we might do so for security. Depending on where this data may eventually get hosted, a PhP server, an HTML server, etcetera; Do any of these special characters allow for call outs to a larger or less secure environment. For example in SQL, an implicit call to a SQL Stored Procedure (SPL) requires the presence of parenthesis; no parenthesis in the source data, no call outs to SQL SPL.

The question of token delimiters is our only real question on this simple example; what determines a word that is then available to be identified as a user-defined pattern class or simple pattern class. Since we are needing to discover input strings that lead with a common word and then a space, we should at least delimit on the space character.

If you have a sharp eye, you'll notice we don't SEPLIST on "'" (a single right quote) or the "!" characters. This has to do with how tokens are first separated *but then also how they are reassembled*. If "'" and "!" are in the SEPLIST, you'll inherit the need for more lines of code as you pick up extra spaces surrounding the "'" and "!" that you don't want. (Extra spaces around "&" were okay.)

After the PRAGMA lines begin our real Pattern Action blocks. The first real Pattern Action block begins with the line,

C | ** ;

This line specifies that anytime we receive the user-defined token of "C"/any common word (from our Classification File, the user-defined pattern class of "C", representing the words "The", "A" and "An"), then followed by anything (represented by "**"), perform this action block.

Contrast this line by that which starts the next pattern action block, denoted by,

** ;

"**" would catch [any] larger/remaining input string.

Note: Notice we check for the more restrictive line first "C | ** ;", then the (basically catches every line regardless via "**").

For just the "C | ** ;" pattern action block, there are 8 (count) pattern action commands. These include,

COPY_S [2] temp ;

This line calls to copy, while preserving spaces (_S), the portion of the larger input string represented by the "**" simple pattern class; basically, everything after the first word, which is either "The", "A", or "An". "[2]" represents the portion of the larger input string identified by the "**", the second token.

This second token is copied to a new variable named "temp"; we could just as easily called this variable "Bob", "temp" is not a pattern action language keyword.

After the first copy, we concatenate a comma and a space to the variable entitled, "temp"; represented by the two (count) CONCAT command lines.

Then we concatenate the first token, which was of the user defined type "C", currently held in the [1] token space, also to the variable entitled, "temp".

The fourth line copies the variable entitled "temp", to the output column entitled, CompanyName. And the fifth line copies a character constant of "Y", to the output column entitled, IfModCompanyName (if you are receiving a modified Company Name: Y-Yes, N-No).

Per our design, we really only have two types of input lines; those that start with a common word and those that do not. And we only really need this first test to determine same. If effect, we really only need the

EXIT pattern action statement that follows. We don't need the two RETYPE commands, and they are offered only for further detail; The RETYPE commands remove, in this case, the two identified tokens "[1]" and "[2]" from any further processing. Since these lines are followed by an EXIT, the RETYPE(s) are not required.

Note: The entries in the Pattern Action file (the individual Pattern Action blocks), are executed for every input line that is received. Each Pattern Action block is checked in sequential order for a match against the larger input line. The input line may match zero or more Pattern Action blocks.

In our example, a single input line will either match the first Pattern Action block, or it will not. Because the first Pattern Action block contains an "EXIT" command, a match here will disregard the remainder of the entries in the Pattern Action file.

In our example, there are only two Pattern Action blocks; hence, if we reach the second Pattern Action block, an EXIT is not required here since we have already reached the end of file.

In larger, more common and complex examples, we will match a given Pattern Action block, and RETYPE any tokens we have finished processing to a type "0"/NULL; this form of the RETYPE command removes these tokens from further processing. This usage is common, for example, when we have a larger input string with say a street address, then City, then State. You would commonly start with Pattern Action blocks to process *and remove* (RETYPE) the street address, then State, then City.

Returning to creation of this example custom rule set

Returning to the creation of our current custom rule set creation, we are currently in the DataStage/QualityStage Designer program, and have opened the newly created custom rule set property dialog box, equal to that as displayed in Figure23-4.

4. Alter the contents of the newly created customer rule set, RS01_MYCOMPANY.
    a. Optional step-

       The "Options" button displayed in Figure23-4 allows you to set the editor you use to alter the component pieces of a rule set. In the examples displayed above, we used the Vi(C) for Windows editor, but any favorite editor you wish is allowed. An editor similar to Windows Notepad is the default.
    b. In Figure23-4, Double-click the entry for Dictionary.

Alter the contents of this file to equal those as displayed in Figure23-5.

Select Save when you are done.

c. In Figure23-4, Double-click the entry for Classifications.

Alter the contents of this file to equal those as displayed in Figure23-6.

Select Save when you are done.

d. In Figure23-4, Double-click the entry for Patterns.

Alter the contents of this file to equal those as displayed in Figure23-7.

Select Save when you are done.

e. Save your work thus far.

Under the absolute worst of circumstances, you can so badly alter/ruin the contents of the three files we worked with above that you may forcibly exit the product and lose all of your work. For that reason, we will save our work thus far.

