Note

Before using this information and the product that it supports, read the information in "Notices and trademarks" on page 311.
## Contents

**Chapter 1. Introduction to data cleansing** ......................................................... 1
- InfoSphere QualityStage methodology ....................................................... 1
- Phase one: Understand goals and requirements ........................................... 2
- Phase two: Analyze source data ................................................................. 2
- Phase three: Design and develop jobs ......................................................... 3
- Phase four: Evaluate results ....................................................................... 5

**Chapter 2. Analyzing source data** ................................................................. 7
- Character Investigate option ...................................................................... 7
- Word Investigate option ........................................................................... 8
- Word investigation and rule sets ............................................................... 8
- Setting up to generate investigation reports .............................................. 9
  - Setting up an Investigate job ................................................................. 10
  - Configuring the source data file ............................................................ 11
  - Configuring the Investigate stage .......................................................... 12
  - Configuring the target file ..................................................................... 16
- About stage properties .............................................................................. 16
  - Setting stage properties ........................................................................ 17
  - Configure the Mapping tab ................................................................... 18
  - Investigation reports ............................................................................... 19
  - Report format for the Investigate stage ................................................... 19
  - Creating investigation reports ............................................................... 20
  - Running and viewing the report ............................................................. 21

**Chapter 3. Making data consistent through standardization** ......................... 23
- Source data preparation ............................................................................ 23
- Standardization workflow ........................................................................ 24
- Creating the Standardize stage job ............................................................ 25
  - Rule sets that are used by the Standardize stage .................................. 26
- Setting up the Standardize job ................................................................. 28
- Configuring the Standardize stage ............................................................. 28
- Predefined standardization jobs ............................................................... 32
- Standardize and correct international address data .................................... 33
  - Countries and regions supported by the MNS rule sets ......................... 33
  - Configuring the MNS stage .................................................................... 34
- Standardization Quality Assessment (SQA) reports .................................. 36
  - Adding the SQA stage ........................................................................... 36
  - Configuring the Standardize stage to generate SQA report columns ....... 37
  - Configuring the SQA stage .................................................................... 38
  - Creating SQA reports ............................................................................ 39
  - Running SQA reports ............................................................................ 40
  - Viewing SQA reports ............................................................................ 41
  - How to interpret the SQA reports ......................................................... 41

**Chapter 4. Matching data** ........................................................................... 51
- Matching goals ......................................................................................... 51
- Matching: key concepts ........................................................................... 51
  - Record linkage and the matching process ............................................. 51
  - Consider the level of information content ............................................. 52
- How InfoSphere QualityStage matches records ...................................... 53
  - Match types ......................................................................................... 54
  - Match column selection ....................................................................... 57
  - Match passes ....................................................................................... 58
  - Blocking ............................................................................................... 59
  - Weights and record comparisons ........................................................... 62
    - Composite weights ............................................................................. 63
    - Computation of weights .................................................................... 63
    - Cutoff values ..................................................................................... 65
  - Guidelines when determining probability ............................................... 66
  - Preparing to use the Match Designer .................................................... 67
  - Defining the test environment ............................................................... 68
  - Updating configuration information ...................................................... 70
  - Using the Match Designer .................................................................... 71
    - Creating match specifications ............................................................ 72
    - Testing results of match passes .......................................................... 116
    - Reporting for match specifications .................................................... 130
  - Designing matching jobs ...................................................................... 132
    - Generating frequency information: Match Frequency stage ............... 132
    - Matching one source: One-source Match stage ................................. 136
    - Matching two sources: Two-source Match stage ............................... 143
    - Predefined matching jobs ................................................................. 147
  - Match Statistics reports ....................................................................... 149
    - Creating Match Statistics reports ....................................................... 149
    - Running and viewing Match Statistics reports ................................... 150

**Chapter 5. Consolidating duplicate records to create the representative record** ... 151
- Creating a survive record ....................................................................... 152
- Setting up a survive job ........................................................................... 152
- Configuring the survive job .................................................................... 153
  - Defining simple survive rules ............................................................... 154
  - Defining complex survive expressions ................................................... 155
- Applying survive rules ............................................................................ 156
  - About the Survive stage rule processing .............................................. 157
  - Survive stage rule examples ................................................................. 158

**Chapter 6. Rule sets applied in the data cleansing process** ............................ 163
- Introduction to rule sets ........................................................................... 163
  - How rule sets are used .......................................................................... 163
  - Features and benefits of rule sets ......................................................... 163
- Objects within rule sets ........................................................................... 164
  - Classifications ...................................................................................... 164
  - Lookup tables ...................................................................................... 168
  - Output columns .................................................................................... 169
  - Rules .................................................................................................... 172
  - Rule set extensions ............................................................................... 178
  - Overrides ............................................................................................... 178
- Categories of predefined rule sets ............................................................ 179
  - Country or region identifier rule set ....................................................... 180
  - Domain preprocessor (PREP) rule sets ............................................... 182
  - Domain-specific rule sets .................................................................... 188
  - Validation rule sets ............................................................................. 192

© Copyright IBM Corp. 2004, 2013
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing rule sets</td>
<td>201</td>
</tr>
<tr>
<td>Accessing rule sets</td>
<td>202</td>
</tr>
<tr>
<td>Creating rule sets</td>
<td>205</td>
</tr>
<tr>
<td>Developing rule sets</td>
<td>205</td>
</tr>
<tr>
<td>Copying rule sets</td>
<td>209</td>
</tr>
<tr>
<td>Provisioning rule sets</td>
<td>209</td>
</tr>
<tr>
<td>Modifying rule sets</td>
<td>209</td>
</tr>
<tr>
<td>Standardization rule set tester</td>
<td>235</td>
</tr>
<tr>
<td>Selecting a rule set editor</td>
<td>236</td>
</tr>
<tr>
<td>Reference: Rule sets</td>
<td>237</td>
</tr>
<tr>
<td>Predefined rule sets</td>
<td>237</td>
</tr>
<tr>
<td>Characteristics of Japanese rule sets</td>
<td>243</td>
</tr>
<tr>
<td>Indian address rule sets</td>
<td>245</td>
</tr>
<tr>
<td>ISO territory codes</td>
<td>247</td>
</tr>
<tr>
<td>Chapter 7. Reference: Stage Editor</td>
<td>255</td>
</tr>
<tr>
<td>user interface</td>
<td></td>
</tr>
<tr>
<td>The stage page</td>
<td>259</td>
</tr>
<tr>
<td>General tab</td>
<td>259</td>
</tr>
<tr>
<td>Properties tab</td>
<td>259</td>
</tr>
<tr>
<td>Advanced tab</td>
<td>262</td>
</tr>
<tr>
<td>Link ordering tab</td>
<td>263</td>
</tr>
<tr>
<td>NLS map tab</td>
<td>264</td>
</tr>
<tr>
<td>NLS locale tab</td>
<td>264</td>
</tr>
<tr>
<td>Input page</td>
<td>264</td>
</tr>
<tr>
<td>Input General tab</td>
<td>265</td>
</tr>
<tr>
<td>Input Properties tab</td>
<td>265</td>
</tr>
<tr>
<td>Input Partitioning tab</td>
<td>265</td>
</tr>
<tr>
<td>Input tab Format tab</td>
<td>267</td>
</tr>
<tr>
<td>Input page: Columns tab</td>
<td>277</td>
</tr>
<tr>
<td>Advanced tab</td>
<td>290</td>
</tr>
<tr>
<td>Output page</td>
<td>291</td>
</tr>
<tr>
<td>General tab</td>
<td>292</td>
</tr>
<tr>
<td>Properties tab</td>
<td>292</td>
</tr>
<tr>
<td>Format tab</td>
<td>292</td>
</tr>
<tr>
<td>Output page columns tab</td>
<td>300</td>
</tr>
<tr>
<td>Mapping tab</td>
<td>301</td>
</tr>
<tr>
<td>Advanced tab</td>
<td>303</td>
</tr>
<tr>
<td>Appendix A. Product accessibility</td>
<td>305</td>
</tr>
<tr>
<td>Appendix B. Contacting IBM</td>
<td>307</td>
</tr>
<tr>
<td>Appendix C. Accessing and providing feedback on the product</td>
<td>309</td>
</tr>
<tr>
<td>documentation</td>
<td></td>
</tr>
<tr>
<td>Notices and trademarks</td>
<td>311</td>
</tr>
<tr>
<td>Index</td>
<td>317</td>
</tr>
</tbody>
</table>
Chapter 1. Introduction to data cleansing

IBM® InfoSphere® QualityStage® provides a methodology and development environment for cleansing and improving data quality for any domain.

Your organization’s data contains valuable information that your organization needs in order to conduct business, whether it is managing customers and products, managing operations, evaluating corporate performance, or providing business intelligence. InfoSphere QualityStage helps you deliver and maintain data quality so that your organization can rely upon its corporate data investment.

Data is high quality when it is up-to-date, complete, accurate, and easy to use. Depending on your organizational goals, high quality data can mean any of the following items:

- Your customer records do not include duplicate records for the same person.
- Your inventory records do not include duplicates for the same materials.
- Your vendor records do not include vendors you no longer use or suppliers no longer in business.
- You can be confident that Paul Allen and Allen Paul are records for two different customers, not the result of a data entry mistake.
- Your employees can find the data they need when they need it. Confident that they are working with high quality data, they do not need to create their own individual version of a database.

Whether your organization is transitioning from one or more information systems to another, upgrading its organization and its processes, or integrating and leveraging information across the enterprise, your goal is to determine the requirements and structure of the data that will address the organizational goal. Data that is restructured to conform to these new requirements is called cleansed data (sometimes referred to generally as data reengineering).

InfoSphere QualityStage methodology

Knowledge of the overall workflow helps you streamline your data cleansing implementation.

Creating cleansed data is a four-phase and iterative approach as shown in the following diagram:

Phase one
Understand organizational goals and how they determine your requirements

Phase two
Understand and analyze the nature and content of the source data

Phase three
Design and develop the jobs that cleanse the data

Phase four
Evaluate the results
Phase one: Understand goals and requirements

Phase one of the data cleansing workflow is to understand your organizational goals and requirements.

Phase one of the workflow helps you:

- Translate high-level mission directives into specific data cleansing assignments
- Make assumptions about the requirements and structure of the cleansed data

Not every organization has the same needs and objectives for data quality. Before you start designing your data cleansing project, you should understand the organizational goals that are driving the data-cleansing need and how they define your data cleansing assignment (the effective goal). This insight helps you gain a sense of the complexity of the intended cleansed data and provides a context in which you can make decisions throughout the workflow.

The success of a data cleansing project benefits from well-defined requirements for the output data results. As a best practice, provide opportunities throughout the every phase for domain experts and knowledge holders, who understand the organizational requirements of the data, to review the output results, to help iteratively refine requirements, and ultimately to approve the results. This collaborative process helps you meet the organizational requirements, increasing your chances for successful quality results.

Phase two: Analyze source data

Phase two of the data cleansing workflow is to learn about your source data, prepare your source data, and understand the quality of the source data.

Phase two of the workflow helps you:

- Identify whether the source data has the basic structure that your target data requires
- Understand the content of the source data
- Create the input data used in the next phase

Phase two helps you begin understanding the size and complexity of the project for creating cleansed data. If the granularity and structure of the source data closely matches your initial impression of the structure and requirements of the target data, then data cleansing will be less complex. The degree of difference contributes to your project complexity.

Most organizations think they know what data they have. But if you analyzed your data to determine how complete it is, how much of the information is duplicated, and what types of anomalies exist within each data field, you might be
surprised. Over time, data integrity weakens. The contents of fields stray from their original intent. The label might say Name, but the field might also contain a title, a tax ID number, or a status, such as Deceased. This information is useful, but not if you cannot locate it.

**Step one: Prepare for data cleansing**

Preparing for working in IBM InfoSphere QualityStage entails:
- Having general knowledge about the information in the source data
- Knowing the format of the source data
- Developing business rules for use iteratively throughout the data cleansing process, which are based on the data structure and content

**Step two: Investigate the source data**

Investigating helps you understand the quality of the source data and clarify the direction of succeeding phases of the workflow. In addition, it indicates the degree of processing you will need to create the cleansed data.

By investigating data, you gain these benefits:
- Gain a better understanding of the quality of the data
- Identify problem areas, such as blanks, errors, or formatting issues
- Prove or disprove any assumptions you might have about the data
- Learn enough about the data to help you establish business rules at the data level

InfoSphere QualityStage provides stages and reports to help you perform one or more of the following functions on your data:
- Organizing
- Parsing
- Classifying
- Analyzing patterns

This process produces input data for phase three, where you build your data cleansing jobs.

**Phase three: Design and develop jobs**

Phase three is to design jobs or sequences of jobs that generate the cleansed data you need. You then run these jobs on the data that was produced in the previous phase.

Designing the components that are required to build data quality jobs with InfoSphere QualityStage involves one or more of the following steps.

**Step one: Standardizing data**

Standardizing data involves preparing and conditioning data. InfoSphere QualityStage provides several features to assist you in standardizing data, such as stages and reports, to help you perform one or more of the following functions on your data:
- Implement enterprise or industry data-quality standards
- Improve addressability of data that is stored in a free form
- Prepare data for all of its uses (display, matching, reporting)
- Parses free-form or fixed-format columns into single-domain data elements to create a consistent representation of the input data
• Makes each data element have the same content and format
• Normalizes data values to standard forms
• Standardizes spelling formats and abbreviations
• Prepares data elements for more effective matching
• Performs phonetic coding (NYSIIS and SOUNDEX), which can be used in the match processing
• Review data results and statistics (Standardization Quality Assessment [SQA] reports)

Step two: Matching data
After the data is standardized, you are ready for matching. You match data to identify either duplicates or cross-references to other files. Your data cleansing assignment determines your matching strategy. After you know what you are looking for, whether it is to match individuals, match companies, perform householding, or reconcile inventory transactions, you can design a matching strategy to meet these goals.

Matching identifies all records in one source (the input source) that correspond to similar records (such as a person, household, address, and event) in another source (the reference source). Matching also identifies duplicate records in one source and builds relationships between records in multiple sources. Relationships are defined by business rules at the data level.

InfoSphere QualityStage provides Match stages (and a Match Designer that provides a test environment to produce match specifications for the Match stages) to help you perform one or more of the following functions on your data:
• Find similar and duplicate data
• Consolidate views
• Cross reference data to other sources
• Enrich existing data with new attributes from external sources

Step three: Identifying surviving data
After the data matching is complete, you identify which records (or columns of a set of duplicate records) from the match data survive and become available for formatting, loading, or reporting.

Survivorship facilitates that the best available data survives and is correctly prepared for the target destination. Thus, survivorship consolidates duplicate records, creating a best-of-breed representation of the matched data, enabling organizations to cross-populate all data sources with the best available data.

In this step, when you have duplicate records, you must make these decisions:
• To keep all the duplicates
• To keep only one record that contains all the information that is in the duplicates

InfoSphere QualityStage provides survivorship to help you perform one or more of the following functions on your data:
• Resolve conflicts with records that pertain to one entity
• Optionally create a cross-reference table to link all surviving records to the legacy source
- Supply missing values in one record with values from other records on the same entity
- Resolve conflicting data values on an entity according to your business rules
- Enrich existing data with data from external sources
- Customizes the output to meet specific organizational and technical requirements

### Phase four: Evaluate results

Phase four of the data cleansing workflow is to evaluate the results of the previous phases and to identify any organizational process improvements. The success of a cleansing project comes from iterative reviews and refinements throughout each phase.

In this phase of the workflow, you look at the results of the process and determine whether you need to perform any of these activities:
- Revisit a previous phase
- Refine some of the conditions
- Repeat the process, starting from the phase you revisit

If your data quality goals are simple, you might be satisfied with the results of the cleansing iteration. Otherwise, you might have to repeat this workflow, making different decisions and further refinements with each iteration.

Although evaluating your results is the final step of the data cleansing process, for a well-designed and developed job or a sequence of jobs, you are evaluating each step of the way. The results of each step might affect the direction you take in the following step. As a best practice, you evaluate the results of each phase as you complete it. The results of one step might impact the direction that you take in the next step. This process fine tunes a job and its stage components to achieve the highest quality data.

At the end of the design and development of your jobs, you can evaluate the entire process. What did you learn about your data? What did you learn about your cleansing process or about your data collection process? Your evaluation might help you make changes to your next data cleansing project, help you apply the jobs, or help your organization make changes to its business rules or even to its organizational goals.

Evaluating the results of your cleansing process can help your organization maintain data management and ensure that corporate data supports the organizational goals.
Chapter 2. Analyzing source data

You use the Investigate stage to analyze the quality of the source data. The Investigate stage helps you determine the business rules that you can use in designing your data cleansing project.

The Investigate stage indicates the degree of processing needed to create the target cleansed data. Investigating data identifies errors and validates the contents of fields in a data file. This investigation lets you identify and correct data problems before they infect new systems.

The Investigate stage analyzes data by determining the number and frequency of unique values, and classifying or assigning a business meaning to each occurrence of a value within a column. The Investigate stage has the following capabilities:

- Assesses the content of the source data. This stage organizes, parses, classifies, and analyzes patterns in the source data. It operates on both single-domain data columns as well as free-form text columns such as address columns.
- Accepts a single input link from any database connector supported by InfoSphere DataStage®, a flat file or data set, or from any processing stage. It is not necessary to restrict the data to fixed-length columns, but all input data must be alphanumeric.
- Produces output for one or two output links, depending on whether you are preparing information for one or two reports. Character investigations produce information for a column frequency report and word investigations produce information for both pattern and token reports. The Investigate stage performs a single investigation.

The Investigation reports, which you can generate from the IBM InfoSphere Information Server Web console by using data processed in the investigation job, can help you evaluate your data and develop better business practices.

Character Investigate option

You can examine single-domain columns by using the character investigation options in an Investigate stage to analyze and classify data. You can investigate multiple columns individually (Character Discrete Investigate) or integrated as one unit of data (Character Concatenate Investigate).

Investigation of columns provides information that you can use in a later phase of the data cleansing process. For example, you can use character investigation to verify the reliability of columns that you might specify as reference columns in match criteria.

The investigation process generates output that includes information about the frequency of values in a specific column.

Character Discrete Investigate

The Character Discrete Investigate option analyzes multiple single-domain columns. You can use this option to investigate a large number of columns with little effort.
The Character Discrete Investigate option produces output that treats each column as a separate value (token) for frequency count and pattern analysis.

**Character Concatenate Investigate**

The Character Concatenate Investigate option performs cross-column correlations between multiple columns to determine relationships. With this option, you select two or more columns from anywhere in the record (the columns do not have to be contiguous) to be investigated as a single data column.

To create the pattern analysis, the values are concatenated with no spaces between the values.

**Word Investigate option**

The Word Investigate option parses free-form data columns into individual values (previously called tokens) that are analyzed to create patterns.

To create the patterns, the Word Investigate option uses rule sets that are provided with InfoSphere QualityStage or rule sets that you created or customized. You can specify how you want values to be evaluated by the investigation process by selecting a rule set that meets your requirements. The Word Investigate option uses the specified rule set to determine how to identify values, assign class types to those values, and create patterns.

The Word Investigate option also provides frequency counts on the values. The output that the Word Investigate option generates is used by two reports, frequency pattern and frequency word, which can be viewed in the web console.

**Word investigation and rule sets**

When you specify the Word Investigate option in an Investigate stage, you select the rule set by which you want your columns investigated. You then select one or more columns to examine.

The Word Investigate option uses rule sets that are provided with InfoSphere QualityStage or rule sets that you created or customized on input data such as names and postal addresses. Examples of rule sets that are used in the Investigate stage are domain-specific rule sets such as the USNAME, USADDR and USAREA rule sets.

The Word Investigate option parses the free-form data column into individual elements or values (previously called tokens). A value is a word, number, or mixture that is separated by one or more spaces or special characters. The process compares each value with values that are classified in the classifications table for that rule set.

If the value matches the word in the classifications table, the investigation process assigns the value to the class that is specified in the classifications table. Classes are represented by one-character labels that are used to form patterns. For values that do not match any values in the classification table, the investigation process examines the pattern and assigns default classes. Pattern classes are described in detail in the *IBM InfoSphere QualityStage Pattern-Action Reference*.
Setting up to generate investigation reports

To generate investigation reports requires that you set up and configure the Investigate stage job.

Before you begin

Before you set up an Investigate stage job, prepare and specify the source data files.

Procedure

1. Set up the Investigate stage job on the Designer client canvas.
2. Configure the Investigate stage job by determining one of the following types of investigations to perform:
   • Character investigation, which is used on single-domain columns.
   • Word investigation, which is used on free-form columns.
   The type of investigation you choose also determines the type of report you generate.
3. Specify the columns you want to investigate.
4. Complete the action that corresponds to the type of investigation that you chose:
   • For a character investigation, select the type of investigation and prepare masks for each column.
   • For a word investigation, select the rule set to use in classifying patterns or words.
5. Set Stage properties.
6. Choose either a Sequential File stage or an ODBC Enterprise stage for the output of the Investigate stage. Use the ODBC Enterprise stage if you want to output the report columns to a database table. These stages are required as output stages only when you intend to generate reports from the Web console. If you do not intend to generate reports, you can use any output stage.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Action</th>
</tr>
</thead>
</table>
| If you choose Sequential File stage | 1. From the Palette, click File, drag the Sequential File stage onto the canvas, and link it to the Investigate stage.  
2. Double-click the Investigate stage.  
3. In the Output > Mapping tab, map the columns from the Investigate stage to the output columns of the Sequential File stage and click OK. Ensure that you map all the output columns and preserve the order in which they are listed.  
4. Double-click the Sequential File stage.  
5. In the Input > Properties tab, set the following property:  
   • In the File field, enter the file path for your output data.  
6. In the Input > Format tab, set the following properties and click OK:  
   • In the Delimiter field, select Comma.  
   • In the Quote character field, select Double.  
7. Save, compile, and run the job. |
**Setting up an Investigate job**

You organize the stages on the Designer client canvas with an input source data file, one or two output files, and one or more Investigate stages linked together.

**About this task**

When you set up the Investigate job, you can add various stages and files. The output of the Investigate stage is to a file. With this example of setting up an Investigate job, you are adding the following icons to the canvas:

- Source data file
- Three target files
- Two Investigate stages
- DataStage Copy stage

**Procedure**

1. Create a new parallel job or open an existing parallel job.
2. If the Palette is not displayed, click View > Palette to display it.
3. From the Palette, click Data Quality > Investigate stage icon.
4. Grab the icon and drop it toward the middle of the canvas.
   
   You can add more than one Investigate stage if you add a Copy stage (see step 6) to duplicate the data going to each Investigate stage.
5. To set up the input and output stages, perform the following steps:
   a. On the Palette, click File > SequentialFile icon.
   b. Grab the Sequential file and drop it to the left of the Investigate stage on the canvas.
      
      This file becomes the data source file.
   c. On the Palette, click File > Data Set and drop it to the right of the Investigate stage on the canvas.
This file becomes the output file.

d. For a Word Investigation, add a second Data Set file to the right of the Investigate stage and beneath the output file.

This file becomes another output file for a second report. Use two output stages only if you are doing a Word investigation and creating both pattern and token reports.

Generally your source data comes from a file or database, but you can also use other stages to preprocess it before inputting it to the Investigate stage.

6. If you want to add a Copy stage, click Palette > Processing to locate a Copy stage.

7. Grab the Copy stage and drop it in front of the Investigate stages to copy data to various outputs.

8. To add links to the stages, perform the following steps:
   a. Right click the source file and drag from the source file to the first stage on the canvas (Copy or Investigate).
      If the link appears red, you need to extend the line until it touches each stage you are linking.
   b. In the same manner, connect all the stages that you placed on the canvas.

9. Rename the stages and links by following these steps:
   a. Click on the stage default name. A box surrounds the name.
   b. Type a significant name in the box. Do not use spaces.
   c. Click outside the box to set the new name.

**What to do next**

You need to configure the stages to set up the columns that you wish to investigate.

**Configuring the source data file**

In this example, you configure the Sequential file as the source data file.

**Procedure**

1. Double-click the input **Sequential file** to open the Properties page.
2. Click **Source > File** to activate the **File** field.

3. Click > in the File field and select **Browse for File** from the menu.
4. Locate the directory where you installed the source data.
5. Click the file to select it. The file name appears in the **File** field.
6. Click **View Data**.
7. Click the file to select it. This button runs the Data Browser to show you the data within the source data file. The names of the columns are displayed across the top of the Data Browser.
8. Next, click **Column**.
9. Click **Load** and the Table Definitions window opens. You load the column data into the source file.
10. Click **Table Definitions** and locate the directory where the table definitions are stored.
11. Click **OK** to close the Columns page.
Configuring the Investigate stage
The Investigate stage takes the data from the source file and analyzes each record column-by-column.

Before you begin
Set up an investigate job before you configure the stage.

About this task
When you configure the Investigate stage, you determine whether to use a Character investigation or a Word investigation.

Procedure
1. Double-click the Investigate stage.
   The Character Discrete Investigate page opens as the default. The available columns from the input link are shown in the Available Data Columns pane.
   From this window, you can configure one of the following pages:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Setting stage properties” on page 17</td>
<td>Stage properties prepares the framework for cleansing the data using the Investigate stage.</td>
</tr>
<tr>
<td>“Setting up a character investigation”</td>
<td>A Character investigation explores a single-domain column (contains one data element such as a pension identity number, telephone number, date, or postal code). The result of a Character investigation provides a frequency distribution and pattern analysis of the values (previously called tokens).</td>
</tr>
<tr>
<td>“Setting up a word investigation” on page 14</td>
<td>A Word investigation parses free-form data columns into individual values. To create the patterns, Word investigation uses a set of rules for classifying personal names, business names, and addresses.</td>
</tr>
</tbody>
</table>

2. Click Stage Properties
   Before you run the Investigate stage, you need to set the Stage Properties.
3. Select an investigate option.
4. After configuring the Investigate stage, click Compile.
5. Click Run to process the job.

Setting up a character investigation
A character investigation analyzes multiple single-domain columns. For a concatenate character investigation, select columns from anywhere in a record to be investigated as a single data column.

About this task
The following procedure applies to both character concatenate and character discrete investigations.
Procedure
1. In the Investigate stage window, click Character Discrete Investigate or Character Concatenate Investigate.

2. Under Available Data Columns, select a column to investigate.

3. Click Add To Selected Columns.
   The column appears in the Scheduled Process box and the Mask Column Selection window opens. By default, all characters at each position in the column are set to T (type).

4. For every position in the column, adjust the mask according to one of the following options:
   • To set all characters in the column to display one investigation category, click the appropriate button (All T, All C, or All X).
   • To set the mask for individual characters in the column, click a particular character until the desired mask type appears.

   For more information about column masks, see "Column masks."

5. Click OK to close the Mask Column Selection window after you set the mask.

6. Repeat steps 2-5 for additional columns you want to investigate.
   • For a character discrete investigation, each selected column is investigated separately.
   • For a character concatenate investigation, all selected columns are concatenated, based on the chosen masks.

7. To change the number of samples or the frequency cutoff, click Advanced Options.
   The Investigate Advanced Options window opens.

8. In the window, select one of the following options:
   • Specify the number of samples of source data to include in the report for each pattern.
   • To skip patterns that appear infrequently, specify a frequency cutoff level. Patterns with frequencies under the specified level do not appear in the report.

9. Click OK.
   You can change these settings at any time.

10. Click OK to close the stage.

Change the mask for any given column:
Procedure
1. Select the column in the Scheduled Process pane.
2. Click Change Mask.
3. Edit the mask as described in step 4 in the previous procedure.

Remove a selected column from the investigation:
Procedure
1. Select the column in the Scheduled Process pane.
2. Click Delete. The column is returned to the Available Columns pane.

Column masks:

For character investigations, you use column masks to select the characters that are included in the frequency count or pattern analysis and the characters that are displayed as part of samples in the pattern report.
To use the Column Mask Selection window, apply a mask symbol to each character in the selected columns. You can use the following mask characters:

- C. Displays the character and includes it in the frequency count and pattern analysis.
  Use the C column mask to inspect the values in your columns and to certify that false data does not appear in a column such as 99999 for a postal code or 111111111 for a national ID number.

- T. Displays the type of character and includes the character in the frequency count and pattern analysis.
  Use the T column mask when you want to inspect the type of data in a character position, for example, telephone numbers nnn-nnn-nnnn or (nnn)-nnn-nnnn.

- X. Skips the character, does not include it in the frequency count or the pattern analysis, but includes it in the sample data.
  Use the X column mask to include data from the column in the sample but not as a value (previously called token) or part of the value for investigation.
  For example, you set up an Investigate job to analyze the first two characters of a ZIP code (USA postal code) to determine the frequency distribution based on a state (each state is defined by the first two characters of the ZIP code). You would set the column mask for the ZIP code to CCXX. The pattern column of the pattern report displays only the first two characters. The frequency count would be based on the number of records in the file that start with the first two characters of the ZIP code. In the value column, you would see all five characters of the ZIP code in the sample.
  You can also use the X column mask with the Character Concatenate option to specify one or more columns to appear as part of the sample only. From the previous example, you could also select the state columns setting the column mask to X for all characters. The pattern report displays the frequency counts for the first two characters of the ZIP code and the full five characters of the ZIP code along with the state in the sample column.

### Setting up a word investigation

A word investigation parses free-form data columns into individual values (tokens), which are analyzed to create patterns. Word investigation uses a set of rules for classifying personal names, business names, and addresses.

#### Procedure

1. In the Investigate Stage window, click **Word Investigate**.
   The window displays the Word Investigate columns.

2. In the **Rule Set** field, browse to select a rule set for your investigation from
   **Repository > Standardization Rules > Country > Rule Sets**.

3. From Available Columns, select one or more columns, and then click to move them to the Standard Columns pane.
   To change the column selection, select one or more columns in the Standard Columns pane and click **Delete Column**.

4. To rearrange the order of the standard columns, select a column and click either **Move Down** or **Move Up**.

5. To change the investigation options, click **Advanced Options**. The Investigate Advanced Options window opens.
   If necessary, make the changes and click **OK**.
6. In the Output data set area, select one or both options:
   - Pattern Report. Outputs the columns of a frequency pattern report to one link.
   - Token Report. Outputs the columns of a frequency word report to one link.
     You can change which link each report goes to when you edit the stage properties. You cannot send columns for both reports to the same link.

7. Click OK when you have completed setting up your word investigation.

Results

You need to edit the stage properties before you compile and run the job.

Specify advanced options for word investigations:

When you configure the advanced options for a word investigation, you determine how the token report appears by the choices you make.

Before you begin

First, set up a word investigation.

About this task

You control which values (tokens) appear in the token report and how they appear. You can also specify record delimiters. You do this by choosing from the options in the Investigate Advanced Options window.

Procedure

1. Click Advanced Options. The Investigate Advanced Options window opens.
2. Select one or more of the following options for controlling how values appear in the report:
   - Treat Successive Unclassified Words As One Word. Strips out the spaces between unclassified words (concatenating them into one word).
   - For example, CHARLES DE GAULLE becomes CHARLESDEGAULLE. This option reduces the number of patterns in both the token and pattern reports.
   - Include Unclassified Numeric Tokens. Lists all number values.
   - For example, when investigating an address column, you probably do not want to see house and apartment numbers but you might want to see numbers if you are investigating part numbers.
   - Include Unclassified Alphas. Includes all word values that are not in the classifications. If you do not select this option, the report only includes values from the classifications.
   - Include Mixed Types and Punctuated Words. Includes values with leading or trailing numerics, such as 199TH and 42ND.
3. Select one of the following options for displaying values in the report:
   - Standard Abbreviation. The standardized representation of the value from the classifications. (This is the default choice.)
   - Original Spelling. The form as the value appears in the data file.
   - Corrected Spelling. Corrects any misspellings as long as the classifications have weights assigned to the values.
4. In the Other Options area, edit the lists to add or remove special characters:
   - Separator List. Lists all special characters that separate values.
• **Strip List.** Lists all special characters from the Separator List that are not to be values. For example, the pound sign (#) by default is not part of this list; therefore, APT#3A is three values: APT, #, and 3A.

  **Note:** The space special character is included in both lists. Click **Restore Defaults** to reset all values to their default values.

5. In the Statistical Options area, select the output of the pattern reports from one of the following:
   
   • **No. of Samples.** Specify the number of samples of source data you want to include in the report for each pattern. By default, InfoSphere QualityStage provides one sample for each unique pattern in the pattern report. You can increase the number of samples displayed for each unique value. For example, you might want to see four samples for each value.
   
   • **Frequency Cutoff.** If you do not want to report on patterns that appear very infrequently, specify a frequency cutoff level. Patterns with frequencies under the specified level do not appear in the report. The default setting of one means that all patterns are displayed. You might not want to see the low frequency patterns, the ones that appear only once or twice.

6. Click **OK** to approve your changes and close the window.

### Configuring the target file

The data is written to a target file when you run a job.

**Before you begin**

Before you configure the target file, set up the job.

**About this task**

If you do not intend to generate investigation reports in the Web console, you can use any output stage for the output of the Investigate stage. If you intend to generate reports, use a Sequential File stage or an ODBC Enterprise stage for the output. For more information about setting up to generate investigation reports, see the *IBM InfoSphere QualityStage User's Guide*.

**Procedure**

1. Double-click the target file on the Designer canvas.
2. Click **Target > File.**
3. Specify the file where you want to write the data produced by running the job.
4. Click **OK.**

### About stage properties

Setting stage properties for the Investigate stage prepares the framework for cleansing data.

When you set stage properties, you can gain access to the following information:

• The input columns available for investigation
• The columns output with each report
• The link to which each report is written
• Parallel processing options
• Buffering options
The Stage editor has the following tabs:
- Stage Page. Lets you specify general information about the stage.
- Input Page. Lets you specify details about the data being input for investigation.
- Output Page. Lets you specify details about the report columns being output.

In addition, each Page has tabs that are tailored to that stage.

The Stage page includes the following tabs:
- General. Provides a text box in which you can enter an optional description of the stage. This is valuable information for others who might need to understand your job or its metadata.
- Properties. Specifies how the stage operates.
- Advanced. Specifies how the stage executes.
- Link Ordering. Changes which output links get a certain report. Used in the Word Investigate when you want to produce pattern and token reports. Select an output link name and click the up or down arrow.

### Setting stage properties

You set up the properties for the Investigate stage.

**About this task**

It is best to configure your input columns before setting up word or character investigations. If you set up a character or word investigation but then change the available columns on the Input page, you might need to set up the investigation again.

**Procedure**

1. In the Investigate Stage window, click **Stage Properties**.
2. To configure the Stage Properties page, follow these steps:
   a. Click **General** to add a description of the Investigate stage.
   b. Click **Properties** to add optional properties.
   c. Click **Advanced** to set the following actions:
      - Execution mode. Specify **Parallel** or **Sequential**.
      - Compatibility mode. Specify **Auto**, **Combinability**, or **Don't combine**.
      - Preserve partitioning. Specify **Propagate**, **Clear**, or **Set**.
   d. Click **Link Ordering** to set which output links get a certain report.
3. To configure the Input page, follow these steps:
   a. Click **General** to add a description of the stage.
   b. Click **Partitioning** to specify details about how the incoming data is partitioned or collected before it is acted on.
   c. Click **Columns** to view the metadata from the source data file.
   d. Click **Advanced** to view how InfoSphere DataStage buffers data being input to or output from this stage.
      - By default, InfoSphere DataStage buffers data in such a way that no deadlocks can arise. A deadlock situation occurs when stages are mutually dependent and are waiting input from another stage and cannot output until it is received.
4. To configure the Output stage, follow these steps.
Configure the Mapping tab

Specifies the columns that comprise an output report.

About this task

You do not need to load column definitions into the output stage or stages of your Investigate job in advance. Instead, use the Mapping tab to specify which report columns are output.

When you set up a character investigation or a frequency pattern report for a word investigation, the Mapping tab automatically displays all the columns for the column frequency report or the frequency pattern report.

- If you choose to create a frequency word report for a word investigation, three columns prepared for the frequency word report appear in the Columns list.
- If you choose to create both a frequency pattern report and a frequency word report for a word investigation, the pattern report columns appear in the Mapping tab when you select the appropriate output link in the Output name field. The token report columns then appear in the Mapping tab when you select the other link in the Output name field.

Procedure

1. Click the Mapping tab.
2. In the Output name field, select an output link name.
3. Arrange the columns in the order in which you want them to appear in the report as follows.
   a. Drag each column from the Columns list on the left to the Link pane on the right.
   b. Or, select multiple columns.
   c. Right-click and select Copy.
   d. From the Link pane, right-click and select Paste to move the columns to the Link pane.
4. If you want to output two reports from a Word investigation, select the other link from the Output name field and repeat step 3.

   Note: When you drag columns in this manner to create output table definitions, the table definitions are not automatically saved in the metadata repository. For best metadata management, on the Columns tab of the Output page, click Save and save the newly created table definition.

Results

Once you have configured the Mapping tab and set other desired properties on the Output page, you are ready to save, compile, and run your Investigate job.
Investigation reports

The data from the Investigate job creates investigation reports.

You access the Investigation reports from the IBM InfoSphere Information Server Web console. The Investigate stage produces the following graphical reports:

**Column Frequency Report**
- Presents information about frequency values by using statistical analysis.
- You create the data for this report by running a Character Discrete Investigate stage job or Character Concatenate Investigate stage job.

**Frequency Pattern Report**
- Presents pattern information for a specific column by using statistical analysis. You create the data for this report by running a Word Pattern Investigate stage job.

**Frequency Word Report**
- Presents the most commonly occurring word values from columns that are analyzed for patterns. You create the data for this report by running a Word Token Investigate stage job.

You can request a report or multiple reports to run on a specific schedule. You can customize each report by adding your company logo, a description of the report, the report name, and name of the originator of the report. The report is written to a specific file location in your choice of the following formats:
- HTML
- PDF
- RTF
- TXT
- XLS
- XML

**Report format for the Investigate stage**

The columns in the Investigate stage output data are interpreted by the report template and written by using the column format. The data for each report is output on a different link from the stage.

These types of reports are produced by the Investigate stage:
- The Frequency Pattern and Column Frequency reports are prepared for both word and character investigations. The reports have the following columns:
  - **Column Name**
    - The name of the input column in which the pattern is found for character discrete investigations or the names of all selected input columns for other investigations.
  - **Summary**
    - A graph that represents the count of values and the pattern.
  - **Pattern Value**
    - A generated pattern that describes the data.
  - **Count**
    - A frequency count indicating how many times this pattern was encountered across the entire input data.
  - **Percent**
    - The percentage of the input data that matches this pattern.
Cumulative %
The percentage of a portion of the input data that matches the pattern in
order to focus on the most prevalent patterns. For example: the first 80
patterns represents 60% of the total.

Sample
One or more examples of input data that match the pattern.

- The Frequency Word report is prepared only for word investigations. It analyzes
individual values (previously called tokens) or words that are found in the
selected input columns. The report has the following columns:

Word Token Summary
A graph that represents the count of values and the pattern.

Word Token
An individual value or word that is found inside the selected input
columns.

Count
A frequency count that indicates how many times this value is
encountered across the entire input data.

Classification status
The classification of the value that is based on the selected rule set
classification table. (Unclassified values, if selected, get a question mark
? for alpha, or a carat ^ for numeric.)

By default, InfoSphere QualityStage provides one sample for each unique pattern
in the Pattern Value column of the Pattern Frequency report. You can click
Advanced Options in the Investigate stage and choose a larger number of samples
to display for each unique pattern. For example, you may want to see four samples
for each pattern.

In both the Pattern Frequency and Word Frequency reports, you can limit the
displayed frequencies. If you do not want to see the low frequency patterns, the
ones that appear only once or twice, you can set a cutoff count for the frequency
within Advanced Options.

Creating investigation reports
You use investigation reports to evaluate the results of your Investigate job and to
help develop the next stage of your data cleansing process.

Before you begin
- Create an Investigate stage job to process the input data that is then written to a
target file.
- Set up the application server to run investigation reports. Refer to the IBM
InfoSphere Information Server Planning, Installation, and Configuration Guide for
more information about setting up application servers to run investigation
reports.

Procedure
1. Open the Web console and click Reporting.
2. From the Navigation pane, click Report Templates > QualityStage > View
   Report Templates.
3. Select any one of the reports.
4. From the right task pane, click Create New Report. The report template for
   the report you selected is displayed.
5. In the **Name** field, type a name for the report. The name is displayed in the generated reports as the Report Name.

6. In the **Description** field, enter a description for the report. The description is displayed in the generated reports as the Report Description.

   The **Creator** field contains the name that was entered in the Web console **User Name** field as the report originator.

7. Optional: Specify a folder location for the report in the **Save Within** field.

8. Enter the values in the Report Settings parameters.
   a. Select the InfoSphere QualityStage project from the list.
   b. Select your job from **Job Source**, and click **Retrieve Values** to populate the **Stage Name** field with related stages from the selected job.
   c. Select the stage from **Stage Name** for the job that is selected in step a, and click **Retrieve Values** to populate the **Select One or More Columns** field with related columns from the selected stage.
   d. Select columns from the **Select One or More Columns** field for the stage that is selected in step b.
   e. Select **Pattern** or **Count** from **Order by (Pattern/Count)**. This selection orders the report data by using pattern values or count of values.
   f. Select **Ascending** or **Descending** from **Sort By (Descending/Ascending)**. This selection defines the sorting order from step d. The report is sorted according to this value.
   g. Choose between 1 and 25 pattern values from **No Of Patterns in Graph**.
   h. Optional: Enter a comment or description in the **Customer Description** field.

9. Click **Format** to open the **Output Format** parameter and select the format for your report.

10. Click **Settings** to specify Expiration date and History policy.

11. Click **Finish**, then choose what you want to do with the report that you created from the list of options.

### Running and viewing the report

You select Reports in the Web Console Navigation pane to access the reports that you create.

**Before you begin**

First, you need to select the parameter settings for the report.

**Procedure**

1. From the Navigation pane, click **Reports > View Reports** and select the report that you want to run.
   
   If required parameters are missing, the Parameters pane opens. The missing parameter values are displayed in white.

2. From the task pane, click **Run Now**. The report is queued for running. When the report run starts, the run date and run duration appear in the Running pane.

3. To refresh the run statistics, click **Refresh**.

4. After the report processes, click **Reports > View Reports** in the Navigation pane.

5. From the list of reports, select a report.
6. From the task pane, click **View Report**. The report is displayed in the format that you selected when you created the report.
Chapter 3. Making data consistent through standardization

Standardizing data helps you make the source data internally consistent; that is, each data type has the same kind of content and format.

The Standardize stage builds on the interpretation of the data during the Investigate stage. The Standardize stage reformats data and creates a consistent data presentation with fixed and discrete columns, according to your company requirements.

The Standardize stage uses the data content and placement within the record context to determine the meaning of each data element. Common examples of data elements that can be identified are name, address, city, state, and postal code.

To correctly parse and identify each element or value (previously called token), and place them in the appropriate column in the output file, the Standardize stage uses rule sets that are designed to comply with standards or conventions. For example, you can standardize data names (individuals and businesses) and addresses to comply with the conventions of a specific country. The rule sets that are used by the Standardize stage can assimilate the data and append additional information from the input data, such as gender. These rule sets are the same as those used in the Investigate stage.

Standardized data is important for the following reasons:

- Effectively matches data
- Facilitates a consistent format for the output data

The Standardize stage parses free-form and fixed-format columns into single-domain columns to create a consistent representation of the input data.

- Free-form columns contain alphanumeric information of any length as long as it is less than or equal to the maximum column length defined for that column.
- Fixed-format columns contain only one specific type of information, such as only numeric, character, or alphanumeric information, and have a specific format.

The Standardize stage takes a single input, which can be a link from any database connector supported by InfoSphere DataStage, a flat file or data set, or any processing stage. It is not necessary to restrict the data to fixed-length columns.

The Standardize stage has only one output link. This link can send standardized output and the raw input to any other stage.

Source data preparation

As you plan your project, you need to prepare the source data to realize the best results.

IBM InfoSphere QualityStage accepts all basic data types (non-vector, non-aggregate) other than binary. Non-basic data types cannot be acted upon in InfoSphere QualityStage except for vectors in the match stages. However, non-basic data types can be passed through the InfoSphere DataStage and QualityStage stages.
You can use various processing stages to construct some columns before using the columns in a stage that you use for data cleansing. In particular, create overlay column definitions, vector columns, and concatenated columns as explicit columns in the data before you use them.

For example, you do not need to declare the first three characters of a five-character postal code column as a separate additional column. Instead, you can use a Transformer stage to add the column to the source data explicitly before using the column in a stage that you use for data cleansing.

**Note:** Be sure to map missing values to null.

Conform the actual data to be matched to the following practices:

- Make the codes used in columns the same for both data source and reference source.
  
  For example, if the Gender column in the data source uses M and F as gender codes, the corresponding column in the reference source should also use M and F as gender codes (not, for example, 1 or 0 as gender codes).

- Whatever missing value condition you use (for example, spaces or 99999) must be converted in advance to the null character. Conversion can be done using the InfoSphere DataStage Transformer stage. If you are extracting data from a database, make sure that nulls are not converted to spaces.

Use the Standardize stage to standardize data such as individual names or postal addresses. Complex conditions can be handled by creating new columns before matching begins.

### Standardization workflow

When you standardize data, the workflow depends on your data cleansing goals, data domain, and experience with pattern-action language.

You can prepare for the standardization process in the following ways:

- Ensure that you understand the data quality requirements for the domain. If a subject matter expert for the domain is not available, you might need to conduct research about the domain.

- Prepare a representative sample data set.

- Analyze the source data by running a job that includes the Investigate stage.

The following diagram shows a workflow for the standardization process.
Creating the Standardize stage job

The Standardize stage creates multiple columns that you can send, with the input columns, to the output link.

About this task

Any columns from the original input can be written to the output along with additional data created by the Standardize stage based on the input data (such as a SOUNDEX phonetic or NYSIIS codes). The Match stage and other stages can use the output from the Standardize stage. You can use any of the additional data for blocking and matching columns in Match stages.

Procedure

1. Specify the input data.
2. Decide which rule sets to use against which columns:
   - Select a COUNTRY rule set.
   - Select an existing InfoSphere QualityStage rule set or a newly created rule set.
3. Assign each selected rule set to a standardization process, and specify the order of these standardization processes as appropriate.
4. Configure the output columns in the Standardize stage to determine where the output is sent.
5. Save and compile the Standardize job.
6. Click Run to process the job.

**Rule sets that are used by the Standardize stage**

You can apply rule sets in the Standardize stage to create output columns that are consistent, meet industry standards, and that you can use in a variety of ways for data matching.

Rule sets check and normalize input data. The following categories of rule sets are available:

- Country or region identifier rule sets read area information and attempt to identify the associated country or region.
- Domain preprocessor rule sets evaluate mixed-domain input, such as free-form name and address information, and categorize the data into domain-specific column sets.
- Domain-specific rule sets process free-form data from a single domain such as name, address, or area information.
- Validation rule sets generate business intelligence and reporting fields, and are applied to common business data such as dates, email addresses, and phone numbers.

The rule sets that are provided with InfoSphere QualityStage are designed to provide optimum results. However, if the results are not satisfactory, or if you want to create rule sets for other data domains, you can create a new rule set, copy an existing rule set, or modify an existing rule set. You can modify rule set behavior by enhancing the rule set in the IBM InfoSphere QualityStage Standardization Rules Designer, adding user overrides, or editing the rule set files directly.

**Standardize processing flow for records for the USA**

The following diagram illustrates the Standardize stage processing flow using domain preprocessor and domain-specific rule sets to standardize the records commonly found in the United States.

Because input files are rarely domain-specific, domain preprocessor (PREP) rule sets are critical when preparing a file for standardization.
The same workflow is representative of other countries used with the Standardize stage.

**Input Source**
United States (U.S) Records

**Standardize Stage**
U.S. Preprocessor (PREP) Rule Set

**Standardize Stage**
Domain-specific rule sets:
- USNAME
- USADDR
- USAREA

**Output Records**
Include Name, Address, and Area data columns

**Using literals for required values**
Literals are symbols or abbreviations that are entered when the value is missing in the record. Literals can be either a prefix or suffix of a column.

If the input records do not include critical entries, you can insert the required values as a literal, which appears in the output. You insert the literal using the Standardize Rule Process window.

For example, the input records lack a state entry because all records are for the state of Vermont. To include the state in the standardized records, you would insert the literal `VT` between the city name and the postal code.

If input records have an apartment number column containing only an apartment number, you could insert a # (pound sign) literal between the unit type and the unit value.

Literals cannot contain any spaces and must be inserted between columns. You cannot include two contiguous literals for a rule set.

The only special characters that you can use in a literal are:
- `#` pound sign
- `%` percentage
- `^` caret
- `&` ampersand
- `<>` angle brackets
- `/` slash
For domain preprocessor rule sets, you must insert column delimiters using literals.

### Setting up the Standardize job

Organize the stages on the Designer client canvas with an input source data file, one output source, and a Standardize stage linked together.

### About this task

The output of the Standardize stage is one set of columns that include both the standardized columns and the input columns. You can choose to map any or all of these columns to the output link in the InfoSphere QualityStage job.

### Procedure

1. Create a new parallel job or open an existing parallel job.
2. If the Palette is not displayed, click View > Palette to display it.
3. From the Palette, click Data Quality and locate the Standardize stage icon.
4. Grab the icon and drop it near the middle of the Designer client canvas.
5. To set up the input and output stages, complete the following steps:
   a. Select the appropriate icon that represents the input and drop the icon to the left of the Standardize stage on the Designer canvas. The input can be any object that supports output links.
      The object becomes the source data file. Your source data comes from a file or database, but you can also use other stages to preprocess it before inputting it to the Standardize stage.
   b. Repeat step a and drop another icon, such as a Sequential file, to the right of the Standardize stage on the Designer client canvas.
      This file becomes the output or target file.
6. To add links to the stages, complete the following steps:
   a. Right click the source file and drag from the source file to the Standardize stage on the canvas.
      If the link appears red, you need to extend the line until it touches each stage you are linking.
   b. In the same manner, connect all the stages that you placed on the canvas.
7. Rename the stages and links by following these steps:
   a. Right-click on the stage and select Rename. A highlighted box appears around the default name.
   b. Type a significant name in the box. Do not use spaces.
   c. Click outside the box to set the new name.

### What to do next

You need to configure the Standardize stage to set up the columns and rules for processing the data.

### Configuring the Standardize stage

When you configure the Standardize stage, you apply a rule set to one or more columns.
**Before you begin**

Before you can configure the Standardize stage, first set up the job.

**About this task**

A Standardize stage implements one or more standardization processes. A standardization process is implemented by applying a rule set to one or more columns. The process may include the use of literals and names handling. A rule set is used only once in a Standardize stage.

You can add and modify standardize processes in the Standardize stage.