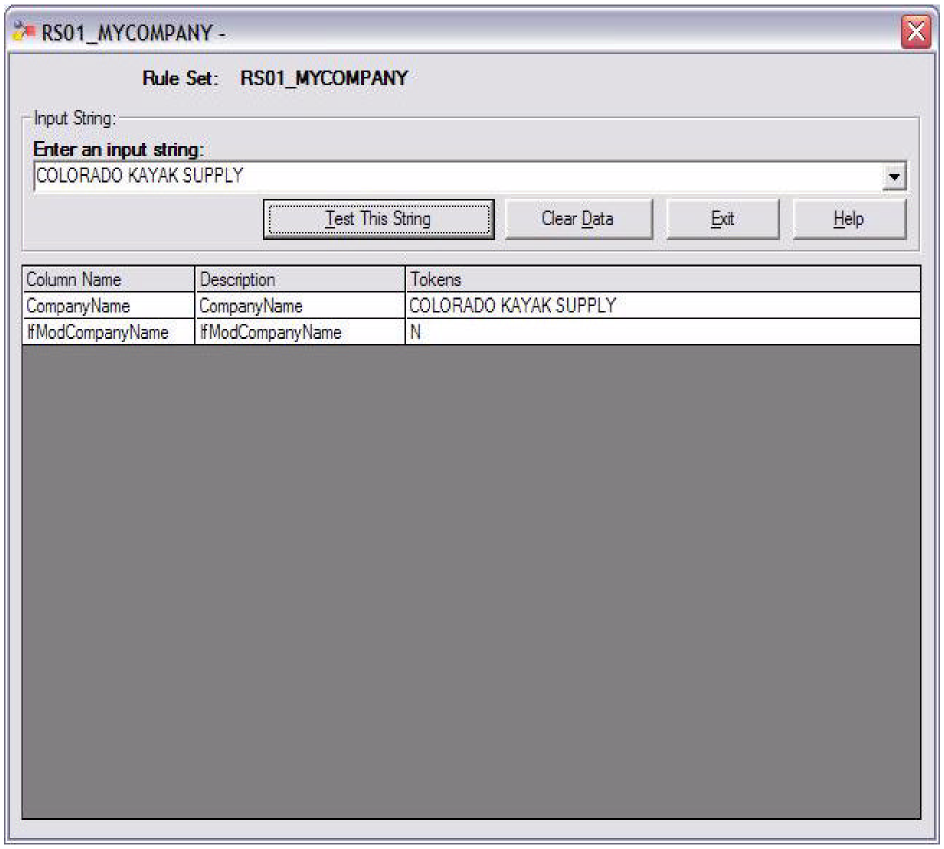
In Figure23-4, Click OK to save your work.

Re-enter the properties dialog for this custom rule set as you did in Step-3 above.

f. Test your rule set.

In Figure23-4, click the button entitled, Test.

This action produces the display as shown in Figure23-8.



COLORADO KAYAK SUPPLY

Column Name	Description	Tokens
CompanyName	CompanyName	COLORADO KAYAK SUPPLY
IfModCompanyName	IfModCompanyName	N

Figure 23-8 Testing a Rule Set, custom or pre-built.

Test the Company Names,

The Sport's Spot

A Big Blue Bicycle Company

Colorado Kayak Supply

And check your results. You should receive,

Sport's Spot, The Y

Big Blue Bicycle Company, A Y

Colorado Kayak Supply N

When you are done, Click OK to exit the rule set properties dialog box.

23.3 Create a DS/QS Job that uses this custom rule

Now that we have created our QualityStage custom rule, we can create a DataStage/QualityStage Job that uses the Standardize stage, and invokes our rule/custom-rule. Figure23-9 displays the DataStage/QualityStage Job we create in this section.

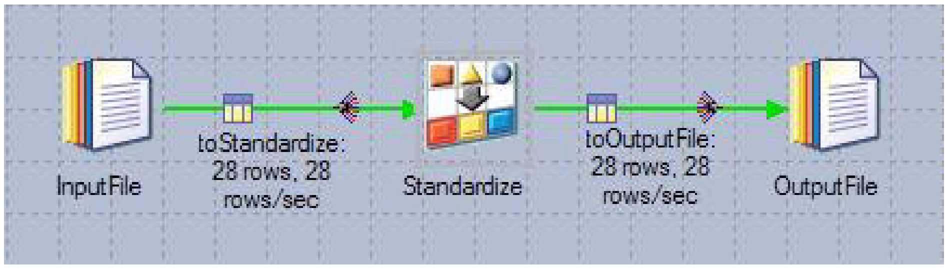

InputFile
toStandardize: 28 rows, 28 rows/sec
Standardize
toOutputFile: 28 rows, 28 rows/sec
OutputFile

Figure 23-9 DS/QS Job to make use of rule set, Standardize stage.

5. Before we can make use of the newly created custom rule set, we must *Provision All* it. We wont define *Provision All / Provisioning* other than to say that it is required.

   In the Repository view of the DataStage/QualityStage Designer program, locate the entry for our newly created standardization rule, example as shown in Figure23-3.

   Right-click the entry for this rule set entitled, SET, and Select, Provision All.

   This action should be performed after any modifications to a given rule set.

6. Create a new DataStage/QualityStage Job equal to that as displayed in Figure23-9.

   a. Save this newly created DataStage/QualityStage Job with a given name.

   a. Set the properties on the Input File so that you may read the sample data file we created in Step-1 above.

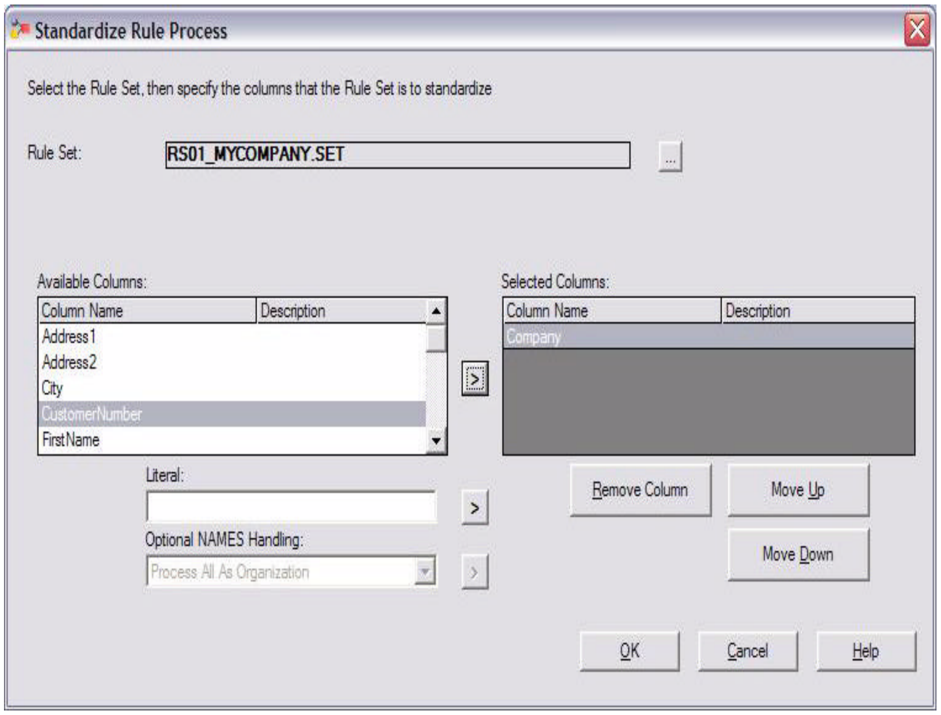
   b. Set the Output File properties.

   c. Double-click the Standardize stage to access its Properties dialog box.

      i. In the Drop Down List Box entitled, Default Standardize Output Format, Select, "Preserve the case".

      ii. Click the TAB entitled, New Process.

      This action produces the display as shown in Figure23-10.



Standardize Rule Process

Select the Rule Set, then specify the columns that the Rule Set is to standardize

Rule Set: RS01_MYCOMPANY.SET

Available Columns:

Column Name	Description
Address1	
Address2	
City	
CustomerNumber	
FirstName	

Selected Columns:

Column Name	Description
Company	

Literal:

Optional NAMES Handling: Process All As Organization

Figure 23-10 Standardize stage, New Process dialog box.

iii. In Figure23-10, Click the ellipsis button "..." to the right of the text entry field entitled, Rule Set. Browse to and Select your newly create rule entitled, RS01_MYCOMPANY.

If you can't find this specific rule set, you likely forgot to Provision All it earlier.

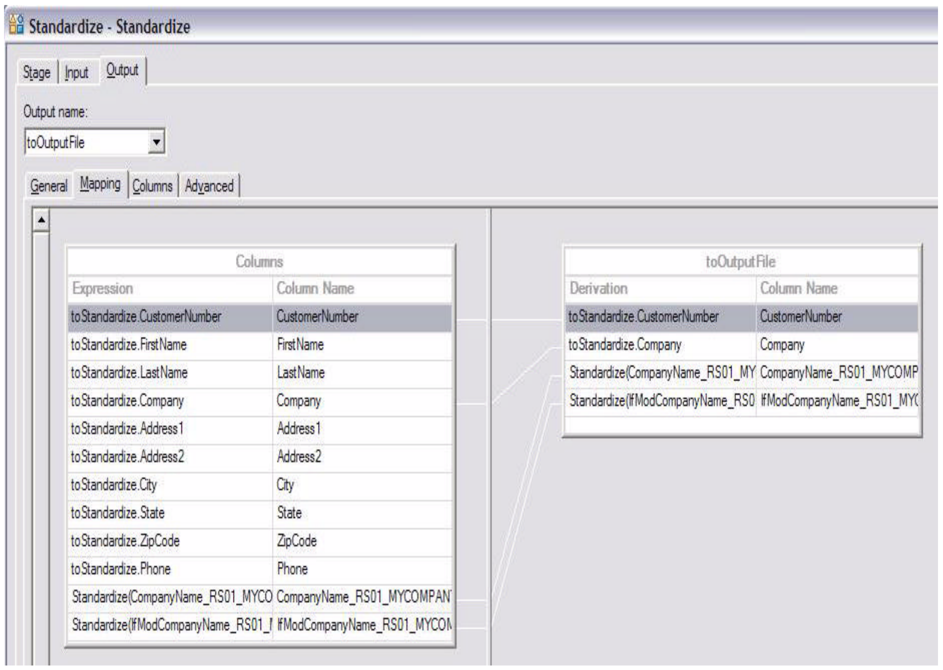
iv. Move the sample input column entitled Company, from the left side of the display to the right; from the Available columns to the Selected Columns.

v. Click OK to close the Standardize Stage -> New Process dialog box.

vi. Click the TAB entitled, Stage Properties -> Output -> Mapping.

Drag and Drop and columns to the right side of the dialog box as displayed in Figure23-11.

The net here is we want the 2 newly outputted columns from our rule, plus perhaps the primary key of the record (CustomerNumber), and the before image of the Company Name for comparisons.