**Procedure**

1. Set up the source data file.
2. Double-click the **Standardize stage** icon to open the Standardize Stage window.
3. Click **New Process** to open the Standardize Rule Process window.
4. In **Rule Set** field, click ![ellipsis icon].
5. In the Open window, navigate to the **Standardization Rules > Rule Set**.
6. Locate the rule set you want and select it.
   The selected rule appears in the **Rule Set** field.
7. Follow the action for the rule set that you selected:
<table>
<thead>
<tr>
<th>Rule set</th>
<th>Action</th>
</tr>
</thead>
</table>
| NAME        | 1. Select any column name in the **Available Columns** field. The **Optional NAMES Handling** list becomes available.  
2. Select one of the following options:  
   • **Process All as Individual**. All columns are standardized as individual names.  
   • **Process All as Organization**. All columns are standardized as organization names.  
   • **Process Undefined as Individual**. All undefined (unhandled) columns are standardized as individual names.  
   • **Process Undefined as Organization**. All undefined (unhandled) columns are standardized as organization names.  
   NAMES handling options enhance performance by eliminating the processing steps of determining the type of name.  
   This option is useful when you know the type of name information that your input file contains. For example, if you know that your file contains only organization names, specify **Process All as Organization**. Every name entry in the file will be treated as an organization name. Even if you have an entry that is obvious to you as the name of an individual, it will be parsed as an organization.  
3. Click to move the selection into the **Selected Columns** list. The NAMES handling appears first in the Selected Columns list.  
4. Go to Step 8.                                                                                     |
| COUNTRY     | 1. In the **Literal** field, enter ZQXXZQ, where XX is the two-character country code. If the COUNTRY rule set cannot determine to which country or region input information belongs, the data is assigned to the country or region patterns specified in the Literal field, as a default delimiter.  
2. Click to move the selection into the **Selected Columns** list. The literal appears first in the Selected Columns list.  
3. Go to Step 8.                                                                                      |
Rule set | Action
--- | ---
PREP | 1. Use delimiters to describe the data in columns. In the **Literal** field, enter a delimiter value without spaces that corresponds to the column that you intend to move from the Available Columns list into the Selected Columns list. For example, you can use the `ZQADDRZQ` literal for an address column.
2. Click `>` to move the literal to the Selected Columns list. The literal precedes the column that it corresponds to.
3. Go to Step 8 and then repeat the previous step until each selected column is preceded by a literal.

ADDR, AREA, VDATE, VEMAIL, VTAXID | Go to Step 8.

8. From the Available Columns list, select one or more columns to which to apply the rule and then click `>` to move the selection to the Selected Columns list.

9. Optional: Adjust the order or selection of columns:
   a. Select a column in the Selected Columns list.
   b. Click **Move Up** or **Move Down**.

   **Note:** The order of the literal is important. The literal must come first when moving the order of the selected columns. For example, when processing address information, the address should appear in the same order that is expected by the mailing service of that country or region.

10. To remove a column from the Selected Columns list, select the column and click **Remove Column**.
11. When all desired columns for the rule set are listed, click **OK**.
    The window closes and the rule set and columns to be standardized are displayed in the Standardize Stage window.
12. To continue setting processes for the stage, you can:
   - Click **New Process** to add another rule.
   - Select a process and click **Modify Process** to change your choices.
   - Select a process and click **Delete Process** to delete it.
   - Select a process and click **Move Up** or **Move Down** to change the order in which the processes are executed.
13. Select a format from the **Default Standardized Output Format** list.
14. To close the Standardize Stage window at any time, click **OK**.

**Results**

Continue with Setting Stage Properties to map output link.
Predefined standardization jobs

You can use standardization jobs that are provided with IBM InfoSphere QualityStage to generate standardized data and the frequency information for that data. This data can be used as input for predefined matching jobs or for tests that you run in the Match Designer.

Predefined standardization jobs are in the following file:

\[ installation_directory\Clients\Samples\DataQuality\MatchTemplates\Jobs\PredefinedJobs.dsx \]

where \textit{installation_directory} is the root installation directory of the InfoSphere Information Server client tier.

You can import the jobs, and then view them in the Designer client repository tree by choosing \textbf{Data Quality > Match Templates > Jobs}.

The following table lists the predefined standardization jobs and the purpose for each job.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
Job name & Purpose \tabularnewline
\hline
MT10_StandardizeMatchDataInput & Create input for a one-source match and input for the MT20_CreateSQAReportInput_USNAME job. \tabularnewline
\hline
MT20_CreateSQAReportInput_USNAME & Generate data for a Standardization Quality Assessment (SQA) report that summarizes the results of a standardization job that uses the USNAME rule set. \tabularnewline
\hline
MR10_StandardizeMatchDataInput & Create data input for a two-source match. \tabularnewline
\hline
MR20_StandardizeMatchReferenceInput & Create reference input for a two-source match. \tabularnewline
\hline
\end{tabular}
\caption{Predefined standardization jobs}
\end{table}

Usage notes

You can run predefined standardization jobs as they are configured initially or you can use the jobs as models for creating jobs that meet your specific standardization requirements.

If you run the jobs as they are configured initially, you can use the sample input data that is provided.

Changes that you make to the predefined jobs overwrite the initial configurations for those jobs.

If you create a job that is based on one of the predefined jobs, ensure that you copy the predefined job and give it a unique name. If you use input data with columns that are different from the initial configuration, you must link each rule set in the Standardize stage to the columns in the input data.

The predefined standardization jobs use only US domain-specific rule sets.
Before you edit or run a predefined standardization job, read the annotation for that job.

**Standardize and correct international address data**

The Multinational Standardize stage (MNS) stage standardizes international address data.

You can configure the MNS stage to standardize street level and city level address data for any number of countries. The MNS stage uses the MNS rule sets.

**About the MNS stage**

The MNS stage uses country names, abbreviations, or ISO country codes in the input file to apply country-appropriate standardization rules. Thus, you can standardize all the records without having to consolidate records from each country into separate files and to standardize the records using country-specific rule sets.

For most countries, the MNS stage does the following tasks:
- Separates street-level address information from city-level information
- Assigns city, locality, province/state, postal code, and postal code add-on to separate fields
- Assigns ISO country codes (2 and 3 byte versions)
- Assigns city name phonetics with enhanced New York State Information and Intelligence Systems (NYSIIS) and reverse Soundex to match incorrectly spelled city names

**Countries and regions supported by the MNS rule sets**

The MNS rule sets are used in the MNS stage. The rule sets support a specific set of countries.

Countries and regions supported by the MNS rule sets are as follows:

<table>
<thead>
<tr>
<th>Country or Region</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUSTRALIA</td>
<td>AU</td>
</tr>
<tr>
<td>AUSTRIA</td>
<td>AT</td>
</tr>
<tr>
<td>BELGIUM</td>
<td>BE</td>
</tr>
<tr>
<td>BRAZIL</td>
<td>BR</td>
</tr>
<tr>
<td>CANADA</td>
<td>CA</td>
</tr>
<tr>
<td>DENMARK</td>
<td>DK</td>
</tr>
<tr>
<td>FINLAND</td>
<td>FI</td>
</tr>
<tr>
<td>FRANCE</td>
<td>FR</td>
</tr>
<tr>
<td>GERMANY</td>
<td>DE</td>
</tr>
<tr>
<td>GREAT BRITAIN</td>
<td>GB</td>
</tr>
<tr>
<td>ITALY</td>
<td>IT</td>
</tr>
<tr>
<td>LUXEMBOURG</td>
<td>LU</td>
</tr>
<tr>
<td>MEXICO</td>
<td>MX</td>
</tr>
<tr>
<td>NETHERLANDS</td>
<td>NL</td>
</tr>
</tbody>
</table>
Configuring the MNS stage

With the MNS stage, you can define the input file configuration as you build the job. Use the configuration that most closely matches your data source.

Before you begin

The MNS stage processes only address and postal code information. So you must remove the extraneous data from the file before you run the job.

Before you configure the MNS stage, add a preprocessor stage such as the Standardize stage to the Designer client canvas to remove data unrelated to address specifics from the address columns. For example, “in care of,” “contact,” or “attention to” information is acceptable as part of the address but notes such as ‘undeliverable” “requires signature” ”secure entry” are not acceptable.

Also, the MNS stage works best when the input data contains a country code. You can use the COUNTRY rule set in a Standardize stage to assign country codes.

About this task

When you configure the input data for the MNS stage, organize the input columns in one of the following ways:

- One combined street, city, state, and postal code column, and one country column
- One combined city, state, and postal code column, with one or more street columns and one country column
- One to five individual address columns that include city, state, street, postal code, and country information

Each record must have a country or territory indicator and can be in any of the following formats:

- Entire country or territory name
- An abbreviation (such as FR for France)
- ISO country or territory code

If the territory code does not match the expected country-level, territory-level, or street-level formats for the indicated territory, the data does not standardize, and the data is written to the output file unchanged. For example, if the record identifier is GB and the address format is that of France, the record does not standardize.
A record can contain contact information such as care of, attention, department, and so on. This information is standardized and placed in the contact information column of the output.

To configure an MNS stage:

**Procedure**

1. Configure the source file with the address, city, state or province, country or territory, and postal code. An example of how you can define input data by using a sequential file is as follows:
   a. Double-click the **Sequential File** icon to open the Properties page.
   b. Select the **Source > File** to activate the **File** field.
   c. Select **Browse for file** from the menu and locate the file where you installed the source data.
   d. Select the file. The file name is shown in the **File** field.
   e. Click **Column**.
   f. Click **Load** to open the Table Definitions window.
   g. Locate the source file metadata that you imported into the **Repository > Table Definitions** directory.
   h. Select the file to load the columns metadata into the source file.
   i. Click **OK** to close the Table Definitions window.
   j. Click **OK** to close the source file stage.

2. Double-click the **MNS stage** icon to open the MNS window.

3. Select one of the Input Entry Options based on the configuration of your input file.
   The fields that are activated on the lower right side of the stage window depend on the option that you select.

4. Select a column and move it into the appropriate field.

5. Click **Stage Properties > Output > Mapping**.

6. In the Columns pane, follow these steps:
   a. Right-click the pane and select **Select All** from the context menu.
   b. Right-click the pane and select **Copy** from the context menu.
   c. Right-click the Output pane and select **Paste Columns** from the context menu.

   The columns from the left pane are pasted into the output link pane with linking lines.

7. An example of how to configure the output to a target data set file is as follows:
   a. Double-click the target **Data Set** icon to open the Properties Stage window and click **Input > Properties**.
   b. Select the folder **Target > File** and select **Browse for file**.
   c. Specify the file where you want to write the output data.
   d. Click **OK** to close the Properties Stage window.

8. Click Compile.
   If the job compiles successfully, continue with the next step. If not, the Compile log specifies where the compile failed.

9. Click **Tools > Run Director** to open DataStage Director.

10. Click Run to run the job.
Standardization Quality Assessment (SQA) reports

To assess the results of your standardization process and verify that the results meet your data quality objectives, you can use a Standardization Quality Assessment (SQA) stage.

The SQA stage processes records that have been processed through the Standardize stage. The SQA stage gathers data about the standardization process which you use to create SQA reports. You create SQA reports in the IBM InfoSphere Information Server Web console.

The SQA reports provide an executive summary of the standardization results. You can send the reports to team members and stakeholders who do not have direct access to the standardization job results, or access to the IBM InfoSphere DataStage and QualityStage Designer (where you generate data for the reports).

Use the reports as part of an ongoing quality control of production processes, or as part of standardization rule set development or modification. For example a technical member of the data quality team configures a standardization job, produces output, and then needs to share the results with a subject matter expert. The subject matter expert has the domain knowledge needed to determine the success of the standardization results. The technical person can send SQA reports to the subject matter expert for feedback and signoff.

SQA reports are intended to be viewed in pairs. The high-level report and low-level report complement each other, although each type of report is generated separately:

**Standardization Quality Assessment (SQA) report**
Provides statistics about the standardization process, such as how much data remains unstandardized. You can produce this report as an HTML file, a PDF file, or an RTF file. For the most accurate rendering of graphics in the report, view the RTF file with Microsoft Word or Microsoft Word Viewer.

**Standardization Quality Assessment (SQA) Record Examples report**
Provides examples of processed records. You can produce this report only as a Microsoft Excel (.xls) file.

### Adding the SQA stage

To generate data for the Standardization Quality Assessment (SQA) reports, add the SQA stage to a job that processes output from the Standardize stage. Optionally, you can add the SQA stage directly to a Standardize stage job.

### Before you begin

Before you add the SQA stage, complete the following tasks:

- Create a Standardize stage job
- Set up the Standardize job
- Configure the Standardize stage job

### Procedure

1. Open the job that you want to add the SQA stage to. You can add the SQA stage to any job in which you have a stage that contains the output data from
your standardization job. You can link the Standardize stage directly to the SQA stage, but remember that the Standardize stage can support only one output link.

2. If the palette is not displayed, click View > Palette to display it.
3. From the palette, click Data Quality.
4. Drag the SQA stage on to the canvas.
5. Link the stage that contains output data from the standardization job to the SQA stage.

**Configuring the Standardize stage to generate SQA report columns**

After you add the SQA stage to a job, you must configure the Standardize stage to generate additional columns that are used to create SQA reports.

**Before you begin**

Before you configure the Standardize stage to generate SQA report columns, complete the following task:

- **Add the SQA stage to a job that processes the output of the standardization job**
  Optionally, you can add the SQA stage directly to a Standardize stage job.

**About this task**

The SQA stage requires additional output columns from the Standardize stage to generate data for the SQA reports. When you alter the Standardize stage to produce additional output columns, consider the implications that additional output columns have for your job. When the Standardize stage generates additional data for the SQA stage, a column named qsSqaStl_rule set name is added for each standardization process that is configured in the Standardize stage. This column contains the subset input data field that is processed by that particular rule set for that input record. For example, target databases that the Standardize stage writes to must be altered to account for the additional columns that are generated.

**Procedure**

1. Open the standardization job for which you want to create an SQA report.
2. Double-click the Standardize stage and click Stage Properties.
3. In the Stage > Properties tab, add the SQA Report Output property and verify that the property value is set to True.
4. Map the output columns of the Standardize stage to the input columns of the stage whose data will be used by the SQA stage. Remember that the SQA stage can directly receive output from the Standardize stage. In addition, the SQA stage can receive output from another stage (such as a Sequential File stage) which reads data from a file that contains the output of the standardization job.

  **Note:** You must preserve the order of the output columns when you map them.
5. If the SQA stage is not directly linked to the Standardize stage, ensure that you reconfigure the necessary stage or stages to accommodate the additional output columns from the Standardize stage.
6. Run the standardization job after you configure it to generate the additional columns for the SQA stage.
Configuring the SQA stage

When you configure the Standardization Quality Assessment (SQA) stage, you select a column from the output of the stage that contains the output of the standardization job. The SQA stage generates data from that column which the Web console uses to generate reports.

Before you begin

Before you configure the SQA stage, complete the following task:

- Add the SQA stage to a job
- Configure the Standardize stage

Procedure

1. Open the job that contains the SQA stage.
2. Double-click the SQA stage icon to open the stage editor.
3. In the Stage > Properties tab, select the Unhandled Data Column option and select the column from the Standardize stage which contains data that was not fully parsed. The format for the unhandled data column is as follows: UnhandledData_rule set. A Standardize stage can process data using more than one rule set. Each rule set has an unhandled data column. If you select any column other than an unhandled data column, you will receive a runtime error because you selected a noncompliant column.
4. Choose either a Sequential File stage or an ODBC Enterprise stage for the output of the SQA stage. Use the ODBC Enterprise stage if you want to output the report columns to a database table.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you choose Sequential File stage</td>
<td>1. From the Palette, click File, drag the Sequential File stage onto the canvas, and link it to the SQA stage.</td>
</tr>
<tr>
<td></td>
<td>2. Double-click the SQA stage.</td>
</tr>
<tr>
<td></td>
<td>3. In the Output &gt; Mapping tab, map the columns from the SQA stage to the output columns of the Sequential File stage and click OK. Ensure that you map all the output columns and preserve the order in which they are listed.</td>
</tr>
<tr>
<td></td>
<td>4. Double-click the Sequential File stage.</td>
</tr>
<tr>
<td></td>
<td>5. In the Input &gt; Properties tab, set the following properties:</td>
</tr>
<tr>
<td></td>
<td>• In the File field, enter the file path for your output data.</td>
</tr>
<tr>
<td></td>
<td>• If the first row of your output data is a header row, in the First Line is Column Names field, select True.</td>
</tr>
<tr>
<td></td>
<td>6. In the Input &gt; Format tab, set the following properties, and then click OK:</td>
</tr>
<tr>
<td></td>
<td>• In the Delimiter field, select Comma.</td>
</tr>
<tr>
<td></td>
<td>• In the Quote character field, select Double.</td>
</tr>
<tr>
<td></td>
<td>• In the Null field value field, enter &quot;.&quot; If the Null field value field is not shown, select Field defaults and then click Null field value from the list of properties.</td>
</tr>
<tr>
<td></td>
<td>7. Save, compile, and run the job.</td>
</tr>
</tbody>
</table>
Stage | Action
--- | ---
If you choose ODBC Enterprise stage
Note: To use the ODBC Enterprise stage, you must first set up a database table and the ODBC connection. Refer to the IBM InfoSphere Information Server Planning, Installation, and Configuration Guide for details on configuring access to ODBC data sources.
1. From the Palette, click Database, drag the ODBC Enterprise stage onto the canvas, and link it to the SQA stage.
2. Double-click the SQA stage.
3. In the Output > Mapping tab, map the columns from the SQA stage to the output columns of the ODBC Enterprise stage and click OK. Ensure that you map all the output columns and preserve the order in which they are listed.
4. Double-click the ODBC Enterprise stage.
5. In the Input > Properties tab, set the following properties and click OK:
   - In the Table field, enter the name of the database table to which you want to send the columns for the SQA reports.
   - In the Write method field, select Write.
   - In the Write mode field, select Replace.
   - In the Data source field, enter the name of the ODBC driver that was used when the ODBC connection was set up.
   - In the Truncate Column Names field, select False.
6. Save, compile, and run the job.

Creating SQA reports

After a standardization job is created, you can create a Standardization Quality Assessment (SQA) report to review example results from the Standardize stage.

Before you begin

Before you create the standardization reports, complete the following task:
- **Configure the SQA stage**
- Set up the application server. See the information about setting up application servers for SQA or investigation reports in the IBM InfoSphere Information Server Planning, Installation, and Configuration Guide.

Procedure

1. In the Web console, click the Reporting tab.
3. Select either Standardization Quality Assessment (SQA) or Standardization Quality Assessment (SQA) Record Examples.
5. Optional: In the Name field, replace the default name with a name that is more meaningful to you. For example, you might choose a name that includes the name of the standardization job.
6. Optional: In the Description field, replace the default description with a description that is more meaningful to you. For example, you might enter a summary of the standardization job.
   The Save-in Folder field contains the name of the default folder where the new report is stored on the server. You can save the report in the default Reports folder or click Browse to specify an existing folder.
7. Select items for the Report Settings parameters:
a. From the Project field, select the project that contains the SQA stage job that generated the data for the report that you want to create.

b. In the Job Source field, click Retrieve Values and select the SQA stage job.

c. In the SQA Stage Name field, click Retrieve Values and select the SQA stage.

d. In the Range of Composition Sets - Start and End fields, enter the numbers of the composition sets that you want to view in the report.

A composition set is a group of records for which the same dictionary columns are populated. For example, suppose that for a set of records, each record has nine fields. Two of those records contain values for the same five dictionary columns and the remaining four dictionary columns are null. Those two records are members of the same composition set.

The reports display the composition sets from the highest percentage of records to the lowest percentage of records that contain values for the same columns. For example, if you want to see only the sets with the second, third, fourth, and fifth highest percentages of records that contain values for the same columns, enter 2 in the Range of Composition Sets - Start field and 5 in the End field. If you enter these values, the report will contain four composition sets.

The default values are 1 and 20, which show the sets with the 20 highest percentages. The maximum number of sets that you can see is 50.

e. Optional: Select Or Display All Available Composition Sets. If you select this check box, every composition set will be displayed in the reports. Depending on your data, the higher number composition sets might contain low percentages of records that contain values for the same columns.

f. Optional: For the record examples report, replace the default value in the Number of Record Examples Per Composition Set (1-50). The default value is 10. This option is available only in the SQA Report Examples template.

g. Select a file format for the report.

h. Optional: Select settings for the Expiration and History Policy.

8. Click Finish to select one of the following options: Save, Run Now, Save and Run Now, Save and Close.

**Running SQA reports**

After a report is created, you run the report. You cannot view example results from the Standardize stage until you run the report.

**Before you begin**

Before you run the SQA reports, complete the following task:

- [Create SQA reports](#)

**Procedure**

1. In the Web console, select the Reporting tab.

2. From the Navigation pane, click Reports > View Reports and select the report that you want to run or view.

3. If the report has never been run before, or if you have changed the report settings, click Run Now. The report is queued for running. When the report run starts, the run date and run duration appear in the Running pane.

4. Optional: To refresh the run statistics, click Refresh.
**Viewing SQA reports**

After a report is run, you view the report. You can view a report immediately after you run it. In addition, you can view any report that was run previously.

**Before you begin**

Before you view the SQA reports, complete the following task:

- **Run an SQA report**

**Procedure**

1. In the Web console, select the **Reporting** tab.
2. From the Navigation pane, click **Reports**.
3. If the report that you want to view was saved in a custom folder, select the folder where your report is saved.
4. Select **View Reports** and select the report that you want to view.
5. Click **View Report Result**. The report opens in the format that you selected when you created the report.

**How to interpret the SQA reports**

When you review Standardization Quality Assessment (SQA) reports, look at the data from the perspective of what is and what is not processed by your job with the Standardize stage.

When you standardize your data, the following flow of actions occur:

1. Your job for the Standardize stage **processes** the data.
2. During the processing, the stage and its rule sets for that job determine how it **handles** the data. Data that your job cannot place in an appropriate column is called **unhandled**.
3. After your job finishes, you use the reports to verify that the rule sets for your job **standardized** the data correctly. The result of your job is standardized data, as defined by your business requirements.

The SQA report and the SQA Record Examples report help you make these analyses:

- Identify data that is not processed by your job (the unhandled data).
- Identify data that is processed by your job, but is not processed correctly.

**Unhandled data**

Data that is not processed by the job (unhandled). The reports show you the percentage of records that contain data that is not standardized. You can also view example records that contain values in the **UnhandledData** column, and all other columns as well.

Review the values in the **UnhandledData** column to decide what to do next:

- If the values are irrelevant to the columns that you want to standardize, you might conclude that the rule set requires no change.
- If you have a high percentage of records that are not standardized by the job, you must decide whether your rule set for the stage is to remain as-is, be modified, or be replaced.

From the SQA report, for example, in Set 2 in Figure 2 on page 42, you can see the percentage of records which contain a value that the job does not
standardize. Set 2 is the only set that contains records for which the **UnhandledData** column is populated with values that the job did not standardize.

<table>
<thead>
<tr>
<th>Composition Sets</th>
<th>Displayed sets comprise 100.00% of the processed records</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set 1</td>
</tr>
<tr>
<td>NameType</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
<tr>
<td>GenderCode</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
<tr>
<td>NamePrefix</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
<tr>
<td>FirstName</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
<tr>
<td>MiddleName</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
<tr>
<td>PrimaryName</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
<tr>
<td>NameGeneration</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
<tr>
<td>NameSuffix</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
<tr>
<td>AdditionalName</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
<tr>
<td>UnhandledData</td>
<td><img src="%E2%9C%93" alt="✓" /></td>
</tr>
</tbody>
</table>

*Figure 2. Example of an SQA report: composition set*

Perhaps you find, by looking at the SQA Record Examples report, that the source data was not properly prepared for standardization. You find that, unlike the other records sets, the record columns in Set 2 are not clearly or consistently defined. Knowing the state of the records in Set 2, you can be confident that the rule set used for standardization is not problematic; the structure of the source data might be the problem.

**Processed data**

Data that is standardized by the job. The rule set for the stage can parse the data, but you must determine whether the values in the dictionary columns make sense. Review the record examples in the reports. Look to see if the job placed values in columns where you do not expect them to be placed.

For example, you know that your records contain the names of many professionals that include a designation such as Ph.D., M.D., CEng., or LL.B. You expect to find record sets with values in the **NameSuffix** column. The data in Set 5 in Figure 2 does not meet your expectation.

By generating an SQA Record Example report, you can see the columns that are a result of standardization, at the record set level. You can see in Figure 3 that the name suffixes are not standardized in the way that you expected. The job affixed the name suffixes LL.B. and CEng. to the primary name and did not placed them in a **NameSuffix** column.

*Figure 3. Set 5 of a Record Examples report*

Through analysis, the reports enable you assess and verify the results that you expect in your standardized data. Analysis of the SQA report might prompt you to review specific records in the SQA Record Examples report. Features within the SQA reports can help you interpret the data.
Standardization Summary

In the SQA report, the first chart is a pie chart that provides a graphical summary of processing results. The results are divided between records that the job fully standardizes, and partially standardizes or does not standardize at all. You can compare the statistics against a goal that you set for the percentage of data that the standardization process handles. However, to ensure that the quality of processing results is high and meets your business requirements, you must still review actual records provided in the SQA Record Examples report.

Frequency of Records by Populated Dictionary Column

The second chart in the SQA report is a bar chart that shows the percentage of total records that contain a value in specific dictionary columns. Check the chart against your expectations. You might expect specific dictionary columns to be populated for every record, or close to 100% because you know that the data is in your records. Alternatively, you might expect a much lower percentage in other dictionary columns. For example, if you are standardizing name data you might expect all records to have a first and primary name but fewer to contain a middle name.

The bar chart also provides the number of unique values for a given dictionary column. If a name domain is likely to contain many unique first names, you might expect the dictionary column, FirstName, to have a high number of unique values. For other columns, such as Name Suffix (Ph.D, M.D., J.D., or LL.B.), you might expect a lower number of unique values because only a limited number of name suffix values exist.

Composition Sets

The third part of the SQA report is a data grid that shows how the job standardized data from several record sets into a set of dictionary columns. Refer to Figure 2 on page 42. The view of records against dictionary columns summarizes output results, but also can provide additional insight into the quality of your data and the standardization of your data.

For example, if 72% of your data records populate to the same dictionary columns, you can then review the corresponding SQA Record Examples report to determine whether the results are accurate. If so, you know that 72% of your data is being properly standardized.

You can also use the chart to find gaps in your data. If a moderate percentage of your records populate a column that reflects data that is important to your business, you might want to find out why the percentage is not as high as you expect and want it to be. The job might not be standardizing the records properly. Or perhaps the records require data enrichment.

Record Examples

The SQA Record Examples report shows you the results of standardization in the context of data records. Using this view, you can interpret the SQA report summary statistics in context.

Unhandled data in the reports - examples

When you analyze the SQA reports, you focus on records that the standardization job did not process (the unhandled data), to determine the success of the standardization process. A review of the data values that the standardization process does not handle can uncover potential issues.
Standardization Quality Assessment (SQA) report

The various sections of the Standardization Quality Assessment (SQA) report show different perspectives of your data. You can see percentages of records and record sets that contain data which is totally or partially unhandled by the job with the Standardize stage. The SQA reports include a summary, a frequency chart, and a composite view.

Standardization Summary

![Pie Chart](image)

- Fully standardized records
- Partially or nonstandardized records

The pie chart shows the percentage of records that were fully standardized by the job and the percentage of records that were either partially standardized or not standardized at all by the job. Nonstandardized records contain at least some data that the job could not process (unhandled data). In some cases, unhandled data is acceptable based on your data quality objectives. In other cases, unhandled data might be data that is fundamental to your business requirements.

In the SQA Record Examples report, look at the values in the UnhandledData column and determine whether you need to change the rule set for the stage so that the job standardizes the data that is currently unhandled by the job. If the percentage of unhandled data is low and the values in the Record Examples report are anomalies, you might decide that your current rule set is good enough and that no adjustments are necessary. However, the percentage of unhandled data might be greater than you think it should be. If you think that the values in the Record Examples report can be standardized by a job, consider modifying the rule set or using another rule set.

Frequency of Records by Populated Dictionary Column

UnhandledData (25)

The bar chart shows the number of unique values (shown in parentheses) for a given dictionary column. The chart also shows the percentage of total records that contain a value in that dictionary column. For example, a column and number such as UnhandledData (25) that shows 3.27% is interpreted as follows:

- The UnhandledData column contains 25 unique values
- 3.27% of the data records that the stage processed contain a value in that column.
Composition Sets

The data grid shows columns of sets that contain records for which the same dictionary columns are populated by the job. The percentage beneath the Set heading is the percentage of total records that fall within that set.

Find the row in the report that represents the UnhandledData column and find the first set that contains a value in the UnhandledData column. If your rule set works well, the first set that contains a value in the UnhandledData column is only a small percentage of your total records.

For example, Set 8 represents 0.94% of the total records that contain values in the UnhandledData column. Look at Sheet 8 in the SQA Record Examples report, which corresponds to Set 8. In Sheet 8, you find examples of values where the job populated the UnhandledData column. The examples can help you determine whether the rule set is handling those values correctly.

Standardization Quality Assessment (SQA) Record Examples report

The SQA Record Examples report contains sheets that correspond to composition sets as summarized in the SQA summary report. The report shows you the results of standardization in the context of data records.

Set 8 of a Record Examples report

<table>
<thead>
<tr>
<th>Compositions Sets</th>
<th>Displayed sets comprise 98.13% of the processed records</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set 1</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
</tr>
<tr>
<td>HouseNumber</td>
<td>✓</td>
</tr>
<tr>
<td>HouseNumberSuffix</td>
<td></td>
</tr>
<tr>
<td>StreetPrefixDirectional</td>
<td></td>
</tr>
<tr>
<td>StreetPrefixType</td>
<td>✓</td>
</tr>
<tr>
<td>StreetName</td>
<td>✓</td>
</tr>
<tr>
<td>StreetSuffixType</td>
<td></td>
</tr>
<tr>
<td>StreetSuffixQualifier</td>
<td></td>
</tr>
<tr>
<td>StreetSuffixDirectional</td>
<td></td>
</tr>
<tr>
<td>RuralRouteType</td>
<td></td>
</tr>
<tr>
<td>RuralRouteValue</td>
<td></td>
</tr>
<tr>
<td>BoxType</td>
<td>✓</td>
</tr>
<tr>
<td>BoxValue</td>
<td>✓</td>
</tr>
<tr>
<td>FloorType</td>
<td></td>
</tr>
<tr>
<td>FloorValue</td>
<td></td>
</tr>
<tr>
<td>UnitType</td>
<td></td>
</tr>
<tr>
<td>UnitValue</td>
<td></td>
</tr>
<tr>
<td>MultiUnitType</td>
<td></td>
</tr>
<tr>
<td>MultiUnitValue</td>
<td></td>
</tr>
<tr>
<td>BuildingName</td>
<td></td>
</tr>
<tr>
<td>AdditionalAddress</td>
<td>✓</td>
</tr>
<tr>
<td>AddressType</td>
<td>✓</td>
</tr>
<tr>
<td>StreetNameNYSIIS</td>
<td>✓</td>
</tr>
<tr>
<td>StreetNameRVSNDX</td>
<td>✓</td>
</tr>
<tr>
<td>UnhandledData</td>
<td>✓</td>
</tr>
<tr>
<td>InputPattern</td>
<td>✓</td>
</tr>
<tr>
<td>ExceptionData</td>
<td>✓</td>
</tr>
<tr>
<td>UserOverrideFlag</td>
<td>✓</td>
</tr>
</tbody>
</table>


Chapter 3. Standardizing data  45
The report shows the record examples that are contained by Set 8 of the processed records. The set numbers correspond to the sheet numbers in the Record Examples report. Compare Set 8 in the Record Examples report to the first page of composition sets in the SQA summary report.

Notice that 0.94% of the total records are contained by this set, which is the same percentage that the summary report shows you for the set. In the Set 8 column of the summary report, notice the check mark in the UnhandledData column. This check mark is your cue to review the data values in the Record Examples report.

In the UnhandledData column of the Record Examples report, try to find a pattern in the unhandled data values that might help you modify your rule set to standardize this data. If it makes sense that the job cannot standardize the data values and you think that your rule set is handling the data in the best way possible, your assessment might be that no modifications to the rule set are necessary because the standardization job is working as you expect.

**Processed data in the reports - examples**

When you review the SQA reports, decide if the values in the dictionary columns of the records make sense. If the values do not make sense, review the patterns of those values to decide how to modify the rule sets to get the results that you need.

**Standardization Quality Assessment (SQA) report**

The following sections of the SQA summary report identify data in your records processed by the Standardize stage:

**Standardization Summary**
Fully standardized records  Partially or nonstandardized records

The pie chart shows the percentage of records that were fully standardized by the job and the percentage of records that were either partially standardized or not standardized at all by the job. Fully standardized means that the job was able to place all the record values into dictionary columns other than the UnhandledData column.

In the Standardization Quality Assessment Record Examples report, look at the values in the fully standardized data columns to decide what to do next:

- If the values in the columns make sense, you might decide that your current rule set is good enough and that no adjustments are necessary.
- If the values in the columns are not consistent with your understanding of the data, adjust the rule set to better standardize the data.

Frequency of Records by Populated Dictionary Column

HouseNumber (528)

The bar chart shows the number of unique values (shown in parentheses) for a given dictionary column. Also shows the percentage of total records that contain a value in that dictionary column. For example, a column and number such as HouseNumber (528) that shows 78.39% means that the HouseNumber column contains 528 unique values and 78.39% of the processed data contain a value in that column.

Composition Sets
## Compositions Sets

<table>
<thead>
<tr>
<th>Displayed sets comprise 98.13% of the processed records</th>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
<th>Set 4</th>
<th>Set 5</th>
<th>Set 6</th>
<th>Set 7</th>
<th>Set 8</th>
<th>Set 9</th>
<th>Set 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>HouseNumber</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HouseNumberSuffix</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>StreetPrefixDirectional</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>StreetPrefixType</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>StreetName</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>StreetSuffixType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>StreetSuffixQualifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>StreetSuffixDirectional</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>RuralRouteType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>RuralRouteValue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>BoxType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>BoxValue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>FloorType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>FloorValue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>UnitType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>UnitValue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>MultiUnitType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>MultiUnitValue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>BuildingName</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>AdditionalAddress</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>AddressType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>StreetNameNYSiIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>StreetNameRVSNDX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>UnhandledData</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>InputPattern</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ExceptionData</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>UserOverrideFlag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

The data grid shows columns of sets that contain records for which the same dictionary columns are populated by the job. Additionally, the percentage shown beneath the set heading is the percent of total records that fall within that set.

For example, **Set 1** represents 72.03% of the total records whose values populate the same eight dictionary columns. This percentage does not mean that the values in each column are the same, only that the same columns in this particular set are populated by the job. The remaining sets continue in descending percentages of the total data.

The heading of the Composition Sets page shows you the percentage of the processed records that are represented by the composition sets in the report. The higher the set number, the smaller the percentage of total records in a set that the stage processed. For example, in Set 20 the percentage is typically less than 1%. Set 20 might contain only two records for which the same columns are populated by the job.

### Standardization Quality Assessment (SQA) Record Examples report

The SQA Record Examples report contains sheets that correspond to composition sets as summarized in the SQA summary report.

**Set 2 of a Record Examples report**
The report shows the record examples that are contained by Set 2 of the records processed by the stage. The set numbers correspond to the sheet numbers in the Record Examples report.

Compare Set 2 in the Record Examples report to the first page of composition sets in the SQA summary report. Notice that 11.69% of the total records are contained by this set, which is the same percentage that the summary report shows you for the set.

In this example, the fifth dictionary column in the summary report, **StreetName**, contains a check mark. In the Record Examples report, the first column of the report table contains the input records. The other columns of the report are the dictionary columns. **StreetName** is the first dictionary column for the input records in Set 2 of the Record Examples report. Each input record contains a value in the **StreetName** column. The remaining check marks in the Set 2 column of the summary report follow the same pattern as the **StreetName** column. The second check mark represents the second dictionary column in the Record Examples report, and so on, until the Record Examples report shows you the data values for the **UserOverrideFlag** column. The subsequent sheets correspond to the remaining sets in the summary report.

Use the Record Examples report to verify that the data values make sense for a particular column. If the values are not what you expect, modify the rule set to better handle the data.
Chapter 4. Matching data

Matching in IBM InfoSphere QualityStage is a probabilistic record linkage system that automates the process of identifying records that are likely to represent the same entity. The matching process improves the integrity of your data.

With matching, you can identify duplicates in your data, group records on any set of criteria, and build relationships among records in multiple files despite variations in the representation of the data and missing or inaccurate information.

Matching goals

Your cleansing assignment determines your matching strategy. When you know what you are looking for, you can design a matching strategy to meet your goals.

Some typical goals for matching include the following ones:
- Identifying duplicate records for entities such as individuals, companies, suppliers, products, or events
- Grouping records with the same or similar values, such as householding
- Enriching existing data with new attributes from external sources
- Locating entities in a data warehouse
- Reconciling inventory or transactions

Next steps

After you identify your goals, you define the requirements of what constitutes a match. You choose which columns to compare and how to compare the columns.

You then create and test customized match specifications by using the Match Designer. These match specifications are used by the Two-source Match and One-source Match stages in InfoSphere QualityStage jobs.

Matching: key concepts

Matching automates the process of record linkage. The purpose of matching is to identify records that have a high probability of representing the same real-world entity.

Probabilistic record linkage uses statistical properties of values to calculate the likelihood that records correspond to the same entity. The information content, completeness, reliability, contextual frequency, and representation of the data are considered to create a cumulative assessment of confidence.

Understanding conceptual information is necessary to define match specifications. You need to know how to evaluate the results, estimate probabilities, set thresholds, and perform related tasks. Some understanding of the theory of record linkage is required.

Record linkage and the matching process

Record linkage is the methodology of identifying records that correspond to the same entity such as a person, household, or product.
In practice, you compare record pairs and classify them into one of these sets: matched pairs and nonmatched pairs.

Statistical methods of record linkage are required, due to these reasons:
- Columns contain errors or missing values.
- Data can be unreliable.
- You want to find the matches with a reasonable statistical assurance.

**Scenario for record linkage**
A scenario illustrates how record linkage works.

Consider two sources of data. Each source consists of a number of records, and the records contain some number of columns. Typically, each record corresponds to an entity, and the columns are attributes identifying the entity, such as name, address, age, and gender. The objective of the record linkage or matching process is to identify and link the records on each source that correspond to the same entity. The records do not contain totally reliable unique identifiers that make the matching operation trivial. Also, the individual columns are all subject to error.

The columns in common between the two sources are useful for matching. Not all columns, however, contain an equal amount of information, and error rates vary. For example, a column such as Gender has only two value states, and consequently, cannot impart enough information to identify a match uniquely. Conversely, a column such as FamilyName imparts much more information, but it might frequently be reported or transcribed (keyed) incorrectly.

You use weights to measure the contribution of each column to the probability of making an accurate classification. Record linkage has the following states; a record pair is classified as follows:

**Match** The composite weight is above a threshold (cutoff) value.

**Nonmatch** The composite weight is below a second threshold value.

**Undecided situation** The composite weight is between the first and second thresholds.

**Consider the level of information content**
Matching is more than just a comparison of available data. Both the information content and the representation of the data being compared are considered.

One or more columns in one record must have equivalent columns in the other record to compare them. For example, in order to match on family name and age, both records must have columns containing family name and age information. Although for a two-source match, the metadata for the comparable columns does not need to be identical.

For a record linkage project to be feasible, it is possible for a human to examine the record pairs and declare with reasonable certainty which of the pairs are a match or a nonmatch. For example, if the only column in common between two sources is gender, you do not then conclude that, because the gender agrees, the pair represents the same individual.
Information content measures the significance of one column over another (discriminating value). For example, a gender code contributes less information than a tax identification number.

Information content also measures the significance of one value in a column over another. In the United States, John contributes less information than Dwezel in a GivenName column. The given name, John, in the United States is much more common than the given name, Dwezel. Significance is determined by the reliability of the value and the ability of the value to discriminate a match from a nonmatch. And any comparison of records needs enough information to reach a reliable conclusion. For example, two identical customer records that contain only the family name, Smith, and that miss values in every other name and address column, do not have enough information to determine that the records represent the same person.

**How InfoSphere QualityStage matches records**

Matching is a probabilistic record linkage implementation. Matching is an open system that you can tune to create the best match for your needs.

**Weights, scores, and thresholds**

The matching process assigns numeric scores called weights to the comparison of individual data elements. The scores measure the contribution of each data element to the overall or composite weight. The composite weight, in turn, is the sum total weight of all defined comparisons. Some data elements contribute more weight because they are more critical to the match or more reliable than others. Statistical properties of the data elements and tuning parameters determine the weight of the contributions.

The weight for a data element is generated by using one of the comparison functions that are available in matching. Among the comparison functions are exact functions and functions that provide a full spectrum of error-tolerant or fuzzy matching functions. You can adjust the resulting weight of a given comparison function to reflect the importance of the data element to its domain and to the overall comparison.

The composite weight is compared against a set of thresholds (also called cutoffs) to determine how to translate the weight into a measurement of confidence. The confidence indicates the likelihood that a matching pair of records was identified.

**Computational loads of large volumes of data**

A matching process consists of one or more passes. Each pass employs a technique called blocking to reduce the computational load by focusing on pairs of records that are more likely to be matches. The pass also specifies the columns and comparison functions that are used to compare records.

For data sources of a reasonable size, it is not feasible to compare all record pairs, because the number of possible pairs is the product of the number of records in each source. For example, when you have two sources with as few as 1000 records each there are 1,000,000 combinations of records, one from each source. But there are, at most, only 1000 possible matches (if there are no duplicates on the sources). Therefore, the set of matched pairs contains, at most, 1000 pairs, and the set of nonmatched pairs contains the remaining 999,000 pairs.
There are many more nonmatched pairs than matched pairs. Only when you look at pairs of records that have a high probability of being matches, and ignore all pairs with very low probabilities, does it become feasible in a computational sense to link large volumes of data.

**Match types**
Matching provides several ways to process either a single source or two sources of data. You can apply the matching types on sequential files, data sets, records from relational databases, or as a real-time service.

**Two-source matching**
Two-source matching involves a data source and a reference source.

The following types are available for two-source matching:
- Two-source many-to-one
- Two-source many-to-one, multiple
- Two-source many-to-one, duplicate
- Two-source one-to-one

For two-source matching, one or more columns on the data source must have equivalent columns on the reference source. For example, to match on family name and age, both sources require columns containing this information. The location, name, and length of a column in the data source can be different from its equivalent column in a reference source.

**One-to-one matching and examples:**
In one-to-one matching, one record from the data source can be assigned to only one record from the reference source, and vice versa. Each record pertains to a single individual or event.

When a reference record is matched, the reference record cannot match to any other records and is removed from subsequent passes. After record pairs are scored and possible matches are determined, a linear-sum-assignment algorithm is used to optimize the assignment of matched pairs.

Examples of one-to-one matching are as follows:
- A sample of individuals counted in the United States Census are matched to a Post Enumeration Survey. The test objective is to determine which individuals and households are present in both the census and survey.
- The National Highway Traffic Safety Administration matches highway traffic accident reports, which are completed by law enforcement officers, to injury records, which are completed by emergency medical service and hospital personnel. Match provides a way to measure how highway safety countermeasures affect medical outcomes.
- Matching files that contain administrative data are useful in a number of areas. For example, if juries are chosen from both driver license records and voter registration rolls, the two files can be matched to identify individuals that are listed in both files. Steps can be taken to ensure that individuals that are listed both files are not twice as likely to be selected for a jury as individuals listed in only one file.
Matching administrative files can also provide data that is missing from one set of files. For example, a driver's license file could provide information regarding age, gender, or race, while a voter's file might provide a national identity number.

**Many-to-one matching and examples:**

You use many-to-one matching to match a single data source to a reference source. A reference record can match to many records on the data source.

The *many-to-one, multiple* type reports on all reference records that tie for the best match score for a data record. One reference record is chosen as a match and the others that have an identical weight to the matched pair are flagged as duplicates. The *many-to-one, duplicate* type identifies the best record as a match and flags all other reference records, that have a weight above the duplicate cutoff threshold, as duplicates.

Examples of many-to-one matching are as follows:

- Matching a purchased mailing list against a list of customers. The mailing list can have duplicates, or many lists might be integrated. Consequently more than one record on the mailing list can match to the list of customers.
- Matching a file of sales activities or accounts receivable to the file of customers. There might be more than one sale for any given customer.
- Address matching to enrich data records. You can match a data source containing postal addresses to a reference source to obtain geographic coordinates, census tract numbers, special area codes, and other information. For example, 103 Main St. can match to a record like 101-199 Main St. There can be more than one user record within that block; therefore multiple records on the data source might match to a single record on the reference source.
- Matching a file that contains data about park visitors to census reference sources to obtain the census tracts where the visitors reside. You can use the information to prepare a demographic profile of the visitors.
- Matching a customer file to a geographic coordinate file to produce an automated map that shows the location of the customers.
- Matching a file of ambulance calls or fires to a file containing fire district codes to produce a map that shows the locations of such events.

**One-source matching**

One-source matching groups records that have similar attributes into sets.

The following types are available for one-source matching:

- One-source dependent
- One-source independent
- One-source transitive

When you *deduplicate* a data source, you first identify groups of records that share common attributes. The identification allows you to correct, merge, or eliminate the duplicate entries. But one-source matching is not only for deduplication. One-source matching also provides the capability of grouping records even in the following situations:

- The records do not have unique identifiers, such as tax identification numbers.
- The content is subject to error.
One-source matching determines the relationship among the match passes, the records that are processed by each of the passes, and how the groups are formed. In most cases, you choose the dependent match type, because you want duplicates removed from consideration, so that they do not match to other records in subsequent passes. However, the independent match type is useful when you want to link people or organizations regardless of address. For example, you can link each doctor with all the locations where the doctor works. The transitive match type is useful if you want to account for inconsistent data entry in columns that assist in duplicate identification, for example, date of birth and driver’s license numbers.

The passes process and group records based on the type that you select:

**Dependent**

The passes process the data sequentially. In each pass, groups are built around master records. The groups that are formed in all the passes for the same master record are combined to create the final group for the master. Each duplicate record in a group matches the group master record in one of the match passes. The master records and nonmatched records from a pass are made available for the subsequent pass. Duplicates are taken out of consideration, so that they are not assigned to more than one group. Existing master records are given priority in group construction in subsequent passes.

**Independent**

Each pass processes all the input records. Like the one-source dependent match type, in each pass, groups are built around master records. But because each pass processes all records, a record can be a member of a group from more than one of the passes. (Similarly, a record can be a master in a group that is built in one pass while being a duplicate in a group that is built in another pass.) The groups from all the passes are merged, so that groups that have a record in common form a single group.

If record A is in a group with record B, and record B is in a different group with record C, then the two groups are merged so that records A, B, and C are all in the same group. (A record ends up in no more than one group.) Groups are merged until all groups that have records in common are merged. At the pass level, the relationship that determines group membership is that of records matching a master record. However, for the merge process, the relationship is one of group membership. Therefore, members in a group can be connected by a chain of relationships and do not necessarily all match a common master.

**Transitive**

Like the one-source independent match type, the one-source transitive match type processes all the input records, in each of the passes. But unlike the one-source independent match type, the one-source transitive match type uses all pairs that have a score above the match cutoff in a pass, not just pairs that are grouped in a pass. Using only grouped pairs discards high scoring record pairs if the records end up in different groups.

The one-source transitive match type does not discard the information about high scoring record pairs that are in different groups. The one-source transitive match type builds groups so that all records that score above the match cutoff in any pass are in the same group. For example, if record A and record B scored above the match cutoff in a pass, and record B and
record C scored above the match cutoff in a pass (possibly the same pass),
then records A, B, and C are added to the same group. (A record ends up
in no more than one group.)

Like the one-source independent match type, members in a group can be
connected by a chain of relationships and do not necessarily all match a
common master. But the one-source transitive match type chain can extend
further because it uses all the pairs that score above the match cutoff.

Master record selection and group construction:

One-source matching builds groups of records by finding records that are good
matches to a master record.

Match processes one block of records at a time. Each record in a block of records is
compared to every other record in the block. A master record is designated. Each
master record is used to create a group. When the block is completely read,
another block is read, until the end of the input data is reached.

The following list describes the process:
1. All pairs of records within a block are scored.
2. From the records that have not been added to a group, the record pair with the
   highest composite weight over the cutoff thresholds is selected.
3. The record from the pair with the highest score when compared to itself is
designated as the master record.
4. Other records are then added to create the group. A record whose composite
   weight when compared to the master record is above the match cutoff is
   assigned to the group as a *match duplicate*. A record whose weight is between
   the clerical and match cutoffs is assigned to the group as a *clerical duplicate*.
5. The process is repeated within the block until there are no remaining pairs of
   ungrouped records whose weight is above either of the cutoffs.

Examples of one-source matching:

One-source matching is often used to deduplicate lists and identify groups of
related records.

Examples of one-source matching are as follows:
- Grouping all hospital charges for a patient.
- Finding all members of a household. All persons at a specific address or
  building are grouped.
- Deduplicating a file of customer invoices to identify individual customers.
- Deduplicating a mailing list that was created by merging multiple files.
- Removing duplicate records that pertain to the same individual, household,
  event, product, or part, from a data source before updating or creating a data
  warehouse.

Match column selection

For any given pass, you might or might not want to use some match columns for
blocking columns. In general, if a column is a blocking column, then you do not
make it a match column.

If all passes use the same comparisons, then the weights generated are identical for
each pass. You might want to make some blocking columns match columns to
keep all pass comparisons the same. Making the comparisons the same might make setting cutoff weights easier, but might not be a good idea in terms of statistical accuracy.

Specifying only blocking columns and no matching columns creates an exact match. All record pairs agreeing on the blocking columns are considered to be matches and duplicates.

It is a good idea to compare all columns in common in a two-file match. Often, people want to omit columns that are not reliable. However, it is often useful to include unreliable columns and assign a low m-probability to the columns, so that there is not much penalty for mismatches.

Decreasing the number of match comparisons might result in more matches but might also reduce the quality of some of the matches. Fewer comparisons can decrease the ability of the matching process to differentiate the correct matches from the incorrect matches.

When you use a blocking column that contains a manufactured or encoded value, such as a Soundex code or part of a value (for example, the first three characters of a family name), the underlying values are not necessarily the same. If matching the underlying values must be part of the consideration, be sure to include the underlying values as a match comparison.

**Match passes**

In a match pass, you define the columns to compare and how to compare them. You also define the criteria for creating blocks.

You use multiple passes to implement complementary or independent business rules, to help overcome the complexities of processing large data volumes, and to compensate for data errors or missing values in the blocking columns.

Strategies for multiple pass include the following actions:

- Make the early passes the most restrictive, and use the most reliable columns for blocks. Loosen the blocking constraints in successive passes as the number of nonmatched records decreases.
- Use different blocking constraints with different perspectives or views. Also, use similar match comparisons so that pairs that are missed by the blocking conditions of one pass are found by the blocking conditions of another pass.
- Use different sets of match comparisons to implement complementary match rules.
- Select a strategy that produces small blocks of records in the match passes. Smaller blocks are many times more efficient than larger blocks. Use multiple passes to define different blocks.

The strategy that you choose to match data depends on your data cleansing goals. After you decide on the criteria, you can design a matching strategy to meet the goals. For example, if your goal is to match households, you might use name, address, and birth date data to determine a match.