Output name: toOutputFile

Columns

Expression	Column Name
toStandardize.CustomerNumber	CustomerNumber
toStandardize.FirstName	FirstName
toStandardize.LastName	LastName
toStandardize.Company	Company
toStandardize.Address1	Address1
toStandardize.Address2	Address2
toStandardize.City	City
toStandardize.State	State
toStandardize.ZipCode	ZipCode
toStandardize.Phone	Phone
Standardize(CompanyName_RS01_MYCO	CompanyName_RS01_MYCOMPAN
Standardize(fModCompanyName_RS01_I	fModCompanyName_RS01_MYCOM

toOutputFile

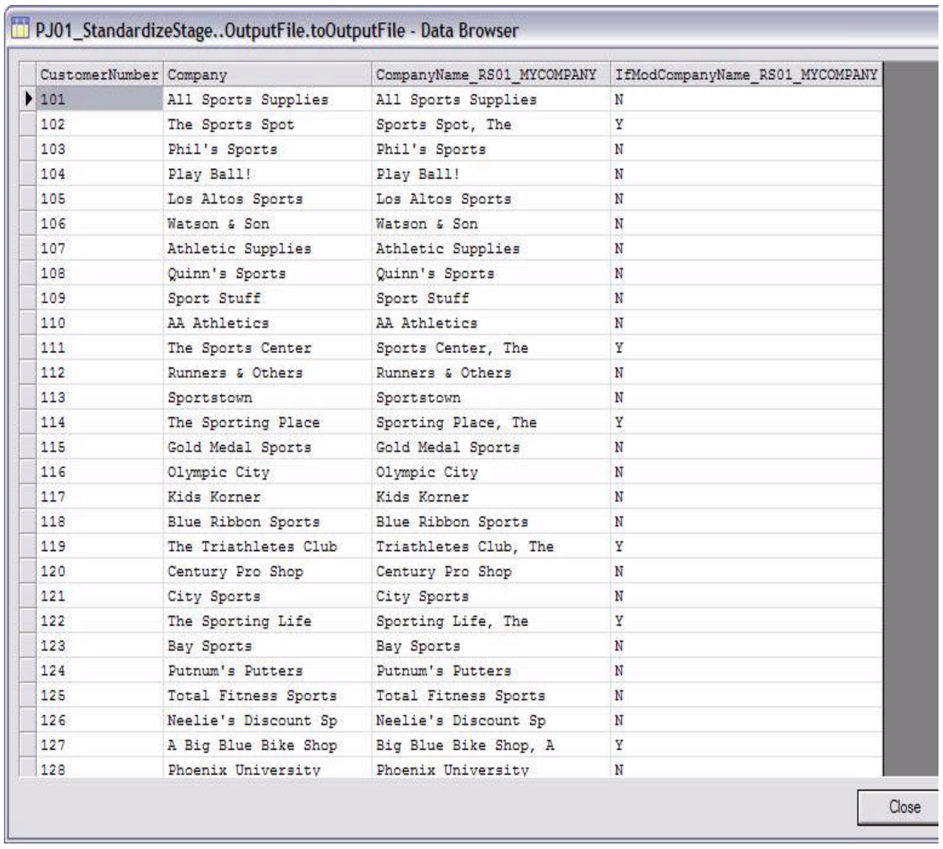
Derivation	Column Name
toStandardize.CustomerNumber	CustomerNumber
toStandardize.Company	Company
Standardize(CompanyName_RS01_MY	CompanyName_RS01_MYCOMP
Standardize(fModCompanyName_RS0	fModCompanyName_RS01_MY(

Figure 23-11 Standardize stage -> Stage Properties -> Output -> Mapping TAB.

vii. Click OK twice to exist the properties dialogs for this stage.

viii. Save, Compile and Test this Job.

A successful test of this Job appears in Figure23-12.


PJ01_StandardizeStage..OutputFile.toOutputFile - Data Browser
Close

CustomerNumber	Company	CompanyName_RS01_MYCOMPANY	IfModCompanyName_RS01_MYCOMPANY
101	All Sports Supplies	All Sports Supplies	N
102	The Sports Spot	Sports Spot, The	Y
103	Phil's Sports	Phil's Sports	N
104	Play Ball!	Play Ball!	N
105	Los Altos Sports	Los Altos Sports	N
106	Watson & Son	Watson & Son	N
107	Athletic Supplies	Athletic Supplies	N
108	Quinn's Sports	Quinn's Sports	N
109	Sport Stuff	Sport Stuff	N
110	AA Athletics	AA Athletics	N
111	The Sports Center	Sports Center, The	Y
112	Runners & Others	Runners & Others	N
113	Sportstown	Sportstown	N
114	The Sporting Place	Sporting Place, The	Y
115	Gold Medal Sports	Gold Medal Sports	N
116	Olympic City	Olympic City	N
117	Kids Korner	Kids Korner	N
118	Blue Ribbon Sports	Blue Ribbon Sports	N
119	The Triathletes Club	Triathletes Club, The	Y
120	Century Pro Shop	Century Pro Shop	N
121	City Sports	City Sports	N
122	The Sporting Life	Sporting Life, The	Y
123	Bay Sports	Bay Sports	N
124	Putnum's Putters	Putnum's Putters	N
125	Total Fitness Sports	Total Fitness Sports	N
126	Neelie's Discount Sp	Neelie's Discount Sp	N
127	A Big Blue Bike Shop	Big Blue Bike Shop, A	Y
128	Phoenix University	Phoenix University	N

Figure 23-12 Successful result from test of first Job.

23.4 Create a more complex custom rule

The QualityStage custom rule set we created in Section 23.2, "Create the first custom rule set, Company Name" on page9 is a good first rule. It covers;

– The 5 entities that form a QualityStage custom rule set; 2 mandatory, and then one optional entity we used.

– How to Test and Provision All a rule set.

– Other.

In this section, we are going to create a rule that requires more pattern action blocks, and uses more of the simple pattern action classes; hopefully giving a better sense of how those work. For this example, we will standardize U.S.

telephone numbers, which per our input sample set may arrive in 6 or more input patterns. For example,

(303)555-1212 Extn: 555

(303)555-1212 Ex: 555

(303)555-1212

555-1212 Extn: 555 (For these and below, default to a zero filled area code.)

555-1212 Ex: 555

555-1212

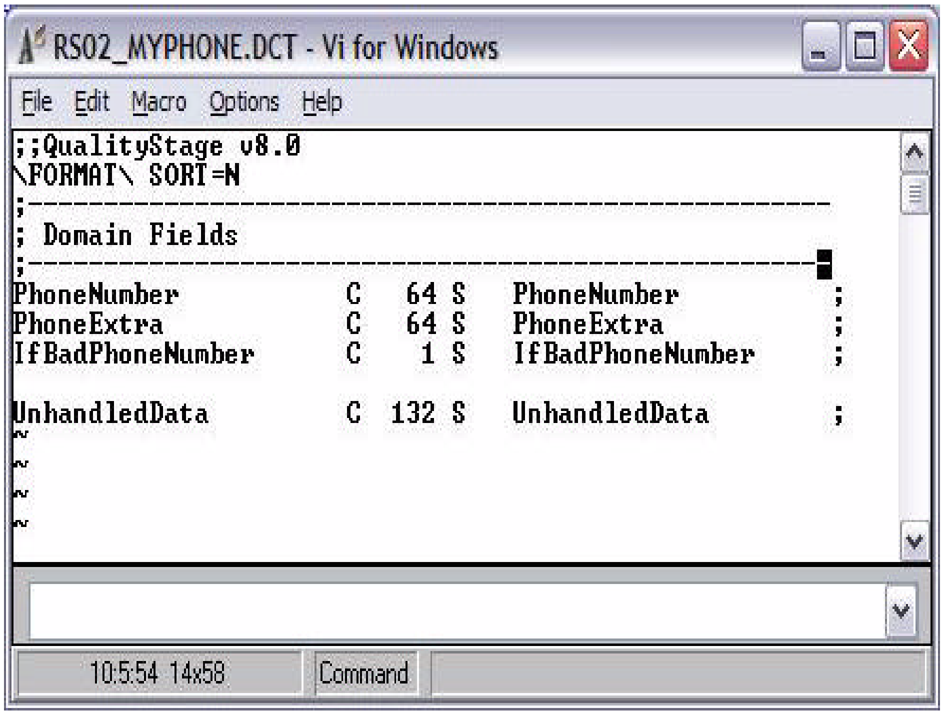
Complete the following steps:

7. Create a new custom rule set.
    a. Create a new custom rule set as we did in Step-2 above.

       Save this rule with a given name. We called ours, RS02_MYPHONE.
    b. Alter the contents of the Dictionary File, Classification File, and Pattern (Action) File as we did in Steps,3-4 above.
        i. The Dictionary File should resemble that as displayed in Figure23-13.

           With no new technology being presented in Figure23-13, no code review will follow this example.

           The output column name "UnhandledData" is not a keyword, and while it is a name of common practice, it is a name of our choosing.


RS02_MYPHONE.DCT - Vi for Windows
File Edit Macro Options Help
10:5:54 14x58
Command

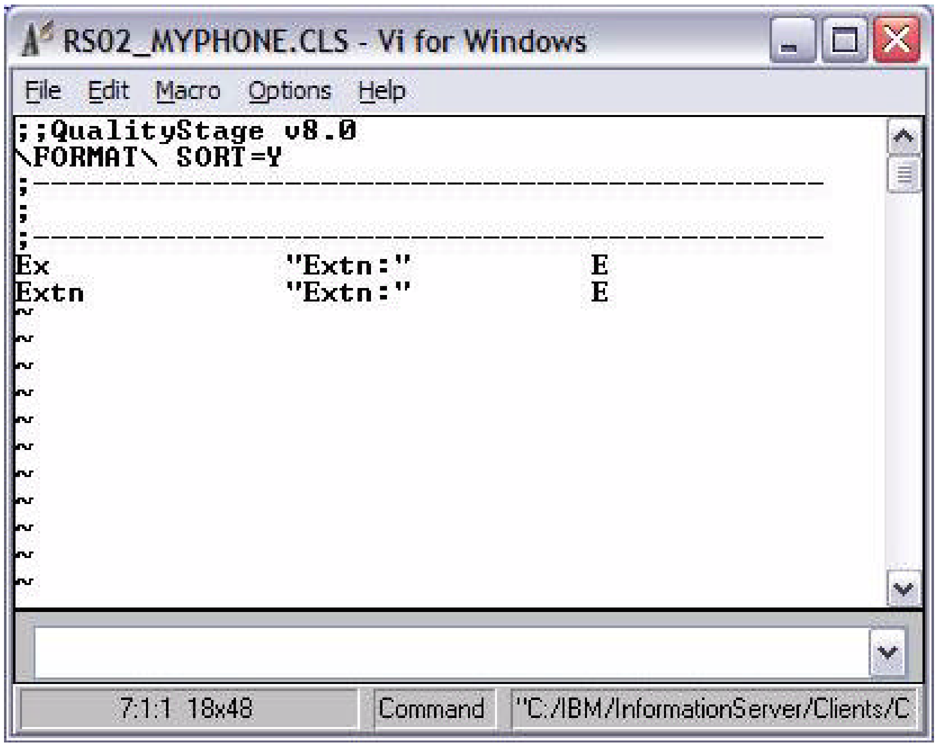
```
;;QualityStage v8.0
\FORMAT\ SORT=N
;-------------------------------------------------------
; Domain Fields
;------------------------------------------------------
PhoneNumber            C   64 S   PhoneNumber        ;
PhoneExtra             C   64 S   PhoneExtra         ;
IfBadPhoneNumber       C    1 S   IfBadPhoneNumber   ;

UnhandledData          C  132 S   UnhandledData      ;
~
~
~
~
```