**Match pass examples**

Using common column names, the examples in this topic suggest which blocking columns that you might want to use in the first, second, and third match passes.
In the examples, notice how sometimes columns are used whole (birth date) or divided into parts (birth year). Where necessary, this strategy of using parts of columns is accomplished by creating additional columns.

The following examples assume that you have two sources containing a date, given name, family name, and gender.

**Date, given name, family name, and gender columns**
- match pass 1: date and gender
- match pass 2: Soundex of family name and first two characters of given name
- match pass 3: Soundex of given name, year, and month (from the date column)

**Date, family name, city, and postal code columns**
- match pass 1: date and gender
- match pass 2: postal code
- match pass 3: Soundex of family name and first two characters of city

**National identity number, family name, given name, and birth date columns**
- match pass 1: national identity number
- match pass 2: birth date
- match pass 3: Soundex of family name, birth year

**Family name, middle initial, given name, gender, and birth date (year, month, day) columns**
- match pass 1: family name, gender, and birth date
- match pass 2: birth month, birth day, the first character of the given name, and middle initial

### Blocking

Blocking is an important factor in match performance and efficiency.

*Blocking* provides a method of focusing the scope of record pairs to examine. For sources of reasonable size, it is infeasible to compare all record pairs, because the number of possible pairs is the product of the number of records on each source. Blocking provides a method of limiting the number of pairs being examined.

Blocking partitions the sources into mutually exclusive and exhaustive subsets, and the matching process searches for matches only within a subset. If the subsets are designed to bring together pairs that have a higher likelihood of being matches and ignore those that are less likely matching pairs, successful matching becomes computationally feasible for large data volumes.

Records in a block already match exactly on one or several blocking columns. Blocking is an essential step to effective matching in your data cleansing project.

**Blocking considerations**

Blocking is a method used to increase performance and efficiency, but it is important to balance the need for efficiency with the need to meet your quality objectives.

You have a potential trade-off between these issues:
The computational cost (examining too many records) that blocking helps to resolve.

The potential to increase the rate of false negatives (when a record pair that does represent the same entity is not a match) because the records are not members of the same block.

However, well planned and implemented blocking and pass strategies can help mitigate most of these concerns.

When you select columns for blocking, think through the following considerations:

- Choose columns with reliable data.
  - Choose columns that make business sense to meet your objective. If you want to identify unique customers, then blocking by house number is not the best choice.
  - The character discrete investigate reports help you select the blocking columns. The reports tell you how often a column is populated. If you choose columns with reliable data, then you are truly grouping like records, because the data values are reliable.

- Choose columns with a good distribution of values.
  - Some columns do not have many possible values. For example, gender, as a column value, typically does not make blocks small enough.
  - Sometimes columns do not have a good distribution of values. For example, if your data is from only a few states or provinces, then the states or provinces might not be the best columns to use.

- A combination of columns might be preferable.
  - Use enough columns to keep the block size manageable.

**Blocking analogy**

You have the task of matching into pairs many recently washed socks that are in a box. You might select one sock and search through the entire box, sock by sock to find the partner of the sock. But most likely you decide to seek a more efficient method. You can sort or “block” socks by characteristic. You can sort by color first. Then you need to compare only white socks to white socks and not waste time comparing a white sock to a blue sock.

What if you found one pink sock that was potentially white originally but is now discolored? Due to color error, you do not put the pink sock with the white socks in your first pass of sorting socks. Using match terms, the pink sock does not make the block of white socks. In subsequent match passes, after you sort by color, you sort the remaining, unpaired socks by size and shape. In one of these subsequent passes, you might find the most likely match for the pink sock. Similarly, multiple passes help overcome the problem of records not making the correct block group.

**Blocking guidelines**

The general guidelines in this topic can help you select useful blocking columns.

Follow these guidelines as you plan and implement blocking:

**Use blocking columns like sort keys**

All records that have the same value in the blocking columns are eligible for comparison during the matching phase.

**Make blocks as small as possible**

One to two hundred records per source is a good size. Efficiency decreases
as block size increases. Blocking partitions the sources into subsets that make computation feasible. Using small blocks results in better system performance when processing records. To create an effective blocking strategy, use several columns for blocking in each pass.

**Implement a blocking strategy that is consistent with your matching goals**

Although small blocks of several columns are preferable, a blocking scheme that is highly restrictive can create too many blocks. The problem with too many blocks is that records that are potential targets for comparison are distributed in multiple blocks. If the records are distributed in multiple blocks, then the records are not compared within the pass, and your match results might be compromised.

**Avoid block overflow**

Block overflow occurs if, during a pass, more records are grouped in a particular block than the number that you specified for the block overflow limit. When block overflow occurs, all records in the block are skipped by the pass.

You can configure the block overflow setting. The default setting for the block overflow is 10,000 records. For a two-source match process, many-to-one match, the reference block size is the only block that is constrained by the overflow setting. If a block overflow occurs, examine the blocking criteria. Instead of raising the block overflow limit immediately, understand why the overflow occurs. Determine if one of the blocking columns has many different values (high cardinality).

If you raise the block overflow limit without understanding that you have a problem with your data, you might inadvertently hide a problem with your data or your blocking strategy. Raise the block overflow if you know why the block overflow occurred and if you have a logical reason to do so.

**Define missing values for blocking columns**

Define missing values for blocking columns. Blocks that have columns with missing values are skipped in the matching process. Convert all generic stand-ins for missing values (such as UNKNOWN or 99999) to nulls. When missing values are not converted to nulls, the matching process does not identify the values as missing. As a result, costly block overflows can occur. For example, if a national identity number is present in only half the records but the missing values are reported as spaces instead of nulls, the blank numbers form one large block. The large block might cause excessive computation and block overflow.

**For sources with limited information, use a reverse Soundex code**

The reverse Soundex is formed by looking at the name backwards and computing a Soundex. For example, the reverse of JONES would be SENOJ. Because the Soundex algorithm preserves the first letter, running a reverse Soundex allows for errors at the beginning of names.

**Blocking examples**

The best blocking columns are columns with the largest number of values possible and the highest reliability. These general guidelines can help you to select useful blocking columns.
Individual identification numbers

Identification numbers are typically reliable. In a first pass, use individual identification numbers such as national identity numbers, medical record numbers, claim numbers, and so forth, even if the numbers are missing or in error in a sizable percentage of the records.

For example, sources contain a national identity number in 50 percent of the records. Pass 1 is blocked by national identity number. Match skips all records with no national identity number. The skipped records are applied to the second pass. However, a fairly large percentage of the records are matched easily.

If there are several identification numbers, use them on the first two passes. After that, try other columns. Identification numbers are ideal for blocking columns, because they partition the records into many sets.

Birth dates

Birth dates are excellent blocking columns.

For example, by using the InfoSphere DataStage Transformer stage, you can separate birth dates into these columns: BirthYear, BirthMonth, and BirthDay. For larger sources (over 100,000 records), use all three columns as a first-pass blocking column. For smaller sources, use BirthYear, BirthMonth, and an additional column such as Gender. Subsequent passes can use blocks containing BirthDay.

Event dates

Event dates, such as an accident date, claim date, hospital admission date, and so on, are useful as blocking columns.

Names

A phonetic encoding (such as Soundex or NYSIIS codes) of the family name is a useful blocking column. For large sources, combine this code with the first letter of the given name or birth year. Remember, different cultures use different conventions for family names, so do not rely exclusively on them.

Addresses

Postal addresses present a wealth of information for blocking. For example, postal codes and a phonetic encoding (Soundex or NYSIIS) of street name or city name are all excellent choices.

Weights and record comparisons

The information content of the data determines which record pairs are matches and which are nonmatches, both for an automated process like matching in IBM InfoSphere QualityStage and for manual matching.

Each column within a record provides some information. The weight is a measure of the information content of the data and the likelihood that a record pair matches. Taken together, all the columns and comparisons determine the status of the pair that is being examined by the matching process.
Some columns provide more information more reliably than others. For example, it does not make sense to compare just the gender column and assert that if there is agreement, the record pair represents the same individual. However, it is more reasonable to compare records by using an individual identification number, and assert that if the number agrees then the record pair represents the same individual. You can use the discriminating power of each column to measure and predict matches.

**Composite weights**

For each record pair that you want to compare, a composite weight is computed.

The *composite weight* is the sum of the individual weights for all the match comparisons. It is the measure of confidence that the two records are a match.

**Computation of weights**

Match weights measure the extent to which two records match based on the designated matching criteria. Record pairs with higher weights are a stronger match than record pairs with lower weights.

The *odds ratio* is the ratio of the odds that an event occurs in one group to the odds that the event occurs in another group. An odds ratio of one (1) implies that the event is equally likely in both groups. An odds ratio greater than one (>1) implies that the event is more likely in the first group. An odds ratio less than one (<1) implies that the event is less likely in the first group.

The weight calculation for a column uses the odds ratio of the likelihood that the values are the same and the record pair is a match versus the likelihood that the odds of the values are the same and the record pair is a nonmatch. Using such an odds ratio minimizes the number of clerical pairs for the chosen cutoff values.

**Agreement and disagreement weights**

For each match comparison, the matching process calculates an agreement weight and a disagreement weight.

The match weight is derived by using the agreement and disagreement weights established for that column.

If a match comparison agrees for the record pair being compared, the agreement weight is added to the composite weight. The agreement weight is a positive value. If a match comparison disagrees for the record pair being compared, the disagreement weight is added. The disagreement weight is a negative value. Therefore, agreement weights add to the composite weight, and disagreement weights subtract from the composite weight. The higher the score is, the greater the agreement is.

*Partial weight* is assigned for non-exact or fuzzy matches.

Missing values have a default weight of zero.

**Reliability and chance agreement**

Each column has probabilities that are associated with it. Those probabilities are called the m and u probabilities and are sometimes referred to as m-prob and u-prob.
Reliability: a discussion of m-prob

The *m probability* is the probability that a column value agrees with its pair value given that the record pair being examined is a matched pair. The m probability is effectively one minus the error rate of the column. For example, in a sample of matched records, if gender disagrees 10% of the time due to transcription errors or being misreported, the m probability for this variable is 0.9 (1 - 0.1).

A high m probability (0.9) indicates that the data in a given column is considered highly reliable. The higher the m probability is, the higher the disagreement weight is. A disagreement between two values in a reliable column is penalized more highly than for an unreliable column for which you set a low m probability (0.1).

If a column is important, then the m probability can be given higher values. If the m probability is high, it is equivalent to saying that a disagreement of values from that column is a rare event in a matched pair. Consequently the penalty for a nonmatch is high.

Chance agreement: a discussion of u-prob

The *u probability* is the probability that a column value agrees with its pair value given that the record pair being examined is a nonmatched pair. Because there are so many more possible nonmatched pairs than matched pairs, the u probability is effectively the probability that the values agree at random.

Match uses a *frequency* analysis to determine the probability of a chance agreement for all values.

Rare values bring more weight to a match.

For example, the probability that two gender values agree at random is about 0.5. Given a uniform distribution, there are four possible combinations of the two values:

*Table 2. Possible combinations of the gender values for a pair of records*

<table>
<thead>
<tr>
<th>File A</th>
<th>File B</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>F</td>
</tr>
<tr>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>F</td>
<td>M</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

The gender agrees in two of the four combinations. Therefore, the pair has a u probability of 0.5.

Weights calculations

Column weights reflect both the reliability of the data and the possibility for chance agreement.

The agreement weight for a column is calculated as the logarithm to the base two of the ratio of the m probability and u probability, as shown in the following equation:

$$\log_2(\frac{m\text{ probability}}{u\text{ probability}})$$
The disagreement weight for a column is calculated as shown in the following equation:
\[
\log_2\left(\frac{1 - m \text{ probability}}{1 - u \text{ probability}}\right)
\]

**Agreement weight example**
Gender and national identity numbers demonstrate how differences in frequency or reliability change the agreement weight for columns.

To see how the weight computation translates into actual values, consider the values for the columns, gender and the national identity number. In this example, the gender has a 10 percent error rate, and national identity number has a 40 percent error rate.

The \(m\) probability for gender is 0.9. The \(u\) probability is 0.5. Therefore, the weight for gender is shown in the following example:
\[
\log_2 \left(\frac{m}{u}\right) = \frac{\ln(m/u)}{\ln(2)} = \frac{\ln(0.9/0.5)}{\ln(2)} = 0.85.
\]

Conservatively, assume that the probability of a chance agreement of the national identity number is one in 10 million. Given \(m\) as 0.6 (40% error rate in matched pairs), the weight for national identity number is \(\ln(0.6/0.0000001) = 22.51\).

Therefore, the weight for a match on the gender column is 0.85, and a match on the national identity number is 22.51. The weights capture what you might intuitively know about the columns.

**Cutoff values**
Match and clerical cutoffs are thresholds that determine how to categorize scored record pairs.

Your goal of setting cutoffs is to minimize uncertainty in the match results while you limit the number of false categorizations.

Record pairs with composite weights equal to or greater than the match cutoff are considered matches. Record pairs with composite weights equal to or greater than the clerical cutoff but less than the match cutoff are called *clerical pairs*. The matching process is uncertain whether clerical pairs are matches or nonmatches. Pairs with composite weights below the clerical cutoff are considered nonmatches. You can set cutoffs at the same value, so that you eliminate clerical records.

You can set a high cutoff threshold to limit the results to better quality matches, though possibly fewer matches. A lower threshold can produce more matches, but some of these matches might be of lesser quality. Business requirements help drive decisions. Results can vary depending on whether you take a conservative or more aggressive approach to defining the cutoff values.

For example, matching for the purpose of docking a person’s pay might require a more conservative approach than deduplicating a mailing list for shopping catalogs. As a best practice, keep in mind the business purpose when you tune the match settings.

The composite weights assigned to each record pair create a distribution of scores that range from very high positive to very high negative. The graph in Figure 4 on page 66 focuses on the area of a histogram where the number of low scoring pairs tails off and the high scoring pairs starts to increase. In this area of the graph, there is not a high likelihood that pairs are either matches or nonmatches.
You set the cutoff values to tell the matching process how to handle pairs in this range. Differences in the distribution of pairs help to determine the settings. The detail of the graph of matched versus nonmatched records relate to the cutoff points. You typically set cutoffs on the down slope of the nonmatched and the up slope of the matched. Where you set the cutoff is influenced by both the business objective and the tolerance for error.

The weights between the vertical lines form a gray area, where one cannot say whether the pair is matched or not. You want to have enough variables to distribute the matched versus nonmatched groups further apart (minimize the uncertain pairs). You know that you developed a good match strategy when what is in the clerical area are records with mostly blank, missing, and default values.

![Figure 4. Histogram of weights](image)

The fewer records in the clerical area, the fewer the cases to review, but the greater the probability of errors.

*False positives* are cases in which records are classified as matched records but really are nonmatch records. *False negatives* are cases in which records are classified as nonmatch records but are matched records.

The goal of setting cutoffs is to minimize the number of clerical pairs and limit the number of false negatives and positives. You fine tune the results depending on the goals of your organization.

**Guidelines when determining probability**

You can use these guidelines as you assign probabilities.

The higher the m probability is, the greater the disagreement weight is. Therefore, if a column is important, give the m probability higher values. If the m probability is high, it is equivalent to saying that a disagreement of that column is a rare event in a matched pair, and consequently the penalty for a nonmatch is high. The weights computed from the probabilities are visible in the data viewer of the Match Designer so that you can inspect the results.

Use the following guidelines when determining m probabilities.

- Give high m probabilities to the columns that are the most important and reliable.
Give lower m probabilities to the columns that are often in error or incomplete.
The m probability must always be greater than the u probability and must never
be zero or 1.

Agreement or disagreement between data values is more significant for reliable
data and less significant for unreliable data.

As a starting point, you can guess the u probability because the matching process
replaces any guess with actual values. A good estimate is to make the u probability
$1/n$ values, where $n$ is the number of unique values for the column. By default, the
u probability for each comparison is calculated automatically by the matching
process using the frequency information from the Frequency stage. This calculated
u probability is important for columns with non-uniform distributions.

The frequency information allows match to vary the weights according to the
particular values of a column. Rare values bring more weight to a match. For
example, in using a column such as FamilyName, the values Smith and Jones are
common in the United States. However, a value such as Alcott is relatively rare in
the United States. A match on the rare family name, Alcott, gets a higher weight
than a match on the more common family names because the probability of chance
agreement on Alcott is relatively low compared to chance agreements on other
values such as Smith or Jones.

For columns with a uniform distribution of values and a high number of different
values (such as individual identification numbers), it is better not to generate
frequency information. Specify the vartype NOFREQ in the Variable Special
Handling window of the Match Designer.

Even exact matching is subject to the same statistical laws as probabilistic
matching. It is possible to have two records that contain identical values and yet
do not represent the same entity. You cannot make a definitive determination when
there is not enough information.

---

**Preparing to use the Match Designer**

Before you use the Match Designer to define and test the match specification,
ensure that the Match Designer database is configured and that your sample data
is prepared.

**Before you begin**

Refer to the *IBM InfoSphere Information Server Planning, Installation, and Configuration
Guide* for more information about preparing for the Match Designer database.

**Setting up your Match Designer database**

- To enable the Match Designer test environment and run test match passes, you
  need a database to receive output from those activities.
  - You can use an existing database or create a new one.
  - This database can be located on any accessible computer. It can be collocated
    with IBM InfoSphere Information Server on the client or engine tier. In
    addition, it can be configured on a separate system.
  - You need to create this database only one time, which is when you first install
    IBM InfoSphere DataStage and IBM InfoSphere QualityStage. Unless you elect
to change it, the same database can be reused during the development of
successive match specifications.
- This database is not required during the runtime of a match job. It is required by the Match Designer only during the development of a match specification. You can export a completed match specification and run it on a computer that did not participate in the development of that match specification. That runtime computer does not require a connection to the Match Designer database.

- To connect with the Match Designer database, you need an ODBC data source name (DSN) on each client that hosts the InfoSphere DataStage and QualityStage Designer client. Refer to the IBM InfoSphere Information Server Planning, Installation, and Configuration Guide for more information about configuring the Match Designer and defining a DSN.

- That same DSN that is available on the client must also be available on the engine tier.

- If the client and engine tiers of InfoSphere Information Server are installed on the same computer, you need only one DSN.

**Note:** During installation, InfoSphere Information Server installs various ODBC drivers that you can use when you define the DSN that is required by the Match Designer.

**Preparing your data**

- Typically, you want to standardize and format your data source and reference source before preparing the sample data and frequency data that are required for Match Designer input. Include all the columns that you plan to use for blocking and matching. You might want to replace non-null values such as spaces, zeros, or nines with null values in blocking columns.

- Prepare Match Designer sample input data from the data source in the data set format by using the InfoSphere DataStage Data Set stage. If you are preparing a specification for a Two-source Match stage, also prepare sample input data from the reference source in the data set format by using the Data Set stage.

- Using the Match Frequency stage, prepare frequency input data that was generated from the standardized, formatted data. This frequency information must be in the data set format that you create by using the Data Set stage. If you are preparing a match specification for a Two-source Match stage, also prepare reference data frequency information in the data set format by using the Data Set stage.

- Using the InfoSphere DataStage and QualityStage Designer client, import or create table definitions that define the columns in the input data for the Match Designer.

**Defining the test environment**

To test match passes in the Match Designer, configure the match specification test environment.

**About this task**

This task must be done once for each match specification that you create. Also, the test environment must be updated if you change any information such as name, path, or content of the frequency and sample data set files.

The test environment consists of the following items.

**Sample information**

You need data to use all the features of the Match Designer. The Match
Designer processes more efficiently if you use moderate data volumes. Use random samples from your production sources in the Match Designer to ensure that the sample data is a good representation of the source data. Sample data must be in IBM data set file format. To create a data set file, use your sample data as input to a Data Set stage. The Data Set stage produces a data set file as output.

**Frequency information**
You need to define the data frequency data set and, for a reference match, the reference frequency data set to serve as input to your match specification. Frequency data sets are created by using a Match Frequency stage and contain frequency distributions for the data columns that participate in the match. Also, specify a maximum frequency value. This value is the maximum number of frequencies that are used when you run a test pass. The default value of 100 is typically sufficient.

**Test results database**
You need to define a database to hold test results from the Match Designer (Match Designer database).

**Execution environment**
These optional settings for experienced Match Designer users are a means of controlling or fine-tuning the way that match passes run on the engine tier. Environment variables can be especially useful as debugging aids when additional, detailed information is needed to diagnose a problem. Refer to the *IBM InfoSphere DataStage and QualityStage Parallel Job Advanced Developer’s Guide* for reference details on the environment variables that affect the setup and operation of parallel jobs.

**Procedure**
1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification for which you want to define the test environment.
3. In the Match Designer, click **Configure Specification > Test Environment** to open the Test Environment window.
4. In the **Sample Information** area, specify the sample input data from the source data.

   **Remember:** The sample data must be in the IBM InfoSphere DataStage Data Set format.
   a. In the **Data Sample Data Set** field, enter a file name or click ... to select a data set file.
   b. For two-source match specifications only, in the **Reference Sample Data Set** field, enter a file name or click ... to select a data set file of the reference data.
5. In the **Frequency Information** area, specify the frequency input data from the source data.

   **Remember:** The frequency data must be in the InfoSphere DataStage Data Set format.
   a. In the **Data Frequency Data Set** field, enter a file name or click ... to select a data set file.
   b. For two-source match specifications only, in the **Reference Frequency Data Set** field, enter a file name or click ... to select a data set file of the reference data.
6. In the **Maximum Frequency Value** field, enter the maximum number of frequencies to use when you run a test pass.

   **Note:** Regardless of this value, the number of frequencies that are used never exceeds the maximum number of frequencies that are output by the Match Frequency stage that generated the frequency file.

7. In the **Test Results Database** area, specify the connection information for the database that stores test pass results by using one of the following options.

<table>
<thead>
<tr>
<th>Connection</th>
<th>Action</th>
</tr>
</thead>
</table>
   | Existing data connections       | 1. Click **Load** to select a data connection object from the metadata repository.  
                                   | 2. Test your connection to the data source.                              |
   | New connections                  | 1. If a current connection exists, select **Clear** to specify a new connection.  
                                   | 2. From the **ODBC Data Source Name** list, select a DSN.              
                                   | 3. Enter your user name and password for the database for the DSN.    
                                   | 4. Test your connection to the data source.                            
                                   | 5. Save the connection to the metadata repository as a data connection object.  
                                   | The Match Designer can create only new data connections. To modify a data connection that you created, you must use the InfoSphere DataStage and QualityStage Designer. |

8. Optional: In the **Configuration File Location** field, enter a file name or click ... to select a configuration file other than the default configuration file (APT_CONFIG_FILE) for the parallel engine. Configuration files enable you to set dynamically at run time the degree of parallelism and resources that are used by parallel jobs such as test match passes.

9. Optional: In the **Environment Variables** field, enter an environment variable and a value in the format of `ENVIRONMENT VARIABLE=VALUE`. Refer to the *IBM InfoSphere DataStage and QualityStage Parallel Job Advanced Developer’s Guide* for reference details on the environment variables that are available for affecting the setup and operation of parallel jobs. Ensure that the environment variable and value are syntactically correct because the Match Designer does not validate them. If they are incorrect, you will receive an error when you run a test pass or update the test environment.

10. Click **Update**. If the update is successful, the latest test environment settings are saved and the Test Environment window closes. If the update is not successful, the Test Environment window remains open and you can save, modify, or discard the current settings.

### Updating configuration information

If you make any changes to the test environment or the files that it uses, you must update the test environment configuration.

### About this task

When you change any information in the Test Environment window or when the content of the input data set or frequency data set changes, the Match Designer...
must regenerate the test environment for the match specification.

**Procedure**
1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification for which you want to update the configuration information.
3. Click **Configure Specification > Test Environment**.
4. Select the configuration settings that you want to update.
5. Click **Update** to apply test environment changes. The next time you run a test pass, you will see these changes.

---

**Using the Match Designer**

You use the Match Designer to define and test the criteria for matching data.

The Match Designer is a data-centric design environment for the repetitive process of defining a match, running it against sample data, viewing data results and statistics, and fine-tuning the match until you achieve your matching objectives.

In the Match Designer, the results of a match job are displayed as statistics, data grids, and charts. You can use built-in capabilities of the Match Designer to search, sort, and drill down into the results.

By using the Match Designer, create match specifications and associated match passes for one-source or two-source matches that you then deploy in the One-source Match or Two-source Match stages. Optionally, if you want to restrict the columns for which frequency information is generated, deploy a match specification in the Match Frequency stage.

A match run is essentially a two-step process:
1. You isolate subsets of records to process, by using a blocking strategy.
2. You then examine the scores for those records.

Blocking creates non-intersecting subsets of data in which the records in the set have a higher probability of matching each other rather than matching other records. Creating a block allows the match process to look for matches only in records that have the same values in the blocking columns.

With the Match Designer, you determine which columns are to be examined for the matching process.

Use the completed match specification in the following stages:
- One-source Match stage
- Two-source Match stage
- Match Frequency stage. Using a match specification is optional for this stage. A match specification restricts the columns for which frequency data is generated. Use a match specification if you want frequency data for some columns but not others.

You apply the results from these match stages to the next phase of your data cleansing project, such as a Survive stage or a database load activity.
Creating match specifications
Create a match specification to define the criteria that you want to use to implement a specific matching strategy.

Creating a one-source match specification
A one-source match specification provides the criteria that are used to group records that have similar attributes.

Procedure
1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Click File > New.
3. Select Data Quality > Match Specification and click OK.
4. Create a match specification by using one of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a basic match specification by using the Match Specification Setup Wizard.</td>
<td>1. Click Help me get started. 2. Complete the pages of the wizard.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| Create a match specification in the Match Designer. | 1. Click **Take me to the Match Designer**. The Match Designer opens with an untitled specification and a default pass with the temporary name, **MyPass**. **Note:** The match type is set to *Two-source Many-to-one Multiple* and a button with two data files is shown.
| | 2. Click **Two-source Many-to-one Multiple** and select one of the one-source match types.
| | 3. Click **(Define input for this specification)**.
| | 4. In the Input Columns window, click **Load**.
| | 5. In the Table Definitions window, select the table definition that you want to use and click **OK**. The columns from the table definition are displayed in the Input Columns window.
| | 6. Click **OK**. The table definition name shows under the data button.
| | 7. Click **Save > Specification**.
| | 8. Select or create a folder in which to save the match specification. A default Match Specifications folder is provided, but you can create an alternative file structure that suits the needs of your project.
| | 9. Enter a name for the match specification and click **Save**. The specification is saved to the folder and the specification name is displayed in the title of the Match Designer window.

**What to do next**

After you define one or more match passes and test a match specification, you must provision the specification before you can use it in an One-source Match or Two-source Match stage. To provision the specification, right-click the specification in the InfoSphere DataStage and QualityStage Designer **Repository** tree, and click **Provision All**.

Multiple match specifications can be provisioned at the same time.

**Creating a two-source match specification**

A two-source match specification provides the criteria for identifying related records in two data sources.
**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Click **File > New**.
3. Select **Data Quality > Match Specification** and click **OK**.
4. Create a match specification by using one of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Create a basic match specification by using the Match Specification Setup Wizard. | 1. Click **Help me get started**.  
2. Complete the pages of the wizard. |
| Create a match specification in the Match Designer. | 1. Click **Take me to the Match Designer**.  
The Match Designer opens with an untitled specification and a default pass with the temporary name, **MyPass**.  
*Note:* The match type is set to **Two-source Many-to-one Multiple** and a button with two data files is shown.  
2. Click **Two-source Many-to-one Multiple** and select one of the two-source match types.  
3. Click .  
4. In the Data Table Definition pane of the Input Columns window, click **Load**.  
5. In the Table Definitions window, select the data table definition that you want to use and click **OK**. The window closes and the columns from the table definition are displayed in the Input Columns window.  
6. In the Reference Table Definition pane, click **Load**.  
7. In the Table Definitions window, select the reference table definition that you want to use and click **OK**. The reference table definition must be different from the data table definition.  
8. Click **OK**. The table definition names show under the data button.  
9. Click **Save > Specification**.  
10. Select or create a folder in which to save the match specification. A default Match Specifications folder is provided, but you can create an alternative file structure that suits the needs of your project.  
11. Enter a name for the match specification and click **Save**. The specification is saved to the folder, and the specification name is displayed in the title of the Match Designer window. |
What to do next

After you define one or more match passes and test a match specification, you must provision the specification before you can use it in a One-source Match or Two-source Match stage. To provision the specification, right-click the specification in the InfoSphere DataStage and QualityStage Designer Repository tree, and click Provision All.

Multiple match specifications can be provisioned at the same time.

Defining match passes

A match specification consists of one or more match passes.

When defining a match pass, you specify which columns in your source data to use for blocking or matching. Blocking columns determine which records are compared. Those records are called a block. Matching columns establish how records within the block are compared. In addition to matching columns, the match comparison that you select affects the way that records within the block are compared.

Adding match passes:

Add match passes as part of the larger process of creating a match specification.

About this task

New match specifications contain a default match pass with the temporary name, MyPass.

Procedure

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification to which you want to add a match pass.
3. Add a match pass by using one of the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>To add a new match pass</td>
<td>1. Click Add Pass &gt; New Pass.</td>
</tr>
<tr>
<td></td>
<td>2. Select or create a folder in which to save the match pass.</td>
</tr>
<tr>
<td></td>
<td>3. In the Item name field, enter a name for the new match pass and click Save.</td>
</tr>
<tr>
<td>To copy an existing match pass</td>
<td>1. Click Add Pass &gt; Copy Pass.</td>
</tr>
<tr>
<td></td>
<td>2. Select the match pass that you want to copy and click OK.</td>
</tr>
<tr>
<td></td>
<td>3. In the Save Match Pass Definition As window, select or create a folder in which to save the match pass.</td>
</tr>
<tr>
<td></td>
<td>4. In the Item name field, enter a name for the new match pass and click Save.</td>
</tr>
</tbody>
</table>

If you already have a pass in the match specification, the new pass is displayed to its right. When the specification runs, the passes run in the order that they are displayed from left to right.
4. Optional: To change the order in which the passes run when a match job uses this specification, press Ctrl and click the pass to move it to the left or right.

Specifying blocking columns:

Blocking columns are used to create subsets or blocks of input data records that are likely to be associated. The records that have the same values in the blocking columns are compared to only one another. Blocks make the matching process faster and more efficient.

About this task

If you want exact matching, specify only the blocking columns. If you specify only the blocking columns, all pairs of records that contain the same values in the blocking columns are considered matches.

Procedure

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass for which you want to specify blocking columns. In the Blocking Columns pane, you can add, modify, or delete blocking columns.
3. To add a blocking column to the match pass, click **Add**.
4. In the Match Blocking Specification window, complete the following steps.
   a. In the **Name** field, enter a string to identify this blocking selection. If you leave this field blank, the name of the blocking specification will default to the name of the selected blocking column.
   b. Select a column in the Available Data Columns table.
   c. Optional: Right-click the selected column and click **Properties** to view the frequency information in the Column Details window.
   d. Select one of the following blocking comparisons.
      • Character Comparison. Use when the column is alphanumeric.
      • Numeric Comparison. Use when the column is only numeric.
   e. If you want to add more blocking columns, click **Apply**.
   f. When you are done adding blocking columns, click **OK**.
5. To modify a blocking column, complete the following steps.
   a. Select a column and click **Modify > Blocking Specification**.
   b. In the Match Blocking Specification window, change the name of the blocking column, select a different available data column, or change the blocking comparison.
   c. Click **OK**.
6. Optional: To enter the maximum allowable size for a block, click **Modify > Set Overflow Values**. Block overflow occurs when the number of records contained in a single block exceeds the overflow threshold for that match pass. The default threshold is 10,000 records, although this threshold can be set as high as 40,000. Records in overflow blocks are not matched and are categorized as residuals for the match pass. To determine if a block overflow occurred during a match pass test run, review the OVERFLOW blocks statistic in the Pass Statistics tab. If a block overflow occurred, investigate to understand why it happened. After you understand the problem, you can improve the blocking criteria to prevent block overflow. If the blocking criteria are appropriate, increase the overflow threshold.
7. Optional: To change the order in which the blocking columns run, select a column and click **Move Up** or **Move Down**. Reordering blocking columns does not affect match pass results. However, you might want to reorder blocking columns to emphasize the relative significance of certain data columns and improve the readability of your match pass definition.

**Adding match commands:**

Within match commands, you specify matching columns and match comparisons for two-source and one-source match specifications.

**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass for which you want to add match commands.
3. To add a match command to the match pass, click **Add**.
4. In the **Name** field, enter a name for the match command. If you leave this field blank, the name of the match command defaults to the names of the selected columns.
5. From the **Available Comparison Types** list, select a match comparison. Any additional parameters and modes required by the match comparison that you select show in the Command Options pane.
6. From the Available Data Columns table, select a matching column and click **>**. Repeat as necessary to supply the number of data columns that are required by the comparison that you selected.

**Note:** Although you can select the same columns for blocking and matching in the same match pass, it is not advisable unless you have specific goals for those selections.

7. For a two-source match specification, from the Available Reference Columns table, select a reference column and click **>**. Repeat as necessary to supply the number of reference columns that are required by the comparison that you selected.

8. In the **m-prob** field of the **Command Options** area, enter a value for the m probability. The default value is 0.9.

For very important columns, use 0.999. For unimportant columns, use 0.8. The higher that you set the m probability, the greater the penalty for values that do not match. You can think of the m probability in two ways:

- Reflects the error rate of a column. For example, if values in a column do not match 10% of the time, set **m-prob** to 0.9.
- Forces a column to be more important. For example, if you want a column to have a high penalty if the values do not match, set **m-prob** at a higher value such as 0.95. In that case, the higher value indicates that you think the values in that column are unlikely to be different. If the values are different, you want to give them a high penalty, which is reflected in a high disagreement weight.

9. In the **u-prob** field of the **Command Options** area, enter a value for the u probability. The default value is 0.01.

If you want to change the default value, the following list is a rough guide for deciding which u probability value to use:
For most data, use the default value of 0.01. The matching process computes an accurate probability by using the frequency data that contains information from all the values in a column.

For age, use 0.02.

For gender, use 0.5. The probability that the gender variable agrees at random is about 0.5

10. Enter any required and optional parameters by following these steps.
   a. Optional: If your comparison choice allows reverse matching, select **Reverse** to assign the agreement weight when the columns disagree or the disagreement weight when the columns agree.
   b. Optional: Click **Vectors** to display and compare column vectors. **Columns** is selected by default.
   c. Optional: Click **Weight Overrides** to specify weight overrides.
   d. In the Command Options pane, enter or select required parameters.

11. If you want to add more match commands, click **Apply**.

12. If you are done adding match commands, click **OK**.

13. Optional: To change the order in which the match commands run, select a command and click **Move Up** or **Move Down**. Reordering match commands does not affect match pass results. However, you might want to reorder match commands to emphasize the relative significance of certain data columns and improve the readability of your match pass definition.

**Vectors in the Match Command window:**

You can select **Vectors** in the Match Command window to compare vectors on the data source to vectors on the reference source for a Two-source Match or compare vectors on the data source for a One-source Match.

To create vectors for use in the Match Designer, refer to the IBM InfoSphere DataStage Server Job Developer's Guide for details on using the Make Vector stage.

Vector matching is available with some match comparisons. Comparisons that can be used for matching vectors are marked with an asterisk (*) in the list of match comparisons. If you select a match comparison that supports vector matching, select **Vectors**.

The weight for a vector comparison does not exceed the weight that results from a single column comparison. This limit keeps the weight for a vector comparison from dominating the weights for single column comparisons in the composite weight.

**Match comparisons:**

Match comparisons analyze the values in columns to calculate the contribution to the composite weight. From the more than 24 available comparisons, choose an exact or error-tolerant comparison that suits both your matching objectives and your data.

Comparisons marked with an equal sign (=) can be used for both one-source matches and two-source matches. All other comparisons apply only to two-source matches.
Comparisons that can also be used for matching vectors are marked with an asterisk (*). If you want to create vectors for use in the Match Designer, refer to the IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide for details on using the Make Vector stage.

Table 3. Match comparisons that apply to characters

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR* =</td>
<td>Compares data values on a character-by-character basis. This comparison is often used to catch spelling mistakes or inverted letters.</td>
</tr>
<tr>
<td>LR_CHAR</td>
<td>Compares place information from a data source with geocoding reference files by using a left-right character string comparison.</td>
</tr>
<tr>
<td>LR_UNCERT</td>
<td>Compares place information from a data source with geocoding reference files by using a left-right string comparison algorithm based on information theory principles.</td>
</tr>
<tr>
<td>UNCERT* =</td>
<td>Evaluates the similarity of two character strings by using an algorithm that is based on information theory principles.</td>
</tr>
</tbody>
</table>

Table 4. Match comparisons that apply to numbers

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS_DIFF* =</td>
<td>Compares the absolute difference between two numbers to a value that you specify.</td>
</tr>
<tr>
<td>CNT_DIFF* =</td>
<td>Compares two strings of numbers and assigns agreement or disagreement weights based on the number of differences between the numbers in the strings. Weights are prorated according to the magnitude of the disagreement.</td>
</tr>
<tr>
<td>DATE8* =</td>
<td>Compares dates in the format of YYYYMMDD by measuring the number of days difference between two dates. Weights are prorated according to the magnitude of the disagreement.</td>
</tr>
<tr>
<td>DELTA_PERCENT* =</td>
<td>Compares columns in which the difference is measured as a percentage of the value that is compared. One use for DELTA_PERCENTAGE is comparing age.</td>
</tr>
<tr>
<td>DISTANCE =</td>
<td>Computes the distance between two points and prorates the weight based on the distance between the points. You can use this comparison for matching geographic coordinates where the farther the points are from each other, the lesser the weight that is applied.</td>
</tr>
</tbody>
</table>
### Table 4. Match comparisons that apply to numbers (continued)

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC *</td>
<td>Converts two strings to integers and then performs a numeric comparison of the integers. When converting the strings to integers, leading and trailing spaces are ignored.</td>
</tr>
<tr>
<td>PRORATED *</td>
<td>Compares numeric columns and allows them to disagree by an absolute amount that you specify. Any difference between 0 and the specified amount receives a weight proportionally equal to that difference.</td>
</tr>
<tr>
<td>TIME *</td>
<td>Compares values in IBM InfoSphere DataStage and QualityStage time or character columns in the format of HHMM or HHMMSS. TIME assigns proportionate weights to time differences that fall between an exact match and the maximum difference that you allow.</td>
</tr>
</tbody>
</table>

### Table 5. Match comparisons that apply to strings

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULT_ALIGN</td>
<td>Scores the similarity of two sequences of terms. This comparison combines your knowledge of how similar the terms are, the order of the similar terms, and the proximity of the similar terms. You can use MULT_ALIGN to compare addresses where the sequences of terms are in different orders.</td>
</tr>
<tr>
<td>MULT_EXACT</td>
<td>Compares all words in one column of a record with all words in the same column of a second record.</td>
</tr>
<tr>
<td>MULT_RANGE</td>
<td>Compares a string in one column to a range of strings in another column. You can use MULT_RANGE to compare single house numbers to a list of house number ranges.</td>
</tr>
<tr>
<td>MULT_UNCERT</td>
<td>Compares all words in one column of a record with all words in the same column of a second record by using a string comparison algorithm based on information theory principles.</td>
</tr>
<tr>
<td>NAME_UNCERT *</td>
<td>Compares two strings. First, it right-truncates the longer string so that it contains the same number of characters as the shorter string. If that comparison is not an exact match, it evaluates the similarity of the strings by doing an UNCERT comparison. You can use NAME_UNCERT to compare given names, where one of the name strings is shorter than the other.</td>
</tr>
</tbody>
</table>
Table 5. Match comparisons that apply to strings (continued)

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREFIX =</td>
<td>Compares two strings on a character-by-character basis after right-truncating the longer string so that it contains the same number of characters as the shorter string.</td>
</tr>
</tbody>
</table>

Table 6. Match comparisons that apply to intervals

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN_DINT</td>
<td>Compares an alphanumeric string from a data source to two alphanumeric intervals from a reference source. You can use this comparison to compare house numbers with census, Etak, GDT DynaMap, postal code, or other files.</td>
</tr>
<tr>
<td>AN_INTERVAL</td>
<td>Compares a single number from a data source to an interval or range of numbers from a reference source by using an alphanumeric odd-even interval comparison.</td>
</tr>
<tr>
<td>D_INT</td>
<td>Compares a numeric string from a data source to two numeric intervals from a reference source. You can use this comparison to compare house numbers with census, Etak, GDT DynaMap, or postal code files.</td>
</tr>
<tr>
<td>D_USPS</td>
<td>Compares an alphanumeric house number from a data source to two alphanumeric house number intervals from a reference source by using a left-right interval comparison. Control columns indicating the odd-even parity of the reference intervals are required.</td>
</tr>
<tr>
<td>INT_TO_INT =</td>
<td>Compares an interval from a data source to an interval from a reference source. The results match if an interval in one file overlaps or is fully contained in an interval in another file.</td>
</tr>
<tr>
<td>INTERVAL_NOPAR</td>
<td>Compares a single number from a data source to an interval from a reference source. The single number must be within the interval (inclusive of the end points) to be considered a match. The odd-even parity of the single number does not need to agree with the parity of the beginning value of the interval.</td>
</tr>
<tr>
<td>INTERVAL_PARITY</td>
<td>Compares a single number from a data source to an interval from a reference source. The odd-even parity of the number must agree with the parity of the beginning value of the interval.</td>
</tr>
</tbody>
</table>
Table 6. Match comparisons that apply to intervals (continued)

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USPS</td>
<td>Compares an alphanumeric house number from a data source to two alphanumeric house number intervals from USPS ZIP Code files or other reference sources that can contain alphanumeric ranges. Odd-even parity control information such as the USPS ZIP+4 control column is required.</td>
</tr>
<tr>
<td>USPS_DINT</td>
<td>Compares an interval from a data source to two intervals from a reference source for columns that contain an address primary number. This match comparison can be used to compare information from a USPS ZIP+4 file to geographic reference files such as the Census Bureau TIGER file, GDT Dynamap files, or Etak MapBase files. Odd-even parity control information such as the USPS ZIP+4 control column is required.</td>
</tr>
<tr>
<td>USPS_INT</td>
<td>Compares an interval from a data source to an interval from a reference source for columns that contain address primary number. The results match if the interval in the data source overlaps any part of the interval from a reference source and the odd-even parity agrees. Parity control information such as the USPS ZIP+4 control column is required.</td>
</tr>
</tbody>
</table>

Reverse matching:

With some match comparisons, you can reverse the weights.

By default, the agreement weight is assigned whenever the columns agree and the disagreement weight is assigned whenever the columns disagree. But with the following match comparisons you can use reverse matching to assign the agreement weight whenever the columns disagree and the disagreement weights whenever the columns agree.

- CHAR
- CNT_DIFF
- DATE
- DELTA_PERCENT
- INT_TO_INT
- NUMERIC
- PREFIX
- PRORATED
- UNCERT

You invoke reverse matching in the Match Command window of the Match Designer. First select one of the appropriate match comparisons, then select an available column, and select Reverse.
For the comparison requiring arguments, such as PRORATED, the roles of the agreement weight and disagreement weight are reversed. For example, the full agreement weight is assigned if the columns are different to a degree greater than the parameter specified, and the full disagreement weight is assigned if the columns are equal.

ABS_DIFF comparison:

Compared the absolute difference between two numbers to a tolerance that you specify.

If the numbers differ by less than or equal to the value that you specify in the Param 1 field, a prorated weight is assigned. If the numbers differ by more than the value in the Param 1 field, the full disagreement weight is assigned.

Required Columns

The following data source and reference source columns are required:

- Data. A column from the data source that contains numeric values.
  
  You can use this match comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the IBM InfoSphere DataStage and QualityStage Parallel Job Developer's Guide for details on using the Make Vector stage.

- Reference. A column from the reference source that contains numeric values (only applies to a two-source match).

Required Parameters

One parameter is required and one parameter is optional.

- **Param 1.** The maximum absolute value difference that can be tolerated. If you specify a value for only the Param 1 parameter, the value can be tolerated for either reference source value greater than data source value or data source value greater than reference source value. If you specified both parameters, the value for the Param 1 parameter is the maximum value tolerated for reference source value greater than data source value.

- **Param 2.** (Optional) The maximum absolute value difference that can be tolerated when reference source value is less than data source value.

Example

Suppose that you want to compare age in two files. If you want a tolerance of up to 10 years, specify 10 for the Param 1 parameter. Each absolute value of the difference that is closer to 10 subtracts 1/11 of the weight range from the agreement weight. The weight range is the difference between the agreement weight and disagreement weight. This equation means that for a difference of one year, the agreement weight is penalized 1/11 of the weight range. For a difference of 10 years, the agreement weight is penalized 10/11 of the weight range. For a difference of 11 years, the agreement weight is penalized 11/11 of the weight range, which means that the full disagreement weight is assigned.

You would specify 5 for the Param 2 parameter if you want a five year tolerance when the reference source value is less than the data source value.
AN_DINT comparison:

Compares an alphanumeric string from a data source to two alphanumeric intervals from a reference source. You can use this comparison to compare house numbers with census, Etak, GDT DynaMap, postal code, or other files.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a Two-source Match stage job, create two dummy file inputs instead of files that contain frequency information.

Required Columns

The following data source and reference source columns are required:

- Data. A column from the data source that contains numeric values.
- Reference. The reference column that contains the beginning value of the first interval (such as the left side of the street) from the reference source.
- Reference. The reference column that contains the ending value of the first interval from the reference source.
- Reference. The reference column that contains the beginning value of the second interval (such as the right side of the street) from the reference source.
- Reference. The reference column that contains the ending value of the second interval from the reference source.

Example

A single house number, which might contain alpha characters, is compared to two intervals. One interval represents the left side of the street and the other represents the right side of the street.

For example, 123A is compared to the intervals 101-199 and 100-198. For a number to match to an interval, both the parity (odd/even) and the range must agree. This comparison causes a special flag to be set to indicate whether the left or the right interval matched.

The beginning number of an interval can be higher than the ending number and still match. Files can have a high address in the FROM column and a low address in the TO column. For example, 153 matches both the range 200-100 and the range 100-200.

AN_INTERVAL comparison:

Compares a single number from a data source to an interval or range of numbers from a reference source by using an alphanumeric odd-even interval comparison.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a Two-source Match stage job, create two dummy file inputs instead of files that contain frequency information.

Required Columns

The following data source and reference source columns are required:

- Data. A column from the data source that contains the numeric value.
• Reference. The reference column that contains the beginning value of the interval.
• Reference. The reference column that contains the ending value of the interval.

Example

The numbers for comparison can contain alphanumeric suffixes or prefixes. The number must agree in parity with the low range of the interval. For example, an interval such as 123A to 123C is valid and contains the numbers 123A, 123B, and 123C.

A single number on the data source is compared to an interval on the reference source. If the number on the data source is odd, the beginning number of the range on the reference source must also be odd to be considered a match. Similarly, if the number on the data source is even, the beginning number of the range on the reference source must be even to be considered a match.

Interval match comparisons are primarily used for geocoding applications, such as postal address matching. For example, suppose you are matching 123A Main St to the range 121 to 123C Main St. The single number on the data source must be within the interval, inclusive of the end points, to be considered a match.

The beginning number of the interval can be higher than the ending number and still match. The files have a high address in the FROM column and a low address in the TO column. For example, 153 matches both the range 200-100 and the range 100-200.

CHAR comparison:

Compares two strings on a character-by-character basis. This comparison is often used to catch spelling mistakes or inverted letters.

If one string is shorter than the other, CHAR comparison pads the shorter column with trailing blanks to match the length of the longer column. Any mismatched character causes the disagreement weight to be assigned.

Required Columns

The following data source and reference source columns are required:
• Data. The character string from the data source.
  You can use this comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide for details on using the Make Vector stage.
• Reference. The character string from the reference source (only applies to a two-source match).

CNT_DIFF comparison:

Compares two strings of numbers and assigns agreement or disagreement weights based on the number of differences between the numbers in the strings. Weights are prorated according to the magnitude of the disagreement.
**Required Columns**

The following data source and reference source columns are required:

- **Data.** The column that contains the number from the data source.
  
  You can use this comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the *IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide* for details on using the Make Vector stage.

- **Reference.** The column that contains the number from the reference source (only applies to a two-source match).

**Required Parameter**

The following parameter is required:

**Param 1.** Indicates the number of differences that will be tolerated before the entire disagreement weight is assigned.

**Example**

You can use the CNT_DIFF comparison to count keying errors in columns. Some of these keying errors can include dates, telephone numbers, file or record numbers, and national identity numbers. For example, you have the following birth dates appearing on both files, and you suspect that these numbers represent the same birth date with a data entry error on the sixth number:

19670301
19670801

**Example**

The full agreement weight is always assigned if no keying errors are found. If you specify 1 and one keying error is found, the weight assigned is calculated as follows:

\[
\text{agreement weight} = \frac{1}{2} (\text{agreement weight} + \text{disagreement weight})
\]

Two or more errors result in the disagreement weight. The disagreement weight is always a negative number. Thus, one error would yield a partial weight.

If you specify 2, the errors are divided into thirds. One error results in assigning the agreement weight minus 1/3 the weight range from agreement to disagreement. Two errors would receive the agreement weight minus 2/3 the weight range, and so on. Thus, the weights are prorated according to the seriousness of the disagreement.

**D_INT comparison:**

Compares a numeric string from a data source to two numeric intervals from a reference source. You can use this comparison to compare house numbers with census, Etak, GDT DynaMap, or postal code files.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a two-source match stage job, create two dummy file inputs instead of files that contain frequency information.
**Required Columns**

The following data source and reference source columns are required:

- **Data.** A column from the data source that contains numeric values.
- **Reference.** The reference column that contains the beginning value of the first interval (such as the left side of the street) from the reference source.
- **Reference.** The reference column that contains the ending value of the first interval from the reference source.
- **Reference.** The reference column that contains the beginning value of the second interval (such as the right side of the street) from the reference source.
- **Reference.** The reference column that contains the ending value of the second interval from the reference source.

**Required Mode**

A mode is required. Choose one of the following modes:

- **ZERO_VALID.** Indicates that a value of 0 in a Data or Reference column is a valid value. A blank value in a Reference column means that the range value that it represents is the same as the range value in the companion Reference column.
- **ZERO_NULL.** Indicates that a value of 0 in the Data column is missing data. A value of 0 or a blank value in the Reference column that represents the ending range means that the ending range value is the same as the beginning range value in the Reference column that represents the beginning range.

**D_USPS comparison:**

Compares an alphanumeric house number from a data source to two alphanumeric house number intervals from a reference source by using a left-right interval comparison. Control columns indicating the odd-even parity of the reference intervals are required.

The D_USPS comparison requires the column names for the house number (generally on the data source), two intervals for house number ranges on the reference source, and control columns that indicate the parity of the house number range.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a two-source match stage job, create two dummy file inputs instead of files that contain frequency information.