Figure 23-13 Second rule, RS02_MYPHONE, Dictionary File.

ii. The Classification File should resemble that as shown in Figure23-14.

Only one slightly new technology is displayed in Figure23-14. In Figure23-14 we see that the abbreviation for Extension, "Ex" is standardized as "Extn:', just as "Extn" is. This is another, albeit smallish, use case of the Classification File.


RS02_MYPHONE.CLS - Vi for Windows
File Edit Macro Options Help
7:1:1 18x48
Command
"C:/IBM/InformationServer/Clients/C

```
;;QualityStage v8.0
\FORMAT\ SORT=Y
;---------------------------------------------
;
;---------------------------------------------
Ex               "Extn:"          E
Extn             "Extn:"          E
```

Figure 23-14 Second rule, RS02_MYPHONE, Classification File.

iii. The Pattern Action File should resemble that as shown in Example23-1.

A code review follows with select lines and comments.

Example 23-1 Second Rule, RS02_MYPHONE, Pattern Action File.

```
;;QualityStage v8.0
;-----------------------------------------------
; Parser Rules
;-----------------------------------------------

\PRAGMA_START
SEPLIST   "~`!@#$%^&*()_-+={}[]|\\:;\"'<>,.?/ "
STRIPLIST "~`@#$%^&*()_-+={}[]|\\:;\"<>,.?/ "
\PRAGMA_END
```

```
;-----------------------------------------------
; (415)555-1212 Ex  555
;
; Or basically any punctuation as separator.
;-----------------------------------------------


^ [ {} LEN = 3 ] | ^ [ {} LEN = 3 ] | ^ [ {} LEN = 4 ] | E | **  ;

CONCAT    "("     temp1                     ;
CONCAT    [1]     temp1                     ;
CONCAT    ")"     temp1                     ;
CONCAT    [2]     temp1                     ;
CONCAT    "-"     temp1                     ;
CONCAT    [3]     temp1                     ;

CONCAT_A  [4]     temp2                     ;
CONCAT    " "     temp2                     ;
CONCAT    [5]     temp2                     ;

COPY      temp1   {PhoneNumber}             ;
COPY      temp2   {PhoneExtra}              ;

RETYPE    [1]     0                         ;
RETYPE    [2]     0                         ;
RETYPE    [3]     0                         ;
RETYPE    [4]     0                         ;
RETYPE    [5]     0                         ;

CONCAT    "N"     {IfBadPhoneNumber}        ;
CONCAT    "n/a"   {UnhandledData}           ;
EXIT                                        ;


;-----------------------------------------------
; (415)555-1212
;
; Or basically any punctuation as separator.
;-----------------------------------------------


^ [ {} LEN = 3 ] | ^ [ {} LEN = 3 ] | *^ [ {} LEN = 4 ] | $     ;

CONCAT    "("     temp1                     ;
CONCAT    [1]     temp1                     ;
CONCAT    ")"     temp1                     ;
CONCAT    [2]     temp1                     ;
CONCAT    "-"     temp1                     ;
CONCAT    [3]     temp1                     ;
```

```
COPY     temp1   {PhoneNumber}            ;
CONCAT   "n/a"   {PhoneExtra}             ;

RETYPE   [1]     0                  ;
RETYPE   [2]     0                  ;
RETYPE   [3]     0                  ;

CONCAT   "N"     {IfBadPhoneNumber}       ;
CONCAT   "n/a"   {UnhandledData}          ;
EXIT                                ;


;----------------------------------------------
; (415)555-1212 ..more..
;
; Or basically any punctuation as separator.
;----------------------------------------------


^ [ {} LEN = 3 ] | ^ [ {} LEN = 3 ] | ^ [ {} LEN = 4 ] | **     ;

CONCAT   "("     temp1              ;
CONCAT   [1]     temp1              ;
CONCAT   ")"     temp1              ;
CONCAT   [2]     temp1              ;
CONCAT   "-"     temp1              ;
CONCAT   [3]     temp1              ;

COPY     temp1   {PhoneNumber}            ;
COPY     [4]     {PhoneExtra}             ;

RETYPE   [1]     0                  ;
RETYPE   [2]     0                  ;
RETYPE   [3]     0                  ;
RETYPE   [4]     0                  ;

CONCAT   "N"     {IfBadPhoneNumber}       ;
CONCAT   "n/a"   {UnhandledData}          ;
EXIT                                ;


;----------------------------------------------
; 555-1212 Ex  555
;
; Or basically any punctuation as separator.
;----------------------------------------------
```

```
^ [ {} LEN = 3 ] | ^ [ {} LEN = 4 ] | E | **                    ;

CONCAT    "(000)"  temp1                    ;
CONCAT    [1]      temp1                    ;
CONCAT    "-"      temp1                    ;
CONCAT    [2]      temp1                    ;
CONCAT    " "      temp1                    ;

CONCAT_A  [3]      temp2                    ;
CONCAT    " "      temp2                    ;
CONCAT    [4]      temp2                    ;

COPY      temp1    {PhoneNumber}            ;
COPY      temp2    {PhoneExtra}             ;

RETYPE    [1]      0                        ;
RETYPE    [2]      0                        ;
RETYPE    [3]      0                        ;
RETYPE    [4]      0                        ;

CONCAT    "Y"      {IfBadPhoneNumber}       ;
CONCAT    "n/a"    {UnhandledData}          ;
EXIT                                        ;


;----------------------------------------------
; 555-1212
;
; Or basically any punctuation as separator.
;----------------------------------------------


^ [ {} LEN = 3 ] | *^ [ {} LEN = 4 ] | $                    ;

CONCAT    "(000)"  temp1                    ;
CONCAT    [1]      temp1                    ;
CONCAT    "-"      temp1                    ;
CONCAT    [2]      temp1                    ;

COPY      temp1    {PhoneNumber}            ;
CONCAT    "n/a"    {PhoneExtra}             ;

RETYPE    [1]      0                        ;
RETYPE    [2]      0                        ;

CONCAT    "Y"      {IfBadPhoneNumber}       ;
CONCAT    "n/a"    {UnhandledData}          ;
EXIT                                        ;
```

```
;----------------------------------------------
; 555-1212  ..more..
;
; Or basically any punctuation as separator.
;----------------------------------------------


^ [ {} LEN = 3 ] | ^ [ {} LEN = 4 ] | **                      ;

CONCAT    "(000)"  temp1                      ;
CONCAT    [1]      temp1                      ;
CONCAT    "-"      temp1                      ;
CONCAT    [2]      temp1                      ;

COPY      temp1    {PhoneNumber}              ;
CONCAT    [3]      {PhoneExtra}               ;

RETYPE    [1]      0                          ;
RETYPE    [2]      0                          ;
RETYPE    [3]      0                          ;

CONCAT    "Y"      {IfBadPhoneNumber}         ;
CONCAT    "n/a"    {UnhandledData}            ;
EXIT                                          ;


;----------------------------------------------
;  None of the above
;----------------------------------------------

**                                            ;

COPY      [1]      {UnhandledData}            ;

CONCAT    "n/a"    {PhoneNumber}              ;
CONCAT    "n/a"    {PhoneExtra}               ;

CONCAT    "Y"      {IfBadPhoneNumber}         ;
EXIT                                          ;
```

Code review for Example 23-1;

- There is a lot of repetition between each of the Pattern Action blocks, and for that reason, we'll give attention to any first time or new areas within this file.
- The first Pattern Action block begins with,

  ^ [{} LEN = 3] | ^ [{} LEN = 3] | ^ [{} LEN = 4] | E | ** ;

  ^ is the simple pattern class for a token that is entirely numeric.

  [{} LEN = 3] will be true (match) only if the numeric is 3 characters long. We weren't specifically asked to perform this test, but its easy to do and a nice extra feature to add.

  In effect, the SEPLIST and STRIPLIST pull out any parenthesis, slashes, hyphens, etcetera, that may accompany a U.S. formatted phone number. All we have to check for then are three tokens similar to, 303555 1212, and that is represented by the simple pattern class set of, "^ | ^ | ^".

  "E" checks for our user-defined pattern class of "Ex" or Extn". And "**" Returns anything after that token.

  The remaining Pattern Action blocks are largely subsets of this first, and most restrictive pattern.

Note: The Pattern Action blocks in nearly every Pattern Action File move from most restrictive to least restrictive, generally.

In this example we move from most restrictive, (303)555-1212 Ex: 555, to a pure (well formatted) number telephone number, (303)555-1212. Then we move from the case of a well formatted phone number, to a phone number with something unknown following it. I.e, "(303)555-1212 Ring once, hang up, call back".

After we handle full U.S. based 10 digit phone numbers (numbers with an area code), then we have to 7 digit phone numbers; an older and less common form of U.S. based phone number.

- All of the Pattern Action commands are largely repeats from our first rule set.

Again we don't really need the RETYPE commands because each Pattern Action block contains an EXIT command.