**Required columns**

The following data source and reference source columns are required:

- **Data.** A column from the data source that contains numeric or nonnumeric values.
- **Reference.** (1) The reference column that contains the beginning value of the first interval (such as the left side of the street) from the reference source.
- **Reference.** (2) The reference column that contains the ending value of the first interval from the reference source.
- **Reference.** (3) The reference column that contains the beginning value of the second interval (such as the right side of the street) from the reference source.
• Reference (4) The reference column that contains the ending value of the second interval from the reference source.
• Reference (Control1) The odd/even parity for the range defined with reference columns (1) and (2).
• Reference (Control2) The odd/even parity for the range defined with reference columns (3) and (4).

The control information from the USPS ZIP + 4 code is:
• O. The range represents only odd house numbers.
• E. The range represents only even house numbers.
• B. The range represents all numbers (both odd and even) in the interval.
• U. The parity of the range is unknown.

Example

A house number on the data source is first compared to the interval range defined with reference source columns (1) and (2). If the parity of house number agrees with the code defined with Control 1 and with the parity of the house number defined with reference source column (1), and the intervals overlap, it is considered a match. If not, the house number on the data source is next compared to the interval defined with reference source columns (3) and (4).

DATE8 comparison:

Compares dates in the format of YYYYMMDD by measuring the number of days difference between two dates. Weights are prorated according to the magnitude of the disagreement.

The IBM InfoSphere DataStage and QualityStage date fields and character date fields with the date format of YYYYMMDD are supported.

Required Columns

The following data source and reference source columns are required:
• Data. The date from the data source.
  You can use this comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide for details on using the Make Vector stage.
• Reference. The date from the reference source (only applies for a two-source match).

Note: An invalid date is treated as a missing value.

Required Parameters

At least one of the following parameters are required:
• **Param 1**. Indicates the maximum number of days difference that will be tolerated before the entire disagreement weight is assigned. If you specify a value for only the **Param 1** parameter, this difference is the number of days that can be tolerated for either reference source date greater than data source date, or data source date greater than reference source date.
• **Param 2.** (Optional) The maximum number of days difference that can be tolerated when reference source date is less than data source date.

**Example**

Suppose you are matching on birth date and specified a 1 for the **Param 1** parameter. The full agreement weight is always assigned if no difference is found. If you specify 1 and the birth dates differ by one day, the weight assigned is calculated as follows:

\[
\text{agreement weight} - \frac{1}{2} (\text{agreement weight} + \text{disagreement weight})
\]

Two or more days difference results in the disagreement weight. The disagreement weight is always a negative number. Thus, one day difference would yield a partial weight.

If you specify 2, the errors are divided into thirds. One day difference results in assigning the agreement weight minus 1/3 the weight range from agreement to disagreement. Two days difference would receive the agreement weight minus 2/3 the weight range, and so on. Thus, the weights are prorated according to the seriousness of the disagreement.

Suppose you want to match highway crashes to hospital admissions. A hospital admission cannot occur before the accident date to be related to the accident. You might specify a 1 for the **Param 1** parameter, which allows the admission date to be one day later (greater) than the crash date, and a 0 for the **Param 2** parameter, which does not allow an admission date earlier than the crash date.

**DELTA_PERCENT comparison:**

Compares columns in which the difference is measured as a percentage of the value that is compared. One use for DELTA_PERCENTAGE is comparing age.

**Required Columns**

The following data source and reference source columns are required:

- **Data.** The value from the data source.
  
  You can use this comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the *IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide* for details on using the Make Vector stage.

- **Reference.** The value from the reference source (only applies for a two-source match).

**Required Parameters**

One parameter is required and one parameter is optional.

- **Param 1.** The maximum percentage difference that can be tolerated. If you specify a value for only the **Param 1** parameter, this value is the percentage that can be tolerated for either reference source value greater than data source value or data source value greater than reference source value. If you specified both parameters, the value for the **Param 1** parameter is the maximum percentage tolerated for reference source value greater than data source value.

- **Param 2.** (Optional) The maximum percentage difference that can be tolerated when reference source value is less than data source value.
Example

A one-year difference for an 85 year-old is less significant than a one-year difference for a three year-old, but a 10% difference for each is more meaningful.

Suppose you are comparing age in two files. If you want a tolerance of a 10% difference in the values, specify 10 for the Param 1 parameter. A 1% difference subtracts 1/11 of the weight range (the difference between the agreement weight and disagreement weight) from the agreement weight. A 10% difference subtracts 10/11 of the difference in the weight range.

You would specify 5 for the Param 2 parameter if you want a 5% tolerance when the reference source value is less than the data source value.

DISTANCE comparison:

Computes the distance between two points and prorates the weight based on the distance between the points. You can use this comparison for matching geographic coordinates where the farther the points are from each other, the lesser the weight that is applied.

Note: The distance is calculated as a Pythagorean distance, which is the square root of the sum of the squares of the coordinates.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a Two-source Match stage job, create two dummy file inputs instead of files that contain frequency information.

Required Columns

The following data source and reference source columns are required:

- Data. The X coordinate from the data source.
- Data. The Y coordinate from the data source.
- Reference. The X coordinate from the reference source.
- Reference. The Y coordinate from the reference source.

Note: Coordinates must be positive or negative integers; decimal places are not permitted.

Required Parameter

The following parameter is required:

Param 1. The maximum distance to be tolerated.

Example

The distance is in the units of the coordinates. For example, if the coordinates are in thousandths of a degree, a maximum distance of 100 tolerates a distance of 0.1 degrees.

If the distance between the points is 0, the agreement weight is assigned. If the distance is 0.05 degrees, the midpoint between the agreement and disagreement
weight is assigned. If the distance is greater than 0.1 degree, the disagreement weight is assigned.

**INT_TO_INT comparison:**

Compares an interval from a data source to an interval from a reference source. The results match if an interval in one file overlaps or is fully contained in an interval in another file.

You might use this match comparison for comparing hospital admission dates to see if hospital stays are partially concurrent. In addition, you might use this match comparison for matching two geographic reference files containing ranges of addresses.

You can use this comparison with reverse matching.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a Two-source Match stage job, create two dummy file inputs instead of files that contain frequency information.

**Required Columns**

The following data source and reference source columns are required:

- **Data.** The data column that contains the beginning value of the interval.
- **Data.** The data column that contains the ending value of the interval.
- **Reference.** The reference column that contains the beginning value of the interval.
- **Reference.** The reference column that contains the ending value of the interval.

**Required Modes**

A mode is required. Choose one of the following modes:

- **ZERO_VALID.** Indicates that a value of 0 in a Data or Reference column is valid. A blank value in a Reference column means that the range value that it represents is the same as the range value in the companion Reference column. A blank value in a Data column means that the range value that it represents is the same as the range value in the companion Data column.

- **ZERO_NULL.** Indicates that a value of 0 in the Data column is missing data. A value of 0 or a blank value in the Reference column that represents the ending range means that the ending range value is the same as the beginning range value in the Reference column that represents the beginning range. A value of 0 or a blank value in the Data column that represents the ending range means that the ending range value is the same as the beginning range value in the Data column that represents the beginning range.

**Example**

The following example illustrates interval-to-interval comparisons.

Assume that the interval from the data source is 19931023 to 19931031.

The interval from the reference source matches or does not match depending on whether the interval falls within the data source interval.
• From 19931025 to 19931102, matches because 19931031 falls within the interval on the reference source
• 19930901 to 19931225, matches because the interval from the data source falls within the interval on the reference source
• 19930920 to 19931025, matches because 19931023 falls within the interval on the reference source
• 19931030 to 19940123, matches because 19931031 falls within the interval on the reference source
• 19930901 to 19930922, does not match because the interval from the data source does not overlap the interval on the reference source

**INTERVAL_NOPAR comparison:**

Compares a single number from a data source to an interval from a reference source. The single number must be within the interval (inclusive of the end points) to be considered a match. The odd-even parity of the single number need not agree with the parity of the beginning value of the interval.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a Two-source Match stage job, create two dummy file inputs instead of files that contain frequency information.

**Required Columns**

The following data source and reference source columns are required.

• Data. The number from the data source.
• Reference. The reference column that contains the beginning value of the interval.
• Reference. The reference column that contains the ending value of the interval.

**Required Mode**

A mode is required. Choose one of the following modes:

• ZERO_VALID. Indicates that a value of 0 in a Data or Reference column is a valid value. A blank value in a Reference column means that the range value that it represents is the same as the range value in the companion Reference column.
• ZERO_NULL. Indicates that a value of 0 in the Data column is missing data. A value of 0 or a blank value in the Reference column that represents the ending range means that the ending range value is the same as the beginning range value in the Reference column that represents the beginning range.

**Example**

The beginning value of the interval can be larger than the ending value. For example, if the number from the data source is 153, it matches the interval 200-100 and the interval 100-200.

**INTERVAL_PARITY comparison:**

Compares a single number from a data source to an interval from a reference source. The odd-even parity of the number must agree with the parity of the beginning value of the interval.
If the number on the data source is odd, the beginning range number from the reference source must also be odd to be considered a match. Similarly, if the number on the data source is even, the beginning range from the reference source must also be even to be considered a match.

This match comparison is used primarily for geocoding applications to compare a house number from the data source to an interval of addresses from the reference source. Reference sources such as ZIP code files have a single odd or even interval. With a parity check, you can specify different intervals for each side of the street.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a Two-source Match stage job, create two dummy file inputs instead of files that contain frequency information.

**Required Columns**

The following data source and reference source columns are required:

- Data. The number from the data source.
- Reference. The reference column that contains the beginning value of the interval.
- Reference. The reference column that contains the ending value of the interval.

**Required Mode**

A mode is required. Choose one of the following modes:

- ZERO_VALID. Indicates that a value of 0 in a Data or Reference column is a valid value. A blank value in a Reference column means that the range value that it represents is the same as the range value in the companion Reference column.
- ZERO_NULL. Indicates that a value of 0 in the Data column is missing data. A value of 0 or a blank value in the Reference column that represents the ending range means that the ending range value is the same as the beginning range value in the Reference column that represents the beginning range.

**Example**

The beginning value of the interval can be larger than the ending value. For example, if the number from the data source is 153, it matches the interval 199-101 and the interval 101-199.

*LR_CHAR comparison:*

Compares place information from a data source with geocoding reference files by using a left-right character string comparison.

A single column on the user data file must be matched to the two columns on the reference source on a character-by-character basis.

Census Bureau Tiger files and other geographic reference sources contain a left ZIP code and a right ZIP code, a left city code and a right city code. The left code applies if there was a match to the left address range interval and the right code applies if there was a match to the right address range.
Required Columns

The following data source and reference source columns are required.
- Data. The column from the data source.
- Reference. The left field (ZIP code, city, and so on) from the reference source.
- Reference. The right field (ZIP code, city, and so on) from the reference source.

Required Mode

A mode is required. Choose one of the following modes:
- EITHER. The contents of the data source column must match either of the reference source columns specified (or both) to receive the full agreement weight.
- BASED_PREV. Use the result of a previous D_INT comparison to decide which column to compare.

If you specify the EITHER mode, the data source column must match either of the reference source columns to receive an agreement weight. If you specified the BASED_PREV mode, the data source column must match to the first reference source column of a previous D_INT comparison or of a similar double interval comparison in which the data source matched to the left interval, or the data source column must match to the first reference source column of the previous D_INT in which the data source matched to the right interval. If neither the left nor the right interval agrees, the missing weight for the column is assigned.

**LR_UNCERT comparison:**

Compares place information from a data source with geocoding reference files by using a left-right string comparison algorithm based on information theory principles.

Census files and other geographic reference sources contain a left postal code and a right postal code, a left city code and a right city code, and so forth.

Required Columns

The following data source and reference source columns are required.
- Data. The column from the data source.
- Reference. The left column (city code for example) from the reference source.
- Reference. The right column (city code for example) from the reference source.

Required Parameter

The following parameter is required:

**Param 1.** The minimum threshold is a number between 0 and 900. The following guidelines explain how the number is interpreted.
- 900. The two strings are identical.
- 850. The two strings can be considered the same.
- 800. The two strings are probably the same.
- 750. The two strings are probably different.
- 700. The two strings are different.
A higher value for the Param 1 parameter causes the match to tolerate fewer differences than it would with a lower value for the Param 1 parameter.

The assigned weight is proportioned linearly between the agreement and disagreement weights. For example, if you specify 700 and the score is 700 or less, then the full disagreement weight is assigned. If the strings agree exactly, the full agreement weight is assigned.

For another example, suppose you specify a value of 850 for the Param 1 parameter, which means that the tolerance is relatively low. A score of 800 would get the full disagreement weight because it is lower than the parameter that you specified. Even though a score of 800 means that the strings are probably the same, you have established a low tolerance.

Required Mode

A mode is required. Choose one of the following modes:
- EITHER. The contents of the data source column must match either of the reference source columns specified (or both) to receive the full agreement weight.
- BASED_PREV. Use the result of a previous D_INT comparison to decide which column to compare.

If you specify the EITHER mode, the data source column must match either of the reference source columns to receive an agreement weight. If you specified the BASED_PREV mode, the data source column must match to the first reference source column of a previous D_INT comparison or of a similar double interval comparison in which the data source matched to the left interval, or the data source column must match to the first reference source column of the previous D_INT in which the data source matched to the right interval. If neither the left nor the right interval agrees, the missing weight for the column is assigned.

MULT_ALIGN comparison:

Scores the similarity of two sequences of terms. This comparison combines your knowledge of how similar the terms are, the order of the similar terms, and the proximity of the similar terms. You can use MULT_ALIGN to compare addresses where the sequences of terms are in different orders.

Three independent scores factor into the final score:
- Similarity of the terms
- Order of similar terms in their original sequence
- Proximity of similar terms in their original sequence

Required Columns

The following data source and reference source columns are required:
- Data. The character string from the data source.
- Reference. The character string from the reference source (only applies to a two-source match).
Parameters

The following three parameters control the relative importance that each of the three independent scores has to the final score. Assign the highest number to the score that is the most important to you. For example, if you enter a value of 200 for MatchMix, 100 for OrderMix, and 100 for CompactMix, that means that the similarity score is twice as important as the order score and proximity score. It also means that the order score and proximity score are equally important.

MatchMix
Enter a positive integer that represents the relative importance of the similarity score for all of the matched terms.

OrderMix
Enter a positive integer that represents the relative importance of the order score for matched terms that score at or above the value that you enter for the FactorCutoff parameter.

CompactMix
Enter a positive integer that represents the relative importance of the proximity score for matched terms that score at or above the value that you enter for the FactorCutoff parameter.

The following parameters control the similarity score:

MatchParm
Enter a positive integer from 0-900 which represents the weight that is used by the UNCERT match comparison to determine its tolerance of errors. This parameter is an indication of the tolerance of the comparison. Higher numbers mean that the comparison is less tolerant of differences in the strings. MatchParm is similar to the Param 1 parameter for the UNCERT comparison. Use these values as a rough guideline:
- 900. The two strings must be identical.
- 850. The two strings can be safely considered to be the same.
- 800. The two strings are probably the same.
- 750. The two strings are probably different.
- 700. The two strings are almost certainly different.

The assigned weight is proportioned linearly between the agreement and disagreement weights. For example, if you specify 700 and the score is 700 or less, then the full disagreement weight is assigned. If the strings agree exactly, the full agreement weight is assigned.

As another example, suppose you specify 850 for the MatchParm, which means that the tolerance is relatively low. A score of 800 would get the full disagreement weight because it is lower than the parameter that you specified. Even though a score of 800 means that the strings are probably the same, you require a low tolerance.

MultType
Select one of the following values that determines how you want the match to normalize the score for two sequences of terms when the sequences do not contain the same number of terms:
- 0 – Maximum number of words in the two sequences
- 1 – Minimum number of words in the two sequences
- 2 – Number of words in the first sequence
- 3 – Number of words in the second sequence
• 6 – Minimum number of words plus \( x \), where \( x \) is the result of the ExtraTerms computation.

**ExtraTerm**

When the MultType value is 6, enter a positive integer for the percent of the difference between the greater and lesser of the two word counts to add to the minimum word count. An ExtraTerm value of 0 is equivalent to a MultType value of 1. An ExtraTerm value of 100 is equivalent to a MultType value of 0.

**MatchRange**

Enter a positive integer for the percent of the number of terms in the longer of the two sequences (percentage of the maximum word count). The resulting number of terms establishes a comparison radius that determines how different the position of two terms in their respective sequences can be and still be compared. For example, if the longer sequence contains 20 terms and you enter 50 for the MatchRange parameter, the match compares only the terms that are within 10 positions of each other.

**OutOfRangeScore**

Enter a positive integer for the percent of the default or rare value disagreement weight that is used to calculate a missing term weight. All terms in the shorter sequence must be scored against something. If all of the terms in the longer sequence that are within the range that is determined by the MatchRange parameter are paired with other terms, the value of the OutOfRangeScore parameter is used as the score for the unpaired terms.

This parameter controls which pairs of matched terms are used in the calculations of the order and proximity scores:

**FactorCutoff**

Enter a positive integer for the percent of the default or rare value agreement weight that is used to set a cutoff point for matched terms that are scored for order and proximity. Setting a cutoff score eliminates marginally positive and negative scores because those terms are really not matching. For example, for a FactorCutoff of 33, the lowest-scoring third of the term pairs will not be scored for order and proximity.

The following parameter controls the order score:

**OrderParm**

The value of this parameter determines the order score tolerance for errors. Enter a positive integer for the percent of the difference between the default agreement and disagreement weights that is used to penalize each out-of-order matched term. A lower number translates to more tolerance and a higher number translates to less tolerance.

The following parameters control the proximity score:

**GapOpen**

Enter a positive integer for the percent of the default or rare value agreement weight that is used to determine the proximity score penalty for the occurrence of each gap between matched terms.

**GapExtend**

Enter a positive integer for the percent of the default or rare value agreement weight that is used to determine the proximity score penalty for each additional space in a gap.
Example

The following examples illustrate how term order and term proximity are scored.

In the first example, the order score is higher for the first pair because all matched terms are in the same order.

Apartment 4-B Building 5
Apartment 4-B Building 5
Building 5 Apartment 4-B
Apartment 4-B Building 5

In the next example, the proximity score is higher for the first pair of terms because the second pair has a term that interrupts the sequence of matched terms.

Building 5 Apartment 4-B
Apartment 4-B Building 5
Building 5 Apartment 4-B
Apartment 4-B Upstairs Building 5

MULT_EXACT comparison:

Compares all words in one column of a record with all words in the same column of a second record.

Required Columns

The following data source and reference source columns are required.

- Data. The character string from the data source.
- Reference. The character string from the reference source.

Example

This comparison is similar to array matching, except that the individual words are considered to be the array elements. This match comparison allows matching of free-form text where the order of the words might not matter and where there might be missing words or words in error. The score is based on the similarity of the columns.

For example, the first address matches the second address if all words are inspected.

Building 5 Apartment 4-B
Apartment 4-B Building 5

MULT_RANGE comparison:

The MULT_RANGE comparison matches a single house number to a list of house number ranges.

Required Columns

The following data source and reference source columns are required.

- Data. The character string from the data source.
- Reference. The character string from the reference source.
Example

Each range must be separated by a pipe symbol (|). The tilde (~) is used to indicate the ranges, since the hyphen might be a legitimate address suffix (123-A). The prefix "B:" can be used to signify both odd and even numbers in the range. Otherwise, the parity of the low number is used.

101~199 | B:201~299|456|670 ½| 800-A~898-B|1000~

The following ranges result from the previous example.
- 101 to 199 odd numbers only
- 201 to 299 both odd and even number
- 456 (the one house number only)
- 670 ½ (the one house number only)
- 800-A to 898-B even numbers only
- All even house numbers 1000 or greater.

MULT_UNCERT comparison:

Compares all words in one column of a record with all words in the same column of a second record by using a string comparison algorithm based on information theory principles.

Required Columns

The following data source and reference source columns are required:
- Data. The character string from the data source.
- Reference. The character string from the reference source.

Required Parameter

The following parameter is required:

Param 1. The cutoff threshold is a number between 0 and 900.
- 900. The two strings are identical.
- 850. The two strings can be safely considered to be the same.
- 800. The two strings are probably the same.
- 750. The two strings are probably different.
- 700. The two strings are almost certainly different.

A higher Param 1 value causes the match to tolerate fewer differences than it would with a lower Param 1 value.

Example

The assigned weight is proportioned linearly between the agreement and disagreement weights. For example, if you specify 700 and the score is 700 or less, then the full disagreement weight is assigned. If the strings agree exactly, the full agreement weight is assigned.

As another example, suppose you specify 850 for the MatchParm, which means that the tolerance is relatively low. A score of 800 would get the full disagreement weight because it is lower than the parameter that you specified. Even though a score of 800 means that the strings are probably the same, you require a low tolerance.
Example

The following examples show that the MULT_UNCERT comparison is the best choice to match these addresses.

Building 5 Apartment 4B
Apartment 4-B Building 5

NAME_UNCERT comparison:

Compares two strings. First, it right-truncates the longer string so that it contains the same number of characters as the shorter string. If that comparison is not an exact match, it evaluates the similarity of the strings by doing an UNCERT comparison. You can use NAME_UNCERT to compare given names, where one of the name strings is shorter than the other.

NAME_UNCERT is a two-part comparison. First, it compares two strings on a character-by-character basis after truncating the longer string so that it contains the same number of characters as the shorter string. Second, if those strings are not an exact match, it evaluates the similarity of the strings by using an algorithm that is based on information theory principles.

Required Columns

The following data source and reference source columns are required:

- Data. The given name from the data source.

  You can use this comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the IBM InfoSphere DataStage and QualityStage Parallel Job Developer's Guide for details on using the Make Vector stage.

- Reference. The given name from the reference source (only applies for a two-source match).

Required Parameter

The following parameter is required:

- **Param 1**. The minimum threshold, which is a number 0 - 900. In other words, a higher value for the **Param 1** parameter causes the match to tolerate fewer differences than it would with a lower value for the **Param 1** parameter.

  - 900. The two strings are identical.
  - 850. The two strings can be considered the same.
  - 800. The two strings are probably the same.
  - 750. The two strings are probably different.
  - 700. The two strings are different.

Example

The assigned weight is proportioned linearly between the agreement and disagreement weights. For example, if you specify 700 and the score is 700 or less, then the full disagreement weight is assigned. If the strings agree exactly, the full agreement weight is assigned.

Suppose you specify 850 for the MatchParm, which means that the tolerance is relatively low. A score of 800 would get the full disagreement weight because it is
lower than the parameter that you specified. Even though a score of 800 means that the strings are probably the same, you require a low tolerance.

Example

NAME_UNCERT uses the shorter length of the two names for the comparison and does not compare any characters after that length.

For example, the following two sets of givens names would be considered exact matches:

AL  ALBERT
W  WILLIAM

This result is different from the CHAR comparison, where these two names would not match. With NAME_UNCERT, length is computed by ignoring trailing blanks (spaces). Embedded blanks are not ignored.

These two sets of names would not match for the UNCERT comparison either. UNCERT factors in variables such as the number of deletions between the strings which would probably result in an assignment of the full disagreement weight.

**NUMERIC comparison:**

Converts two strings to integers and then performs a numeric comparison of the integers. When converting the strings to integers, leading and trailing spaces are ignored.

**Required Columns**

The following data source and reference source columns are required.

- **Data.** Column from the data source.
  
  You can use this comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the *IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide* for details on using the Make Vector stage.

- **Reference.** Column from the reference source (only applies for a two-source match).

**PREFIX comparison:**

Compares two strings on a character-by-character basis after right-truncating the longer string so that it contains the same number of characters as the shorter string.

**Required Columns**

The following data source and reference source columns are required.

- **Data.** The string from the data source.
  
  You can use this comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the *IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide* for details on using the Make Vector stage.

- **Reference.** The string from the reference source (only applies for a two-source match).
Example

This comparison uses the shorter length of the two strings for the comparison and does not compare any characters after that length.

For example, a last name of ABECROMBY might be truncated to ABECROM. The PREFIX comparison considers these two representations to be an equal match. This comparison is different from the CHAR comparison, where these two names do not match. The length is computed by ignoring trailing blanks (spaces). Embedded blanks are not ignored.

PRORATED comparison:

Compares numeric columns and allows them to disagree by an absolute amount that you specify. Any difference between 0 and the specified amount receives a weight proportionally equal to that difference.

A difference of zero between the two columns results in the full agreement weight being assigned. A difference greater than the absolute amount results in the disagreement weight being assigned. Any difference between zero and the specified absolute amounts receives a weight proportionally equal to the difference.

Required Columns

The following data source and reference source columns are required:

• Data. The numeric column from the data source.

  You can use this comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide for details on using the Make Vector stage.

• Reference. The numeric column from the reference source (only applies for a two-source match).

Required Parameters

The following parameters are required:

Parameter 1 is required, Parameter 2 is optional:

• Param 1. The maximum absolute value difference that can be tolerated. If you specify only Param 1, that value is the difference that can be tolerated for either reference source value greater than data source value or data source value greater than reference source value.

  If you specify both parameters, Param 1 is the difference that can be tolerated for reference source value greater than data source value.

• Param 2. (Optional) The absolute value difference that can be tolerated when reference source value is less than data source value.

Example

Small differences between the columns receive slightly less than the full agreement weight. Large differences receive weights closer to the disagreement weight.

Suppose that you specify 15 for Param 1. Suppose the value from the reference source is greater than the value from the data source by a difference of 18. The comparison receives the full disagreement weight. Suppose the value from the
reference source is greater than the value from the data source by a difference of 8. In that case, the comparison receives a weight exactly between the agreement and disagreement weight.

**Example**

Suppose you are comparing two dates and specify 5 for **Param 1** and 7 for **Param 2**. The reference source value can exceed the data source value by five days, but the data source value can exceed the reference source value by seven days.

**TIME comparison:**

Compares values in IBM InfoSphere DataStage and QualityStage time or character columns in the format of HHMM or HHMMSS. TIME assigns proportionate weights to time differences that fall between an exact match and the maximum difference that you allow.

Time values must be specified in 24-hour format, where 0 is midnight and 2359 (HHMM) is 11:59 PM or 235959 (HHMMSS) is 11:59:59 PM. Times can cross midnight since the difference is always the shortest way around the clock. You can specify an acceptable maximum time difference in minutes.

**Required Columns**

The following data source and reference source columns are required:

- Data. The time from data source.
  
  You can use this comparison with vectors and reverse matching. To create vectors for use in the Match Designer, refer to the *IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide* for details on using the Make Vector stage.

- Reference. The time from reference source (only applies for a two-source match).

**Required Parameters**

Parameter 1 is required, Parameter 2 is optional:

- **Param 1.** The maximum time difference that can be tolerated. If you specify a value for only the **Param 1** parameter, this value is the difference that can be tolerated for either data source time greater than reference source time or reference source time greater than data source time. If you specify both parameters, the value for the **Param 1** parameter is the difference that can be tolerated for reference source time greater than data source time.

- **Param 2.** (Optional) The maximum time difference that can be tolerated when reference source time is less than data source time.

**Example**

A difference of zero between the two times results in the full agreement weight being assigned. A difference greater than the absolute amount results in the disagreement weight being assigned. Any difference between zero and the specified maximum time difference receives a weight proportionally equal to the difference.

Suppose you specify 10 for the **Param 1** parameter. If the times differ by 12 minutes, the comparison receives the full disagreement weight. If the times differ
by 5 minutes, the comparison receives a weight between the agreement and
disagreement weight.

Example

If you want the tolerances to be different for the data and reference sources, use
both parameters. Suppose you specify 20 for the **Param 1** parameter and 14 for the
**Param 2** parameter. In that case, the reference source value can exceed the data
source value by 20 minutes, but the data source value can exceed the reference
source value by 14 minutes.

**UNCERT comparison:**

Evaluates the similarity of two character strings by using an algorithm that is
based on information theory principles.

The weight assigned is based on the difference between the two strings being
compared as a function of the string length, the number of transpositions, and the
number of unassigned insertions, deletions, or replacement of characters. String
length is an important consideration because longer words can tolerate more errors
than shorter words can. In other words, you have a better chance of understanding
a longer word if it has the same number of errors as a shorter word.

**Note:** If you used the UNCERT match comparison with IBM InfoSphere
QualityStage Version 8.0, or earlier, and you want to retain that behavior, set the
QSM_MAT_UNCERT_VERSION environment variable. Setting
QSM_MAT_UNCERT_VERSION ensures that the UNCERT comparison behaves
consistently with the version of InfoSphere QualityStage that is reflected in the
value that you specify for the environment variable. The values are:

- 7.5 for Version 7.5 behavior
- 8.0 for Version 8.0 and 8.1 behavior

Refer to the *IBM InfoSphere DataStage and QualityStage Designer Client Guide* for
details on environment variables and specifying a job parameter for parallel jobs.

**Required Columns**

The following data source and reference source columns are required:

- **Data.** The character string from the data source.
  
  You can use this comparison with vectors and reverse matching. To create
  vectors for use in the Match Designer, refer to the *IBM InfoSphere DataStage and
  QualityStage Parallel Job Developer’s Guide* for details on using the Make Vector
  stage.

- **Reference.** The character string from the reference source (only applies for a
two-source match).

**Required Parameter**

The following parameter is required:

**Param 1.** The cutoff threshold, which is a number 0 - 900.

- 900. The two strings are identical.
- 850. The two strings can be safely considered to be the same.
- 800. The two strings are probably the same.
750. The two strings are probably different.
700. The two strings are almost certainly different.

A higher value for the Param 1 parameter causes the match to tolerate fewer differences than it would with a lower value for the Param 1 parameter.

Example

The assigned weight is proportioned linearly between the agreement and disagreement weights. For example, if you specify 700 and the score is 700 or less, then the full disagreement weight is assigned. If the strings agree exactly, the full agreement weight is assigned.

As another example, suppose you specify 850 for the MatchParm, which means that the tolerance is relatively low. A score of 800 would get the full disagreement weight because it is lower than the parameter that you specified. Even though a score of 800 means that the strings are probably the same, you require a low tolerance.

Adding QSM_MAT_UNCERT_VERSION environment variable:

If you used the UNCERT match comparison with IBM InfoSphere QualityStage version 8.0, or earlier, and you want to retain that behavior, set the QSM_MAT_UNCERT_VERSION environment variable. Setting QSM_MAT_UNCERT_VERSION ensures that the UNCERT comparison behaves consistently with the version of InfoSphere QualityStage that is reflected in the value that you specify for the environment variable.

About this task

Refer to the IBM InfoSphere DataStage and QualityStage Designer Client Guide for details on environment variables and specifying a job parameter for parallel jobs.

Procedure

1. Open your job in the Designer client.
2. Select Edit > Job Properties and click the Parameters tab.
3. Click Add Environment Variable and select QSM_MAT_UNCERT_VERSION.
4. Enter the value that corresponds to the behavior that you want to retain:
   - 7.5 for Version 7.5 behavior
   - 8.0 for Version 8.0 and 8.1 behavior
5. Click OK.

USPS comparison:

Compares an alphanumeric house number from a data source to two alphanumeric house number intervals from USPS ZIP Code files or other reference sources that can contain alphanumeric ranges. Odd-even parity control information such as the USPS ZIP+4 control column is required.

The USPS comparison requires that the data source contains the column names for the house number and the reference source contains a low house number range, a high house number range, and a control column, indicating the parity of the house number range.
Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a Two-source Match stage job, create two dummy file inputs instead of files that contain frequency information.

**Required Columns**

The following data source and reference source columns are required:

- **Data.** The house number from the data source.
- **Reference. (1)** The ZIP+4 column primary low house number for the beginning of the range from the reference source.
- **Reference. (2)** The ZIP+4 column primary high house number for the ending of the range from the reference source.
- **Reference. (Control)** The odd/even parity for the range defined with (1) and (2).

The control information from the USPS ZIP+4 code is:

- **O.** The range represents only odd house numbers.
- **E.** The range represents only even house numbers.
- **B.** The range represents all numbers (both odd and even) in the interval.
- **U.** The parity of the range is unknown.

**USPS_DINT comparison:**

Compares an interval from a data source to two intervals from a reference source for columns that contain an address primary number.

This match comparison can be used to compare information from a USPS ZIP+4 file to geographic reference files such as the Census Bureau TIGER file, GDT Dynamap files, or Etak MapBase files. Odd-even parity control information such as the USPS ZIP+4 control column is required.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a Two-source Match stage job, create two dummy file inputs instead of files that contain frequency information.

**Required Columns**

The data source requires an address primary low number, an address primary high number, and an address primary odd/even control. The USPS ZIP Code file contains this information. The reference source requires two primary low numbers, two primary high numbers, and two primary odd/even controls, one for each side of the street.

The following data source and reference source columns are required:

- **Data. (1)** The beginning of the street address range from the data source.
- **Data. (2)** The ending of the street address range from the data source.
- **Reference. (3)** The beginning of the street address range for one side of the street (such as from left) from the reference source.
- **Reference. (4)** The ending of the street address range for one side of the street (such as from left) from the reference source.
- **Reference. (5)** The beginning of the street address range for the other side of the street (such as from right) from the reference source.
• Reference. (6) The ending of the street address range for the other side of the street (such as to right) from the reference source.
• Data. (Control) The odd/even parity for the range defined with (1) and (2).
• Reference. (Control) The odd/even parity for the range defined with (3) and (4).
• Reference. (Control) The odd/even parity for the range defined with (5) and (6).

The control information from the USPS ZIP+4 code is:
• O. The range represents only odd house numbers.
• E. The range represents only even house numbers.
• B. The range represents all numbers (both odd and even) in the interval.
• U. The parity of the range is unknown.

How It Works

Agreement weight is assigned when:
• The odd/even control is set to E, O, or B on both the data source and the reference source
• The odd/even control is set to E or O on one source and to B on the other source (such as E on the data source and B on the reference source)

Disagreement weight is assigned when the parity is on one source is set to E or O and on the other source is set to the opposite; that is, either the data source to E and the reference source to O or the data source to O and the reference source to E.

If all strings are numeric, the comparison performs an integer interval comparison; otherwise, the comparison performs an alphanumeric interval comparison.

The interval on the data source is first compared to the first interval defined with reference (3) and reference (4). If the odd/even parity agrees, that is, if the data source control matches control (1) or control (2), and the intervals overlap; the intervals are considered a match.

In the table, the data source interval matches the interval on the reference source defined by reference (3) and reference (4) and because the odd/even parity is compatible (odd on the data source and both on the reference source), and the interval 101-199 overlaps with 123-299.

<table>
<thead>
<tr>
<th>Source</th>
<th>Begin range</th>
<th>End Range</th>
<th>Odd/Even Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>data interval (1) and (20)</td>
<td>101</td>
<td>199</td>
<td>O</td>
</tr>
<tr>
<td>reference interval (3) and (4)</td>
<td>123</td>
<td>299</td>
<td>B</td>
</tr>
<tr>
<td>reference interval (5) and (6)</td>
<td>124</td>
<td>298</td>
<td>B</td>
</tr>
</tbody>
</table>

If the interval on the data source does not match the first interval on the reference source, the data source interval is compared with the interval on the reference source defined by reference (5) and reference (6) for a match.
**USPS_INT comparison:**

Compares an interval from a data source to an interval from a reference source for columns that contain address primary number. The results match if the interval in the data source overlaps any part of the interval from a reference source and the odd-even parity agrees. Parity control information such as the USPS ZIP+4 control column is required.

Frequency information is not taken into account when this match comparison is used but a two-source match requires four input streams. If you use this match comparison with a Two-source Match stage job, create two dummy file inputs instead of files that contain frequency information.

**Required Columns**

Both sources require an address primary low number, an address primary high number, and an address primary odd/even control.

The following data source and reference source columns are required.
- Data. The beginning of the street address range from the data source.
- Data. The ending of the street address range from the data source.
- Reference. The beginning of the street address range from the reference source.
- Reference. The ending of the street address range from the reference source.
- Data. The odd/even parity for the data source.
- Reference. The odd/even parity for the reference source.

The control information from the USPS ZIP + 4 code is:
- O. The range represents only odd house numbers.
- E. The range represents only even house numbers.
- B. The range represents all numbers (both odd and even) in the interval.
- U. The parity of the range is unknown.

**How It Works**

Agreement weight is assigned when:
- The odd/even control is set to E, O, or B on both the data source and the reference source
- The odd/even control is set to E or O on one file and to B on the other file (such as E on the data source and B on the reference source)

Disagreement weight is assigned when the parity on one source is set to E or O and on the other source is set to the opposite; that is, either the data source to E and the reference source to O or the data source to O and the reference source to E.

If all strings are numeric, the comparison performs an integer interval comparison; otherwise, the comparison performs an alphanumeric interval comparison.

**Specifying weight overrides:**

If you need additional control over the way that weights are calculated, use weight overrides.
About this task

You can use weight overrides to independently control the weights for each match command.

When you work with weight overrides, be aware that the term reference record in the Weight Overrides window does not necessarily denote a reference record in a two-source match. Rather, the term refers to any candidate record that is compared to a master record to determine if they match. In this context, a reference record can be a record in the same file (a one-file match) or it can be a record in a different file (a two-file match).

When you use one of the following weight overrides for a one-source match specification, the values for Data Source and Reference Source are for the two records that are compared in the same source. If you specify a value for Data Source Missing Weight, also specify a value for Reference Source Missing Weight.

- Reference Source Missing Weight
- Both Missing Weight
- Conditional Reference Source Value

The override options are as follows:

Replace
Replaces the weight that is calculated for this match command with the weight that you specify.

Add
Adds the value that you specify to the weight that is calculated for this match command. You can reduce the calculated weight by adding a negative number.

Scale
Like Replace override, but Scale preserves probabilistic scoring.

You can use the Scale override option to control the relative weight contributions of each column or to help present scores in a more intuitive way. You can also use Scale to normalize scores from different passes so that they reflect the same range of values. The resulting scaled weights will range between the high and low values that you specify as the agreement and disagreement weights for the override. If either the agreement or disagreement weight is omitted, the default weight is applied.

To understand how Scale preserves probabilistic scoring, you need to understand the distinction between rare values and common values. Rare values are values that do not exist in the frequency data table that was generated by the Match Frequency stage. Common values are values that do exist in the frequency data table. Scale replaces the weight that is calculated for rare values with the weight that you specify. Scale replaces the weight that is calculated for common values with a weight that is scaled. The common value weight is scaled, so that after the rare value weight is replaced, the previous relationship between the values is preserved.

Procedure
1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass that contains the match command for which you want to specify a weight override.
3. From the Match Commands pane, select the match command and click Modify.

4. In the Match Command window, click Weight Overrides.

5. In the Compose Weight Override pane, select one of the override options. The override options are Replace, Add, or Scale.

6. Optional: Enter a value for the Conditional Data Source Value [AV] or the Conditional Reference Source Value [BV], or both, as shown in the following table. If you enter a value, it means that the weight overrides are conditional. They are applied only if a value in the column of the candidate record is equal to the value that you specify. If you select ALL or if the argument is missing, the weight overrides apply to all the non-null values in the column.

Attention: When conditional weight overrides are combined with non-conditional weight overrides, you must specify the conditional weight overrides first in the Summary of Weight Overrides table.

The weight overrides are tested in the order that they are listed in the Summary of Weight Overrides table. The first override that is satisfied is applied. Indicate override precedence by the order that you list the overrides in the table. In particular, ensure that overrides are listed in order of decreasing specificity, so that overrides that are more specific are tested first.

<table>
<thead>
<tr>
<th>Conditional weight override</th>
<th>In single quotation marks (&quot;), enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional Data Source Value [AV]</td>
<td>The value of the column in the data record for which you want the weight overrides to be applied. Alternately, select ALL.</td>
</tr>
<tr>
<td>Conditional Reference Source Value [BV]</td>
<td>The value of the column in the candidate record for which you want the weight overrides to be applied. Alternately, select ALL. The candidate record can be a record in the same file as the data record (a one-file match). Or, the candidate record can be a record in a different file (a two-source match).</td>
</tr>
</tbody>
</table>

7. Enter values for one or more of the other weight overrides.

<table>
<thead>
<tr>
<th>Weight override</th>
<th>Enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement Weight [AW]</td>
<td>An agreement weight if the values for the column agree and are not missing.</td>
</tr>
<tr>
<td>Disagreement Weight [DW]</td>
<td>A disagreement weight if the values for the column disagree and are not missing.</td>
</tr>
<tr>
<td>Data Source Missing Weight [AM]</td>
<td>A weight when the value on the data source is missing.</td>
</tr>
<tr>
<td>Reference Source Missing Weight [BM]</td>
<td>A weight when the value on the reference source is missing.</td>
</tr>
<tr>
<td>Both Missing Weight [XM]</td>
<td>A weight when values are missing on both sources.</td>
</tr>
</tbody>
</table>

8. Click Add Override. The override is displayed in the Summary of Weight Overrides table.

9. Optional: To remove an override, select the override and click Delete Override.

10. Click OK.
Weight override examples:

Weight overrides can be used to modify calculated weights or to calculate weights for missing values or specific combinations of values.

The following examples illustrate the benefits of changing the default weights that are assigned by a match command.

Addresses

You can add a weight of 1.0 to Both Missing Weight [XM], so that a missing value on both sources of data receives a slightly higher weight than a missing value on only one source. The result is that 123 Broadway Ave compared to 123 Broadway Ave. receives a higher weight than 123 Broadway Ave compared to 123 N Broadway Ave. The missing value is N.

<table>
<thead>
<tr>
<th>Original address</th>
<th>Matching address</th>
<th>Discarded address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123 Broadway Ave</td>
<td>123 Broadway Ave.</td>
<td>123 N Broadway Ave</td>
</tr>
</tbody>
</table>

Telephone numbers

For a one-source match, invalid telephone numbers must not be matched. Certain combinations of numbers take the form of a phone number but are not possible valid phone numbers. For example, the number 1111111111 is never a valid phone number.

For the telephone number 1111111111, you replace the calculated Agreement Weight [AW] with a -10 for a data source value of 1111111111.

Business location matches

When you deduplicate a business list by address, the suite number is important, since a high-rise office building can house many businesses. Thus, if the suite does not match, you want to penalize the match. If the suite is missing on one of two records, you cannot be sure that it refers to the same unit.

For example, you add a weight of -20 to the Disagreement Weight [DW] and weights of -9 to both the Data Source Missing Weight [AM] and the Reference Source Missing Weight [BM].

Nine is subtracted from the weight if the suite is missing on one of the records and 20 is subtracted from the weight if the suite number disagrees. If the suite number is missing from both records, it is probable that the location does not have suites or apartments.

Conditional overrides that force nonmatches

You can use a conditional weight override to force all records that contain a certain value into the nonmatched category. For example, your first match pass might be configured to match a Customer Code column. You want any record that contains a value of 1 in that column to not match any other records.

To accomplish this forced nonmatch, define two conditional overrides. The overrides replace the calculated Agreement Weight [AW] and Disagreement
Weight [DW] with a negative value such as -20. The weight override (-20) is low enough to ensure that the composite weight from the other fields is overwhelmed.

For the first override, set the Conditional Data Source Value [AV] value to 1 and the Conditional Reference Source Value [BV] value to ALL. For the second override, set the Conditional Reference Source Value [BV] value to 1 and the Conditional Data Source Value [AV] value to ALL.

The first override causes IBM InfoSphere QualityStage to assign a weight of -20 to any comparison where the Conditional Data Source Value [AV] record has a value of 1. In this case, the Conditional Reference Source Value [BV] value that is compared does not matter. In addition, it does not matter if the Conditional Reference Source Value [BV] value agrees or disagrees with 1.

Conversely, the second override causes InfoSphere QualityStage to assign a weight of -20 to any comparison where the Conditional Reference Source Value [BV] record has a value of 1. In this case, the Conditional Data Source Value [AV] value that is compared does not matter. In addition, it does not matter if the Conditional Data Source Value [AV] value agrees or disagrees with 1.

Viewing frequency information for selected columns:

In the Match Designer, the Column Details window displays frequency information and detailed information such as probabilities and weights for a selected column.

About this task

Understanding that a particular data value occurs more or less frequently can have an impact on how you tune your match. For instance, you might discover that an invalid value is common in your data. If you find such a value, you might want to provide a weight override to penalize the score for the match when that value occurs in a record.

Procedure

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass that contains the columns for which you want to view frequency information.
3. Select one of the following locations from which you want to view frequency information and perform the corresponding steps.

<table>
<thead>
<tr>
<th>Location</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Match Blocking Specification window</td>
<td>1. In the Blocking Columns pane, select a blocking column and click <strong>Modify &gt; Blocking Specification.</strong>&lt;br&gt;2. Right-click an available column and click <strong>Properties.</strong></td>
</tr>
<tr>
<td>The Match Command window</td>
<td>1. In the Match Commands pane, select a match command and click <strong>Modify.</strong>&lt;br&gt;2. Right-click an available column and click <strong>Properties.</strong></td>
</tr>
<tr>
<td>Location</td>
<td>Action</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| The column details in either the Blocking Columns pane or Match Commands pane | 1. Expand the blocking columns or the match commands.  
2. Right-click a data or reference column and click **Properties**.                                                                 |
| The Test Results data grid                                              | 1. In the data grid, right-click the heading of one of the matched columns.  
2. Click **Properties**.                                                                                                      |
| The Results Display window (only applicable for Two-source Match specifications) | 1. In the Pass Definition tab, select **Results Display**.  
2. Right-click a column and click **Properties**.                                                                                  |

The Column Details window opens. The column name, description, and type are shown along with a frequency distribution of the data values from this column.

4. To close the window, click **Cancel**.

*Defining variable special handling:*

You can assign one or more special handling flags to selected columns when they participate in match passes.

*About this task*

The following special handling actions are available.

- **CLERICAL**
  Use when you want a disagreement on the column to cause the record pair to be considered a clerical pair, even if the composite weight is above the match cutoff. For example, in motor vehicle accident matching, the records need to be reviewed if the county code does not match. If one or both values for the column are missing, the record pair is classified as a clerical pair.
  - **CLERICAL [MISSINGOK]**
    Use **MISSINGOK** if a missing value probably is not the cause of the record pair being considered to be forced into clerical review.

- **CRITICAL**
  Used when a disagreement on the column causes the record pair to automatically be considered a nonmatch. For example, if you never tolerate an error on birth date, the column can be made critical. If one or both values for the column are missing, the record pair is rejected.
  - **CRITICAL [MISSINGOK]**
    Use **MISSINGOK** if it is acceptable that one or both values are missing.

- **NOFREQ**
  Typically use when a column has high cardinality, such as a national identification number. **NOFREQ** indicates that no frequency analysis must be performed.

- **CONCAT**
  Use when you want to concatenate columns to form one frequency count. For example, health diagnostic codes can be highly correlated to age and gender. It is not likely that a 20-year old male will develop breast cancer. You can concatenate as many as four columns. When any one of these columns
participants in a match, the specified values are concatenated and looked up in the frequency table. Thus, the concatenated columns form one value in the table.

Special handling is set at the match specification level, which means that it is applicable to all match passes within a match specification.

Procedure
1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification that contains the column for which you want to define special handling.
3. Click Configure Specification > Variable Special Handling.
4. In the Define Special Handling area, select Data Columns or Reference Columns that corresponds to the location of the column for which you want to assign special handling.
   a. Select an available action.

   Note: With a two-source match specification, you can use only the NOFREQ and CONCAT special handling actions.

   b. From the list of available columns, select a column and click .
   c. If you selected CONCAT, repeat the previous step. You can concatenate as many as four columns.
   d. Click Add. The columns and the actions show in the Summary of Special Handling area.
5. Repeat the previous step until you add all the special handling that you need.
6. If you need to remove a special handling action from the Summary of Special Handling are, select the action and click Remove.
7. Click OK to close the window.

Assigning match pass cutoffs:

Cutoffs are thresholds that determine how the result of each record pair comparison is classified. The composite weight from each comparison then determines, relative to the cutoffs, whether that record pair is considered a match, a clerical pair, a duplicate, or a nonmatch.

About this task

Depending on the match type, you can set up to three of the following match cutoffs.

- **Match.** Record pairs with composite weights equal to or greater than this cutoff are considered matches.

- **Clerical.** Record pairs with composite weights that are less than the match cutoff but equal to or greater than the clerical cutoff can be reviewed to determine whether they are a match or a nonmatch. Record pairs with composite weights that are less than the clerical cutoff are considered nonmatches and participate in the following pass. You can eliminate the clerical review category by setting the clerical cutoff equal to the match cutoff. Also, this cutoff is not available with the one-source independent or one-source transitive match type.

- **Duplicate.** You can set a duplicate cutoff for use with only the two-source many-to-one-duplicate match type. This cutoff is optional and must be greater
than the match cutoff. Each reference source record in a record pair that has a composite weight which is equal to or greater than the duplicate cutoff is categorized as a duplicate record.

After you test the match pass, you can adjust cutoff values by using the alternative method of dragging the cutoff handles that are available in the frequency/weight histogram, which is part of the Test Results pane in the Pass Definition tab.

Procedure
1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass for which you want to assign cutoffs.
3. In the Cutoff Values pane, enter values in one or more of the following fields as appropriate.
   - Match
   - Clerical
   - Duplicate
4. In the Pass Definition tab, click Save Pass to save the selected match pass. You can also save the selected match pass or all match passes by clicking Save > Current Pass or Save > All Passes in the Compose tab.

Using the Match Pass Holding Area:

Use this area of the Match Designer to save versions of match pass definitions. It serves as a place to keep match passes while you experiment with different combinations of match passes or different definitions.

About this task

Passes in the holding area are not run as part of the match specification, but you can test and modify them on a standalone basis. You can move match pass definitions between the holding area and the match specification.

You might want to move a match pass to the holding area to assess the impact of its absence. Also, you might move a match pass to the holding area if you anticipate a need for it in the future.

You can open, modify, and test passes when they are in the holding area, and you can easily move them back into the flow of the match specification as required.

When you test all match passes in the Match Designer, anything in the match pass holding area is not run. Similarly, when you use the match specification in a Two-source Match stage, One-source Match stage, or Frequency Match stage within a job, passes in the Match Pass Holding Area are not run.

Also, when you test a match pass from the Match Pass Holding Area, the pass that is shown is considered as the first and only match pass.

Procedure
1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification for which you want to use the Match Pass Holding Area.
3. If the Match Pass Holding Area is not displayed in the Match Designer, click Show Pass Holding Area to display it.
4. Press Ctrl and click a pass to drag it to or from the Match Pass Holding Area. Alternately, right-click the pass and select either **Move to Holding Area** or **Return from Holding Area**.

5. To test a match pass in the holding area, select the pass and click one of the following options.
   a. Select the pass.
   b. In the **Pass Definition** tab, click the **Test Pass** arrow and select either **Test with output of previous pass** or **Test with sample input data set**.
   c. Click **Test Pass**.

**Testing results of match passes**

As you develop the match passes that make up a match specification, you run, review, and refine these match passes, by using the Match Designer. The match-pass testing process is often iterative that ensures that the match specification meets your matching objectives, before it is incorporated into a production job.

After you define one or more match passes and configure the test environment, you run test passes. Test results are displayed as statistics, data grids, charts, and other analytic tools. Evaluate these test results to decide if you need to modify cutoffs, blocking columns, or match criteria. Also, you might determine that additional match passes are needed.

For example, your plan might be to include a match pass that matches on national identification number followed by a match pass that matches on the date of the birth. When you review the results of the first match pass, you determine that your matching goals are achieved and that a second match pass is not needed. Alternatively, the results might indicate that you need to match on a different column for the second pass. Testing each match pass helps you choose the appropriate sequence of activities.

The following list includes some of the choices that you can make:

- Adjust the cutoff values by moving the appropriate cutoff handle in the histogram or by changing the cutoff values in the **Pass Definition** tab. Then test the pass again with the new cutoff values and with any other changes that you make.
- Test a pass by using the input data sample or the output of the previous match pass. Testing with the output of the previous pass lets you evaluate how effectively your passes work together in the match specification.
- Declare one test of the pass as a baseline and then review your subsequent tests against the baseline to see the impact of your changes on subsequent match pass results.
- You can run all active passes for the current match specification by clicking **Test All Passes** on the **Compose** tab. Alternatively, you can open an individual pass and run it by clicking **Test Pass** on the **Pass Definition** tab.

**Testing match passes**

After you define one or more match passes, you test those passes to determine how effectively they meet your matching goals. Also, you determine if they need to be adjusted and retested.
**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass that you want to test.
3. Click the **Test Pass** arrow and select one of the following options. For the one-source independent match type, only the Test with sample input data set option is available.
   - Test with output of previous pass
   - Test with sample input data set
4. Click **Test Pass**. The test runs. When it completes, the Test Results pane is displayed.
5. Optional: To change the view of the **Pass Definition** tab, select from the following options:

<table>
<thead>
<tr>
<th>Tab area</th>
<th>Action</th>
</tr>
</thead>
</table>
| Blocking Columns, Match Commands, and Cutoff Values panes | • To minimize, click the triangle above the Blocking Columns pane.  
  • To restore to the original size, click the triangle next to **Blocking Columns, Match Commands, Cutoffs** at the bottom of the tab. |
| Test Results pane | • To minimize, click the triangle next to **Test Results** at the top left of the Test Results pane.  
  • To restore to the original size, click the triangle next to **Data** at the bottom of the tab.  
  • To resize to the left, drag the left border of the Test Results pane.  
  • To resize to the top, drag the border of the Match Pass pane. |

6. To preserve the test data, click **Save Pass**.

**Evaluating test results**

The Match Designer provides information in various formats to help you evaluate test pass results. These formats include the test results data grid, charts, statistics, column-level metadata, and the Weight Comparison window. This information helps you assess the current test results and plan for subsequent match passes.

**Before you begin**

Test a match pass.

**About this task**

For example, after you test the match passes, the following process shows the steps that you might take to evaluate the test results:

1. Click the Test Results pane and the **Pass Statistics** tab to view the test results.
2. Click the **Total Statistics** tab to view the combined statistics for all the match passes.
3. In the Test Results pane, select a subset of records in the data grid and right-click **Compare Weights**. For those selected records, view the weight that the matching columns contribute to the composite match weight.
These areas of the Match Designer enable you to perform the following actions:

- Explore the match results
- View statistics about the test run
- Create charts
- Establish baseline results and compare previous test results to current test results
- View the column details such as agreement and disagreement weights by right-clicking a column
- View the weight that the matching columns contribute to the composite match weight

Procedure

1. To help determine cutoff values, sort the Test Results data grid in descending order by weight.
   a. Examine the sort results.
   b. If you begin to question the match results at a certain weight range, set the clerical cutoff to that weight.
2. If it seems that questionable matches are occurring at higher weight ranges, give the match columns higher m probabilities. Large m probabilities like 0.99999 are appropriate for very important columns. This value assigns a large penalty when values in this column disagree.
3. Right-click the Test Results data grid and include nonmatched records in the output display. If it seems that some of these records should have been classified as matches, consider lowering your match cutoff, clerical cutoff, or both.
4. Optional: After you evaluate the results, you might want to modify some elements of one or more passes, test the passes, and then evaluate them again.

Results

When you are satisfied with the results of all passes in a match specification, you are ready to use it in a One-source Match or Two-source Match stage.

Test Results pane:

This pane displays the histogram and the data grid. If you hover over the Test Results title bar, you see a tool tip that lists the number of records displayed, the grouping format and sort order you have chosen, and whether nonmatched records are included in the display.

Data grid:

The data grid shows the results of match passes that you test in the Pass Definition tab of the Match Designer.

The Test Results data grid displays records and columns of matched data. The data grid varies depending on whether the match passes were in a one-source match specification or a two-source match specification. By default, nonmatched records are not displayed, and records are sorted in ascending order by Match Set ID. You can sort the data grid based on a different column and display nonmatched records.
**Working with the data grid:**

By clicking different areas of the Test Results pane data grid, you can see more information or customize your view.

The following table shows the options that are available in the Test Results data grid. To view the data grid options, perform the actions that correspond to the option that you select.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
</table>
| Sort rows by one column or sort rows by multiple columns | • To sort rows by one column in ascending or descending order, right-click a column heading.  
• To sort rows by multiple columns, right-click the data grid and click **Sort by Multiple Columns**. |
| Clear rows | • Click each selected row to clear it.  
• Right-click the data grid and click **Clear All Selections**. |
| Identify the columns that participate in match commands, blocks, or user alignment columns  
**Note:** The column header for each blocking column, matching column, or user alignment is not the name of the participating columns. The heading is the name of the match command, blocking column, or user alignment that uses that column. | To view the name of the participating columns, hover over the column heading. |
| Display column metadata | • Right-click a column heading and click **Properties**.  
• For two-source match specifications, you can also choose between the data source and the reference source columns. |
<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display or exclude nonmatched records</td>
<td>To display nonmatched records for one-source match specifications, right-click the data grid and click one of the following items:</td>
</tr>
</tbody>
</table>
| **Note**: By default, nonmatched records are not displayed. | - Group by Match Pairs > Include Nonmatched Records  
- Group by Match Sets > Include Nonmatched Records                                                                                                                                                                                                                       |
| Nonmatched records are displayed at the end of the data grid. When you sort by a given column, nonmatched records are sorted but remain clustered at the end of the data grid. |                                                                                                                                                                                                                                                                                                                                     |
| To remove nonmatched records for one-source match specifications, right-click the data grid and click one of the following items: | - Group by Match Pairs > Exclude Nonmatched Records  
- Group by Match Sets > Exclude Nonmatched Records                                                                                                                                                                                                                     |
| Resize column display                       | Drag a column border.                                                                                                                                                                                                                                                                                                                |
| Reorder the display of columns              | Click a column heading and select a different column from the list. The selected column is displayed in the new position.                                                                                                                                                                                                           |
| Select one or more rows                     | - Click anywhere in the row.  
- To select multiple contiguous rows, press Shift and click the rows.  
- To select multiple noncontiguous rows, press Ctrl and click the rows.                                                                                                                                                                                             |
| Group matching records by match sets or match pairs | Right-click the data grid and click **Group by Match Sets** or **Group by Match Pairs**.                                                                                                                                                                                                                                             |
| Search the results grid                     | Right-click the data grid and click **Search Data**.                                                                                                                                                                                                                                                                                   |
| Compare weights for two or more records in the results grid | - Right-click the data grid and click **Compare Weights**.  
- To compare weights by using the match criteria that is in effect for the most recent run of the current match pass, select **Based on Last Match Run**.  
- To compare weights by using modified match criteria that was not tested yet, select **Based on Current Match Settings**.                                                                                           |

*Working with the Weight Comparison window:*

The Weight Comparison window displays the contribution of each matched column to the composite weight. In addition to records from existing match sets, you can compare records that are not considered matches based on the current match criteria.
By clicking different areas of the Weight Comparison window, you can see more information or customize your view.

The following table lists the options that are available in the Weight Comparison window.

**Note:** The initial presentation of weight comparison results is in descending order by composite weight.

Some nonmatched records show a default score of -99.99. They also show no column-level weight contributions, agreement weights, or disagreement weights.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define sort order by composite weight</td>
<td>Right-click the pass definition column (the first column).</td>
</tr>
<tr>
<td>Define sort order by column value or column weight</td>
<td>Right-click any column except the first one and click <em>Sort by column value</em> or <em>Sort by column weight</em>.</td>
</tr>
<tr>
<td>Display column properties</td>
<td>Right-click a column head and click <em>Properties</em>. Properties are available for all data and reference columns that participate in the current match pass.</td>
</tr>
</tbody>
</table>
| Freeze columns when you scroll horizontally | • Hover over the column splitter that is positioned initially between the first and second columns.  
   • When the pointer changes, drag the splitter to the right of the column that you want to freeze. |
| Reorder the display of columns              | Click a column heading and drag to a different position. The first column cannot be moved. |
| View the full value of a truncated cell     | Hover over the cell.                                                  |

**Searching test results data:**

By using the Search Data window, you can search for a specific data value in the columns of the Test Results data grid.

**Before you begin**

**Test a match pass.**

**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass that you want to search.
3. Right-click the data grid and click *Search Data*. Alternatively, right-click the data grid in a specific column to search only that column.
4. In the **String** field, enter the string that you want to search for.
5. From the **In** list, select *All Columns* or a specific column.
6. Click **Find**. The test results data grid scrolls automatically to the first record that contains the search string. You might need to scroll right or left to see the result.

7. Click **Find** to search for each additional instance of the specified value. When the search reaches the end of the data grid, you receive a message that indicates that the value could not be found.

8. To close the Search Data window, click **Cancel**.

**Using the Test Results data grid sort capabilities:**

The Test Results data grid provides various sorting and grouping options to help you evaluate match pass test results.

The following options are available:
- By default, the initial display is sorted in ascending order by the Set ID column.
- For one-source match specifications that are grouped by match sets, the sets are sorted by the value of the master record.
  - To group the master record and its duplicates together, right-click the data grid and click **Group by Match Sets**.
  - For one-source match specifications that are grouped by match pairs, the sets are sorted by the value of the duplicate records.
  - To display the master record for each set followed by a single duplicate, right-click the data grid and click **Group by Match Pairs**. (This action can display the same master record multiple times.)

  **Note:** If you want to scroll the data grid based on the position of the **Selected Data** handle in the Test Results histogram, select **Group by Match Pairs**.

- To group columns and sort on values within those columns, right-click the data grid and click **Sort by Multiple Columns**.
- To sort on multiple blocking columns and group sets of records compared against one another, right-click the data grid and click **Sort by Combined Blocking Columns**.
- Nonmatched records are sorted by the criteria that you choose, but they are always presented as a group at the bottom of the data grid.

**Aligning the test results display for two-source match specifications:**

In the Results Display window, you can customize your view of match pass test results for two-source match specifications. Customizing your view of columns in ways that are meaningful to you makes it easier to evaluate match results and tune the matching criteria.

**Before you begin**

[Test a match pass for a two-source match specification.](#)

**About this task**

When you test match passes, the matched pairs are displayed in the **Test Results** area of the **Pass Definition** tab.

You can add to this display by using the Results Display window to align additional combinations of data and reference columns. You can display values for one or more data columns and pair them with one or more reference columns.
The choices that you make in the Results Display window do not affect the execution of a match pass or match stage. Instead, the Results Display window simply creates visual links between data and reference columns that help you evaluate match results and tune your matching criteria.

**Restriction:** You cannot change or delete the default alignments for blocking and matching columns that are defined in the Results Display window.

**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the two-source match specification and select the match pass for which you want to align the test results display.
3. In the **Pass Definition** tab, click **Results Display**.
4. Add an alignment by using one of the following methods.
   - To automatically align all data and reference columns that have the same name and that were not used in a previous alignment, click **Auto-Align All**.
   - To manually align data and reference columns, perform the following steps.
     a. From the **Data Input Columns** list, select one or more columns.
     b. From the **Reference Input Columns** list, select one or more columns.
     c. Click **Add Alignment**. The selected pairing of data and reference columns is displayed in the Summary of Column Alignments pane, under the list of blocking columns and match command columns. Blocking columns are highlighted with a light shading. Match commands are highlighted with a darker shading.
     d. To manually add more alignments, repeat these steps.
5. To edit a selected alignment, click **Edit**. This action removes the alignment from the Summary of Column Alignments pane and selects its columns in the Define New Column Alignment pane.
6. To clear all alignments, click **Clear**.
7. To delete an alignment, click **Delete**.
8. To change the order of display of a selected alignment, click **Move Up** or **Move Down**. Changing the order of alignments changes the column order in the test results display but does not affect match pass test results.
9. To close the window, click **OK**.
10. To view these alignments in the Test Results pane, retest the modified match pass or click the match pass button to refresh the test results display.

**Results**

Values for each alignment that you create are displayed in **User Alignments** columns on the right side of the Test Results data grid.

**Viewing weight contributions of columns:**

When you evaluate the results of a test match pass, it is helpful to view the individual score that a match column contributes to the composite match weight of a record.

**Before you begin**

[Test a match pass]
About this task

By selecting records in the Test Results data grid, you can view the scores for each match column in the Weight Comparison window. You can also use the Weight Comparison window to explore potential changes in matching criteria and to compare records that are currently distributed across different match sets.

The first record that you select is always the designated master, which is the basis of the weight comparison. Each remaining record is categorized by the weight comparison process as a duplicate, clerical, nonmatched, or reference duplicate. The categorization is based on the composite weight of the record relative to the current cutoff settings.

Procedure

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass for which you want to view weight contributions of selected columns.
3. In the Test Results data grid, select at least two records according to one of the following options. Selected records can be contiguous or noncontiguous. Also, selected records can include records from multiple match sets.

<table>
<thead>
<tr>
<th>Match specification</th>
<th>Select</th>
</tr>
</thead>
</table>
| For a two-source match specification | 1. Any data record (record type MA or CA)  
2. One or more reference records (record types MB, DB, or CB) |
| For an one-source match specification | Any combination of records from any match sets (record types MP, DA, CP, or RA) |

4. Right-click in the data grid and click a comparison option. The Based on Last Match Run option applies the match criteria for the most recent run of the current match pass. The Based on Current Match Settings option uses any modified match criteria that you applied to the pass definition but did not test.

5. To close the Weight Comparison window, click OK.

Test Results data grid columns:

The columns in the Test Results data grid vary depending on whether you test match passes for a one-source match specification or a two-source match specification.

For a one-source match specification, the following columns are displayed in the following default order:

- **SetID.** A unique ID number for each set of matched records. Records are sorted initially in ascending order by the SetID column.
- **Record Type.** The status of a record.
  - The following record types are within a set of matched records:
    - **MP.** Master record
    - **DA.** Duplicate record
    - **CP.** Record that requires clerical review
  - **RA.** Nonmatched record
• **Weight.** The composite match weight, which is the sum of the individual column-level weights for all the match columns in a given record.
• **DataID.** A unique identification number that is assigned to each record.
• All columns that are used in match commands.
• All columns that are used in blocks.
• All other columns that are in the data source.

For a two-source match specification, the following columns are displayed in the following order:
• **SetID.** A unique ID number for each set of matched records. Records are sorted initially in ascending order by SetID.
• **Record Type.** The status of a record that is within a set of matched records. The record type is one of the following items:
  – **MA.** Data record
  – **MB.** Primary reference record
  – **DB.** Duplicate reference record
  – **CA.** Data record for a match set that requires clerical review
  – **CB.** Primary reference record for a match set that requires clerical review
  – **DB.** Duplicate reference record for a match set that requires clerical review
• **Weight.** The composite match weight, which is the sum of the individual column-level weights for all the match columns in a given record.
• **DataID.** A unique identification number that is assigned to each data record.
• **RefID.** A unique identification number that is assigned to each reference record.

**Note:** A reference record can match to more than one data record. As a result, multiple instances of the same RefID can be displayed in the Test Results data grid.

• **Match Command.** One column for each match command. Each column shows the value for the data source column that is paired with the value for each reference column in the match command.
• **Blocking Column.** One column for each blocking column created. Each column shows the value for the data source column that is paired with the value for the reference column.
• **User Alignment.** One column for each user-defined alignment that you create by using the Results Display window.

**Histogram:**

The Frequency/Weight histogram at the top of the Test Results pane is a graphical representation of the distribution of weights assigned by the run of a match pass.

**Using the Frequency/Weight histogram:**

The Match Designer histogram displays the distribution of the composite weights.

**Before you begin**

[Test a match pass.]
About this task

If you move the **Current Data** handle on the histogram to a new weight, it automatically scrolls the data grid to the location of the data with the selected weight. Likewise, if you reposition the selection within the data display, the **Current Data** handle is repositioned in the histogram.

Move the **Current Data** handle by either of the following actions:

- To display records of a certain weight, move the **Current Data** handle along the **Weight axis**. **Ascending by Weight Sort** moves the **Current Data** handle to the lowest detail weight value.

  **Note:** For one-source specifications, the **Current Data** handle is available only when the data display is in match pair order. To display in match pair order, right-click the data display and click **Group by Match Pairs**.

- To adjust the Clerical Cutoff or Match Cutoff settings, move the cutoff handle along the **Weight axis**. The changed cutoffs show in the Cutoff Values pane.

The following list contains some points to remember about cutoffs:

- The clerical cutoff is a composite weight above which record pairs are considered to be possible matches. Record pairs with weights between the match and the clerical cutoff are known as clericals and typically are reviewed to determine whether they are matches or nonmatches. If you do not want to review clericals, make the match cutoff weight equal to the clerical cutoff weight.

- Cutoff weights can be negative values, if you want. However, when you set cutoff weights to negative values, this setting creates extremely inclusive sets of matched records. The histogram displays the distribution of the composite weights. If you use negative values for cutoff weights, this histogram shows many values at highly negative weights, because most cases are nonmatched pairs. However, record pairs that are obvious disagreements are not a large part of the matching process, and thus, negative weights are not often shown.

- There is another large group of values at highly positive weights for the matched cases. The cutoff values for the match run can be set by inspecting this histogram. Make the clerical review cutoff the weight where the spike in the histogram reaches near the axis. Set the other cutoff weight where the nonmatched cases start to dominate. Experiment and examine the test results as a guide for setting the cutoffs.

- For a two-source many-to-one duplicate match type, there is an additional cutoff weight called a duplicate cutoff. This cutoff is optional. If you use the duplicate cutoff, set it higher than the match cutoff weight. If more than one record pair receives a composite weight that is higher than the match cutoff, these records are declared duplicates if their composite weight is equal to or greater than the duplicate cutoff.

**Procedure**

**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.

2. Open the match specification and select the match pass for which you want to use the histogram.

3. In the histogram, display the weight that you want to see by moving the **Current Data** handle or scrolling through the records in the data grid. The position and availability of the **Current Data** handle depends on the following factors:
One-source match specifications
   The Current Data handle is visible only if you group by match pairs, not by match sets. The handle is initially positioned based on the weight of the first duplicate record.

Two-source match specifications
   The initial position of the Current Data handle is determined by the weight of the first data record.

Both one-source and two-source match specifications
   If you scroll through the Test Results data grid, the Current Data handle is repositioned based on the selected record.

4. Inspect the data in the Test Results data grid to see if the matches are adequate.
5. Repeat the two previous steps to find the most appropriate match cutoff and clerical cutoff points.

   Note: Do not allow the clerical cutoff value to exceed the match cutoff value. If the clerical cutoff value exceeds the match cutoff value, you receive an error message and must cancel or reduce the clerical cutoff value.

6. Move the Match Cutoff and Clerical Cutoff handles to the values that you want, which are shown in the Cutoff Values pane.
7. Test the match pass again with the new cutoff settings.

Pass Statistics tab:

   This tab displays the statistics about the current test, the statistics about a baseline run (if you created one), and a graphical illustration of the pass statistics.

   Reviewing statistics for a match pass:

   The Match Designer Pass Statistics tab provides options that you can use to evaluate your match pass test results.

   Before you begin

   Test a match pass.

   About this task

   Statistics for each test run of the match pass are saved in the Pass Statistics tab.
   You can set one test run as the baseline run and compare subsequent runs to it. Additionally, you can test all the passes and compare their statistics in this tab.

   You can view the cumulative statistics for all passes in the Total Statistics tab.

   Procedure

   1. Open the IBM InfoSphere DataStage and QualityStage Designer.
   2. Open the match specification and select the match pass for which you want to view statistics.
   3. Click the Pass Statistics tab.
   4. To expand the Pass Statistics area, drag the top border of the Match Pass pane.
Creating and setting baseline test runs of match passes:

The baseline run provides a control test run of a match pass to which you can compare subsequent test runs.

**Before you begin**

Test a match pass.

**About this task**

The statistics for a match pass can be saved in the Pass Statistics tab. You can use the table to compare statistics for the most current match pass test run and a selected baseline run.

For each type of statistical data, the table in the Pass Statistics tab shows the following values:

- Current value
- Baseline value
- Difference between values

The Delta column displays arrows that indicate which statistics changed and whether their values increased or decreased.

**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass for which you want to create and set a baseline run.
3. Click the **Pass Statistics** tab.
4. Click **Save Current Statistics**. The statistics from the current match pass test are saved as a baseline run that you can access from the **Baseline Run** list.
5. Select a run from the Baseline Run list. You can select the baseline run that you created, or you can select a baseline run with statistics that were saved from a previous match pass test run.
6. If necessary, click **Refresh** to update the data in the table.

**Displaying graphs and charts of pass statistics:**

The Match Designer Pass Statistics pane shows data about the match pass in graphical format.

**Before you begin**

Test a match pass.

**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification and select the match pass for which you want to display graphs and charts.
3. Click the **Pass Statistics** tab.
4. In the Chart column of the table, select each type of statistic that you want to display.
5. Select a chart type, chart style, and click the **Refresh Chart** button.
Tip: If you select many types of statistics to display, the Pie chart type can become difficult to read. In such a case, select the With Small Sectors chart style.

6. Optional: Click Print Chart.

Total Statistics tab:

This tab displays the results of each match-pass test run and the combined results of all the match passes. If you run consecutive match passes, you can evaluate each match pass individually or evaluate the combined results.

Reviewing total statistics:

The Match Designer Total Statistics tab shows statistical data in a graphical format for all the passes that you run.

About this task

The cumulative statistics are valuable only if you test multiple passes consecutively in the order that they appear in the match specification. The Total Statistics tab displays the following information:

- Cumulative statistics for the current runs of all the match passes in the match specification
- Individual statistics for the current run of each pass
- Charts that compare the statistics for the current run of all the match passes

Remember: The Total Statistics tab does not display information about match passes in the Match Pass Holding Area.

Procedure

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the match specification for which you want to review total match pass statistics.
3. Click Test All Passes to run all active match passes in order, which uses the output of the one match pass as the input of the subsequent match pass.
4. Click Total Statistics.

Results

The Total Statistics table shows additive values for the current runs of all the match passes. The Match Pass tables show values for the current run of each match pass.

Displaying total statistics charts:

The Total Statistics tab shows a Total Statistics chart that displays information for the runs of all the match passes.

Before you begin

Test all active match passes and review total statistics.
Procedure
1. In the Chart column of the Cumulative Statistics table, select each statistic option that you want to display.
2. From the Chart By list, select one of the following options:
   - Statistic: Creates a chart that shows statistics for the options that you chose.
   - Pass: Creates a chart that shows statistics for the match passes.
3. Select a chart type, chart style, and click the Refresh Chart button.
   
   **Remember:** If you select many types of statistics to display, the Pie chart type can become difficult to read. In such a case, select the With Small Sectors chart style.
4. Optional: Click Print Chart.

**Reporting for match specifications**
To review the detailed contents of a match specification, create a match specification report. For example, you might create a report to decide which match specification to use for a match stage.

**Creating match specification reports**
After you create a match specification, you can create a report that describes all the active match passes for a project. Some contents of a match specification are blocking columns, matching columns, match comparisons, and cutoff values.

**Before you begin**
- Configure your Web browser to work with the IBM InfoSphere Information Server Web console. Refer to the IBM InfoSphere Information Server Administration Guide for more information about configuring your Web browser.
  - Create a match specification for a Two-source Match stage or a One-source Match stage.
  - Define the test environment.
  - Define match passes.

**About this task**
The match specification report describes all the active match passes, which are the passes that are not in the holding area. If you want to review all the passes in a match specification, ensure that the holding area does not contain any passes.

**Procedure**
1. Log in to the IBM InfoSphere Information Server Web console.
2. Click the Reporting tab.
5. Optional: In the Name field, replace the default name with a name that is more meaningful to you. For example, you might choose a name that includes the name of the match specification that you will select for this report.
6. Optional: In the Description field, replace the default description with a description that is more meaningful to you. For example, you might enter a summary of the match specification that includes the job that it is used for.
The **Save-in Folder** field contains the name of the default folder where the new report is stored on the server. You cannot specify a different folder when you create a report. When you view the report, you have the option to save it to the folder of your choice.

7. Configure the report settings.
   a. From the **Project** field, select the project that contains the match specification that you want to create this report for.
   b. In the **Match Specification** field, click **Retrieve Values** and select the match specification that you want to create this report for.
   c. Select a file format for the report.
   d. Optional: Select settings for the **Expiration** and **History Policy**.

8. Click **Finish** to select one of the following options: **Save**, **Run Now**, **Save and Run Now**, or **Save and Close**.

### Running match specification reports

After a report is created, you run the report. You cannot view the results that show you the contents of the selected match specification until you run the report.

#### Before you begin

**Create a match specification report**

#### About this task

One reason to run a report is to create the initial results that show the contents of the match specification. The other reason that you might want to run a report is to generate updated results for the selected match specification. If a match specification was changed after an initial report was run, you must run the report again to view changes to the match specification.

#### Procedure

1. Log in to the IBM InfoSphere Information Server Web console.
2. Click the **Reporting** tab.
3. In the Navigation pane, click **Reports > View Reports** and select the report that you want to run. If the report that you want to view was saved in a custom folder, select the folder where your report is saved.
4. Click **Run Now**.
5. Optional: In the **Report Settings > Parameters** pane, set undefined runtime parameters for the report. The report is queued for running. When the report starts running, the run date and run duration are displayed in the Running pane.
6. Optional: To refresh the run statistics such as the duration of the run, click **Refresh**.

#### Viewing match specification reports

After a report is run, you can view the report. You can view one report or any report that previously was run.

#### Before you begin

**Run a match specification report**
**Procedure**

1. Log in to the IBM InfoSphere Information Server Web console.
2. Click the Reporting tab.
3. In the Navigation pane, click Reports > View Reports and select the report that you want to view. If the report that you want to view was saved in a custom folder, select the folder where your report is saved.
4. Click View Report Result. The report opens in the format that you selected when you created the report.
5. Optional: Save the report. If you save the report, it is saved to the local disk.

---

**Designing matching jobs**

Use match specifications when designing IBM InfoSphere QualityStage jobs that use the Two-source Match, One-source Match, or optionally the Match Frequency stage.

**Generating frequency information: Match Frequency stage**

The Match Frequency stage generates frequency distributions, which are needed for input to the matching process. Distributions are generated independently from the matching job.

The stage takes input from a database, file, or processing stage, and generates the frequency distribution of values for columns in the input data.

You use frequency information along with input data when you run the One-source Match or Two-source Match stages. Having the frequency information generated separately means you can reuse it as necessary, and you do not have to generate it whenever you run a one-source or two-source matching job.

**Inputs to the Match Frequency stage**

The Match Frequency stage accepts one input link for the data that you want to generate frequency information for. You can use a match specification if you want frequency data for some columns but not others.

The Match Frequency stage reads data on its input link and generates the frequency distribution of values for columns in the input data. Then, the generated frequency data flows into a match stage or is stored for later use, or both.

While the output from the Match Frequency stage is used specifically for matching data, you typically want to generate frequency data independently from your standardization and matching jobs. By keeping the process of generating frequency data separate, you have the option to run the jobs at predetermined intervals. In addition, you can run these jobs relatively infrequently compared to standardization and matching jobs because frequencies do not necessarily need to be current to produce optimal matching results.

The Match Frequency stage takes one input data file. This data file can be from one of the following places:

- A link from a sample data set file that was generated by the Data Set stage for use in the Match Designer.
- A link from any parallel database, file, or processing stage that carries information that you need for the following stages.
  - A One-source Match stage.
– The data source input of a Two-source Match stage
– The reference source input of a Two-source Match stage

Standardize the input data before you use it with the Match Designer and the match stages.

When you configure the stage, you can choose from the following options.

• Designate a match specification from the repository, so that frequency information is generated only for the match columns in that specification.
• Do not designate a match specification, thereby generating frequency information for all columns of the input.
• Increase the number of frequencies that are reported in the output. The default is 100.

The Match Frequency stage has a single output link that carries four columns.

• qsFreqVal
• qsFreqCounts
• qsFreqColumnID
• qsFreqHeaderFlag

The data in these columns provides the necessary information to give input to the Match Designer and to the One-source Match and Two-source Match stages.

You can link to the input or output of the Match Frequency stage from any parallel database stage, any file stage, and most processing stages.

**Match Frequency stage workflow**

The Frequency Match stage requires input data for which you generate frequency information.

To create a match specification for use in a match job, you need frequency information for the input data and (if you create a two-source match specification) for the reference data. This frequency information from the Match Frequency stage must be output to a Data Set stage. The Data Set stage creates a data set file, which is the only file type that the Match Designer can use to test match passes within a match specification.

A typical workflow for using the Match Frequency stage includes the following tasks.

• Standardize the source data for the data source and (if you intend to create a two-source match job) the reference source.
• Prepare a representative sample data set for the source data to use as input to the Match Designer. If you intend to create a two-source match specification, also prepare a representative sample data set for the reference data.
• Use the Match Frequency stage to generate frequency information for the source data. If you intend to create a two-source match specification, also generate frequency information for the reference data. For the input data, you can use all the source data or only the representative sample source data. If you use all the source data, the job might require more time to run, but the frequency information is potentially more accurate than if you use a sample as the input.
• Along with the sample data, use the output of the Match Frequency stage as input to the Match Designer to create either a two-source match specification or an one-source match specification.
• Optional. If you want to reduce the amount of frequency data that will be used in the match job, you can run the Frequency Match stage job again. However, for this job run, select the match specification that you created. Selecting the match specification in the Frequency Match stage job limits the frequency data to only the columns that will participate in the match job.

• Create an IBM InfoSphere QualityStage job that includes the Two-source Match stage or the One-source Match stage.

• Select the match specification that you created for the job.

Creating Match Frequency jobs
When you create the Match Frequency job on the Designer client canvas, you input a stage that contains the source data into the Match Frequency stage with a single output.

About this task
While you might conceivably take input directly from a Standardize stage and output it directly to a Frequency Match stage in the same job, there are some problems with this approach.

• Performance. Generating frequencies is time consuming. Instead, for example, run the frequencies periodically (monthly, for example) to support a nightly match run.

• Not real time. The Frequency Match stage does not run in real-time mode so this stage is not real-time compatible.

• Unrepresentative data. The data for any particular match run might not be representative. Build the frequencies on a representative data set.

It is better to build modular jobs where the frequency data is generated separately from other jobs such as ones that use the Standardize stage or match stages.

Procedure
1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Click File > New.
3. Select Jobs and Parallel Job and click OK.
4. From the Data Quality palette, select the Match Frequency stage and drag it on the middle of the canvas.
5. From the File palette, select the Sequential File stage and drag it on the canvas to the left of the Match Frequency stage. Link it to the Match Frequency stage. This stage inputs source data to the Match Frequency stage. The source data typically comes from a file or database stage, but you can also use other stages to preprocess it before inputting it to the Match Frequency stage.
6. Select a Data Set stage and drag it on the canvas to the right of the Match Frequency stage. Link the Match Frequency stage to the Data Set stage. This stage receives output from the Match Frequency stage. You can use any file, database, or processing stage. However, if you plan to use the frequency data as input to the Match Designer to create a match specification, you must use an InfoSphere DataStage Data Set stage.
7. Double-click the Sequential File stage.
   a. In the Output > Properties tab, select File = ? and enter or select a file from which the input data will be read. If you used another stage for the input, select the appropriate input and define any additional properties that the source requires.
   b. In the Output > Columns tab, click Load and select a table definition.
c. Click OK and OK.

8. Double-click the Data Set stage (or other stage that you used for the output).
   a. In the Input > Properties tab, select File = ? and enter or select a file to write the output data to. If you used another stage for the output, select the appropriate output and define the properties that the target requires.
   b. Click OK.

9. Optional: Rename the stages and links with meaningful names that reflect their functions in the job or project.

### Configuring the Match Frequency stage

When you configure the Match Frequency stage, determine the source of the input data, edit stage properties, and send the data to a target file.

### About this task

When you use the Match Frequency stage, you can select a match specification to generate frequency information for only the columns that will participate in the match job.

### Procedure

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the job that contains the Match Frequency stage that you want to configure.
3. Double-click the Match Frequency stage.
4. Optional: Click ... to select a match specification.
   a. In the Link Type pane, select Data, if your input link is from the data source. Select Reference, if your input link is from the reference source.
5. If you do not use a match specification, select Do Not Use a Match Specification. The job will generate frequency information for all columns of your input data.
6. Optional: In the Maximum Frequency Entry field, enter 101 - 100,000 to increase the maximum number of frequencies shown in the output. By default, up to 100 different frequencies are output, starting with the most frequent. You might want to increase this number if you are processing large numbers of records.
7. Click Stage Properties.
   a. In the Output > Mapping tab, select the columns and map them to the output link.
8. Click OK and OK.

### Saving the Match Frequency stage table definition

Running a Match Frequency stage job creates a table definition. Save this table definition because you will need to load it when you use the frequency data in a One-source Match stage or Two-source Match stage job.

### Procedure

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the job that contains the Match Frequency stage for which you want to save the table definition.
3. Double-click the Match Frequency stage and click Stage Properties.
4. Click the Output > Columns tab and click Save.
5. In the Save Table Definition window, in the **Table/file name** field, enter a name for the table definition and click **OK**.
6. In the Save Table Definition As window, select the folder where you want to save the table definition and click **Save**.
7. Click **OK** and **OK**.

**Matching one source: One-source Match stage**

The One-source Match stage matches records from a single source file.

An example of grouping records might be that you locate all records that apply to the same individual, household, or event. In addition, you might deduplicate a file to group all invoices for a customer or merge a mailing list.

The One-source Match stage accomplishes the following actions.

- Categorizes all records with weights above the match cutoff as a set of duplicates.
- Identifies a master record by selecting the record within the set that matches to itself with the highest weight. The master record is associated with its set of duplicates.
- Determines that records that are not part of a set of duplicates are nonmatched records. The nonmatched and master records are generally made available for the next pass.
- Excludes duplicates in subsequent passes. However, you can choose the Independent match type if you want duplicates to be included in subsequent passes.

The output of the One-source Match stage can include master records, duplicates above the match cutoff, clerical duplicates, nonmatched records, and statistics about the results of the matching process. You can use this output as input to the Survive stage.

**Inputs to the One-source Match stage**

The One-source Match stage accepts two input links: one for the source data, and one from the frequency information for that data. The One-source Match stage uses a one-source match specification to group and match the data.

To add source data to the One-source Match, you need links from the following sources:

- The data that you want to find matches in. Any stage that produces output can be a source. However, as a best practice, standardize the data before you use the One-source Match stage.
- The frequency information for that data, as generated by the Match Frequency stage. You can input frequency data from the Match Frequency stage as a part of the current job. You can also input frequency data that was output from a previously run Match Frequency job.

When you configure the stage, you must designate an existing match specification from the repository. This match specification must be of the one-source match type and be based on the column definitions of the data you are inputting to the One-source Match stage.

The One-source Match stage matches and groups your input data based upon the match specification. You select the columns to output.
One-source Match stage workflow

The One-source Match stage requires a one-source match specification, standardized data, and frequency information to process data according to particular attributes.

A typical workflow for using the One-source Match stage includes the following tasks.

- Standardize the source data.
- Prepare a representative sample data set from the source data.
- Use the Match Frequency stage to generate frequency information. For use in the Match Designer, use the Data Set stage for the output.
- Use the Match Designer to create a match specification for use in the One-source Match stage.
- Optional. If you want to reduce the amount of frequency data, you can run the Frequency Match stage job again. However, for this job run, use the match specification that you created for the one-source match. Using a one-source match specification limits the frequency data to only the columns that participate in the match job.
- Create an IBM InfoSphere QualityStage job that includes the One-source Match stage, with the source data and the frequency information as input.
- Configure the One-source Match stage, which includes selecting the one-source match specification that you created.
- Use the output information from the One-source Match stage as input to the Survive stage.

Creating One-source Match stage jobs

The One-source Match stage job requires that you add the One-source Match stage to the job and link it to two source stages and up to five output stages.

Procedure

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Click File > New.
3. Select Jobs and Parallel Job and click OK.
4. From the Data Quality palette, select the One-source Match stage and drag it on the middle of the canvas.
5. From the palette, add two input stages. Add one for each of the following data sources.
   a. Data source.
   b. Frequency information for the data source.

   Typically, your source data comes from a file or database but you can also use other stages to preprocess it before inputting it to the One-source Match stage.

6. Link the input stages in the following order.
   a. Data
   b. Data Frequency

7. From the palette, add up to five output stages, one for each of the output options that you intend to use in the One-source Match stage. The following list shows the five output options.
   - Match
   - Clerical
   - Duplicate
8. For the match output options that you intend to use, link the stages in the following order.
   a. Match
   b. Clerical
   c. Duplicate
   d. Nonmatched
   e. Match Statistics

   Note: Verify in the Stage > Link Ordering tab that the link labels and link names are correct.

9. Double-click an input stage.
   a. In the Output > Properties tab, select File = ? and enter or select a file from which the input data will be read. If you used a stage other than a file stage for the input, select the appropriate input and define any additional properties that the source requires.
   b. In the Output > Columns tab, click Load and select a table definition. When you select the table definition for the frequency input, use the table definition that was created when the Frequency Match job was run.
   c. Click OK and OK.
   d. Repeat these steps until all the input stages are configured.

10. Double-click an output stage.
    a. In the Input > Properties tab, select File = ? and enter or select a file to write the output data to. If you used a stage other than a file stage for the output, select the appropriate output and define the properties that the target requires.
    b. Click OK.
    c. Repeat these steps until all the output stages are configured.

11. Optional: Rename the stages and links with meaningful names that reflect their functions in the job or project.

Configuring the One-source Match stage
After you create a job that contains this stage, select from available match specifications and other settings that match records in one data source.

Procedure
1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the job that contains the One-source Match stage that you want to configure.
3. Double-click the One-source Match stage.
4. In the Match Specification field, click ... and select a one-source match specification.
5. Optional: To override any of the match cutoff values from the match specification that you selected, select Override cutoffs and enter new values or job parameters in the Match or Clerical columns.
6. Optional: To set multiple cutoff values for the override match cutoffs, select the values that you want to override and click the arrow icon. Select one of the following options.
### Option Action

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you want to use an existing job parameter</td>
<td>Select <strong>Job Parameter</strong> and an existing parameter.</td>
</tr>
<tr>
<td>If you want to enter a new job parameter</td>
<td>Select <strong>Job Parameter &gt; New</strong> and define the new parameter.</td>
</tr>
<tr>
<td>If you want to enter specific values</td>
<td>Select <strong>Keyed Value</strong> and enter the value.</td>
</tr>
</tbody>
</table>

7. Optional: To return to match cutoff values from the match specification if match cutoff values are overridden, select one of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you want to return all overridden values</td>
<td>Clear <strong>Override.</strong></td>
</tr>
<tr>
<td>If you want to return selected overridden values</td>
<td>Select the values that you want to return. Click <strong>Set selected cutoffs to</strong> and select <strong>Value from match specification.</strong></td>
</tr>
</tbody>
</table>

8. From the Match Type pane, choose a match type.
9. In the Match Outputs pane, select one or more outputs that you want to create. Be aware of the following conditions.
   - Each output that you select must have a corresponding output link.
   - No output can be sent to more than one link.
10. Click **Stage Properties.**
    a. In the **Output > Mapping** tab, select the columns and map them to the output link.
11. Click **OK** and **OK**.

**Match types for the One-source Match stage:**

The One-source Match stage match type determines the relationship among the match passes, the records that are processed by each of the passes, and how the groups are formed.

When you use the One-source Match stage, you select one of the following match types.

- **Dependent.** Passes in a one-source dependent match process the data sequentially. In each pass, groups are built around master records. The groups that are formed in all the passes for the same master record are combined to create the final group for the master. Each duplicate record in a group matches the group master record in one of the match passes. The master records and nonmatched records from a pass are made available for the subsequent pass. Duplicates are taken out of consideration so that they are not assigned to more than one group. Existing master records are given priority in group construction in subsequent passes.

- **Independent.** Each pass in a one-source independent match processes all the input records. Like the one-source dependent match type, in each pass, groups are built around master records. But because each pass processes all records, a record can be a member of a group from more than one of the passes. (Similarly, a record can be a master in a group that was built in one pass while being a duplicate in a group that was built in another pass.) The groups from all the passes are merged so that groups that have a record in common form a single group. If record A is in a group with record B and record B is in a different group with record C, then those two groups are merged so that records A, B,
and C are all in the same group. (A record ends up in no more than one group.)
Groups are merged until all groups that have records in common are merged. At
the pass level, the relationship that determines group membership is that of
records matching a master record. However, for the merge process, the
relationship is one of group membership. Thus, members in a group can be
connected by a chain of relationships and do not necessarily all match a
common master.

- **Transitive.** Like one-source independent matches, each pass in a one-source
  transitive match also processes every record. But unlike a one-source
  independent match, the one-source transitive match type does not create
  pass-level groups. Instead, all record pairs that score above the match cutoff are
  used to produce the groups. Creating pass-level groups would discard the
  information that a record pair's score was above the match cutoff in a pass if
  each record ends up in a different group. The one-source transitive match type
  does not discard that information. It builds groups so that all records that score
  above the match cutoff in any pass are in the same group. For example, if record
  A and record B scored above the match cutoff in a pass and record B and record
  C scored above the match cutoff in a pass (possibly the same pass), then records
  A, B, and C are added to the same group. (A record ends up in no more than
  one group.) Like one-source dependent matches, members in a group can be
  connected by a chain of relationships and do not necessarily all match a
  common master. But the one-source transitive chain can extend further because
  it uses all the pairs that score above the match cutoff.

In most cases, choose the dependent match type, because you want duplicates
removed from consideration so that they do not match to other records in
subsequent passes.

However, the independent option is useful when you want to link people or
organizations regardless of address. For example, you can link together all the
locations where a doctor practices.

The transitive option is useful if you want to account for inconsistent data entry in
fields that assist in duplicate identification, for example, date of birth and driver's
license numbers.

**An example of processing for the dependent and independent match types**

The following example shows how to use the independent match type with the
One-source Match stage. The table shows four records that describe the same
person. You require that all records concerning the same person match without
regard to address.

*Table 7. Four records that describe the same person*

<table>
<thead>
<tr>
<th>Record</th>
<th>Name</th>
<th>Address</th>
<th>Tax ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>William Nickson</td>
<td>123 Rodeo Drive</td>
<td>123456789</td>
</tr>
<tr>
<td>2</td>
<td>Bill Nixon</td>
<td>123 Rodeo Drive</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>B Nickson</td>
<td>978 Sunset Blvd.</td>
<td>123456789</td>
</tr>
<tr>
<td>4</td>
<td>Nickson</td>
<td>456 Western Ave.</td>
<td>123456789</td>
</tr>
</tbody>
</table>

The matching process using this data yields different results depending on the
match type that you choose:

- **Dependent**
– The first pass blocks and matches on Name and Address. Records 1 and 2 are considered a matched pair. Records 3 and 4 are considered nonmatched records.
– If Record 2 (without the TaxID) is selected as the master, and Record 1 is considered a duplicate, then Record 1 is not available for the second pass.
– If the second pass blocks and matches on Name and TaxID, then only Records 3 and 4 match. The result is two groups of matched records: Records 1 and 2, and Records 3 and 4.

- **Independent**
  – The first pass results are the same as the dependent match. Records 1 and 2 are considered a matched pair. Records 3 and 4 are considered nonmatched records.
  – If Record 2 (without the TaxID) is selected as the master record in the second pass, the duplicate record, Record 1, is also compared to the rest of the records. When you block on Name and TaxID, records 1, 3, and 4 match. Since Record 1 matched Record 2 in the first pass, the output is one group with all four records linked.

**An example of processing for the dependent and transitive match types**

The following example shows how to use the transitive match type option with the One-source Match stage. The table shows six records that show a difference of one day between records of the same family name. You require that records of the same family name match if the dates have a difference of one day or less.

*Table 8. Records that show a difference of one day between records of the same family name*

<table>
<thead>
<tr>
<th>Record</th>
<th>Family name</th>
<th>Date</th>
<th>Given name</th>
<th>qsMatch Type</th>
<th>qsMatch Weight</th>
<th>qsMatch Pass Number</th>
<th>qsMatch SetID</th>
<th>qsMatch DataID</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Clifford</td>
<td>19530831</td>
<td>Benn</td>
<td>MP</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Clifford</td>
<td>19530829</td>
<td>George</td>
<td>DA</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>Clifford</td>
<td>19530830</td>
<td>George</td>
<td>DA</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>Clifford</td>
<td>19530731</td>
<td>Thomas</td>
<td>MP</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>Clifford</td>
<td>19530801</td>
<td>David</td>
<td>DA</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>Clifford</td>
<td>19530802</td>
<td>David</td>
<td>DA</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>10</td>
</tr>
</tbody>
</table>

The matching process that uses this data yields different results depending on the match type that you choose:

- **Dependent**
  – The first pass blocks on Family Name and matches on Date by using a date tolerance of one day. Records 5 and 6 are considered a matched pair.
  – If Record 5 is selected as the master record, Record 6 is not available for the second pass and no other records match.

- **Transitive**
  – The first pass blocks on Family Name and matches on Date by using a date tolerance of one day. Records 5 and 6 are considered a matched pair.
  – If Record 5 is selected as the master record, Record 6 is available for subsequent passes and is compared to the rest of the records. Records 6 and 7 are considered a matched pair. Because Record 5 matched Record 6 in the first
pass, the result is one group in which all three records are linked. Records 5 and 6 are matched. Records 6 and 7 are matched. Therefore Records 5, 6, and 7 are within the same match set.

**Match outputs for the One-source Match stage:**

The One-source Match stage takes up to five output links and links to any parallel database or file stage, and most processing stages.

Select from the following output options. Each option is output to a separate output link. Therefore, the number of output links corresponds to the number of output options that you select.

- **Match.** The master records.
- **Clerical.** The duplicates that fall in the clerical range.
- **Duplicate.** The duplicate records that are above the match cutoff.
- **Nonmatched.** The records that are not master, duplicate, or clerical records.
- **Match Statistics.** Summary statistics about the matching results and statistics about the matching process for each match pass.

Use the **Stage Properties > Link Ordering** tab to associate the output options with specific output links. Check the Link Ordering tab to ensure that the records for each output option that you select are output to the link to which you intend.

If you want, you can add other InfoSphere DataStage stages (such as the Funnel stage) to group some or all the output into a single file or table.

The columns that are available for output consist of all the input columns, plus additional columns that are created by the match process.

The nonmatched output includes the following columns.

- **qsMatchDataID.** The data record ID.
- **qsMatchType.** The match ID for the record. One of:
  - **MP.** Master record.
  - **DA.** Duplicate record.
  - **CP.** Record that requires clerical review.
  - **RA.** Nonmatched record.
- **qsMatchSetId.** The match set identifier. (For nonmatched records, this identifier is the same as the data record ID.)

The match, clerical, and duplicate output includes the previous three columns in addition to the following columns.

- **qsMatchWeight.** The weight.
- **qsMatchPattern.** The pattern.
- **qsMatchLRFlag.** "L" for left, "R" for right.
- **qsMatchExactFlag.** "X" if the match is exact.
- **qsMatchPassNumber.** The number of the pass where the match was found.

If you select the **Match Statistics** output option, ensure that you use the default names for the output columns. The statistical output includes the following columns:

- **qsMatchPassNumber.** The number of the pass where the match was found.
• qsMatchStatType. The number used to identify the type of statistic.
• qsMatchStatValue. The value for a particular statistic.
• qsMatchWeight. The weight.

**Matching two sources: Two-source Match stage**

The Two-source Match stage compares two sources of input data (reference records and data records) for matches.

The Two-source Match stage uses the following four sources of data for matches.
• A data source
• Frequency information about the data source, generated by the Match Frequency stage
• A reference source
• Frequency information about the reference source, generated by the Match Frequency stage

**Inputs to the Two-source Match stage**

The Two-source Match stage takes four input sources. The data and reference sources can be from any parallel database, file or processing stage.

The four inputs to the Two-source Match stage come from the following sources.
• Data source
• Reference source
• Frequency information about the data and reference source, generated by the Match Frequency stage as a part of the current job. You can also input frequency data that was output from a Match Frequency job that was run previously.

When you configure the stage, designate an existing two-source match specification from the repository based on the column definitions of the data and reference sources.

The Two-source Match stage matches and groups your input data based upon the match specification. You select which columns to output.

**Two-source Match stage workflow**

The Two-source Match stage requires standardized data and reference data as source data, a two-source match specification, and frequency information for both sources.

A typical workflow for using the Two-source Match stage includes the following tasks.
• Standardize the source data for the data source and the reference source.
• Prepare representative sample data sets from the source data.
• Use the Match Frequency stage to generate frequency information. For use in the Match Designer, use the Data Set stage for the output.
• Use the Match Designer to create a match specification for use in the Two-source Match stage.
• Optional. If you want to reduce the amount of frequency data that will be used in the Two-source Match job, you can run the Frequency Match stage job again. However, for this job run, select the two-source match specification that you
created. Selecting the two-source match specification in the Frequency Match stage job limits the frequency data to only the columns that will participate in the match job.

- Create an IBM InfoSphere QualityStage job that includes the Two-source Match stage, with data source, reference source, and the frequency information for each source as inputs.
- Configure the Two-source Match stage, which includes selecting the two-source match specification that you created.
- Use the output information from the Two-source Match stage as input to the Survive stage.

**Creating Two-source Match stage jobs**

A Two-source Match stage job requires that you add the Two-source Match stage to the Designer client canvas, and link it to data and reference sources and output stages.

**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Click **File > New**.
3. Select **Jobs and Parallel Job** and click **OK**.
4. From the Data Quality palette, select the Two-source Match stage and drag it on the middle of the canvas.
5. From the palette, add four input stages. Add one for each of the following data sources.
   a. Data source.
   b. Frequency information for the data source.
   c. Reference source.
   d. Frequency information for the reference source.

   Typically, your source data comes from a file or database but you can also use other stages to preprocess it before inputting it to the Two-source Match stage.
6. Link the input stages in the following order:
   a. Data
   b. Reference
   c. Data Frequency
   d. Reference Frequency
7. From the palette, add up to seven output stages, one for each of the output options that you intend to use in the Two-source Match stage. The following list shows the seven output options.
   - Match
   - Clerical
   - Data Duplicate
   - Reference Duplicate
   - Data Nonmatched
   - Reference Nonmatched
   - Match Statistics

   For the output stages, you can use any file, database, or processing stage.
8. For the match output options that you intend to use, link the stages in the following order.
a. Match
b. Clerical
c. Data Duplicate
d. Reference Duplicate
e. Data Nonmatched
f. Reference Nonmatched
g. Match Statistics

Note: Verify in the Stage > Link Ordering tab that the link labels and link names are correct.

9. Double-click an input stage.
   a. In the Output > Properties tab, select File = ? and enter or select a file from which the input data will be read. If you used a stage other than a file stage for the input, select the appropriate input and define any additional properties that the source requires.
   b. In the Output > Columns tab, click Load and select a table definition. When you select the table definition for the frequency input, use the table definition that was created when the Frequency Match job was run.
   c. Click OK and OK.
   d. Repeat these steps until all the input stages are configured.

10. Double-click an output stage.
    a. In the Input > Properties tab, select File = ? and enter or select a file to write the output data to. If you used a stage other than a file stage for the output, select the appropriate output and define the properties that the target requires.
    b. Click OK.
    c. Repeat these steps until all the output stages are configured.

11. Optional: Rename the stages and links with meaningful names that reflect their functions in the job or project.

### Configuring the Two-source Match stage

After you create a job that contains this stage, select from available match specifications and other settings that determine matching records in two data sources.

**Procedure**

1. Open the IBM InfoSphere DataStage and QualityStage Designer.
2. Open the job that contains the Two-source Match stage that you want to configure.
3. Double-click the Two-source Match stage.
4. In the Match Specification field, click ... and select a two-source match specification.
5. Optional: To override any of the match cutoff values from the match specification that you selected, select Override cutoffs and enter new values or job parameters in the Match, Clerical, or Duplicate columns.
6. Optional: To set multiple cutoff values for the override match cutoffs, select the values that you want to override and click the arrow icon. Select one of the following options.
### Choice Action

<table>
<thead>
<tr>
<th>Choice</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you want to use an existing job parameter</td>
<td>Select <strong>Job Parameter</strong> and an existing parameter.</td>
</tr>
<tr>
<td>If you want to enter a new job parameter</td>
<td>Select <strong>Job Parameter &gt; New</strong> and define the new parameter.</td>
</tr>
<tr>
<td>If you want to enter specific values</td>
<td>Select <strong>Keyed Value</strong> and enter the value.</td>
</tr>
</tbody>
</table>

7. Optional: To return to match cutoff values from the match specification if match cutoff values are overridden, select one of the following options.

<table>
<thead>
<tr>
<th>Choice</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you want to return all overridden values</td>
<td>Clear <strong>Override</strong>.</td>
</tr>
<tr>
<td>If you want to return selected overridden values</td>
<td>Select the values that you want to return. Click <strong>Set selected cutoffs to</strong> and select <strong>Value from match specification</strong>.</td>
</tr>
</tbody>
</table>

8. From the Match Type pane, choose a match type.

9. In the Match Outputs pane, select one or more outputs that you want to create. Be aware of the following conditions.
   - Each output that you select must have a corresponding output link.
   - No output can be sent to more than one link.

10. Click **Stage Properties**.
    a. In the **Output > Mapping** tab, select the columns and map them to the output link.

11. Click **OK** and **OK**.

### Match types for the Two-source Match stage:

The Two-source Match stage match type determines the relationship among the match passes, the records that are processed by each of the passes, and how the groups are formed.

When you use the Two-source Match stage, you select one of the following match types.

**Many-to-one**

Any reference source record can match many data source records. Any one data source record can match only one reference source record. For example, if 101 Main St. on the data source matches two records on the reference source: 101-199 Main St SW and 101-199 Main St SE, the first reference source record is the matched record and the second reference source record is not considered a match to this particular data source record. It is possible that the second reference source record will match another data source record.

**Many-to-one multiple**

Each reference source record having the same weight as the matched pair when it is scored against the data record is flagged as a duplicate record. Any one data source record might match more than one reference source record. For example, if 101 Main St. on the data source matches to two records on the reference source: 101-199 Main St SW and 101-199 Main St SE, one reference source record is the matched record and the other is the duplicate.
Many-to-one duplicate
Like the many-to-one multiple option, except that additional reference source records that match to a level above the duplicate cutoff value are flagged as duplicates. This functionality means that records with lower weights than the match weight can be flagged as duplicates. For example, if 101 Main St on the data source matches to three records on the reference source: 101-199 Main St SW, 101-199 Main St SE, and 101 Main Rd, you get 101-199 Main St SW as the match, and both of the other addresses might be duplicates.

One-to-one
Matches a record on the data source to only one record on the reference source. A record on the reference source can only match to one data source record.

Match outputs for the Two-source Match stage:
The Two-source Match stage takes up to seven outputs. This functionality lets you send records from each of the following categories to different outputs using these options.

You can send records to different links using one of the following options.
- **Match**: The matched records for both inputs.
- **Clerical**: The clerical review records for both inputs.
- **Data Duplicate**: The duplicates in the data source.
- **Reference Duplicate**: The duplicates in the reference source.
- **Data Nonmatched**: The nonmatched records from the data input.
- **Reference Nonmatched**: The nonmatched records from the reference input.
- **Match Statistics**: Summary statistics about the matching results and statistics about the matching process for each match pass.

Each of these options must connect to a separate link. You can control the links that the outputs go to in the **Stage Properties > Link Ordering** tab.

If you want, you can add other InfoSphere DataStage stages (such as the Funnel stage) to group some or all the output into a single file or table.

The columns that are available for output consist of all the input columns, plus additional columns that are created by the match process. If you select the **Match Statistics** output option, ensure that you use the default names for the output columns.

Predefined matching jobs
You can use matching jobs that are provided with IBM InfoSphere QualityStage to match records from one or two sources of input data. You can run predefined matching jobs after you add match specifications that you create by using the Match Specification Setup Wizard.

Predefined matching jobs are in the following file:

`installation_directory\Clients\Samples\DataQuality\MatchTemplates\Jobs\PredefinedJobs.dsx`
where *installation_directory* is the root installation directory of the InfoSphere Information Server client tier.

You can import the jobs, and then view them in the Designer client repository tree by choosing **Data Quality > Match Templates > Jobs.**

The following table lists the predefined matching jobs and the purpose for each job.

*Table 9. Predefined matching jobs*

<table>
<thead>
<tr>
<th>Job name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT30_BusinessDeduplicationMatch</td>
<td>Identify records in a single data source that represent the same business at the same address.</td>
</tr>
<tr>
<td>MT30_BusinessHouseholdMatch</td>
<td>Group records in a single data source that represent businesses at the same address.</td>
</tr>
<tr>
<td>MT30_IndividualDeduplicationMatch</td>
<td>Identify records in a single data source that represent the same individual at the same address.</td>
</tr>
<tr>
<td>MT30_IndividualHouseholdMatch</td>
<td>Group records in a single data source that represent individuals at the same address.</td>
</tr>
<tr>
<td>MR30_BusinessHouseholdReferenceMatch</td>
<td>Group records from two data sources that represent businesses at the same address.</td>
</tr>
<tr>
<td>MR30_BusinessReferenceMatch</td>
<td>Identify records from two data sources that represent the same business at the same address.</td>
</tr>
<tr>
<td>MR30_IndividualHouseholdReferenceMatch</td>
<td>Group records from two data sources that represent individuals at the same address.</td>
</tr>
<tr>
<td>MR30_IndividualReferenceMatch</td>
<td>Identify records from two data sources that represent the same individual at the same address.</td>
</tr>
</tbody>
</table>

**Usage notes**

Use standardized data and the frequency information for that data as input for the predefined matching jobs. You can generate this data by running the appropriate predefined standardization job. The columns in the input data must be the columns that are produced by US domain-specific rule sets.

You can run predefined matching jobs as they are configured initially or you can use the jobs as models for creating jobs that meet your specific matching requirements.

Changes that you make to the predefined jobs overwrite the initial configurations for those jobs.

If you create a job that is based on one of the predefined jobs, ensure that you copy the predefined job and give it a unique name. When you configure the One-source Match or Two-source Match stage in a predefined job, choose a match specification that applies to that stage. Also, ensure that the specification applies to the columns that are the input to the stage.

Before you edit or run a predefined matching job, read the annotation for that job.
Match Statistics reports

The Match Statistics report presents summary-level statistics about the matching results and statistics about the matching process for each match pass.

Use Match Statistics reports to evaluate the results of the matching process. You can also share the results with team members and stakeholders who do not have direct access to the IBM InfoSphere DataStage and QualityStage Designer, where you generate data for the reports.

You generate the data for these reports by running a job that uses the One-source Match stage or Two-source Match stage. When you configure either stage, choose Match Statistics from the list of match outputs to produce the data that is required for the Match Statistics report.

The output stage that contains data for the Match Statistics report must be a Sequential File stage or ODBC Enterprise stage. When you configure the output stage, specify that the first line in the file contains column names.

You create Match Statistics reports in the IBM InfoSphere Information Server Web console.

Creating Match Statistics reports

After you run a job that contains a One-source Match or Two-source Match stage, you can create a Match Statistics report to view statistics about the results of the matching process.

Before you begin

- Configure the One-source Match or Two-source Match stage in the job for which you want statistics. Ensure that you select Match Statistics from the list of match outputs. Do not change the names of the columns in the output for the statistics.
- Ensure that the job for which you want statistics uses a Sequential File stage or ODBC Enterprise stage as the output stage. When you configure the output stage, specify that the first line in the file contains column names.
- Run the job for which you want statistics.
- Configure your Web browser to work with the IBM InfoSphere Information Server Web console. For more information about configuring your Web browser, see the IBM InfoSphere Information Server Administration Guide.

Procedure

1. Log in to the IBM InfoSphere Information Server Web console.
2. Click the Reporting tab.
5. Optional: Enter a meaningful name and description for the report.
   a. In the Name field, replace the default name with a name that is more meaningful to you. For example, you might choose a name that includes the name of the job or stage that you select for this report.
   b. In the Description field, replace the default description with a description that is more meaningful to you. For example, you might enter a summary of the job that describes the input data in the job.
The **Save-in Folder** field contains the name of the default folder where the new report is stored on the server. You cannot specify a different folder when you create a report. When you view the report, you can save it to the folder of your choice.

6. Configure the report settings.
   a. From the **Project** list, select the project that contains the job for which you want to create the report.
   b. For the **Job Source** list, click **Retrieve Values**, and then choose the job for which you want to create the report.
   c. For the **One-source or Two-source Match Stage Name** list, click **Retrieve Values**, and then choose the stage for which you want to create the report.
   d. Select a file format for the report.
   e. Optional: Change the expiration and history policy settings.

7. Click **Finish**, then choose what you want to do with the report that you created from the list of options.

**Running and viewing Match Statistics reports**

After a report is created, you can run the report. You can then view the report that you ran or any report that was run previously.

**Before you begin**

[Create a Match Statistics report]

**About this task**

You run a Match Statistics report to generate information about the initial results of the matching process. If you change the input data or the configuration of an One-source Match or Two-source Match stage, you can run the report again to generate updated information.

**Procedure**

1. Log in to the IBM InfoSphere Information Server Web console.
2. Click the **Reporting** tab.
3. From the Navigation pane, click **Reports > View Reports**, and select the report that you want to run.
4. Click **Run Now**. The report is queued for running. When the report runs, the run time and run duration are shown in the Running pane.
5. When the report processing is complete, click **Reports > View Reports**.
6. From the list of reports, select the report that you ran.
7. From the task pane, click **View Report Result**. The report opens in the format that you chose when you created the report.
Chapter 5. Consolidating duplicate records to create the representative record

The Survive stage constructs column values from groups of related or duplicate records and stores the column values in the survive record (the best result) from each group.

This "best result" helps you to populate the columns of surviving records with the best available data. The stage implements the business and mapping rules, creating the necessary output structures for the target application and identifies columns that do not conform to load standards.

The Survive job is the last job in the InfoSphere QualityStage workflow and is usually run after the One-source Match stage job. The output from the One-source Match stage, and in some cases the Two-source Match stage, becomes the source data that you use for the Survive stage.

The Survive stage accepts all basic data types (non-vector, non-aggregate) other than binary. The Survive stage accepts a single data source from any of the following groups:

- database connector
- flat file
- data set
- processing stage

The Survive stage requires one input source. If your input is the result of a match stage, you need to set up another stage (for example, a Funnel stage) to combine the master and duplicate records into one input source.

While it is not necessary to process the data through the match stages before you use the Survive stage, the source data must include related or duplicate groups of rows. Also, the data must be able to be sorted on one or more columns that identify each group. These columns are referred to as group keys.

To order the records, you sort on the group key or keys so that all records in a group are contiguous. The Survive stage automatically sorts records if the Pre-sort Input Option is selected in the Survive Stage window. However, the automatic sort provides no control over the order of the records within each group. To control the order within groups, you can pre-sort the input by using the Sort stage.

The Survive stage can have only one output link. This link can send output to any other stage. You specify which columns and column values from each group create the output record for the group. The output record can include the following information:

- An entire input record
- Selected columns from the record
- Selected columns from different records in the group

You select column values based on rules for testing the columns. A rule contains a set of conditions and a list of one or more target columns. If a column tests true against the conditions, the column value for that record becomes the best candidate...
for the target. After each record in the group is tested, the columns that are the
best candidates are combined to become the output record for the group.

To select a best candidate match, you can specify multiple columns, for example:
- Record creation date
- Data source from which the record originated
- Length of data in the column
- Frequency of data in a group

For example, the One-source Match stage identified the following portions of three
records as representing the same person using different variations of the name.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>qsMatchSetID</th>
<th>Given Name</th>
<th>Middle Initial</th>
<th>Family Name</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
<td>JON</td>
<td></td>
<td>SMITH</td>
<td>JR</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>J</td>
<td></td>
<td>SMITHE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>JOHN</td>
<td>E</td>
<td>SMITH</td>
<td></td>
</tr>
</tbody>
</table>

The Survive stage constructs the best record using length analysis on the columns
Given Name, Middle Initial, and Suffix, and using frequency analysis on the
column Family Name, with the following result.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>qsMatchSetID</th>
<th>Given Name</th>
<th>Middle Initial</th>
<th>Family Name</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9</td>
<td>JOHN</td>
<td>E</td>
<td>SMITH</td>
<td>JR</td>
</tr>
</tbody>
</table>

Creating a survive record

You set up and configure the Survive stage to create a survive record.

Before you begin

Before you can add source data to the Survive stage, all input records must be
combined into one input source.

Procedure
1. Set up the Survive job.
2. Configure the Survive stage.
   a. Define survive rules.
   b. Set stage properties.
3. Compile the job.
4. Run the job.

Setting up a survive job

A Survive job requires that you link a single input stage and a single output stage
to the Survive stage.

Before you begin

Open the Designer client.
Procedure
1. Create a new parallel job or open an existing parallel job.
2. If the Palette is not displayed, click View > Palette to display it.
3. Click Data Quality and locate the Survive stage.
4. Drag the Survive stage icon onto the canvas and drop it in the middle of the canvas.
5. In the Palette, perform the following steps:
   a. Select the appropriate icon that represents the input and drop the icon to the left of the Survive stage on the Designer canvas. The input can be any object that supports output links.
      The object becomes the source data file. Your source data comes from a file or database, but you can also use other stages to preprocess it before inputting it to the Standardize stage.
   b. Drop it to the left of the Survive stage icon.
   c. Drag a second appropriate stage on the Designer canvas and drop it to the right of the Survive stage icon.
      This file becomes the output or target file.
   d. Click General and drag the Link to add a link between the Survive stage and the left Sequential file. To add links, you can also right-click the Sequential file and drag the mouse to the Survive stage.
   e. Drag another link from the Survive stage and the right Sequential file.
6. Rename the stages and links.

Configuring the survive job
You need to configure the Survive stage and Sequential files before you run a Survive job.

About this task
When you configure the Survive stage, you choose simple rules that are provided in the New Rules window or you select the Complex Survive Expression to create your own custom rules. You use some or all of the columns from the source file, add a rule to each column, and apply the data. After the Survive stage processes the records to select the best record, this information is sent to the target file.

Procedure
1. Double click the source file.
2. To configure the source file, follow these steps:
   a. From the Properties window, click Source: File to activate the File: field.
   b. Select Browse for file.
   c. Locate the directory where you stored the input data file.
   d. Select the file. The file name displays in the Source: File pane.
   e. Click OK to close the Properties window.
3. Double click the Survive stage icon on the Designer client canvas.
4. To configure the Survive stage, follow these steps:
   a. In the Survive Stage window, click the Select the group identification data column to select the column that contains the ID number for each matched group. If you are using data from a Match stage, this would be the qsMatchSetId column.
b. If your input data is already sorted on the group keys that you specified in substep a, select Don’t pre-sort the input data.
Otherwise, the Survive stage sorts the input data on the column selected in substep a.

c. Click New Rule to add a new survive rule.

d. In the Survive Rules Definition window, select one or more columns from the Available Columns list.

e. Click \( \text{\textgreater} \) to add the selection to the Targets list in the Specify Output Column(s) area.
If you are creating a complex survive expression, select AllColumns to preserve the entire record.

f. Create a simple rule or a rule that uses a complex expression by selecting one of the following procedures:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Defining simple survive rules”</td>
<td>You define a simple rule by selecting a Technique and applying it to a selected column.</td>
</tr>
<tr>
<td>“Defining complex survive expressions” on page 155</td>
<td>You define a complex expression by clicking the Complex Survive Expression button and building a condition for a selected column.</td>
</tr>
</tbody>
</table>

5. Click OK to return to the Survive Stage window.
6. Click Stage Properties > Output > Mapping tab.
   You configure the Mapping columns to specify which target columns you want to send to the output Sequential file. When you select rules for the Survive stage, the Mapping tab displays only the target columns that you selected when creating rules.

7. Copy the columns from the left pane to the right pane.
8. Click OK twice to save your changes and exit the stage.
9. Click Compile. The Compile window displays information about the compile process.
10. When the compile ends without errors, open the IBM InfoSphere DataStage and QualityStage Director and click Run.

**Defining simple survive rules**

To identify a target (data columns) for output records, the Survive stage requires a rule that contains one or more targets and a TRUE condition expression.

**About this task**

In the Survive stage, you define a simple rule by specifying each of the following elements:
- Target column or columns
- Column to analyze
- Technique to apply to the column that is being analyzed
The rule is defined for you depending on the Technique that you select.
**Procedure**

1. In the **Analyze Column** field, click the drop-down menu to select the column you want to analyze.

   You are selecting a column as the target to which to compare other columns. If you selected a single target column such as **AllColumns**, you can select **Use Target** to analyze that column. The values in the selected target column survive depending on criteria set for that column. Only one column is analyzed by each rule, but values for multiple target columns can survive as a result.

2. In the **Technique** field, select the Technique to apply to the column being analyzed.

   Techniques are commonly used survive rules that are compressed into a single, descriptive name. The following table lists the available Techniques and their associated pattern, where *column* is the column name to be analyzed, *DATA* is the contents of the Data column, *c.* is the current record to be analyzed, and *b.* is the best record analyzed so far.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest Field</td>
<td><code>SIZEOF(TRIM(c.&quot;column&quot;)) &lt;= SIZEOF(TRIM(b.&quot;column&quot;))</code></td>
</tr>
<tr>
<td>Longest Field</td>
<td><code>SIZEOF(TRIM(c.&quot;column&quot;)) &gt;= SIZEOF(TRIM(b.&quot;column&quot;))</code></td>
</tr>
<tr>
<td>Most Frequent</td>
<td><code>FREQUENCY</code></td>
</tr>
<tr>
<td>Most Frequent [Non-blank]</td>
<td><code>FREQUENCY</code> (Skips missing values when counting most frequent.)</td>
</tr>
<tr>
<td>Equals</td>
<td><code>c.&quot;column&quot; = &quot;DATA&quot;</code></td>
</tr>
<tr>
<td>Not Equals</td>
<td><code>c.&quot;column&quot; &lt;&gt; &quot;DATA&quot;</code></td>
</tr>
<tr>
<td>Greater Than</td>
<td><code>c.&quot;column&quot; &gt;= &quot;DATA&quot;</code></td>
</tr>
<tr>
<td>Less Than</td>
<td><code>c.&quot;column&quot; &lt;= &quot;DATA&quot;</code></td>
</tr>
<tr>
<td>At Least One</td>
<td><code>1</code> (At least one record survives, regardless of other rules.)</td>
</tr>
</tbody>
</table>

3. If the Technique you selected was Equals, Not Equals, Greater Than, or Less Than, enter a value, field name, or expression in the **Data** field.

   For all other Techniques, enter nothing in the **Data** field.

4. Click **OK** to add the rule. The Survive Rules Definition window closes and the rule appears in the Survive Stage window.

**Results**

You can add, modify, or delete simple rules for all the columns that you want to appear in your target data in the same manner.

If you have multiple rules, you can reorder them in the Survive Stage window.

**Defining complex survive expressions**

To identify a target for output records, the Survive stage requires a rule that contains a target and a TRUE condition expression.
About this task

In the Rule Expression Builder, you define a rule by specifying each of the following elements:
- a current record from the Columns list
- one or both functions
- an operator
- a best record from the Columns list
- one or both functions

Procedure

1. Click **Complex Survive Expression** to display the Rule Expression Builder.
2. To create a complex expression, follow these steps:
   a. From the Functions list, double-click the SIZEOF function. The SIZEOF function determines the number of characters including spaces in a string-type column.
   b. In the Expression field, place the cursor between the parentheses that follow the SIZEOF function.
   c. Double-click the TRIM function. The TRIM function strips the leading and trailing spaces from the string-type column.
   d. In the Expression field, place the cursor between the parentheses that follow the TRIM function.
   e. From the Columns list, double click to select a current record. A current record is a column name preceded by the letter c; for example, c.AddressLine.
   f. In the Expression field, click to the right of all parentheses.
   g. From the Operations list, double-click to select the operation that best compares the two records.
   h. Repeat steps a to d.
   i. From the Columns list, double-click to select a best record. A best record is a column name preceded by the letter b; for example, b.AddressLine.
   j. To check your expression for the correct syntax, click **Check Expression**. If an error in syntax occurred, a message describes the error and where it occurred. The cursor moves to the error in the Expression field. You should correct the error and click **Check Expression** again.
3. Click **OK** to add the rule. The expression checker checks the rule again to ensure that it is syntactically correct.

**Note:** Even though the syntax is correct does not mean that the rule makes sense in the context of the data. You need to evaluate the logic of the rules as you construct them.

Results

The Survive Rules Definition window closes, and the rule appears in the Survive Stage window.

Applying survive rules

The Survive stage analyzes record columns by applying simple or complex rules to the column and selecting the best column based on the rule.
To apply survive rules to the input columns, the Survive stage includes the following items:

- A set of pre-defined Techniques (packaged survive expressions) from which you can select
- The Rule Expression Builder for creating your own complex expressions

To consider a target as the best candidate for the output record requires a rule that comprises one or more targets and a TRUE conditional expression. A condition is made up of:

- Column names
- Constant or literal values
- Operators that specify comparison or arithmetic operations

You can create more than one rule for a target. In addition, you can use any input column for testing the condition. The column can be the target or another column that is not associated with the target.

**About the Survive stage rule processing**

The Survive stage evaluates the columns to the rule and selects those that meet the conditions of the rule as best columns.

The Survive stage reads the first record and evaluates the record according to any rule that you select. The evaluation process uses the following method:

- If the first record has no best columns then the selected rule for the target record is evaluated against all the columns in the record. If a target record passes the test, its columns become best columns and a b appears in front of the column names.
- Each subsequent record in the group is evaluated in relation to the current record. If a target record passes the test then its columns become the best columns and replace any existing best columns. If none of the current columns meets the conditions, the best columns remain unchanged.
- After all records in the group are evaluated, the values that are designated as the best values are combined in the output record. Survive continues the process with the next records.

**Rule processing example**

The following rule states that COLUMN3 of the current record should be retained if the column contains five or more characters and COLUMN1 has any contents.

\[ \text{COLUMN3: (SIZEOF (TRIM c.COLUMN3) } \geq 5) \text{ AND (SIZEOF (TRIM c.COLUMN1) } > 0) \]

This rule is created by selecting COLUMN3 as the target and using the Rule Expression Builder to create the complex expression.

This table shows the number of characters in the three records in the first record group.

<table>
<thead>
<tr>
<th>Record</th>
<th>COLUMN1</th>
<th>COLUMN3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>
Record 1 has two characters in COLUMN3 and three characters in COLUMN1. This record fails the test, because COLUMN3 has less than five characters.

Record 2 has seven characters in COLUMN3 and five in COLUMN1. This record passes the conditions for the rule. The current COLUMN3 (from the second record) becomes the best column.

Record 3 has five characters in COLUMN3 and seven in COLUMN1 and also passes the conditions. COLUMN3 from this record replaces the best value as the new best value.

When you define multiple rules for the same target, the rule that appears later in the list of rules has precedence.

For example, if you define two rules for the target COLUMN1, the record value that meets listed conditions for the second rule becomes the best value. If no target passes the second rule, the best values for the first rule become part of the output record.

**Survive stage rule examples**

A Survive stage rule comprises one or more targets and a true conditional statement. The conditional statement consists of a column name, constant or literal values, and comparison or arithmetic operations.

You can define a simple rule or a rule that uses a complex expression. When you define a rule that uses a complex expression, consider the following requirements:

• Rules can have multiple targets.
• Rules must have only one condition.
• A rule can extend over several lines.
• Parentheses () must be used to group complex conditions.
• Integer constants are indicated with a number such as 9.
• String literals are enclosed in double quotation marks. For example, the string literal MARS is specified as "MARS".

**Rule operators**

The Survive stage supports the following types of operators.

<table>
<thead>
<tr>
<th>Mathematical operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equals</td>
</tr>
<tr>
<td>&lt;&gt; or !=</td>
<td>Not Equals</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater Than</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less Than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater Than or Equals</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less Than or Equals</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 11. List of operators for integer columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mathematical operator</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>+</td>
</tr>
</tbody>
</table>
Table 11. List of operators for integer columns (continued)

<table>
<thead>
<tr>
<th>Mathematical operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division (drops the remainder)</td>
</tr>
<tr>
<td>%</td>
<td>Modulo (evaluates the remainder)</td>
</tr>
</tbody>
</table>

Table 12. List of logical operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>Binary logical &quot;and&quot; (expression1 AND expression2 is true if both expressions are true.)</td>
</tr>
<tr>
<td>OR</td>
<td>Binary logical &quot;or&quot; (expression1 OR expression2 is true if either expression is true.)</td>
</tr>
<tr>
<td>NOT</td>
<td>Unary logical &quot;not&quot; (NOT expression1 is true if expression1 is false.)</td>
</tr>
</tbody>
</table>

Table 13. List of supported Survive stage functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZEOF</td>
<td>The number of characters, including spaces in a string-type; the number of decimal digits in an integer-type column</td>
</tr>
<tr>
<td>TRIM</td>
<td>Strips leading and trailing spaces from string-type columns</td>
</tr>
</tbody>
</table>

Examples

Use the following examples to define rules for the Survive stage by using complex expressions.

First record in a group survives
To assign the first record in the group as the best record, select all of the columns as the target and create the following complex expression:

SIZEOF (TRIM b.column) = 0) AND (SIZEOF(TRIM c.column) >= 0)

where column is a column that always has a value.

This rule should appear before any other rules in the Survive stage.

Date column as a target
In this example, YEAR is a column in the record. With the YEAR column selected as the target, you can specify that the current column survives if the current year is greater than the best year. Create the following complex expression:

c.YEAR > b.YEAR

Multiple targets
In this example, NAME, PHONE, and YEAR are columns in the record. With the NAME and PHONE columns selected as the targets, you can specify that current values in the target columns survive if the current year is greater than the best year. Create the following complex expression:
Using the length of data

If you use the data length in a column, the column must be a string column use type.

In this example, GNAMES and MDNAMES are columns in the record. With the GNAMES and MDNAMES columns selected as the targets, you can specify that the given name and middle name survives based on the longest given name. Create the following complex expression:

\[(\text{SIZEOF (TRIM c.GNAMES)} > \text{SIZEOF (TRIM b.GNAMES)})\]

Using the file from which the record originated

To use the file from which the record originated, you assign a file identifier to each record. You then define the condition for that column.

Multiple rules

If you have multiple rules for the surviving column, the value that satisfies the later rule is the survivor. To ensure that the entire record survives the analysis, define one or more rules that target all of the columns in the record.

In this example, TRUE, TYPE, FREQ, FIRSTACC, and V9 are columns in the record. You define three rules that select all of the columns as targets. The rules use the following complex expressions:

- \(\text{c.TYPE <> "DD"}\)
- \(\text{c.FREQ>b.FREQ}\)
- \(\text{c.FIRSTACC = 1}\)

Then, you define a rule that selects the V9 column as a target and uses the following complex expression:

\(\text{(c.V9 > b.V9)}\)

When these rules are applied, records are processed in the following ways:

- If a record satisfies the last rule that targets all of the columns and the value for column FIRSTACC is 1, the record becomes the best record (b.RECORD).
- If more than one record in the group passes the last rule, the latest record processed that satisfies the rule survives.
- If no records pass the FIRSTACC rule, the last record processed that passes the c.FREQ>b.FREQ rule survives.
- If no records pass the FREQ rule, the last record processed that passes the c.TYPE<> "DD" rule survives.
- If no records pass any of the rules, the surviving record is all blanks.

This set of rules has a rule for one of the columns (V9) in the record to survive. Because the V9 rule appears later in the list of rules than the rules for all of the columns, V9 takes precedence over the value that survives for that column in the record. The following example uses three records in a group with the following values:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FREQ</th>
<th>FIRSTACC</th>
<th>V9</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD</td>
<td>3</td>
<td>2</td>
<td>19990401</td>
</tr>
<tr>
<td>DD</td>
<td>4</td>
<td>1</td>
<td>19990314</td>
</tr>
<tr>
<td>DN</td>
<td>5</td>
<td>4</td>
<td>19980302</td>
</tr>
</tbody>
</table>
The Survive stage processes the records using the rules and the second input record survives the rule that targets all of the columns because FIRSTACC=1, but the first input record provides the surviving value for V9. If the FIRSTACC is not equal to 1 for any of these records, the third record survives the rule that targets all of the columns because it has the highest value for the FREQ column. The following record survives:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FREQ</th>
<th>FIRSTACC</th>
<th>V9</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD</td>
<td>4</td>
<td>1</td>
<td>19990401</td>
</tr>
</tbody>
</table>
Chapter 6. Rule sets applied in the data cleansing process

Rule sets check and normalize input data. You can apply cleansing rules to correct the data as it comes in or correct data in multiple databases.

Introduction to rule sets

You can use predefined rule sets, create new rule sets, or copy and modify existing rule sets to extend the scope of your data cleansing project. You can also enhance rule sets in the Standardization Rules Designer.

How rule sets are used

Use rule sets to enforce consistency and quality in your data. You can apply rule sets to data to resolve issues with common data quality problems such as invalid address fields across multiple geographies.

You develop the rules in rule sets from patterns in your data. You then apply the rule sets as you analyze your input data, standardize your data, and prepare your data for matching.

To use rule sets:
1. Determine what data requires cleansing and create requirement documents that map your data input to output that reflects your data cleansing goal. For example, you might need to correct duplicate customer names and addresses in a customer relationship management database.
2. Design rules to parse and standardize your data.
   a. Identify predefined rule sets that you can use as a basis for your data cleansing
   b. For specialized situations where you need greater cleansing flexibility, create custom rule sets, customize copies of existing rule sets, enhance rule sets in the Standardization Rules Designer, or plan to use overrides with existing rule sets.
3. Test and refine your rule sets.

Features and benefits of rule sets

The rule set architecture provides many features and benefits.

The following lists the features and benefits offered by the rule set architecture.

- Support of your business objectives by maximizing your understanding of the critical information contained within data. The data structures created by the rules provide comprehensive addressability to all data elements necessary to meet data storage requirements and facilitate effective matching.
- Modular design that allows for a “plug and play” approach to solving complex standardization challenges.
- Country- and region-specific design for processing of multinational data through country or region standards. The rules conform to the name and address data conventions used within the country or region that you specify.
- The Standardization Rules Designer, a web-based interface that you can use to enhance rule sets.
• Identification and collection of unhandled data. Also, generation of the corresponding unhandled pattern makes it easier to measure the effectiveness of rule sets.

**Objects within rule sets**

Rule sets consist of a specific set of objects that are used to check and normalize input data.

The following stages use the rule set objects to manipulate data during the data cleansing process:
• Investigate
• Standardize
• Multinational Standardize (MNS)

Each rule set requires the following objects:
• Classifications
• Output columns
• Rules

Rule sets might also include extensions, override objects, and lookup tables.

The rule set description file (.PRC) is no longer used.

Detailed information about the set of objects that make up rule sets is in the *IBM InfoSphere QualityStage Pattern-Action Reference*.

**Classifications**

Classifications strengthen the contextual information that patterns provide by identifying that the underlying values belong to particular categories. Each rule set contains its own set of categories, which are called classes.

In IBM InfoSphere QualityStage, records are represented as patterns. In the same way that a record consists of one or more values, patterns consist of one or more abstract characters, each of which represents a class. For example, a set of address data might include the record 123 N CHERRY HILL ROAD, which is represented by the pattern ^D++T. The following table shows the contextual information that each class in the pattern ^D++T provides.

*Table 14. Example of a standard address pattern with the contextual information that each class provides*

<table>
<thead>
<tr>
<th>Input record</th>
<th>Class label</th>
<th>Contextual information that the class provides</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>^</td>
<td>Value that includes only numbers</td>
</tr>
<tr>
<td>N</td>
<td>D</td>
<td>Street direction</td>
</tr>
<tr>
<td>Cherry</td>
<td>+</td>
<td>Value that includes only letters</td>
</tr>
<tr>
<td>Hill</td>
<td>+</td>
<td>Value that includes only letters</td>
</tr>
<tr>
<td>Road</td>
<td>T</td>
<td>Street type</td>
</tr>
</tbody>
</table>
Patterns contain the following types of classes:

- **Default classes** provide basic information about the type of the value, such as whether the value is comprised of alphabetic characters, numeric characters, or some combination of both.
- **Custom classes** provide stronger contextual information about the type of the value. In a data set that contains retail product information, custom classes might be used to indicate whether an alphabetic value is the name of a product or the name of a brand. The one-character label for custom classes can be any letter in the Latin alphabet or 0, which indicates a null class.

Rule sets use classifications to identify and classify key values. For example, a rule set for address data might use classifications to categorize values that are street types (AVE, ST, RD) or directions (N, NW, S) by providing the following information:

- Standard abbreviations for each word; for example, HWY for Highway
- A list of one-character labels that represent classes and that are assigned to individual data elements during processing

Classifications are added and modified by editing the classifications table (previously called .CLS file), enhancing a rule set in the Standardization Rules Designer, or using the user classification override.

**Class types**

Classes provide contextual information about values. Default classes provide basic information about the type of the value, and custom classes provide stronger contextual information about the type of the value.

**Default classes**

If a value is not assigned to a custom class, the value has one of the default classes, or basic pattern classes, that are shown in the following table.

<table>
<thead>
<tr>
<th>Class label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>Digits only.</td>
</tr>
<tr>
<td></td>
<td>The caret (^) class represents a single number, for example, the number 123. However, the string 1,230 uses three values (previously called tokens): the number 1, a comma, and the number 230.</td>
</tr>
<tr>
<td>?</td>
<td>One or more consecutive words that are not assigned to a custom class.</td>
</tr>
<tr>
<td></td>
<td>The Standardization Rules Designer does not use the ? class.</td>
</tr>
<tr>
<td>+</td>
<td>Letters only.</td>
</tr>
<tr>
<td>&amp;</td>
<td>A single value of any type.</td>
</tr>
<tr>
<td>&gt;</td>
<td>Leading digits, followed by letters.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Leading letters, followed by digits.</td>
</tr>
<tr>
<td>@</td>
<td>Mixed letters and digits.</td>
</tr>
<tr>
<td></td>
<td>For example: A123B, 345BCD789.</td>
</tr>
<tr>
<td>~</td>
<td>Special characters that are not in the SEPLIST, which is the list of characters that indicate where one value in a record ends and the next value begins.</td>
</tr>
<tr>
<td>k</td>
<td>One or more Chinese numeric characters.</td>
</tr>
</tbody>
</table>
**Custom classes**

Custom classes are defined by users. The label for a custom class can be an uppercase alphabetic character or the number 0, which indicates a null class.

A custom class provides stronger contextual information about a value than a default class. For example, if a classification definition does not assign the value ROAD to a custom class, the value is assigned to the + default class. This default class indicates that the value is a single alphabetic word. If a classification definition assigns the value to a custom class that represents street types, which might be represented by the character T, the value provides more contextual information. When this information is provided, you can write rules that address a specific subset of the data and therefore handle that data more effectively.

**The null class**

The null class, which has the label 0, is used in a classification definition or in a RETYPE action to make a value NULL. Because a value with the null class never matches anything, the value is never used in a pattern and is not processed.

If you assign a value to the null class, the value is skipped in the pattern matching process.

You can find more information about the null class and the RETYPE action in the *IBM InfoSphere QualityStage Pattern-Action Reference*.

**Classification definitions**

A classification definition assigns a value to a class. The definition can include additional information about the value and affect other similar values.

A classification definition has the following parts:

**Value**   The string of one or more characters that you want to add a definition for.

**Standard value**

A standardized spelling or representation of the value that can be used as part of an action or condition in a rule. If you do not specify a standard value, it is the same as the value.

The standard value might be an abbreviation or expanded variation of the word. For example, the standard value for WEST might be W, and the standard value for POB might be "PO BOX".

In the classifications table (previously called the .CLS file), the maximum length for a standard value is 25 characters.

In the classification definition for a value in the null class, the standard value is not required.

**Class**   The class that the value is assigned to. The class is represented by a one-character class label. For more information about class types, see "Class types" on page 165.

**Similarity threshold (previously called threshold weight)**

The degree of variation that can exist in the spelling or representation of the value. If you want the classification definition to affect values that are different from the value in the definition, you can set the similarity threshold lower than the default of 900.
The similarity threshold must be an integer in the range 700 - 900. The integers represent the following degrees of variation:

900    Strings must match exactly.
800    Strings are almost certainly the same.
750    Strings are probably the same.
700    Strings are probably different.

When the rule set that contains a classification definition is applied to data, values in the data are compared and a score is assigned. This score indicates the degree of similarity between two values. The string comparison method that is used can take into account phonetic errors, random insertion, deletion and replacement of characters, and transposing of characters.

The score is weighted by the length of the value because small errors in long values are less serious than errors in short values. Because errors in short values cannot generally be tolerated, do not specify a similarity threshold for short values.

**Classifications table (.CLS file)**

In a rule set, the classifications table (previously called the .CLS file) contains a list of classification definitions. A classification definition assigns a value to a class.

The header of the classifications table includes the name of the rule set and the legend for the classifications. The legend indicates the classes that are used in the classification definitions and descriptions for those classes. All of the lines in the header are specified as comments by preceding any text with semicolons. For example, the header in a classifications table for a rule set that handles retail product data might include the following lines:

```
;--------------------------------------------------------
; Retail Product Classification Table
;--------------------------------------------------------
; Classification Legend
;--------------------------------------------------------
; B - Product Brand
; C - Product Color
; N - Product Name
; S - Product Size
; T - Product Type
```

After the header, the file contains the following strings:

```
;::ProductName vn.n
\FORMAT\ SORT=N
```

Do not include any other comments before these lines.

After the header and introductory strings, each line in the classifications table includes one classification definition. In the classifications table, classification definitions use the following format:

```
value standard value class [similarity-threshold] [, comments]
```

In the classifications table, each value must be a single word. Multiple or compound words, such as New York, North Dakota, or Rhode Island, are considered separate values.
Literals in the classifications table

Literals are characters that are entered instead of a string in one of the parts of a classification definition.

Some characters that function as literals are also used as labels for default classes. To specify one of these characters as a literal, you must enter an escape character before the character that you want to use as a literal.

When you enter a classification definition in the classifications table, you can use the literals and escape characters that are shown in the following table.

Table 16. Literals and escape characters in the classifications table

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>The ampersand (&amp;) is a class that indicates a single value of any type. However, you can type the backslash () escape character before the ampersand to use the ampersand as a literal.</td>
</tr>
<tr>
<td>/</td>
<td>Literal.</td>
</tr>
<tr>
<td>/</td>
<td>You can use the backslash () escape character with the forward slash (/) in the same manner that you use the forward slash (/) character.</td>
</tr>
<tr>
<td>-</td>
<td>Literal.</td>
</tr>
<tr>
<td>-</td>
<td>You can use the backslash () escape character with the hyphen in the same manner that you use the hyphen (-) character.</td>
</tr>
<tr>
<td>#</td>
<td>Literal. You must use this character with the backslash () escape character, for example: #.</td>
</tr>
<tr>
<td>()</td>
<td>Literal. The parentheses are used to enclose operands or user variables in a pattern syntax.</td>
</tr>
<tr>
<td>( and )</td>
<td>Use the backslash () escape character with the opening parenthesis or closing parenthesis to filter out parenthetical remarks.</td>
</tr>
</tbody>
</table>

Lookup tables

Lookup tables contain a set of definitions. Rules can reference lookup tables as part of actions or conditions.

Actions or conditions might use a lookup table in the following ways:
- An action or condition can compare a particular value to a value in the lookup table. For example, a condition can stipulate that a rule handles a record only when a value in a particular position in the record is in the lookup table.
- An action can convert a particular value to a value in the lookup table. For example, an action might use a lookup table that contains geographic information to convert numeric place codes to place names.

Some rule sets use one or more lookup tables. For example, a lookup table that contains the gender that is associated with particular names is included in the name rule sets.

Lookup tables are added and modified by adding or editing .TBL files or enhancing a rule set in the Standardization Rules Designer.
Lookup table definitions
A lookup table definition includes the value that is looked up, a returned value, and a similarity threshold.

A lookup table definition has the following parts:

Value  The string of one or more characters that you want to add a lookup table definition for.

Returned value  A string of one or more characters that can be used by an action or condition in a rule instead of the value. If you do not specify a returned value, it is the same as the value.

Similarity threshold (previously called threshold weight)  The degree of variation that can exist in the spelling or representation of the value. If you want the definition to affect values that are different from the value in the definition, you can set the similarity threshold lower than the default of 900.

The similarity threshold must be an integer in the range 700 - 900. The integers represent the following degrees of variation:

- **900**  Strings must match exactly.
- **800**  Strings are almost certainly the same.
- **750**  Strings are probably the same.
- **700**  Strings are probably different.

When the rule set that contains a lookup table definition is applied to data, values in the data are compared and a score is assigned. This score indicates the degree of similarity between two values. The string comparison method that is used can take into account phonetic errors, random insertion, deletion and replacement of characters, and transposing of characters.

The score is weighted by the length of the value because small errors in long values are less serious than errors in short values. Because errors in short values cannot generally be tolerated, do not specify a similarity threshold for short values.

Output columns
Each rule set contains a list of output columns. Standardized data is added to these columns.

Output columns are defined in the dictionary (previously called the .DCT file).

In the IBM InfoSphere QualityStage Standardization Rules Designer, you can add or modify rules that map input data to output columns. You can also specify leading separators, which separate strings in an output column.

Most rule sets include the types of output columns that are shown in the following table.
Table 17. Types of output columns

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Example columns</th>
</tr>
</thead>
</table>
| Business intelligence | Contain the standardized values. These columns are often arranged in an order that is suited to the domain. For example, in the predefined rule set that is applied to US address data, the house number output column is before the street name output column. | • HouseNumber  
• HouseNumberSuffix  
• StreetPrefixDirectional |
| Matching           | Contain information that is used exclusively for matching.                  | • StreetNameSOUNDEX  
• MatchFirstNameNYSIIS  
• CityNameRVSNDX |
| Reporting          | Contain information that is used to evaluate the standardization results.   | • UnhandledData  
• InputPattern  
• ExceptionData |

Dictionary (.DCT file)

The dictionary (previously called the .DCT file) defines the output columns that the rule set provides on the output page of the Standardize stage.

The dictionary holds a list of domain, matching, and reporting fields. Each field is identified by a unique descriptive name. The dictionary also provides the data type, such as character or integer, and length information.

The following example shows the format for the dictionary.

```
fld-identifier field-type field-length
  missing-value-identifier [ description ] [ comments ]
```

The table explains the dictionary format.

Table 18. Dictionary format

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>field-identifier</strong></td>
<td>A descriptive field name that follows the package rules and is unique for all dictionaries.</td>
</tr>
<tr>
<td><strong>field-type</strong></td>
<td>The type of information in the field. For more information about field types see Field types.</td>
</tr>
<tr>
<td><strong>field-length</strong></td>
<td>The field length in characters.</td>
</tr>
</tbody>
</table>
| **missing-value-identifier** | Optional. Serve as placeholders. The possible values are:  
• S - spaces  
• Z - zero or spaces  
• N - negative number, such as -1  
• 9 - all nines, such as 9999  
• X - no missing value  
Generally, use X or S for this argument. |
Table 18. Dictionary format (continued)

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>Optional. Describe the field.</td>
</tr>
<tr>
<td>; comments</td>
<td>Optional. Any additional comments about the field. Comments can continue on separate lines if the comments are preceded by a semicolon.</td>
</tr>
</tbody>
</table>

The following string must appear as the first two lines of a dictionary:

```plaintext
;;QualityStage v8.0
\FORMAT\ SORT=N
```

The following example shows part of a USADDR dictionary:

```plaintext
;;QualityStage v8.0
\FORMAT\ SORT=N
;----------------------------------
; USADDR Dictionary File
;----------------------------------
; Total Dictionary Length = 411
;----------------------------------
; Business Intelligence Fields
;----------------------------------
HouseNumber C 10 S HouseNumber ;0001-0010
HouseNumberSuffix C 10 S HouseNumberSuffix ;0011-0020
StreetPrefixDirectional C 3 S StreetPrefixDirectional ;0021-0023
```

The order of fields in the dictionary is the order in which the fields appear on the output tab in the stage. When you map input data to the output columns, you can change the order.

Field types:

The following field types are supported by the dictionary. In the output, the SQL type is varchar.

Table 19. Field type definitions

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Numeric fields, which are right-aligned and filled with leading blanks.</td>
</tr>
<tr>
<td>C</td>
<td>Alphabetic fields, which are left-aligned and filled with trailing blanks.</td>
</tr>
<tr>
<td>NS</td>
<td>Numeric field in which leading zeros are stripped. For example, you define the house number field as NS and the ZIP Code as N. Use NS because leading zeros are relevant with ZIP codes, but could interfere with matching on the house number.</td>
</tr>
</tbody>
</table>
Table 19. Field type definitions (continued)

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Mixed alphabetic and numeric information, in which numeric values are right-aligned and alphabetic values are left-aligned. Leading zeros are retained. The U.S. Postal Service uses this type of field for house numbers and apartment numbers. For example, a four-character type M field, where b represents a space, is:</td>
</tr>
<tr>
<td></td>
<td>• 102 becomes b102</td>
</tr>
<tr>
<td></td>
<td>• A3 becomes A3bb</td>
</tr>
<tr>
<td>MN</td>
<td>Mixed name, which is often used for representing street names. Field values that begin with a letter are left-aligned. Field values that begin with a number are indented as if the number is a separate four-character field.</td>
</tr>
<tr>
<td></td>
<td>In the following examples, b represents a space. The one-digit numbers are indented three spaces, two-digit numbers are indented two spaces, and so on. The U.S. Postal Service uses this type of field for street names in the ZIP+4 files.</td>
</tr>
<tr>
<td></td>
<td>• MAIN</td>
</tr>
<tr>
<td></td>
<td>• CHERRY HILL</td>
</tr>
<tr>
<td></td>
<td>• bbb2ND</td>
</tr>
<tr>
<td></td>
<td>• bb13TH</td>
</tr>
<tr>
<td></td>
<td>• b123RD</td>
</tr>
<tr>
<td></td>
<td>• 1023RD</td>
</tr>
</tbody>
</table>

Rules

Rules are processes that standardize groups of related records. Rules can apply to records that match the same pattern or to exact strings of text.

When you create or modify a rule, you map values in the input records to output columns, specify actions that manipulate the data, and identify conditions to ensure that rules apply only to the correct records.

Rules are added and modified by editing the pattern-action specification (previously called pattern-action file), enhancing a rule set in the IBM InfoSphere QualityStage Standardization Rules Designer, or using override objects.

In the pattern-action specification, patterns are executed in the order that they appear. A pattern either matches the input record or does not match. If it matches, the actions that are associated with the pattern are executed. If it does not match, the actions are skipped. In either case, processing continues with the next pattern in the file.

Actions

An action is a part of a rule that specifies how the rule processes a record. You can add one or more actions for each value in a record.
You specify actions in the Standardization Rules Designer, pattern-action specification, or user overrides. Regardless of where the action is specified, every action that manipulates a particular record includes the following parts.

Table 20. Parts of an action

<table>
<thead>
<tr>
<th>Part</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>An object that is acted upon</td>
<td>• Value</td>
</tr>
<tr>
<td></td>
<td>• Standard value</td>
</tr>
<tr>
<td></td>
<td>• The first three characters in a value</td>
</tr>
<tr>
<td></td>
<td>• Literal</td>
</tr>
<tr>
<td>One or more manipulations of the object</td>
<td>• Copy the object from one field to a different field</td>
</tr>
<tr>
<td></td>
<td>• Look up the object in a lookup table and convert it to a returned value</td>
</tr>
<tr>
<td></td>
<td>• Concatenate the object with a different object</td>
</tr>
<tr>
<td>A target for the resulting character string</td>
<td>• Output column</td>
</tr>
<tr>
<td></td>
<td>• User variable (pattern-action specification only)</td>
</tr>
</tbody>
</table>

In the pattern-action specification, you can also specify actions that affect how records are processed by the rule set and the order in which they are processed. For example, you can use the CALL action to call subroutines that process particular types of information, such as unit types.

Conditions
A condition specifies requirements that records must meet before the actions in a rule are applied to that record. You can add one or more conditions for each value in a record.

You specify conditions in the Standardization Rules Designer or pattern-action specification. Regardless of where the condition is specified, all conditions have the following parts.

Table 21. Parts of a condition

<table>
<thead>
<tr>
<th>Part</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>An object that the condition applies to</td>
<td>• Value</td>
</tr>
<tr>
<td></td>
<td>• Standard value</td>
</tr>
<tr>
<td></td>
<td>• The first three characters in a value</td>
</tr>
<tr>
<td>Requirements that the object must meet for the rule to apply to that object</td>
<td>• The object equals a particular value</td>
</tr>
<tr>
<td></td>
<td>• The object is in a lookup table</td>
</tr>
<tr>
<td></td>
<td>• The length of an object is greater than or equal to a particular value</td>
</tr>
</tbody>
</table>

For example, your data might contain distinct values that represent calendar dates in the *ccyy mmm dd* format. A current record in this format is 1986 03 16. Suppose that your data quality requirements require a rule that concatenates these values into one value in the format *ccyyymmd*. 
To ensure that the rule applies only to calendar dates in the correct format, you add a series of conditions that requires the length of the first value to be equal to four characters and the length of the second and third values to be equal to two characters. When you add this condition, the rule applies to records such as 1986 03 16, but not to records such as 03 16 86.

**Pattern matching principles**

To obtain correct standardization, you need to understand the concepts of pattern matching and the reasons for matching.

If all elements of an address are uniquely identified by keywords, address standardization is easy. The following example is not subject to any ambiguity. The first field is numeric (house number), the next is a direction, which is uniquely identified by the value (previously called token) N, the next is an unknown word MAPLE, and the last is a street type, AVE:

123 N MAPLE AVE

Most addresses fall into this pattern with minor variations.

123 E MAINE AV
3456 NO CHERRY HILL ROAD
123 SOUTH ELM PLACE

The first numeric value is interpreted as the house number and must be moved to the house number field {HouseNumber}. The direction is moved to the pre-direction field {StreetPrefixDirectional}, the street names to the street name field {StreetName}, and the street type to the {StreetSuffixType} field.

The braces indicate that the reference is to a dictionary field that defines a field in the output data. For example:

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Dictionary field</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric value. The class is ^\d+.</td>
<td>{HouseNumber}</td>
<td>123 3456 123</td>
</tr>
<tr>
<td>Direction. The class is D.</td>
<td>{StreetPrefixDirectional}</td>
<td>E (an abbreviation for East) NO (an abbreviation for North) SOUTH</td>
</tr>
<tr>
<td>Unclassified words. The class is ?.</td>
<td>{StreetName}</td>
<td>MAIN CHERRY HILL ELM</td>
</tr>
<tr>
<td>Street type. The class is T.</td>
<td>{StreetSuffixType}</td>
<td>AV (an abbreviation for Avenue) ROAD PLACE</td>
</tr>
</tbody>
</table>
Rule groups

A rule group is a collection of rules that are applied to records at the same point in the standardization process. To ensure that rules are applied in a particular order, you can organize the rules into rule groups in the Standardization Rules Designer.

Rule groups can contain rules that are applied to records before or after the other actions in the standardization process. A rule group is invoked by a separate action in the pattern-action specification.

For example, the following action invokes the Hardware_Retail rule group. The example includes comments that specify the rule group that is invoked.

```
; Rules for hardware retail products
; ----------------------------------
; ----------------------------
& ; CALL Hardware_Retail SUBROUTINE
CALL Hardware_Retail
```

The Hardware_Retail rule group might include rules like the following rule:

```
B | + | S | C | P ; Common Pattern Found: CALL Post_Process SUBROUTINE then EXIT
COPY_A [1] {ProductBrand}
COPY_S [2] {ProductName}
COPY_A [3] {ProductSize}
COPY_A [5] {ProductUnitPrice}
CALL Post_Process
EXIT
```

For most rule sets, you might need only the following rule groups:
- A rule group for rules that you want to apply to records before all other actions
- A rule group for rules that you want to apply to records after all other actions

In IBM InfoSphere QualityStage Version 9.1 and later, the pattern-action specification for predefined rule sets contains the following rule groups:
- The Input_Overrides rule group contains rules that are applied to records before all other actions in the pattern-action specification.
- The Unhandled_Overrides rule group contains rules that are applied to records after all other actions in the pattern-action specification.

You cannot add rules that are based on pattern-action language to these rule groups.

In the Standardization Rules Designer, you can modify the rules in a rule group, add a rule group, or change the name of a rule group. For a rule set to work correctly, the references to the rule groups in the pattern-action specification must match the information about the rule groups in the Standardization Rules Designer. Before you provision a rule set and apply it in a job, ensure that changes are published from the Standardization Rules Designer and that the pattern-action specification is updated to match the Standardization Rules Designer.

In the pattern-action specification, the rule groups for the Standardization Rules Designer must be added to the following types of rule sets:
- Custom rule sets that you created before IBM InfoSphere QualityStage Version 9.1
- Copies of predefined rule sets that were made before IBM InfoSphere QualityStage Version 9.1
Pattern-action specification (.PAT file)
The pattern-action specification (previously called pattern-action file) is an ASCII file that can be created or updated using any standard text editor.

The pattern-action specification has the following general format:
\POST_START
post-execution actions
\POST_END
\PRAGMA_START
specification statements
\PRAGMA_END

pattern
actions

pattern
actions

pattern
actions

...

There are two special sections in the pattern-action specification. The first section consists of post-execution actions within the \POST_START and \POST_END lines. The post-execution actions are executed after the pattern matching process is finished for the input record.

Post-execution actions include computing Soundex codes, NYSIIS codes, reverse Soundex codes, and reverse NYSIIS codes, and copying, concatenating, and prefixing dictionary field value initials.

The second special section consists of specification statements within the \PRAGMA_START and \PRAGMA_END lines. The only specification statements currently allowed are SEPLIST, STRIPLIST, and TOK. The special sections are optional. If omitted, the header and trailer lines must also be omitted.

Other than the special sections, the pattern-action specification consists of standardization rules. Standardization rules include one or more conditions, such as a pattern, and the associated actions. The pattern requires one line. The actions are coded one action per line. The next pattern can start on the following line.

Blank lines are used to increase readability. For example, it is suggested that blank lines or comments separate one rule from another.

Comments follow a semicolon. An entire line can be a comment line by specifying a semicolon as the first non-blank character; for example:

; This is a standard address pattern
; ^ | ? | T ; 123 Maple Ave

Consider the following input entries:
123 N MAPLE AVE
123 MAPLE AVE

The following sample code shows how the post actions compute a NYSIIS code for street name and process patterns to handle the above input entries:
This example pattern-action specification has a post section that computes the NYSIIS code of the street name (in field {StreetName}) and moves the result to the {StreetNameNYSIIS} field.

The first pattern matches a numeric value followed by a direction followed by one or more unknown words followed by a street type (as in 123 N MAPLE AVE). The associated actions are to:
1. Copy operand [1] (numeric value) to the {HouseNumber} house number field.
2. Copy the standard abbreviation of operand [2] to the {StreetPrefixDirectional} prefix direction field.
3. Copy and retain spaces between the words in operand [3] to the {StreetName} field.
4. Copy the standard abbreviation of the fourth operand to the {StreetSuffixType} street type field.
5. Exit the pattern program. A blank line indicates the end of the actions for the pattern.

The second rule is similar except that the rule handles cases like 123 MAPLE AVE. If there is no match on the first pattern, the next pattern in the sequence is attempted.

**Special characters**

At the beginning of the standardization process, input data is parsed into meaningful values. Special characters are used to identify distinct values and distinguish between values and characters that do not contain useful information.

Rule sets use the following types of special characters:
- Separation characters indicate where one value in a record ends and the next value begins. If a character is in the separation list but is not in the strip list, the character is identified as a distinct value.
- Strip characters are removed from the record. For example, if a period (.) is in the strip list but is not in the separation list, the characters N.W. in the raw data are parsed into the following value: NW

In the pattern-action specification, separation characters are specified in the separation list, which is specified by using the SEPLIST statement. Strip characters are specified in the strip list, which is specified by using the STRIPLIST statement. When input data is parsed, the separation list is applied first.

Any character in both lists separates values but is not identified as a distinct value itself. For example, if a space is in both lists, the characters 123 456 in the raw data
are parsed into the following two values: 123 and 456. Because the space is in both lists, it separates the two values but is not a value itself. When you specify patterns in the pattern-action specification, you cannot include any character that is in both lists in a pattern.

Examples

In this example, the space is in both lists and the hyphen is in the strip list but not the separation list. Hyphens are stripped so that STRATFORD-ON-AVON is considered to be STRATFORDONAVON.

SEPLIST: " !%$,:;\(/#&"
STRIPLIST: " !+*,:-\''"

In this example, the hyphen is in both lists. Because the separation list is applied before the strip list, STRATFORD-ON-AVON in the incoming data is parsed into three values: STRATFORD, ON, and AVON.

SEPLIST: " !%$,:;\(/#&"
STRIPLIST: " !+.*,:-\''"

Rule set extensions

Rule set extensions are created by enhancing a rule set in the Standardization Rules Designer. Rule set extensions change the way that data is processed by a rule set.

Extensions enhance the existing components of a rule set. Before extensions are added, a rule set might contain enough information to standardize all of the data in a domain, only some of the information that is required to standardize data, or no information at all. For example, a classification file might contain classification definitions for some of the values in your data set. In the Standardization Rules Designer, you can add classification extensions that change existing definitions or add classification definitions that are not in the classifications file.

Rule set extensions are stored in the following files:

- The .RCC file contains classification extensions.
- The .RCR file contains rule extensions.
- One or more .RCT files contain lookup table extensions.

Although override object types are available, rule set extensions provide a greater number of standardization options. If a rule set uses overrides, you can convert the overrides to rule set extensions and work exclusively in the Standardization Rules Designer.

Overrides

You can use overrides to alter the processing of the data as specified in the classifications table and the pattern-action specification.

You can indicate, to a limited degree, how you want to handle the data that you identify as overrides. You can also indicate where you want the data to be placed in the output.

You use the Overrides windows in the Designer client to edit the contents of the overrides.
Input overrides take precedence over the behavior of the rule set files. The input overrides are applied against the input pattern before the main processing. The input overrides are as follows:

- User classification
- Input text
- Input pattern

Unhandled overrides are executed after the main processing is complete. The unhandled overrides are as follows:

- Unhandled text
- Unhandled pattern

The domain preprocessor rule sets also have the input overrides that are listed above and two additional input overrides:

- Field text
- Field pattern

The domain preprocessor rule sets do not have unhandled overrides.

### Categories of predefined rule sets

You can choose from various predefined rule sets in your data cleansing project.

If you know already know that the data you are processing is specific to a country or region and the country or region is already identified (for example an ISO code or the country or region name is part of the data) you do not need to use the COUNTRY rule set. If the data you are processing is already grouped into names, addresses, and area information, then you do not need to use the PREP rule set.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region preprocessor</td>
<td>Identifies country or region domain from address data.</td>
<td>The rule set is named COUNTRY</td>
</tr>
<tr>
<td>Domain preprocessor</td>
<td>For a specific country or region, identifies and assigns a data domain to the name, address, and area columns in each record. The objective of these rule sets is to identify if the incoming information belongs to the name, address, or area domain. You can use the output that is generated when this type of rule set is applied as the input to the country or region appropriate domain-specific rule sets.</td>
<td>One for each country or region: PREP (preprocessor)</td>
</tr>
</tbody>
</table>
### Table 23. Categories of rule sets (continued)

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain-specific</td>
<td>For a specific country or region, standardizes each data domain. The most common rule sets are as follows:</td>
<td>Most countries or regions have three domains: NAME, ADDR, and AREA. Some countries or regions, for example Japan, have several more domain-specific rule sets.</td>
</tr>
<tr>
<td></td>
<td>Name including individual names, organization names, attention instructions, and secondary names.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Address including unit number, street name, type, and directions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Area including cities, states, and postal codes (ZIP code in the USA, for example).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Phone including area code and extension numbers.</td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td>For a specific country or region, standardizes and validates the format and value of common business data including:</td>
<td>VTAXID is intended for only the United States. VPHONE supports Canada, the Caribbean, and the United States. VEMAIL and VDATE can be used internationally.</td>
</tr>
<tr>
<td></td>
<td>Phone number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tax ID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E-mail address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>Multinational Standardize</td>
<td>For multiple countries and regions, standardizes and verifies international address data.</td>
<td>The MNS rule sets are used by the MNS stage.</td>
</tr>
</tbody>
</table>

### Country or region identifier rule set

The country or region identifier rule set reads area information and attempts to identify the associated country or region. The rule set name is COUNTRY.

The purpose of the country or region identifier rule set is to assign to each input record the appropriate 2-byte ISO territory code that is associated with the geographic origin of the record’s area information: city, state or province, postal code or ZIP code, and, possibly, country or region name.

The COUNTRY rule set is usually used when the input file represents multiple countries and regions and there is no reliable country or region indicator. If the data includes a value (previously called token) that indicates the country or region, the rule set uses that information to populate the ISO territory output field.

If a country or region name is not found in the data, the rule set evaluates the area information to determine if the address belongs in one of the following countries:
• Australia
• Canada
• France
• Germany
• Italy
• Spain
• United Kingdom
• United States

If the area information does not match any of the country or region patterns, a default delimiter, which is specified in the Literal field in the Standardize stage, is assigned.

The country or region identifier rule set can help you with the following activities:
• Investigation: to determine if an input file contains multi-national data. The word frequency output file provides a distribution of the country or region words.
• Input preparation: to facilitate segmenting the input file into country- or region-specific subsets for country- or region-specific processing. Then, you can use the output field to filter records by country or region in subsequent processes.

After you create this output file, you can use a Filter stage to create a file that contains only one country or region. You can use the file with a domain-preprocessing rule set for the appropriate country or region.

**Country or region code delimiters**
The country or region identifier rule set (COUNTRY) uses a default delimiter when the rule set cannot determine the country or region of origin for a record.

When using the COUNTRY rule set and defining the rule process, you must specify a default delimiter. The delimiter must be defined before you can run the rule in a job.

The default delimiter format is:

```
ZQ<Two-Byte ISO territory code>ZQ
```

For example, default delimiter for the United States of America is ZQUSZQ.

Use the territory code that you think represents most records, the most frequently occurring country or region in the input data. For example, if your data contains some Canadian data but is largely United States-based, specify US as the territory code in the delimiter.

You specify the default delimiter in the Literal field in the Standardize stage.

When the country or region identifier rule set cannot determine the territory code, the default value is taken from the default delimiter and assigned to the record.

**Output data for the COUNTRY rule set**
You can find the following fields in the output data of the COUNTRY rule set:
• A two-byte ISO territory code. The code is associated with the geographic origin of the address and area information in the record.
An identifier flag.
The values are:

Y  The rule set identifies the country or region.
N  The rule set cannot identify the country or region and uses the value of
    the default delimiter.

**Domain preprocessor (PREP) rule sets**

Domain pre-processor rule sets evaluate mixed-domain input, such as free-form name and address information, and categorize the data into domain-specific column sets.

After the proper domains are identified, you can use the domain-specific rule sets to create the appropriate standardized structures. Domain preprocessor rule sets evaluate the mixed-domain input from a file for a specific country.

Domain preprocessor rule sets follow a naming convention that starts with a country or region abbreviation and ends with PREP (which is an abbreviation for preprocessor). See the following table for examples.

<table>
<thead>
<tr>
<th>Rule set name</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>USPREP</td>
<td>United States</td>
</tr>
<tr>
<td>GBPREP</td>
<td>Great Britain</td>
</tr>
<tr>
<td>CAPREP</td>
<td>Canada (English speaking)</td>
</tr>
</tbody>
</table>

**Preparing the input data for the domain preprocessor rule set**

The order of columns in input data is not used by domain preprocessor rule sets to determine data domains: name, address or area. You must use a delimiter to describe the data in columns.

You must insert at least one metadata delimiter for a column in your input record. Delimit every column or group of columns. The delimiter indicates what kind of data you are expecting to find in the column based on one or more of the following:

- Metadata description
- Investigation results
- An informed estimate

You can refer to classifications (.CLS) in your domain preprocessor (PREP) rule set for delimiter names specified for your input data. Some of the delimiter names are listed here:

**ZQPUNTNZQ**
Automatically defaults the entire field to the name domain.

**ZQMIXNZZQ**
Field overrides and field modifications are applied, and then the rule set checks the field for name, address, and area data (in that order). Any information that is not assigned a domain defaults to the name domain.

**ZQNAMEZQ**
Name delimiter. Field overrides and field modifications are applied, and then the rule set checks for common name patterns. If name patterns are
not found, the rule set checks for address and area patterns. If address and area patterns are not found, the field defaults to the name domain.

**ZQPUTAZQ**
Automatically defaults the entire field to the address domain.

**ZQMIXAZQ**
Field overrides and field modifications are applied, and then the rule set checks the field for name, address, and area data (in that order). Any information that is not assigned a domain defaults to the address domain.

**ZQADDRZQ**
Address delimiter. Field overrides and field modifications are applied, and then the rule set checks for common address patterns. If address patterns are not found, the rule set checks for name and area patterns. If name and area patterns are not found, then the field defaults to the address domain.

**ZQPUTRZQ**
Automatically defaults the entire field to the area domain.

**ZQMIXRZQ**
Field overrides and field modifications are applied, and then the rule set checks the field for name, address, and area data (in that order). Any information that is not assigned a domain defaults to the area domain.

**ZQAREAQ**
Area delimiter. Field overrides and field modifications are applied, and then the rule set checks for common area patterns. If area patterns are not found, the rule set checks for name and address patterns. If name and address patterns are not found, then the field defaults to area.

The expected input into a domain preprocessor rule set is as follows:
- At least one metadata delimiter for a column in your input record
- Up to 6 delimiters with fields following each delimiter. The fields indicate which domain the data most likely represents: name, address, or area.
- Multiple input columns can follow one delimiter; for example, ZQAREAQ City State ZIPCode

You must use literals to insert column delimiters. You use the Literal field in the Standardize Rule Process window to insert the delimiters.

The domain preprocessor rule set performs the following actions:
- Checks to see if delimited fields belong in a domain other than the default domain.
- Sends all data out to the appropriate domain output field in the order it was entered.
- Note about the MIX delimiter: most delivered rule sets have minimal processing when a MIX delimiter is specified; user overrides are necessary.

Note: If you fail to enter at least one metadata delimiter for the input record, you receive the following error message: PRE-PROCESSOR ERROR - NO METADATA DELIMITERS WERE SPECIFIED

**Domain preprocessor example**
Column sets and metadata labels do not necessarily provide enough information about data content. Preprocessing categorizes the input data into domain-specific column sets: name, address, and area.
Because input files are typically not domain-specific, these rule sets are critical when preparing a file for standardization.

Columns can contain data that do not match their metadata description. Here is an example that shows the metadata label and the data content:

Name 1
  John Doe

Name 2
  123 Main Street Apt. 456

Address 1
  C/O Mary Doe

Address 2
  Boston, MA 02111

Domains are not specified in the input. The following example shows domain names and data content:

Name  John Doe
NameC/O Mary Doe
Address 123 Main Street Apt. 456
Area Boston, MA 02111

In addition, other problems arise when:
- Information continues across multiple column sets.
- More than one data domain is present within a single column set.

The following set, that shows the domain name and data content, is an example:

Name 1
  John Doe and Mary

Name 2
  Doe 123 Main Street

Address 1
  Apt. 456 Boston

Address 2
  MA 02111

The domain names and data content are as follows:

Name  John Doe and Mary Doe
Address 123 Main Street Apt. 456
Area  Boston, MA 02111

**Objects in domain preprocessor rule sets**
Each rule set has a group of objects associated with it.

The naming convention for the domain preprocessor rule sets is:
For example, here are the objects in the United States domain preprocessor rule set:

**USPREP.CLS**
Classifications table

**USPREP.DCT**
Dictionary

**USPREP.PAT**
Pattern-action specification

**USPREP.FPO**
Field pattern overrides

**USPREP.FTO**
Field text overrides

**USPREP.IPO**
Input pattern overrides

**USPREP.ITO**
Input text overrides

**USPREP.UCL**
User classifications

**Overrides for domain preprocessor rule sets**
The domain preprocessor overrides and their abbreviations let you specify your own custom conditioning rules.

The following table describes the overrides for standard domain preprocessor rule sets.

<table>
<thead>
<tr>
<th>Domain preprocessor override names</th>
<th>Object type abbreviation</th>
<th>Example (United States)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-defined classifications</td>
<td>UCL</td>
<td>USPREP.UCL</td>
</tr>
<tr>
<td>Input pattern overrides</td>
<td>IPO</td>
<td>USPREP.IPO</td>
</tr>
<tr>
<td>Input text overrides</td>
<td>ITO</td>
<td>USPREP.ITO</td>
</tr>
<tr>
<td>Field pattern overrides</td>
<td>FPO</td>
<td>USPREP.FPO</td>
</tr>
<tr>
<td>Field text overrides</td>
<td>FTO</td>
<td>USPREP.FTO</td>
</tr>
</tbody>
</table>

**Domain preprocessor dictionary**
The domain preprocessor rule set contains a dictionary.

There are two types of fields in the dictionary of a domain preprocessor rule set:

- Domain fields
- Reporting fields
Domain columns for domain preprocessor rule sets:

Domain fields are used to organize values (previously called tokens) according to a rule set. In the stage, the fields are mapped to columns.

Domain preprocessor rule sets move every input values to one of the following domain fields:

**NameDomain**
- All input values belonging to the name domain

**AddressDomain**
- All input values belonging to the address domain

**AreaDomain**
- All input values belonging to the area domain

In the Standardize stage, the column names that you see on the output links for the rule set have suffixes of underscore and rule set name. For example, when the USPREP rule set is used, NameDomain, is labeled NameDomain_USPREP in the output link from the Standardize stage.

Reporting columns for domain preprocessor rule sets:

Domain preprocessor rule sets provide reporting fields for quality assurance and post-standardization investigation. In the stage, the fields are mapped to columns.

All domain preprocessor rule sets have the following reporting fields:

**Field[N]Pattern**
- The pattern generated for the delimited column as indicated by number. Field1Pattern is generated for the first delimited column. Field2Pattern is generated for the second delimited column and so on, up to Field6Pattern, which is generated for the sixth delimited column. The pattern is generated from column input values (previously called tokens) and is based on the parsing rules and classifications.

**InputPattern**
- The pattern generated for the entire stream of input values based on the parsing rules and classifications.

**OutboundPattern**
- The pattern image for all values just prior to being written to the output file.

**UserOverrideFlag**
- A flag indicating what type of user override was applied to this record.

**Custom Flag**
- Unused. Available for users to create a flag needed in their project.

In the Standardize stage, the column names that you see on the output links for the rule set have suffixes of underscore and rule set name. For example, when the USAREA rule set is used, UserOverrideFlag, is labeled UserOverrideFlag_USAREA in the output link from the Standardize stage.

User flag descriptions for domain preprocessor rule sets:

Flags used in the domain preprocessor rule set indicate the use of user overrides.
The possible values for the UserOverride flag in the domain preprocessor rule sets are as follows:

- **NO**: The default. No user remasks are used
- **IP**: An input pattern user override is used
- **IT**: An input text user override is used
- **UP**: An unhandled pattern user override is used
- **UT**: An unhandled text user override is used

The CustomFlag is not populated by default. The flag is populated only if you modify the rule set in order to use it.

**Domain masks for domain preprocessor rule sets**
The domain preprocessor rule set attempts to assign a domain mask to each input value (previously called token).

All pattern-actions retype values to one of the domain masks, which are:

- **A**: ADDRESS
- **N**: NAME
- **R**: AREA

The final step in the domain preprocessor is to map the output from the rule set, values, to the domain columns based on their assigned (or defaulted) domain mask.

**Upgrading preprocessor rule sets**
When a new release is available, or when changes are made to the delivered rule sets, you can incorporate the improvements into an existing project.

Do not update the following items:
- Rule set descriptions
- Dictionary
- User override tables

These items are specific to the job, and improper changes can cause processing failure or produce different results.

For the USPREP rule set, you do not change the following items:

- **USPREP.DCT**: Dictionary
- **USPREP.UCL**: User classification
- **USPREP.ITO**: User override input text object
- **USPREP.IPO**: User override input pattern object
- **USPREP.FTO**: User override field text object
**USPREP.FPO**  
User override field pattern object

You can update the following items:

- Classifications
- Pattern-actions
- Lookup tables

These items might have changed from the previous version.

For the USPREP rule set, the items you can change include the following items:

**USPREP.CLS**  
Classifications

**USPREP.PAT**  
Pattern-Actions

**USCITIES.TBL**  
United States city lookup table

Because any changes to the classifications are made through the user classifications (.UCL), you can replace the classifications. However, verify that what is on the current classifications is also on the newly-delivered one. Use this approach with any lookup tables.

Pattern-actions are more complicated to upgrade. During development, changes to pattern-actions are made in two subroutines: Input_Modifications and Column_Modifications. If this is the case, copy those subroutines from the existing pattern-actions, and then paste them in the one where the empty subroutines are found.

Many times, other changes are made outside of the modification subroutines. You can also add those changes to the new pattern-actions.

**Note:** The development of the job is based on the rules it is currently using, so any rules changes could impact the output. If you must upgrade the rules in an existing job, we advise that extensive testing be done before you run the job in production.

**Domain-specific rule sets**

Domain-specific rule sets process free-form data, data not limited by structure constraints. The data that the rule sets process is single-domain data, pertaining to specific types of information content: name, address, or area.

The standard domain-specific rules sets for each country or region are as follows:

**NAME**  
Individual, business, and organization names.

**ADDR**  
Street name, number, unit, and other address information.

**AREA**  
City, state, region, and other locale information.
Objects in domain-specific rule sets
The domain-specific rule set has a group of objects associated with it.

The naming convention for domain-specific rule sets is:

<table>
<thead>
<tr>
<th>Position</th>
<th>Description</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-6</td>
<td>Type of rule</td>
<td>NAME: Name ADDR: Address AREA: Area</td>
</tr>
</tbody>
</table>

For example, here are the objects in the United States NAME rule set:

USNAME.CLS
Classifications table

USNAME.DCT
Dictionary

USNAME.PAT
Pattern-action specification

USNAME.IPO
Input pattern overrides

USNAME.ITO
Input text overrides

USNAME.UCL
User classifications

USNAME.UPO
User pattern overrides

USNAME.UTO
User text override file

You can also use the following lookup tables:

USNAMEMF.TBL
File that contains words to be excluded from the match fields

USFIRSTN.TBL
File that contains the extended version of the given name and the gender (Bob Robert M). If there is no extended version, the second column contains only the gender (Bob M).

USGENDER.TBL
File that contains the gender identification based on the prefix: Mr, Ms, Mrs, or Miss

Overrides for domain-specific rule sets
The domain-specific overrides and their abbreviations let you specify your own custom conditioning rules.

The following table describes the objects used to override the domain-specific rule sets:
Characteristics of domain-specific rule sets
The name, address, and area domain rule sets manipulate specific input and have specific rule set behavior.

Name

The expected input into a name domain rule set is valid individual, organization, or qualifying names, such as DBA (database administer), for the specified country or region.

The name domain rule set performs the following actions based on the process option:

**None**
Attempts to determine if the input is an individual or organization name and standardizes the input accordingly; if the rule set is unable to determine the type of name, the data is put in the unhandled data output field.

The rule set is conservative in what data is handled; for example, a pattern of “++” is unhandled with no process option but is handled when a process option is selected.

**Process All As Individual**
Assumes that all input is individual names and is more liberal in what the rule set allows into the given, middle, and family name output fields.

**Process All As Organization**
Does not try to populate the given and middle names; assumes that all input names are organizations and standardizes them into the family name and suffix output fields.

**Process Undefined as Individual**
Attempts to determine if the input is an individual or organization name and standardizes the input accordingly; if the rule set is unable to determine the type of name, it assumes it to be an individual name and standardizes it into the given, middle, and family name output fields.

**Process Undefined as Organization**
Attempts to determine if the input is an individual or organization name and standardizes the input accordingly; if the rule set is unable to determine the type of name, the rule set assumes the input to be an organization name and standardizes the input into family name and suffix output fields.

The name domain rule set uses classified values (previously called tokens) along with punctuation to determine how names are standardized. In the example name DAVIS CHRISTOPHER, the pattern is: +F. The pattern might be Given Name | Family Name or Family Name | Given Name. The input is left unhandled. However,
if the input name is DAVIS, CHRISTOPHER, then the pattern is +,F.
CHRISTOPHER is the given name and DAVIS is the family name.

**Address**

The expected input into an address domain rule set is valid street and secondary address data for that country or region.

The address domain rule set performs the following actions:
- Follows postal standards closely, when standards are available.
- If a pattern can represent two types of addresses and there is no indicator to distinguish which way to process them, the rule set leaves both addresses unhandled. The user must determine how to process the input by using overrides.

**Area**

The expected input into an area domain rule set is valid high-level address data for the country or region in correct order. For the United States, the correct order is as follows:
1. City name
2. State
3. ZIP code
4. Country

The area domain rule set performs the following actions:
- Standardizes country or region, state or province, and postal code into output fields that, in the stage, are mapped to column.
- Remaining information is put into a city name field with minimal standardization.
- Minimal standardization is applied to the City.
- In the United States, directional information such as North, South, East, West, and the common word, Saint, are expanded. In other countries, major cities that have multiple variations are made to conform.

**Domain-specific dictionary**

The dictionary for a domain-specific rule set contains business intelligence fields, matching fields, and reporting fields. The fields are mapped to columns as specified in the Standardize stage.

**Business intelligence fields for domain-specific rule sets:**

Business intelligence fields help focus on critical information contained within data.

The business intelligence fields vary by domain, and by country or region. The USAREA business intelligence field names are as follows:
- CityName
- StateAbbreviation
- ZIPCode
- Zip4AddonCode
- CountryCode
Matching fields for domain-specific rule sets:

Domain-specific rule sets create fields that facilitate effective data matching.

The matching fields vary by domain, and by country or region. The USAREA matching field names are CityNameNYSIIS and CityNameRVSNDX.

Reporting fields for domain-specific rule sets:

The reporting fields are used for quality assurance and post-standardization investigation.

Domain-specific rule sets have the following reporting fields:

**UnhandledPattern**
- The pattern generated for the remaining values (previously called tokens) not processed by the rule set based on the parsing rules, classifications, and any additional manipulations by the pattern-action language.

**UnhandledData**
- The remaining values not processed by the rule set, with one character space between each value.

**InputPattern**
- The pattern generated for the stream of input values based on the parsing rules and classifications.

**ExceptionData**
- The values that are not processed by the rule set because they represent a data exception. Data exceptions might be values that do not belong to the domain of the rule set or are invalid or default values.

**UserOverrideFlag**
- A flag indicating what type of user override was applied to this record.

Data flags for domain-specific rule sets:

Domain-specific rule sets provide an explanation of override flags.

The possible values for the UserOverride flag in the reporting fields of the domain-specific rule sets are as follows:

**NO**  The default. No user remasks are used

**FP**  A column pattern user remask is used

**FT**  A column text user remask is used

**IP**  An input pattern user remask is used

**IT**  An input text user remask is used

The CustomFlag is not populated by default. The flag is populated only if you modify the rule set in order to use it.

Validation rule sets

Validation rule sets generate business intelligence fields and reporting fields. In the stage, the fields are mapped to columns.

The Validation rule sets are used to standardize common business data including:
**VDATE**
Dates that include day, month, and year.

**VEMAIL**
E-mail addresses that have a user, domain, and top-level qualifier.

**VPHONE**
Country-specific phone numbers.

**VTAXID**
Tax identity or pension fund identity numbers.

**Objects in validation rule sets**
The validation rule set has a group of objects associated with it.

The naming convention is:

<table>
<thead>
<tr>
<th>Position</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Validation rule abbreviation</td>
<td>V</td>
</tr>
<tr>
<td>2 to n</td>
<td>Type of rule</td>
<td>DATE EMAIL PHONE TAXID</td>
</tr>
</tbody>
</table>

For example, here are the objects in the DATE rule set:

**VDATE.CLS**
Classifications table

**VDATE.DCT**
Dictionary

**VDATE.PAT**
Pattern-action specification

**VDATE.IPO**
Input pattern overrides

**VDATE.ITO**
Input text overrides

**VDATE.UCL**
User classifications

**VDATE.UPO**
User pattern overrides

**VDATE.UTO**
User text overrides

You can also use lookup tables in your rule set. The following files are lookup tables used with the VDATE rule set:

**VDATELY.TBL**
Leap year

**VDATEMM.TBL**
Month

**INVDATE.TBL**
Invalid calendar dates
**VDATE rule set**

The VDATE rule set validates the value and standardizes the format of a date.

The following information pertains to the VDATE rule set:

- The rule set applies only to Gregorian calendars.
- Punctuation, such as hyphens or slashes, are removed during the parsing step.
- The rule set checks and flags the following conditions as invalid:
  - Month is equal to or greater than 1 but equal to or less than 12
  - Month is February and day is equal to or less than 29 by using leap year calculations
  - Month is April, June, September, or November and day is equal to or less than 30
  - Month is January, March, May, July, August, October, or December and day is equal to or less than 31
- The output from the rule set is mapped to columns and results in business intelligence columns and reporting columns.
- The standard output format is CCYYMMDD.

**Default parsing parameters for VDATE rule sets:**

The default parsing parameters are:

SEPLIST <space character> ";%:*/+\\()"  
STRIPLIST <space character> ";%:*/+\\()"

**Input date formats for VDATE rule sets:**

There are formats for dates that are required for the input data.

The expected input format for dates and an example for each format are as follows:

`mmdccyy`  
09211991

`mmmddccyy`  
OCT021983

`mmmdccyy`  
OCT21983

`mmddccyy`  
04101986

`mm/dd/ccyy`  
10/23/1960

`m/d/ccyy`  
10/23/1960

`mm/d/ccyy`  
10/3/1960

`mm/dd/ccyy`  
1/13/1960

`mm-dd-ccyy`  
04-01-1960
m-d-ccyy
1-3-1960

mm-d-ccyy
10-3-1960

m-dd-ccyy
1-13-1960

ccyy-mm-dd
1990-10-22

SEPLIST and STRIPLIST are statements you can use to override default formats. You can find more information about SEPLIST and STRIPLIST statements in the IBM InfoSphere QualityStage Pattern-Action Reference.

Examples of input strings and output results are as follows:

1990-10-22
19901022

1/13/1960
19600113

OCT021983
19831002

Business intelligence fields for VDATE rule sets:

Business intelligence fields help focus on date information contained within output data.

If a data value passes validation, this rule set populates the following business intelligence field values:

DateCCYMMDD
Eight numeric bytes; valid date data.

ValidFlag
The value T.

Reporting fields for VDATE rule sets:

The output of the VDATE rule set are data values that do not pass validation.

If a data value fails validation, the rule set populates the reporting fields. The output field InvalidData is populated with the data that did not meet the validation requirements. The output field InvalidReason is populated with one of the following values:

IF Invalid format
IM Invalid month (for example, JJJ011976 instead of JAN011776)
IT Date is on invalid table (for example, 11111111)
MM Invalid numeric month (for example, 13/10/1988)
FB Invalid day for February (for example, 02/30/1976)
M0 Invalid day for months with 30 days
M1 Invalid day for months with 31 days
**VEMAIL rule set**

The VEMAIL rule set identifies the format, components, and completeness of email addresses.

The following information pertains to the VEMAIL rule set:

- All email addresses must have a user, domain, and top-level qualifier.
- The at sign (@) and period (.) are used as delimiters during the parsing step.
- The default classifications for this rule set contain the common domain (for example, ORG, COM, EDU, GOV) and sub-domain qualifiers (for instance, country and state codes).
- The rule set verifies email formats distinguishing local, domain, and top domain components.
- The rule set corrects spaces found in local and domain components but not in the top domain; the output must be formatted to use the corrected information.

**Default parsing parameters for VEMAIL rule sets:**

The default parsing parameters are:

- **SEPLIST** `<space character>`, %, : & \ / + \ ( ) [ ] < > @ . `_.
- **STRIPLIST** `<space character>` `.`

**Parsing examples for the VEMAIL rule set:**

The parsing parameters parse the address into multiple values (previously called tokens) as in the following examples:

<table>
<thead>
<tr>
<th>Input string</th>
<th>Value</th>
<th>Value number</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:John_Smith@abccorp.com">John_Smith@abccorp.com</a></td>
<td>John_Smith</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>abccorp</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>com</td>
<td>3</td>
</tr>
<tr>
<td><a href="mailto:kjones@example.org">kjones@example.org</a></td>
<td>kjones</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>example</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>org</td>
<td>3</td>
</tr>
</tbody>
</table>

The at sign (@) and period (.) are used to separate the data. They are removed during the parsing process.

The Standardize stage does not reassemble the multiple values into a single value before processing. You must append the input email addresses to the end of the data.

**Business intelligence fields for the VEMAIL rule set:**

Business intelligence fields help focus on email information contained within output data.

If a data value is validated, this rule set populates the following business intelligence fields:

- **EmailUser**
  - Alphanumeric
EmailDomain
   Alphanumeric

EmailToplevel
   A qualifier such as EDU, ORG or COM

EmailURL
   Possible Web site; value contains www

ValidFlag
   The value T

Reporting fields for the VEMAIL rule set:

The output of the VEMAIL rule set are data values that do not pass validation.

If a data value fails validation, the rule set populates the reporting fields. The output fields are as follows:

UnhandledData
   Populated with the value that did not meet the validation requirements

InputPattern
   Indicates the pattern generated for the entire stream of input values
   (previously called tokens) based on the parsing rules and classifications

UserOverrideFlag
   Contains a flag indicating what type of user override was applied to the record

The output field InvalidReason is populated with one of the following values:

NA   Input data is "N/A" or NA
@@   Email address contains more than one at sign (@)
.@   Includes a period (.) before the at sign (@)
..   Contains at least two sequential periods (.)
.1   A period (.) is the first character of the email address
IC   Email address contains an invalid character
WW   Email address contains a possible web site. The information is in the EmailURL field

VPHONE rule set

The VPHONE rule set validates the value and standardizes the format of phone numbers from the United States, Canada, and the Caribbean.

The following information pertains to the VPHONE rule set.

- The rule set verifies the following example formats:
  - (nnn) nnn-nnnn
  - nnn nnn nnnn
  - nnn-nnn-nnnn
  - nnnnnnnnnnn
- The rule set checks and flags the condition where a number sequence begins with 1. The condition is invalid.
- The following symbols are removed during the parsing step:
  - comma (,)

Chapter 6. Rule sets applied in the data cleansing process  197
semicolons (;)
periods (.)
colons (:)
slashes (/)
asterisks (*)
backward slashes (\)
quotes ("")
double-slashes (\\)
parentheses ( )
dashes (-)
underscores (_)

**Default parsing parameters for VPHONE rule sets:**
The default parsing parameters are:

SEPLIST `<space character>; . % : & * " / + \ ( )` - `_`

STRIPLIST `<space character>; . . / * " \ ( )` - `_`

**Parsing examples for the VPHONE rule set:**
The following tables show examples of how phone numbers are parsed:

**Table 24. Seven-digit phone number**

<table>
<thead>
<tr>
<th>Input string</th>
<th>Value</th>
<th>Value number</th>
</tr>
</thead>
<tbody>
<tr>
<td>(617) 338-0300</td>
<td>617</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>338</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0300</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 25. Seven-digit phone number with extension as distinct value**

<table>
<thead>
<tr>
<th>Input string</th>
<th>Value</th>
<th>Value number</th>
</tr>
</thead>
<tbody>
<tr>
<td>(617) 338-0300 EXT 316</td>
<td>617</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>338</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0300</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>EXT</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>316</td>
<td>5</td>
</tr>
</tbody>
</table>

**Table 26. Seven-digit phone number with extension as part of the last value**

<table>
<thead>
<tr>
<th>Input string</th>
<th>Value</th>
<th>Value number</th>
</tr>
</thead>
<tbody>
<tr>
<td>617-338-0300 X316</td>
<td>617</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>338</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0300</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>X316</td>
<td>4</td>
</tr>
</tbody>
</table>

The hyphen, space, and parentheses are used to separate the data. After the data is parsed the hyphen, spaces, and parentheses are dropped.

**Validation logic for VPHONE rule sets:**
The VPHONE rule set validates patterns and values based on the following criteria:

- The value has 7 or 10 numeric bytes. Can be over 10 bytes with extensions
- The first 3 bytes are not all zeros (000). If all zeros, they are replaced with blanks
- The value is not listed on the ‘invalid table’, INVPHONE.TBL. Examples of invalid numbers are the number sequence 1234567, or 7 digits of the same number, for example, 2222222.

If the data value fails any one of the validation requirements, the InvalidData and the InvalidReason fields are populated.

**Business intelligence fields for VPHONE rule sets:**

Business intelligence fields help focus on phone information contained within output data.

If the data value passes validation, this rule set outputs the following business intelligence field values:

- **PhoneNumber**
  - An all numeric phone number, excluding extension number.

- **PhoneExtension**
  - An all numeric extension number.

- **ValidFlag**
  - The value T.

**Reporting fields for VPHONE rule sets:**

The output of the VPHONE rule set are data values that do not pass validation.

If a data value fails validation, the rule set populates the reporting fields. The output field InvalidData is populated with the data that did not meet the validation requirements. The output field InvalidReason is populated with one of the following values:

- **IL** Invalid length. Main phone (without extension) must be 7 bytes or 10 bytes.
- **IP** Invalid pattern or format. Main phone (without extension) must be 10 bytes numeric. This logic can be commented out if alphas are to be considered valid values.
- **IT** The data value is listed in INVPHONE.TBL, the invalid table (for example, 1234567)

**Examples**

The following table shows sample input data and the output they produce:

<table>
<thead>
<tr>
<th>Input String</th>
<th>PhoneNumber</th>
<th>ValidFlag</th>
<th>InvalidData</th>
<th>InvalidReason</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001234567</td>
<td>0001234567</td>
<td></td>
<td>0001234567</td>
<td>IT</td>
</tr>
<tr>
<td>(617) 338-0300</td>
<td>6173380300</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>617-338-0300</td>
<td>6173380300</td>
<td>T</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**VTAXID rule set**
The VTAXID rule set validates and standardizes the format of a standard nine-digit tax identification number.

The VTAXID rule set validates the value and standardizes the format of a tax ID or national ID number. The following information pertains to the VTAXID rule set:
- All punctuation is removed during the parsing step.
- The rule set verifies that the input data is nine digits, all numeric, and does not consist of all one number, for example 000-00-0000.

**Default parsing parameters for VTAXID rule sets:**
The default parsing parameters are:

\[\text{SEPLIST} <\text{space character}> ; . % : \& * \" / + \ \\{ \} -\]

\[\text{STRIPLIST} <\text{space character}> ; . / * \\" \\{ \} -\]

**Parsing examples for the VTAXID rule set:**
The following tables show examples of how tax IDs and national ID numbers are parsed:

**Table 27. ID that includes separators**

<table>
<thead>
<tr>
<th>Input string</th>
<th>Value</th>
<th>Value number</th>
</tr>
</thead>
<tbody>
<tr>
<td>051-34-8198</td>
<td>051</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>8198</td>
<td>3</td>
</tr>
</tbody>
</table>

The hyphen and space are used to separate the data. After the data is parsed the hyphen and spaces are deleted.

**Table 28. ID that does not include separators**

<table>
<thead>
<tr>
<th>Input string</th>
<th>Value</th>
<th>Value number</th>
</tr>
</thead>
<tbody>
<tr>
<td>193837485</td>
<td>193837485</td>
<td>1</td>
</tr>
</tbody>
</table>

**Validation logic for VTAXID rule sets:**
The rule set validates patterns and values based on the following criteria:
- The value has 9 numeric characters.
- The first 3 bytes are not all zeros (000).
- The value is not listed on the 'invalid table', INVTAXID.TBL. Examples of invalid numbers are the number sequence 987654321, or digits of the same number, for example, 888888888.

If the data value fails any one of the validation requirements, the **InvalidData** and the **InvalidReason** fields are populated.

**Business intelligence fields for VTAXID rule sets:**

Business intelligence fields help focus on tax identification information contained within output data.

If the data value passes validation, this rule set outputs the following business intelligence field values:

**TaxID** Nine numeric bytes.
ValidFlag
The value T.

Reporting fields for VTAXID rule sets:

The output of the VTAXID rule set are data values that do not pass validation.

If the data value fails validation, the rule set populates the reporting fields. The output field **InvalidData** is populated with the data that did not meet the validation requirements. The output field **InvalidReason** is populated with one of the following values:

**IP**  The data value did not contain nine, and only nine, numeric characters.
**IT**  The data value is listed in INVTAXID.TBL, the invalid table (for example, 555555555).
**Z3**  The first three numeric characters are all zeros.

Examples

The following table shows sample input data and the output they produce:

<table>
<thead>
<tr>
<th>Input String</th>
<th>TaxID</th>
<th>ValidFlag</th>
<th>InvalidData</th>
<th>InvalidReason</th>
</tr>
</thead>
<tbody>
<tr>
<td>000123456</td>
<td>000123456</td>
<td>T</td>
<td>000123456</td>
<td>Z3</td>
</tr>
<tr>
<td>193837485</td>
<td>193837485</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>193-83-7485</td>
<td>193837485</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111111111</td>
<td>1111111111</td>
<td>IT</td>
<td>1111111111</td>
<td>IT</td>
</tr>
<tr>
<td>222-22-2222</td>
<td>222222222</td>
<td>IT</td>
<td>222222222</td>
<td></td>
</tr>
<tr>
<td>A12-09-1234</td>
<td>A12091234</td>
<td>IP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Managing rule sets

You use the IBM InfoSphere DataStage and QualityStage Designer to apply the logic of rule sets within stages of your data cleaning jobs. You use the IBM InfoSphere QualityStage Standardization Rules Designer to enhance standardization rule sets.

You apply the logic of rule sets in the Standardize stage or Investigate stage to determine how columns are parsed and classified into values (previously called tokens). You can also apply the logic of rule sets within international stages such as the Multinational Standardize (MNS) stage.

You edit files within a rule set by using a text editor. You manage rule sets from with the Designer client, in the Rules Management window. You enhance rule sets in the Standardization Rules Designer, a web-based console where you can add and modify classifications, lookup tables, and rules.

After you add or remove rule sets, you can test application of the rule set in rules management. When you modify or add a business rule set, you can test it to ensure that it defines the business rules for data cleansing that you need.

The information that you enter in override tables is applied to rule sets across all projects.
Accessing rule sets

You can install rule sets from an IBM InfoSphere Information Server file directory and then import the rule sets into your project. You can also access the set of predefined rule sets from the repository or view rule sets in the Standardization Rules Designer.

Importing rule sets

You can use asset interchange to import rule sets into your project.

Before you begin

A starter set of rule sets is available to your project by default. You can import more predefined rule sets from the metadata repository by using asset interchange.

Use asset interchange commands to import files. You run the installation commands from the following directory: \IBM\InformationServer\Clients\istools\cli.

Procedure

1. Go to the rule set listing in the directory path: \IBM\InformationServer\Server\PXEngine\QSRules.
2. Locate the asset interchange archive file that you want. The file name format for rule sets is as follows: QSRules_RegionName or GenericName_Version_Release.isx For example, you might see this file: QSRules_Peru_8_5.isx.
3. Import the rule set into your project by typing the following command from a command prompt on the computer where the client tier or engine tier is installed:

   Option               Description
   Windows
   istool.bat import -domain domain_name
                    -username login_name -password password
                    -archive "archive_file_name_with_path"
                    -datastage "server/project"

   Linux, UNIX
   istool import -domain domain_name
                    -username login_name -password password
                    -archive "archive_file_name_with_path"
                    -datastage "server/project"

Example

Here is an example of the command you type to import the rule sets:

   istool import -domain MYSERVER
                   -username dsadm1 -password qsadmin1
                   -archive "opt/IBM/InformationServer/Server/PXEngine/QSRules/QSRules_Mexico_8_5.isx"
                   -datastage "MYSERVER/RulesMgmt"

What to do next

The next step is to open the Designer client and to provision the rule set before you attempt to use it in a job.
**Viewing rule sets from the Designer client**

Rule sets that are stored in the repository are accessed from the InfoSphere DataStage and QualityStage Designer.

**About this task**

You can go directly into the Designer client view rule sets that you can use in your project.

**Procedure**

1. Go to the rule set listing.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| In the Designer client repository view | 1. Expand the navigation tree.  
2. Locate the folder containing the rule sets. You can find several predefined rule sets in the *Standardization Rules* folder.                                                                 |
| In one of the following stages: | Right-click on the stage on the Designer palette and select *Properties*. If rule sets are applied in stages, you see them listed in the window.                                                             |
| • Investigate                  | The Multinational Standardize (MNS) stage uses rule sets indirectly. You cannot view the rule sets from within these stages, but you can view the rule sets in the *MNS Rules* folder. |
| • Standardization              |                                                                                                                                                                                                             |

2. Open the rule set.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| In the Designer client repository view | 1. Expand the rule set folder to view the files.  
2. Right-click the SET folder in domain-preprocessor rule set such as USPREP and select *Properties* to open the Rules Management window.                                                        |
| In one of the following stages: | 1. Select the rule set.  
2. Click *New Process* or *Modify* to open the Standardize Rule Process window.                                                                                                                              |
| • Investigate                  |                                                                                                                                                                                                             |
| • Standardization              |                                                                                                                                                                                                             |

**What to do next**

You can apply the rule set in an InfoSphere QualityStage stage or make changes. Before you attempt to use the rule set in a job, provision the rule set.

**Viewing rule sets in the Standardization Rules Designer**

You can access rule sets to enhance in the Standardization Rules Designer.

**About this task**

To view a particular rule set in the Standardization Rules Designer, you must have one of the following sets of roles:
**Suite Administrator role and DataStage and QualityStage Administrator role**

Users that have this set of roles can log in to the Standardization Rules Designer and view all of the rule sets in each project.

**Suite User role and DataStage and QualityStage User role**

Users that have this set of roles can log in to the Standardization Rules Designer. They can also view all of the rule sets in each project for which they have the DataStage Developer or DataStage Production Manager role.

**Procedure**

View a rule set in the Standardization Rules Designer.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Access the Standardization Rules Designer by using the shortcut on your desktop. | 1. Double-click the Standardization Rules Designer shortcut on your desktop.  
2. Log in to the Standardization Rules Designer.  
3. Expand the server and project for the rule set. |
| Access the Standardization Rules Designer from your browser.          | 1. In your browser, go to `https://host_name:secure_port_number/ibm/iis/qs/StandardizationRulesDesigner/`, where `host_name` is the host name of the server where the Standardization Rules Designer is installed.  
2. Log in to the Standardization Rules Designer.  
3. Expand the server and project for the rule set. |
| Access the Standardization Rules Designer from the Designer client.   | 1. In the Designer client repository tree, expand the folder that contains the rule set to view.  
2. Expand the rule set folder.  
3. Right-click the SET folder for the rule set, and then click Properties.  
4. Complete one of the following steps:  
   • To access the Home page for the rule set, click Take me to the Standardization Rules Designer.  
   • To access the Classifications, Lookup Table, or Rules page for the rule set, expand Rule Set Extensions and double-click the appropriate object. |

The Standardization Rules Designer opens, and a revision is opened for the rule set. You do not have to log in to the Standardization Rules Designer.
What to do next

If you accessed the Standardization Rules Designer from your browser or by using the shortcut on your desktop, open a revision for the rule set by double-clicking the rule set.

Creating rule sets

You can create a unique rule set that applies to a particular business problem.

Procedure

1. Optional: Create a new folder in the Designer client repository. Right-click on the project folder or on an existing folder within the project and select New > Folder.
2. In the Designer client repository tree, right-click on the folder that is to contain the new rule set and select New > Data Quality > Rule Set to open the New Rule Set window.
3. From the New Rule Set window, type a rule set name in the Specify the new Rule Set name field.
4. Add a folder to the repository for all the rule sets that you create.
5. Navigate to the rule set folder.
6. Optional: Type a description of your rule set in the Specify a description (optional) text field.
7. Click OK to add the new rule set to your folder.

Developing rule sets

After you create a rule set, you develop the rule set to address your data cleansing requirements.

Before you begin

- Ensure that your data cleansing requirements cannot be met by using or modifying a predefined rule set.
- Create a rule set.

About this task

Rule set development is an iterative process that often requires collaboration between IBM InfoSphere QualityStage developers and subject matter experts. To develop a rule set that meets your data cleansing requirements, you might need to complete the steps in this task more than once.

For more information about rule set development, see the rule set development package. The rule set development package is in the InfoSphere Information Server file directory, and you can install it from asset interchange.

After you choose special characters and define output columns in the dictionary, you can complete many of the remaining steps in this task by enhancing a rule set in the Standardization Rules Designer. In the Standardization Rules Designer, you can add lookup tables, classify values, and add rules that standardize the data for your domain.
Procedure

1. Choose the special characters to use to parse data into meaningful values. You specify the following types of special characters:
   - Separation characters indicate where one value in a record ends and the next value begins.
   - Strip characters are removed from the record.

2. Define output columns for the standardized data.

3. Classify values in the input data to assign the values to particular categories. Focus on values for which contextual information or transformation is required to process the data correctly. For example, classify values that require standard values. In a rule set for the address domain, you might classify Street with a standard value of ST.

4. In the pattern-action specification, specify rules to standardize groups of related records. Rules include actions, which specify how the record is processed, and conditions, which set requirements that records must meet before the action is applied.

5. Test the rule set. Use one or more of the following methods:
   - To test a single record, use the standardization rule set tester.
   - To evaluate the results of the standardization process when the rule set is applied to your data, run a job that includes the SQA stage. You can generate and view a Standardization Quality Assessment report, which includes statistics about the standardization results and examples of processed records.

Scenario: rule set development for retail product data

You can create and develop standardization rule sets for domains that predefined rule sets do not apply to. This scenario describes how the fictional Sample Outdoor Company develops a rule set for retail product data.

The fictional Sample Outdoor Company sells and distributes products to third-party retailer stores and consumers. The company has decided to create and develop a rule set to standardize its retail product data.

Before developing the new rule set, the IBM InfoSphere QualityStage developer collaborated with a subject matter expert to determine initial requirements for data cleansing. After the developer completes the first iteration of the rule set, the developer will review the standardization results with the subject matter expert to determine where and how to improve the rule set.

Step 1: Choose special characters for parsing

To begin, the developer copies the sample lists of separation and strip characters from the rule set in the rule set development package. The developer ensures that the space character is included in both lists. The following table shows the characters in each list.

<table>
<thead>
<tr>
<th>List</th>
<th>Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separation list</td>
<td>~ ` ! @ # $ % &amp; * ( ) _ - + = [ ] { }</td>
</tr>
<tr>
<td>Strip list</td>
<td>^ + ? &gt; &lt; @</td>
</tr>
</tbody>
</table>
After the initial list is created, the developer consults the requirements that the subject matter expert provided. The developer looks for the following types of requirements, which might require changes to the special character lists:

- Default classes that appear as values in the data. For example, because the at sign (@) is used in email addresses for product suppliers, the developer removes the at sign from both lists.
- Special characters that can be removed from the data. For example, a hyphen (-) is used in the input data in values such as 9-PIECE and ONE-SIZE. Because the hyphen is not required to understand the data, the developer adds the hyphen to the strip list.

**Step 2: Define output columns**

Output columns often include columns for business intelligence, matching, and reporting. In the first iteration, the developer decides to focus on single-domain columns that are used for business intelligence. The developer adds the following output columns to the dictionary:

- ProductBrand
- ProductName
- ProductSize
- ProductColor
- ProductType

**Step 3: Classify values in the input data**

Next, the developer classifies values. Classifying values provides context that is required to process the data correctly. Although multiple strategies for classification are available, the developer chooses to classify only those values that appear most frequently in the data in the first iteration. For example, the developer creates classes for product brands, product colors, and product types in the classifications table.

When the classes are assigned, the developer chooses the one-character label for the class based on the input of the subject matter expert. Because every class is associated with a product, the developer chooses class labels based on the secondary characteristic of the class. The following table shows the class labels that the developer assigns.

<table>
<thead>
<tr>
<th>Class label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Product brand</td>
</tr>
<tr>
<td>C</td>
<td>Product color</td>
</tr>
<tr>
<td>T</td>
<td>Product type</td>
</tr>
</tbody>
</table>

After the values are assigned to classes, the records match patterns that include the new classes. For example, before the values are classified, the record HIBERNATOR PAD 195 CM GREY SLEEPING BAG matches the pattern ++^+++++. After the values are classified, the record matches the pattern B^+CTT.

Next, the developer asks the subject matter expert to review the class labels and values. Because the subject matter expert is not familiar with pattern-action language, the expert opens a revision for the rule set in the Standardization Rules.
Designer. In the Standardization Rules Designer, the expert imports sample data and reviews the patterns that various records match.

After the review, the subject matter expert decides to add a new class, S, which will be used for product size values. For example, after the value CM is assigned to the S class, the record HIBERNATOR PAD 195 CM GREY SLEEPING BAG matches the pattern B+^SCTT.

**Step 4: Specify rules**

Next, the developer decides to create an initial set of rules that apply to the most common patterns in the data. The developer copies the pattern-action specification (previously called pattern-action file) template from the rule set development package and adds three rules for retail product data.

For example, the developer adds the following rule for the most common pattern in the data:

```
B | + | S | C | T ; Common Pattern Found: CALL Post_Process SUBROUTINE then EXIT
COPY_A [1] {ProductBrand}
COPY_S [2] {ProductName}
COPY_A [3] {ProductSize}
COPY_A [5] {ProductType}
CALL Post_Process
EXIT
```

Again, the developer asks the subject matter expert to review the rules. The subject matter expert reviews the rules and decides to use the Standardization Rules Designer to add rule set extensions. The extensions address standardization requirements that are not met by the rules that the developer added in the pattern-action specification.

**Step 5: Test the rule set**

At the end of the first iteration, the developer asks the subject matter expert to publish the revision in the Standardization Rules Designer. The developer then provisions the rule set and applies it to sample input data in a standardization job. Because the job includes an SQA stage, the developer can generate an SQA report to view the job results.

In the SQA report, the standardization summary shows that 39% of the records were fully standardized by the standardization job that used the developed rule set.

The developer can use the SQA report and input from the subject matter expert to determine where and how to improve the rule set.

**Notices:** The Sample Outdoor Company, GO Sales, any variation of the Great Outdoors name, and Planning Sample, depict fictitious business operations with sample data used to develop sample applications for IBM and IBM customers.
These fictitious records include sample data for sales transactions, product distribution, finance, and human resources. Any resemblance to actual names, addresses, contact numbers, or transaction values, is coincidental. Other sample files may contain fictional data manually or machine generated, factual data compiled from academic or public sources, or data used with permission of the copyright holder, for use as sample data to develop sample applications. Product names referenced may be the trademarks of their respective owners. Unauthorized duplication is prohibited.

Copying rule sets
You can make a copy of any rule set that you can then modify.

Procedure
1. Locate the rule set that you want to copy from within the Repository tree in the Designer.
2. Right-click that rule set and select Create copy.
3. Right-click and select Rename to change the name of your copied rule set.
4. Optional: If you want to make a change that cannot be done in the overrides, drag the new rule set to your folder for custom rule sets in the repository. For example, you want to add a field to the dictionary.

Provisioning rule sets
You need to provision new, copied, or customized rule sets from within the Designer client before you run a job that uses them.

About this task
When a rule set is provisioned, the information about the rule set on the engine tier is updated based on information that is stored in the metadata repository. As a result, any jobs in the project that use the rule set do not need to be compiled again. When the jobs are run, the jobs will use the updated rule set.

If you import or export a rule set after you provision the rule set, you do not need to provision the rule set again.

Procedure
1. Open the Designer client, if it is not already open.
2. Locate the rule set that you want to use within the Repository tree in the Designer.
3. Select the rule set.
4. Right-click and select Provision All from the menu.

Results
The rule set is now available to be used in a job.

Modifying rule sets
You can modify rule sets by enhancing rule sets in the Standardization Rules Designer, by using overrides, or by editing the rule set files directly.

How to identify gaps in current standardization practices
When input data changes, existing rule sets might become less effective. To meet new or changed standardization requirements, you change the way that rule sets...
process data by enhancing rule sets in the Standardization Rules Designer, using override object types, or modifying the pattern-action language.

To identify standardization goals that are not met by current rule sets, you can use one or more of the following methods:

- View Investigate reports, which are generated by using data that is processed by the Investigate stage. Investigate reports provide details about the content and quality of source data. For example, an Investigate report might contain details about formatting errors or values that might indicate duplicate records. Use the Investigate stage and Investigate reports for data that is not already processed by standardization rule sets.

- View Standardization Quality Assessment (SQA) reports, which are generated by using data that is processed by the Standardize stage. SQA reports can be used to evaluate current standardization processes. For example, you might see in an SQA report that a large percentage of records were not fully standardized by a standardization job that used the rule set. You can then view the SQA Record Examples report to identify the patterns that are not handled. If a simple, common pattern is not handled, you might choose to enhance the rule set by adding a rule for that pattern in the Standardization Rules Designer.

**How to choose a method for rule set management**

You can modify a rule set by enhancing the rule set in the Standardization Rules Designer, using override object types, or modifying the pattern-action language. The method that you choose depends on your requirements and your experience with the pattern-action language.

**IBM InfoSphere QualityStage Standardization Rules Designer**

The Standardization Rules Designer is a browser-based interface that you can use to enhance standardization rule sets. Use the Standardization Rules Designer to modify a rule set when the following conditions are true:

- You want to use a method that involves less complexity than the pattern-action language or have relatively little experience with the pattern-action language.
- The data that you want to standardize contains relatively few patterns.
- A small number of patterns match a large percentage of the data.
- A relatively small number of actions and conditions can meet your standardization requirements for a particular set of input data.

**Override objects**

Override objects are objects in the rule set that you can edit in the Designer client. These objects are used to alter the processing of data in the classifications table and pattern-action specification (previously called pattern-action file).

Although override objects are available, the Standardization Rules Designer provides a greater number of standardization options. You can convert existing overrides to rule set extensions and work with them in the Standardization Rules Designer.

**Pattern-action language**

Pattern-action language is a language that is used to manipulate data. You can decipher and identify patterns in data, and then specify actions in the pattern-action specification based on the pattern. Modify a rule set by editing the pattern-action language directly when the following conditions are true:
• You are an experienced user of the pattern-action language and can work in the pattern-action specification quickly.
• The data that you want to standardize includes very long or complex patterns or records.
• Your standardization requirements involve rules that are complex or use many actions and conditions.

Considerations for copying a rule set

You cannot edit the pattern-action specification for a predefined rule set directly. To modify a predefined rule set by editing the pattern-action specification, you must copy the rule set and edit the copy of the pattern-action specification.

If you upgrade to a later version of IBM InfoSphere QualityStage, the predefined rule sets might be updated. When predefined rule sets are updated, any copy of the pattern-action specification that you made will not be updated concurrently. To ensure that a predefined rule set that you apply in standardization jobs remains current, modify the original rule set by using only the Standardization Rules Designer.

Strategies for classification

You classify values in the input data to strengthen the contextual information that the values provide and group them into categories.

When you classify values, you can use one or more of the strategies that are listed in Table 31.

Table 31. Strategies for classification

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Benefits</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classify values that appear most frequently in the data.</td>
<td>A relatively small number of classifications affects many patterns.</td>
<td>• In the Standardization Rules Designer, go to the Classifications page. From the list of classes, choose All Values for Default Classes. You can sort the values by frequency and identify the most common values that are assigned to default classes. You can then add definitions for those values that assign the values to a custom class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the Designer client, run a job that includes an Investigate stage with the Word Investigate option. The output of the job includes frequency information about the values. You can use this frequency information to identify values that are assigned to default classes and then assign the values to a custom class.</td>
</tr>
<tr>
<td>Strategy</td>
<td>Benefits</td>
<td>Implementation</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Classify values that are important to your domain.</td>
<td>Data that is important to your organization is accurate and standardized appropriately for your domain.</td>
<td>Consult documents such as company guidelines, domain standards, and glossaries. Ensure that terms from these documents are grouped into appropriate categories. You can use classifications to enforce standard terminology or adapt data to the conventions of a specific country or context.</td>
</tr>
<tr>
<td>Classify values that require standard values.</td>
<td>If your requirements include standardized spellings or representations of certain values, those values must be classified. Because you can often map these values directly to an output column, classifying values in this way makes it easier to add rules.</td>
<td>Consult a subject matter expert or domain standards to identify where standard values must be applied.</td>
</tr>
</tbody>
</table>
### Table 31. Strategies for classification (continued)

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Benefits</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classify values that are often spelled or represented incorrectly in your data.</td>
<td>Increase the accuracy and overall quality of your data.</td>
<td>• In the Standardization Rules Designer, go to the <strong>Classifications</strong> page. From the list of classes, choose <strong>All Values for All Classes</strong>. You can sort the values by frequency and browse for values that are spelled or represented incorrectly. You can classify these values independently. Alternatively, you can edit the classification definition for the value that is represented correctly. If you lower the similarity threshold for the value that is represented correctly, variations of the value that are represented incorrectly might be affected by the definition. • In the Designer client, run a job that includes an Investigate stage with the Word Investigate option. The output of the job presents information about the values that appear in your data. When you identify values that are spelled or represented incorrectly, you can classify these values independently. Alternatively, you can edit the classification definition for the value that is represented correctly. If you lower the similarity threshold for the value that is represented correctly, variations of the value that are represented incorrectly might be affected by the definition.</td>
</tr>
</tbody>
</table>

---

### Customizing rule sets by using overrides

You can change the way that input data is processed by using overrides.

#### About this task

Overrides modify existing rule sets. To use overrides, you must know the entire pattern or the entire text (input data) that you want to override.
Overrides include the rule set file names with three characters appended indicating each type of override.

Because overrides are added only by users, the overrides are empty when you first install InfoSphere QualityStage.

**Modifying rule sets with the classification override:**

From the Rules Management window, you can add one or more classification overrides to categorize values that share common attributes.

**About this task**

When you add classification overrides to a rule set, the Designer client provides the appropriate list of class types automatically based on the rule set that you want to modify.

After you create the override, the next time you apply the rule set, the values (previously called tokens) are classified with the designations that you specified.

**Procedure**

1. In the Rules Management window, double-click **Overrides**.
2. In the **Input Token** field, type the value that you want to override the classification for, such as SSTREET. Type the value as it appears in the input file.
3. In the **Standard Form** field, type the standardized spelling of the value, such as ST.
4. From the **Classification** menu, select the one-character tag that indicates the class of the value.
5. In the **Comparison Threshold** field, type a numeric value that defines the degree of uncertainty to tolerate in the spelling of the value.
   
   For example, the following table shows values for the comparison threshold.

<table>
<thead>
<tr>
<th>Range of comparison threshold values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank or no value</td>
<td>Value must match exactly</td>
</tr>
<tr>
<td>700</td>
<td>High tolerance for difference between the strings (minimum value)</td>
</tr>
<tr>
<td>750 and 800</td>
<td>Allows for one or two letter transformations depending on the length of the word (common threshold value)</td>
</tr>
<tr>
<td>950</td>
<td>Zero tolerance for difference between the strings (maximum value)</td>
</tr>
</tbody>
</table>

6. Click **Add** to add the override to the pane at the bottom of the window.
7. Click **OK** to close the window.

**Customizing preprocessor rule sets by using overrides:**

You can add overrides to data that is provided as input into a preprocessor rule set or field patterns that are identified by the preprocessor rule set.

**Modifying the overrides for input and field pattern:**

For the preprocessor rule sets, the input pattern overrides modify the associated object, *.IPO, and the field pattern overrides modify the associated object, *.FPO.
About this task

The field is defined by the delimiters. For example, you have the following input columns:

- Name
- AddressLine1
- AddressLine2
- City
- State
- Zip

The standardization process specifies the input with four fields, each of which are indicated by the zq qualifier, as shown in the following example:

```
zqnamezq Name zqaddrzq AddressLine1 zqaddrzq AddressLine2 zqareazq city state zip
```

The process has 4 fields, as indicated by the qualifier zq.

The pattern overrides have the following characteristics:

- With the input pattern override, you can specify value (previously called token) overrides that are based on the input pattern. The input pattern overrides take precedence over the pattern-action specification. Input pattern overrides are specified for the entire input pattern.
- With the field pattern override, you can specify value overrides that are based on one field pattern. These overrides are specified for the entire field pattern.

For example, you have the following input pattern:

```
N^+TA+S^+N
```

You must add an override to the input pattern to move the $^+T string to the address domain and the $A+S^$ string to the area domain.

Procedure

1. Right-click the SET folder in a domain-preprocessor rule set, such as USPREP, and select Properties to open the Rules Management window.
2. Double-click Overrides.
3. Click the Input Pattern or Column Pattern tab, as appropriate. Changes that you make in the Column Pattern tab affect the content of the FPO folder. You can see the FPO folder from the Designer client repository tree.
4. From the Classification Legend list, select the first class in the pattern, such as N, and then click Append this code to current pattern.
   The class is added to the Enter Input Pattern or Unhandled Pattern field and is shown in the Current Pattern List section with the default A (Address Domain) under the Override Code column.
5. For each class in the input pattern, such as for classes^, +, T, A, +, S, and ^, repeat the previous step.
   You can also type the classes directly into the Enter Input Pattern or Unhandled Pattern field. If you use the list, it verifies that you enter the correct values.

Changing the override code of a class:

You can change the override code of a class in the Current Pattern List.
Procedure

1. Select the class in the Current Pattern List section.
2. Select a domain from the Dictionary Fields list.
   For example, for the pattern N^+TA+S^, the N^+T string can stay at the default
   A-Address Domain. Change the A+S^ string to the R-Area Domain.

   Note: Alternatively, from the Current Pattern List section, select the class from
   the Token column and the override code from the Override Code column that
   you want to use. For example, select A from the list of classes, and then select R
   from the list of override codes.
3. Click Add to add the override to the Override Summary list.
4. Click OK to save your edits and close the window.
   Whenever you run the rule set on the pattern, it is assigned to the domain that
   you specified.

Modifying overrides for input and field text:

For the domain preprocessor rule sets, the input text and column text overrides
modify the override object (*.ITO) for the input text and the override object (*.FTO)
for the field text.

About this task

The input and field text overrides have the following characteristics:

- With the input text override, you can specify overrides that are based on an
  entire record of input text. These overrides take precedence over the
  pattern-action specification. Because they are more specific, input text overrides
  take precedence over input pattern overrides. Input text overrides are specified
  for the entire input text string. Partial string matching is not allowed.
- With the field text override, you can specify overrides that are based on the
  input text string from a particular field in a record. Since they are more specific,
  field text overrides also take precedence over field pattern overrides. You can
  specify field text overrides only for an entire text string in a field. You cannot
  match a partial string within a field.

For example, you would add an override to the input text object if you wanted to
change the name domain in the following text to the address domain. ZQNAMEZQ is
the name domain delimiter and ZQADDRZQ is the address domain delimiter:
ZQNAMEZQ CHARLES DE GAULLE ZQADDRZQ SQUARE

Procedure

1. Right-click the SET folder in a domain preprocessor rule set, such as USPREP,
   and select Properties to open the Rules Management window.
2. Double-click Overrides.
3. Click the Input Text or Column Text tab, as appropriate.
4. In the Enter Input Tokens or Enter Column Tokens field, type the domain
delimiter and the text for the values (previously called tokens) that you want to
override. For example, type ZQNAMEZQ CHARLES DE GAULLE ZQADDRZQ SQUARE.
   Each word that you enter is displayed in the Current Pattern List section with
   the word itself in the Token column and the default domain, such as A
   (Address Domain), in the Override Code column.
5. Select the value ZQNAMEZQ for the current domain delimiter of the text that you want to override, and then select the override that you want to apply from the Dictionary Fields list. For example, to process the example text as an address, do not change any of the override codes because both ZQNAMEZQ and ZQADDRZQ have the same override code.

6. Repeat the previous step if you need to change additional words.

7. Click Add to add the override to the Override Summary list.

8. Click OK to save your edits and close the window.

When you run the customized rule set on the text, it is processed as you specified. For example, when you run USPREP on the text ZQNAMEZQ CHARLES DE GAULLE ZQADDRZQ SQUARE, the entire text string is handled as an address.

Customizing domain-specific rule sets by using override objects:

You can add overrides for data that is provided as input into a domain-specific rule set or for data that remains unhandled by the domain-specific rule set.

Modifying overrides for input patterns and unhandled patterns:

Input pattern and unhandled pattern overrides for domain-specific rule sets are used to modify the input pattern (*.IPO) and unhandled pattern (*.UPO) override objects.

About this task

The pattern overrides have the following characteristics:

- With the input pattern override, you can specify value (previously called token) overrides that are based on the input pattern. The input pattern overrides take precedence over the pattern-action specification. Input pattern overrides are specified for the entire input pattern.

- With the unhandled pattern override, you can specify value overrides that are based on the unhandled pattern. Unhandled pattern overrides work on values that are not processed by the pattern-action specification. These overrides are specified only for an entire unhandled pattern. The overrides are not specified for partial pattern matching.

For example, you use an input pattern override if you want to designate the following pattern:

\[^+T\]

Procedure

1. Right-click the SET folder in a domain-specific rule set, such as USADDR, or a validation rule set, such as VTAXID, and select Properties to open the Rules Management window.

2. Double-click Overrides.

3. Click the Input Pattern or Unhandled Pattern tab, as appropriate.

4. From the Classification Legend list, select the first class, such as ^, and click Append this code to current pattern to add the class to the Enter Input Tokens or Enter Unhandled Tokens field. The class also is shown under the Current Pattern List section with the default override code AA1 (additional address) information and code 1.

5. Repeat the previous step for the classes + and T.

7. From the Dictionary Columns list, select the option that represents the type, for example HN-House Number.
   The selected row in the Current Pattern List changes to ^HN1.
8. Repeat step 6 and step 7 for the remaining classes.
   For example, select SN - Street Number for +, and ST - Street Type for T.
9. Select the current pattern from the Current Pattern List section, such as T ST1, for the class that you want to specify the override for.
10. Click Standard Value.
    The row T ST1 changes to T ST2, which indicates that the standard value (and includes an override action code) from the classifications is used for this class in this pattern. The rest of the classes are left as the original value.
    Every selection or combination of selections under User Override creates a code that appears next to the selected row in the Current Pattern List section.
11. Click Add to add the override to the Override Summary list.
12. Click OK to save your edits and close the window.

Modifying overrides for input and unhandled text:

For the domain-specific rule sets, the input text and unhandled text overrides modify the override object (*.ITO) for input text and the override object (*.UTO) for unhandled text.

About this task

The input and unhandled text overrides have the following characteristics:

- With the input text override, you can specify overrides that are based on an entire record of input text. These overrides take precedence over the pattern-action specification. Because they are more specific, input text overrides take precedence over input pattern overrides. Input text overrides are specified for the entire input text string. Partial string matching is not allowed.
- With the unhandled text override, you can specify rule overrides that are based on an entire record of unhandled text. Unhandled text overrides work on exact text strings that are not processed by the pattern-action specification. Because they are more specific, unhandled text overrides take precedence over unhandled pattern overrides. Unhandled text overrides can be specified only for the entire unhandled text string. Partial string matching is not allowed.

For example, the following address contains two values (Street and Floor) that use standard values from the classifications table:

100 Summer Street Floor 15

The remaining values are not associated with standard values from the classification object and use their original data values. The following table shows the address represented as values before you add any overrides:

<table>
<thead>
<tr>
<th>Text</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>House number</td>
<td>Original</td>
</tr>
<tr>
<td>Summer</td>
<td>Street name</td>
<td>Original</td>
</tr>
<tr>
<td>Street</td>
<td>Street type</td>
<td>Standard</td>
</tr>
<tr>
<td>Floor</td>
<td>Floor type</td>
<td>Standard</td>
</tr>
<tr>
<td>15</td>
<td>Floor value</td>
<td>Original</td>
</tr>
</tbody>
</table>
Procedure

1. Right-click the SET folder in a domain-specific rule set, such as USADDR, or a validation rule set, such as VTXAXID, and select Properties to open the Rules Management window.
2. Double-click Overrides.
3. Click the Input Text or Unhandled Text tab, as appropriate.
4. In the Enter Input Tokens or Enter Unhandled Tokens field, type the text string that you want to define a pattern override for, such as 100 SUMMER STREET FLOOR 15.
   Each text value is displayed in the Current Token List section under the Token column. Next to each value, the default code of AA (additional address) information plus the action code 1 is shown.
5. Select the first text value, such as 100.
6. From the Dictionary Columns list, select the code that you want, such as HN - House Number.
   The AA1 next to 100 in the Current Token List section changes to HN1.
7. Repeat the previous two steps for each of the remaining text values, for example:

<table>
<thead>
<tr>
<th>Text Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summer</td>
</tr>
<tr>
<td>Street</td>
</tr>
<tr>
<td>Floor</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

8. Select the text value in the Current Token List section, such as STREET, and then click Standard Value.
   The row STREET ST1 changes to STREET ST2, indicating that the standard value from the classifications is used for the value in this text string. The rest of the values are left as the original value.
9. Repeat the previous step for each text value that you want to standardize. For example, repeat the previous step for the text value FLOOR.
   Every selection or combination of selections under User Override creates a code that appears next to the selected row in the Current Token List section.
10. Click Add to add the override to the Override Summary list.
11. Click OK to save your edits and close the window.

Validation overrides:

Validation overrides modify validation rule sets.

The following table describes the object types that override the validation rule sets:

<table>
<thead>
<tr>
<th>Validation override object names</th>
<th>File type abbreviation</th>
<th>Example (United States Address)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-defined classification</td>
<td>UCL</td>
<td>VTXAXID.CLS</td>
</tr>
<tr>
<td>Input pattern overrides</td>
<td>IPO</td>
<td>VTXAXID.IPO</td>
</tr>
<tr>
<td>Input text overrides</td>
<td>ITO</td>
<td>VTXAXID.IPO</td>
</tr>
<tr>
<td>Unhandled pattern overrides</td>
<td>UPO</td>
<td>VTXAXID.UPO</td>
</tr>
<tr>
<td>Unhandled text overrides</td>
<td>UTO</td>
<td>VTXAXID.UTO</td>
</tr>
</tbody>
</table>
Overrides for international address data:

For Multinational Standardize (MNS) rule sets, you can use the MNS overrides.

The following table describes the object types that override the MNS rule sets:

<table>
<thead>
<tr>
<th>MNS address override names</th>
<th>File type abbreviation</th>
<th>Example (United States)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-defined classification</td>
<td>UCL</td>
<td>USMNAD.CLS</td>
</tr>
<tr>
<td>Input pattern</td>
<td>IPO</td>
<td>USMNAD.IPO</td>
</tr>
<tr>
<td>Input text</td>
<td>ITO</td>
<td>USMNAD.ITO</td>
</tr>
<tr>
<td>Unhandled pattern</td>
<td>UPO</td>
<td>USMNAD.UPO</td>
</tr>
<tr>
<td>Unhandled text</td>
<td>UTO</td>
<td>USMNAD.UTO</td>
</tr>
</tbody>
</table>

Action codes for domain-specific, validation, and Multinational Standardize rule sets:

When a rule set is applied to your data, actions are performed based on the code setting that you select in the text and pattern override panels.

The codes are displayed in the Override Code column of the Current Token List section in the domain-specific text and pattern override panels. You adjust the code settings by making selections in the User Override pane.

- **0 (zero)**
  - Drop the current value (previously called token).
- **1**
  - Append a leading character space and then append the original value of the current value to the specified data type.
- **2**
  - Append a leading character space and then append the standard value for the current value to the specified data type.
- **3**
  - Append the original value of the current value to the specified data type, without appending a leading character space.
- **4**
  - Append the standard value for the current value to the specified data type, without appending a leading character space.
- **5**
  - Move all remaining values while using their original values to the specified data type. Leave one character space between each value.
- **6**
  - Move all remaining values while using their standard values to the specified data type. Leave one character space between each value.
- **7**
  - Move all remaining values to the specified data types, while using their original values. Do not leave a character space between each value.
- **8**
  - Move all remaining values to the specified data types, while using their standard values. Do not leave a character space between each value.

Copying overrides:

You can duplicate a rule and modify its values.

Procedure

1. Select the rule that you want to duplicate.
2. Click **Copy** to copy the values of the override to the Current Token List.
3. Modify the values to create a new override.
4. Click **Add** to add the override to the Override Summary list.

**Modifying overrides:**

You can modify overrides.

**Procedure**

1. Select the override that you want to modify.
2. Click **Edit** to temporarily move the override values to the respective editing areas in the upper part of the screen.
3. Modify the values as needed.
4. Click **Add** to move the override back to the Override Summary list.

**Deleting overrides:**

You can delete overrides.

**Procedure**

1. Select your overrides.
2. Click **Delete** to remove the selected overrides from the list.

**Enhancing standardization rule sets by using the Standardization Rules Designer**

Use the IBM InfoSphere QualityStage Standardization Rules Designer to enhance standardization rule sets. After you enhance rule sets in the Standardization Rules Designer, you can apply the enhanced rule sets in a Standardize stage.

In the Standardization Rules Designer, you can add and modify classifications, lookup tables, and rules. You can also import sample data to work with representative examples.

**The lifecycle of a rule set in the Standardization Rules Designer:**

Enhancing a rule set in the Standardization Rules Designer is an iterative process. When you enhance a rule set, you can collaborate with others and store changes that are not ready for use in a job.

The process of enhancing a rule set in the Standardization Rules Designer includes the following phases:

1. To begin, you create a revision for a rule set by editing that rule set in the Standardization Rules Designer.
   
   Rule sets are stored in the metadata repository. When you create a revision in the Standardization Rules Designer, a copy of the rule set is created in a database for the Standardization Rules Designer.

2. Next, the rule set is enhanced through multiple iterations.
   
   You can keep a revision for as long as you want. The changes that you make in the Standardization Rules Designer affect only the copy of the rule set in the Standardization Rules Designer database. You can change the rule set yourself or collaborate with other users. If you collaborate with other users, only one user can open the revision at a time.
   
   To discard changes that you made in the Standardization Rules Designer, you can reset the revision. When you reset the revision, the revision remains open and is updated with the version of the rule set in the metadata repository.
3. When you reach a particular milestone, you publish the revision. Publishing the revision updates the rule set in the metadata repository by applying the changes that you made in the Standardization Rules Designer. When you publish a revision, you can delete the revision or keep the revision and continue to enhance the rule set.

4. When your enhancements are completed, you delete the revision. Deleting the revision discards the copy of the rule set that is stored in the Standardization Rules Designer database. You can begin the lifecycle again by creating a new revision for the rule set.

The following figure shows the phases of the lifecycle.

![Figure 5. Enhancing a rule set with the Standardization Rules Designer](image)

**Importing sample data into the Standardization Rules Designer:**

To inform the changes that you make in the Standardization Rules Designer, you can import a representative sample data set.

**Before you begin**

In the Standardization Rules Designer, open a revision for a rule set that you want to enhance.

**About this task**

Files that contain sample data must meet the following requirements:

- Files must be plain text files that are smaller than 100 MB.
- Files must use UTF-8 character encoding.
- Files must have only one record per line and a maximum of 4097 Unicode characters per record.

After you choose a file to import sample data from, the sample data is parsed by using the separation and strip characters. All of the characters in the sample data are converted to uppercase.

**Procedure**

1. In the Standardization Rules Designer, click the **Home** tab.
2. From the navigation pane, click **Import Sample Data**.
3. Click **Import**.
4. Browse to the file that contains the sample data set, and then click **Open**.

**Characteristics of good sample data:**

In the Standardization Rules Designer, you work with records from sample data when you add or modify rules and classifications. The sample data that you choose informs the changes that you make in the Standardization Rules Designer and the standardization processes that result.

When you choose sample data to work with in the Standardization Rules Designer, ensure that the data has the following characteristics:

- The format and content of the sample data must be representative of the actual data that you plan to standardize. Choose a subset of the data randomly.
- The sample data set can contain only the types of records that require special handling. However, you might want to include records that you do not plan to affect with your work in the Standardization Rules Designer. If these records are affected by the changes that you make in the Standardization Rules Designer, you can adjust the rules and classifications accordingly. For example, if records in the sample data are affected by a rule that is not meant to handle those records, you can add conditions to the rule.
- The sample data set must be large enough to apply to the entire domain. Ensure that the data set includes anomalies or other rare but valid record types.

Remember that the Standardization Rules Designer processes every value in a record. When you prepare sample data, ensure that the sample records contain only the values that records in your actual data will contain at a particular point in the standardization process.

Suppose that you want to add rules in the Standardization Rules Designer that are applied to records after all other actions in the pattern-action specification (previously called pattern-action file). You can note how the records are changed by previous actions in the pattern-action specification and prepare your sample data accordingly. For example, previous actions in the pattern-action specification might remove values from further processing by assigning those values to the NULL class. When you prepare the sample data, you remove the values that are assigned to the NULL class.

After you import a sample data set, verify in the Standardization Rules Designer that the sample data set matches the format and content that you require. You can browse the patterns in the data and ensure that the distribution of patterns matches your expectations for the data set. For example, if the most common pattern in the sample data set is a pattern that you know is relatively rare in your actual data, the sample data set might not represent your actual data adequately.

**Classifying values in the Standardization Rules Designer:**

In the Standardization Rules Designer, you can view the values that are assigned to each custom class and each default class. You can add custom classes and specify the classification definitions that assign a value to a class.

You can add classification definitions manually or import classification definitions from a source. If more than one definition is specified for a value, you can choose which definition is active.
**Viewing how classification changes affect the pattern rule that handles a record:**

When you change the class that a value is assigned to, records that contain that value are processed differently. You can identify the changes to those records by viewing the patterns on the Rules page before and after you change the class.

**Before you begin**

1. Select a record that contains the value that you want to assign to a different class.
2. Identify the pattern for that record.

**About this task**

Pattern rules apply only to records that match a specified pattern. When you change the class that a value is assigned to, you also change the pattern of any record that contains that value. As a result, the record might be affected by a different pattern rule or no longer be affected by any pattern rule.

For example, the record GRANITE HAMMER 30 CM BROWN TOOLS might match the pattern B+^MCT. The following table shows what each class in the pattern represents.

<table>
<thead>
<tr>
<th>Class label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Product brand</td>
</tr>
<tr>
<td>+</td>
<td>Value that includes only letters</td>
</tr>
<tr>
<td>^</td>
<td>Value that includes only numbers</td>
</tr>
<tr>
<td>M</td>
<td>Unit of measure</td>
</tr>
<tr>
<td>C</td>
<td>Product color</td>
</tr>
<tr>
<td>T</td>
<td>Product type</td>
</tr>
</tbody>
</table>

If a rule is defined for the B+^MCT pattern, and you assign the value HAMMER to a different class, the rule will no longer handle the record GRANITE HAMMER 30 CM BROWN TOOLS.

To ensure that records that contain a changed value are handled correctly, you can view the pattern and pattern rules on the Rules page. If the rule that applies to the record does not meet your standardization requirements for the record, you might need to add a new pattern rule or modify an existing rule.

**Procedure**

1. On the Rules page, select the rule group that contains the relevant rules, and then click **Open**.
2. Identify a rule that handles the record containing the value that you want to assign to a different class:
   a. From the list of patterns, click the twistie for the appropriate pattern.
   b. If more than one rule is specified for the pattern, view the **Add Conditions** page for each rule to determine which set of conditions applies to the record.
3. On the Classifications page, assign the value to a different class. You can assign a value to a different class in the following ways:
   - Add a classification definition for a value that has no definition
   - Change the active definition for a value that has two or more definitions
The value is assigned to a different class. Because the classification of the value changes, the pattern for any record that contains the value also changes. For example, if the value HAMMER is changed from the + class to the N class, the pattern for the record GRANITE HAMMER 30 CM BROWN TOOLS changes from B+^MCT to BN^MCT.

4. Repeat step 2 on page 224 for the new pattern for the record. The record is handled by a different rule or is no longer handled by any pattern rule.

**What to do next**

If the rule that handles the record does not meet your standardization requirements for this type of record, add a new pattern rule or modify an existing rule.

*Adding classification definitions manually:*

You can add classification definitions manually to categorize values that share common attributes.

**Procedure**

1. In the Standardization Rules Designer, click the **Classifications** tab.
2. Click **Define Values**.
3. In the Define Values window, specify a classification definition for the value:
   a. Complete the **Value**, **Standard Value**, **Class**, and **Similarity Threshold** fields to specify the classification definition. For more information about the parts of a classification definition, see "Classification definitions" on page 166.
   b. To make the definition active, expand **Status of Definitions** and ensure that **Active definition** is selected. An inactive definition does not affect the value.
   c. Click **OK**.

*Importing classification definitions:*

You can import classification definitions from a CSV file.

**Before you begin**

Ensure that the CSV file that contains classification definitions is formatted correctly. For more information about the format that is required, see "CSV file format for classification definitions" on page 226.

**About this task**

When you import classification definitions, all of the characters in the definitions are converted to uppercase.

**Procedure**

1. In the Standardization Rules Designer, click the **Classifications** tab.
2. From the **Define Values** list, select **Import Definitions**.
3. Specify the file that contains the definitions to import:
   a. In the Import Definitions window, click **Browse**.
b. Browse to the file that contains the classification definitions, and then click Open.

The first ten definitions in the file are shown.

4. To make the definitions active, expand Status of Definitions and ensure that Active definitions is selected. An inactive definition does not affect the value.

5. Click OK.

CSV file format for classification definitions:

When you import classification definitions in the Standardization Rules Designer, files that contain definitions must be text files that contain values that are separated by commas. Each classification definition in the file must be formatted correctly.

CSV file requirements

The entire CSV file must meet the following requirements:
- Files must have only one definition per line.
- Files must use UTF-8 character encoding.
- If leading or trailing white space must be preserved for an individual value, the entire value must be enclosed in double quotation marks.

Definition requirements

Each classification definition can include a maximum of four columns. The following table shows the four columns in the order that they must be specified and lists requirements for each column.

<table>
<thead>
<tr>
<th>Column</th>
<th>Required column</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Yes</td>
<td>The maximum length is 600 characters.</td>
</tr>
<tr>
<td>Standard value</td>
<td>No</td>
<td>If a standard value is not specified, the standard value is the same as the value.</td>
</tr>
<tr>
<td>Class label</td>
<td>Yes</td>
<td>The class label must be one character.</td>
</tr>
<tr>
<td>Similarity threshold</td>
<td>No</td>
<td>If a similarity threshold is not specified, a default of 900 is assigned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value must be an integer in the range 700 - 900.</td>
</tr>
</tbody>
</table>

Examples

In this example, values are specified only for the required columns.

BOX,,C

As a result, default values are assigned for columns that are not specified. The following table shows the classification definition that is shown in the Standardization Rules Designer.
In this example, values are specified for all of the columns. Because the standard value contains white space, it is enclosed in double quotation marks.

NC,"NORTH CAROLINA",S,800

The following table shows the definition that results.

Table 34. Definition that includes values for all columns

<table>
<thead>
<tr>
<th>Value</th>
<th>Standard value</th>
<th>Class label</th>
<th>Similarity threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>NORTH CAROLINA</td>
<td>S</td>
<td>800</td>
</tr>
</tbody>
</table>

**Adding lookup tables in the Standardization Rules Designer:**

In the Standardization Rules Designer, you can add a lookup table and specify lookup table definitions, which rules can reference as part of actions or conditions.

You can add definitions or import definitions from a source. If more than one definition is specified for a value, you can choose which definition is active.

**Adding lookup table definitions:**

You can add lookup table definitions to lookup tables. Rules can reference lookup tables as part of actions or conditions.

**Before you begin**

Add a lookup table to the rule set by using the Standardization Rules Designer. You cannot add definitions to lookup tables that are not part of a rule set extension.

**Procedure**

1. In the Standardization Rules Designer, click the **Lookup Tables** tab.
2. Select the lookup table that you want to add a definition for.
3. Click **Define Values**.
4. In the Define Values window, specify a definition for the value:
   a. Specify a lookup table definition by completing the **Value**, **Returned Value**, and **Similarity Threshold** fields. For more information about the parts of a lookup table definition, see “Lookup table definitions” on page 169.
   b. To make the definition active, expand **Status of Definitions** and ensure that **Active definition** is selected. An inactive definition does not affect the value.
   c. Click **OK**.

**Importing lookup table definitions:**

You can import lookup table definitions from a CSV file.
Before you begin

Complete the following steps:
1. Add a lookup table to the rule set by using the Standardization Rules Designer. You cannot import definitions into lookup tables that are not part of a rule set extension.
2. Ensure that the CSV file that contains lookup table definitions is formatted correctly. For more information about the format that is required, see "CSV file format for lookup table definitions."

About this task

When you import lookup table definitions, all of the characters in the definitions are converted to uppercase.

Procedure
1. In the Standardization Rules Designer, click the Lookup Tables tab.
2. Select the lookup table that you want to import definitions into.
3. From the Define Values list, select Import Definitions.
4. Specify the file that contains the definitions to import:
   a. In the Import Definitions window, click Browse.
   b. Browse to the file that contains the classification definitions, and then click Open.

The first ten definitions in the file are shown.
5. To make the definitions active, expand Status of Definitions and ensure that Active definitions is selected. An inactive definition does not affect the value.
6. Click OK.

CSV file format for lookup table definitions:

When you import lookup definitions in the Standardization Rules Designer, files that contain definitions must be text files that contain values that are separated by commas. Each lookup table definition in the file must be formatted correctly.

CSV file requirements

The entire CSV file must meet the following requirements:
1. Files must have only one definition per line.
2. Files must use UTF-8 character encoding.
3. If leading or trailing white space must be preserved for an individual value, the entire value must be enclosed in double quotation marks.

Definition requirements

Each lookup table definition can include a maximum of three columns. The following table shows the three columns in the order that they must be specified and lists requirements for each column.

<table>
<thead>
<tr>
<th>Column</th>
<th>Required column</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Yes</td>
<td>The maximum length is 600 characters.</td>
</tr>
</tbody>
</table>
Table 35. Columns in a lookup table definition (continued)

<table>
<thead>
<tr>
<th>Column</th>
<th>Required column</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returned value</td>
<td>No</td>
<td>If a returned value is not specified, it is the same as the value. The maximum length is 600 characters.</td>
</tr>
<tr>
<td>Similarity threshold</td>
<td>No</td>
<td>If a similarity threshold is not specified, a default of 900 is assigned. The similarity threshold must be an integer in the range 700 - 900.</td>
</tr>
</tbody>
</table>

Examples

In this example, the returned value and similarity threshold are not specified.

METER,,

As a result, the returned value is the same as the value, and a default value is assigned for the similarity threshold. The following table shows the lookup table definition that is shown in the Standardization Rules Designer.

Table 36. Definition that includes defaults for the returned value and similarity threshold

<table>
<thead>
<tr>
<th>Value</th>
<th>Returned value</th>
<th>Similarity threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>METER</td>
<td>METER</td>
<td>900</td>
</tr>
</tbody>
</table>

In this example, values are specified for all of the columns. Because the value contains white space, it is enclosed in double quotation marks.

"FREIGHT TON",IMPERIAL,800

The following table shows the lookup table definition that is shown in the Standardization Rules Designer.

Table 37. Definition that includes values for all columns

<table>
<thead>
<tr>
<th>Value</th>
<th>Returned value</th>
<th>Similarity threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREIGHT TON</td>
<td>IMPERIAL</td>
<td>800</td>
</tr>
</tbody>
</table>

Adding rules to a rule group:

In the Standardization Rules Designer, you can add pattern rules and exact text rules to a rule group. For each pattern, you can view the records that are handled by a rule and records that are not handled.

Adding conditions in the Standardization Rules Designer:

In the Standardization Rules Designer, you add one or more conditions that values must meet before the actions in a rule are applied to the record.

Before you begin

On the Rules page, click an example record that you want to add a condition for.
About this task

Before actions in a rule are applied to a record, the record must meet all of the conditions of the rule. For example, suppose that all of the records that match a pattern are unhandled. You can choose an example record, add conditions and actions, and apply the rule as a new rule. After you apply the rule, the rule is shown under the pattern. The example records that meet the conditions of the new rule are shown under the rule. Example records that do not meet the conditions are still unhandled.

Procedure
1. Click the Add Conditions tab.
2. In the Add Conditions table, select the row for the value that you want to add a condition for.
3. Click Add.
4. In the Conditions window, select the object you want the condition to apply to from the Object list. When a record matches the pattern, the condition will apply to any value in this position in the record.
5. Specify a condition for the object, and then click OK.

Adding actions in the Standardization Rules Designer:

In the Standardization Rules Designer, you add one or more actions that specify how a rule processes a value.

Before you begin

On the Rules page, click an example record that you want to add an action for.

Procedure
1. On the Define Rule page, right-click a value in the example record, and then click New Action.
2. From the Object list, select the object to act on. When a record matches the pattern and other conditions are met, the rule will process the object that you specify for any value in this position in the record.
3. Optional: Look up the object in a list or lookup table:
   a. Choose a lookup table or list.
   b. Specify how the object that is looked up is processed when it is found and when it is not found in the lookup table or list.
4. From the Output Column list, select the output column to add the resulting string to.
5. Optional: From the Leading Separator list, select a character to separate strings in the output column. You might select a character that is not the default, a space, when you want to add more than one type of information to the output column. For example, if you want to add a family name and a given name to an output column, you might select a comma (,).
6. Click OK.

Applying rule group changes to the pattern-action specification:

When you add a rule group, delete a rule group, or edit the name of a rule group in the Standardization Rules Designer, the pattern-action specification (previously called pattern-action file) must be updated.
About this task

The pattern-action specification contains references to the rule groups that are managed in the Standardization Rules Designer. The pattern-action specification uses the pattern-action language and is intended to be edited primarily by application developers. If you do not have experience with the pattern-action specification, consider contacting an application developer or other experienced user to complete the updates that are required.

For a rule set to work correctly, the references to the rule groups in the pattern-action specification must match the information about the rule groups in the Standardization Rules Designer. Before you provision a rule set and apply it in a job, you must ensure that changes are published from the Standardization Rules Designer and that the pattern-action specification is updated to match the Standardization Rules Designer.

The pattern-action specification for a rule set must be updated when you do any of the following actions in the Standardization Rules Designer:

- You open a revision for a rule set that you created or that you copied from a predefined rule set from Version 8.7 or earlier.
- You add a rule group.
- You change the name of a rule group.
- You delete a rule group.

Predefined rule sets contain default rule groups. If you use a default rule group, you do not need to update the pattern-action specification.

Procedure

1. In the Designer client repository tree, double-click the SET folder for the rule set that you want to update.
2. Double-click Patterns.
3. In the appropriate section of the pattern-action specification, add, delete, or update the reference to the rule group. References to rule groups are made as part of a subroutine. A complete reference has the following format:

   
   
   \[
   \text{PROCESS} \_ \text{RC Rule} \_ \text{Group}\_\text{Name RC} \_ \text{Return}
   \]

   \[
   \text{COPY} \_ \text{CI} \_ \{ \text{UserOverrideFlag} \}
   \]

   \[
   \text{CALL Post\_Process}
   \]

   \[
   \text{EXIT}
   \]

   

   

   where \text{Return}\_\text{Code}\) is the return code value, which indicates whether and how a rule in the rule group applies to a particular input record.

What to do next

Use the standardization rule set tester to verify that the updated rule set processes data correctly and does not contain errors.

Converting overrides to rule set extensions:

If you use overrides in a rule set, you can convert the overrides to rule set extensions before you enhance the rule set in the Standardization Rules Designer. You can use a tool that is installed with the Standardization Rules Designer to convert the overrides.
About this task

The override conversion tool uses .dsx files for both input and output.

When the tool converts overrides to rule set extensions, the overrides are commented out. If no revisions have been opened for the rule set in the Standardization Rules Designer, a copy of the rule set is created. The characters _RC are appended to the rule set name, and the rule set is added to the RC Converted folder in the Designer client repository tree. For example, the name of the DENAME.SET file changes to DENAME_RC.SET. The rule set is added to the RC Converted folder, which is in the DENAME folder.

If you want to convert the overrides, you must convert them before you enhance the rule set in the Standardization Rules Designer for the first time. You cannot convert the overrides later.

Procedure
1. Export the rule set to a .dsx file:
   a. In the Designer client repository tree, right-click the rule set that you want to export, and then click Export.
   b. Ensure that the Exclude read-only items check box is cleared and that the Include dependent items check box is selected.
   c. Specify a name for the exported file, and then click Export.
2. From the client_tier_installation_directory\ASBNode\bin directory, run the following command:
   .\QSBaseDSXConverter.bat -inputDSX input_file -outputDSX output_file
   where input_file is the file path for the input .dsx file and output_file is the file path for the output .dsx file. For example, you can run the following command:
   .\QSBaseDSXConverter.bat -inputDSX C:\My_Test\DENAME.dsx -outputDSX C:\My_Test\DENAME_RC.dsx
3. Complete the conversion process by following the command prompts.
4. Import the .dsx file into the Designer client:
   a. In the Designer client, click Import > DataStage Components.
   b. Specify the file path to the .dsx file that was created by the conversion tool.
   c. Click OK.

What to do next

Update the pattern-action specification so that it references the rule groups that are rule set extensions. References to these rule groups use different syntax than references to rule groups for overrides.

User modification subroutines

Occasionally, you need to make rule set modifications that are more complex than what can be done with user overrides. You must make the modifications by using the pattern-action language. Each rule set used for standardization has subroutines that are reserved for these modifications.

Typically, when you want to modify a portion of the data or pattern, you use one of the following types of user modification subroutines:

Input modifications
Used in COUNTRY, preprocessor and domain-specific rule sets. The field is
blank until you modify it, as needed. This is the only modification subroutine type used in COUNTRY rule sets.

**Field modifications**
Used only in preprocessor rule sets. The field is blank until you modify it, as needed.

**Unhandled modifications**
Used only in domain-specific rule sets. The field is blank until you modify it, as needed.

In the following example, the post office (P.O.) box information precedes street information. In standard rule sets, the P.O. box number follows the street information. In this example, the modification subroutine moves the P.O. box information to P.O. box-related output fields.

```
\SUB Input_Modifications

;PO BOX 123 456 Main Street

B | B | ^ | ^ | & [ {BoxType} = "" & {BoxValue} = "" ]
COPY_A [1] {BoxType}
COPY [3] {BoxValue}
RETYPE [1] 0
RETYPE [2] 0
RETYPE [3] 0

& ; No patterns matched in the Input_Modifications SUBROUTINE
RETURN

\END_SUB
```

You can find more information about writing subroutines in the pattern-action language in the *IBM InfoSphere QualityStage Pattern-Action Reference*.

You cannot use subroutines to modify the MNS rule sets.

**Country or region identifier user subroutines:**

Rules that are added to the input modifications subroutine are performed before any other rules in the COUNTRY rule set.

Add modifications in the pattern-action specification (previously called pattern-action file) if you have determined that certain conditions are completely mishandled or unhandled by the rule set.

The input subroutine section of the pattern-action specification is found at the beginning of the subroutine section or by searching for the following string:

```
;--------------------------------------------------
;Input_Modifications SUBROUTINE Starts Here
;--------------------------------------------------
\SUB input_Modifications
```

**Domain preprocessor user subroutines:**

Subroutines exist within the pattern-action specification for a domain preprocessor rule set.
Input modifications:

Standardization rules that are added to the input modifications subroutine are performed before any other rules.

Modifications should be added here if you have determined that certain conditions are completely mishandled or unhandled by the rule set.

The subroutine section of the pattern-action specification is delimited by a header, as shown here:

;------------