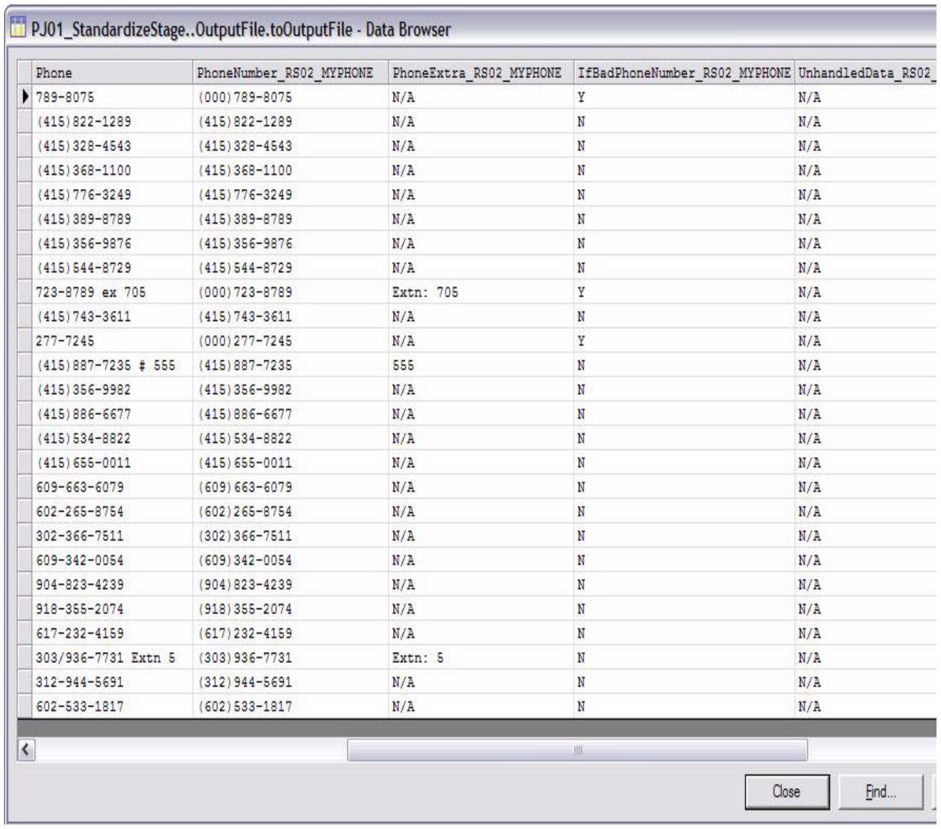
CONCAT_A is a new command to this example. CONCAT_A copies the standardized entry from our Classification file, not the native (observed) Value. In other words we copy "Extn:", not "Ex:".

c. Save, Compile, and Test this new rule set.

- You may exit this new rule set's property dialog boxes when your Dictionary, Classification, and Pattern Action Files equal those as displayed in Figure23-13, Figure23-14, and Example23-1.
- Don't forget to Provision All this new rule.
- You must also add this new rule as a New Process to the Standardize stage, as we did in Steps,6.c.i-vii above.

  The new column we are standardizing is PhoneNumber; add to the output columns list the original PhoneNumber and all columns from the RS02_MYPHONE rule set.

A successful test appears equal to that as displayed in Figure23-15.



PJ01_StandardizeStage..OutputFile.toOutputFile - Data Browser

Phone	PhoneNumber_RS02_MYPHONE	PhoneExtra_RS02_MYPHONE	IfBadPhoneNumber_RS02_MYPHONE	UnhandledData_RS02_
789-8075	(000)789-8075	N/A	Y	N/A
(415)822-1289	(415)822-1289	N/A	N	N/A
(415)328-4543	(415)328-4543	N/A	N	N/A
(415)368-1100	(415)368-1100	N/A	N	N/A
(415)776-3249	(415)776-3249	N/A	N	N/A
(415)389-8789	(415)389-8789	N/A	N	N/A
(415)356-9876	(415)356-9876	N/A	N	N/A
(415)544-8729	(415)544-8729	N/A	N	N/A
723-8789 ex 705	(000)723-8789	Extn: 705	Y	N/A
(415)743-3611	(415)743-3611	N/A	N	N/A
277-7245	(000)277-7245	N/A	Y	N/A
(415)887-7235 # 555	(415)887-7235	555	N	N/A
(415)356-9982	(415)356-9982	N/A	N	N/A
(415)886-6677	(415)886-6677	N/A	N	N/A
(415)534-8822	(415)534-8822	N/A	N	N/A
(415)655-0011	(415)655-0011	N/A	N	N/A
609-663-6079	(609)663-6079	N/A	N	N/A
602-265-8754	(602)265-8754	N/A	N	N/A
302-366-7511	(302)366-7511	N/A	N	N/A
609-342-0054	(609)342-0054	N/A	N	N/A
904-823-4239	(904)823-4239	N/A	N	N/A
918-355-2074	(918)355-2074	N/A	N	N/A
617-232-4159	(617)232-4159	N/A	N	N/A
303/936-7731 Extn 5	(303)936-7731	Extn: 5	N	N/A
312-944-5691	(312)944-5691	N/A	N	N/A
602-533-1817	(602)533-1817	N/A	N	N/A

Figure 23-15   Successful test of rule set, RS02_MYPHONE.

Note: When you create several new Pattern Action blocks at one time, it can become difficult to know which Pattern Action block is actually processing your larger input string; which pattern set did you actually match?

As a development technique, we set a trace variable.

Where we set "IfBadPhoneNumber" to "N" or "Y" in each block (a very easy setting), we temporarily set it to A, B, C, etcetera, uniquely for each block until development is complete.

By having a unique code output for each Pattern Action block, we are then certain which block is firing or giving us grief. for larger or complex rules, we create this logic and may leave it in place inside the final rule set.

Extra credit work at this point:

In the U.S., at least, a common abbreviation for telephone extension might be, "#555"; using the number sign. You can't place number sign in the classification File, because a simple pattern class already exists for it.

As an extra assignment, create a Pattern Action block that processes the number sign. the Simple pattern class would be "\#" and would likely start as the second Pattern Action block. Hint: You may have to alter the SEPLIST or STRIPLIST to complete this task.

23.5 In this document, we reviewed or created:

We examined the (IBM) InfoSphere Information Server (IIS), QualityStage component, Standardize stage. Specifically, we created and tested two QualityStage custom rule sets; a very powerful technology with nearly endless applications to standardize and cleanse data of any source or purpose.

Persons who help this month.

Lady Dayo Joseph and future daughter Danielle, and Andy Wilson.

Additional resources:

The IBM InfoSphere Information Server, QualityStage component, product tutorial.

Legal statements:

Special attributions: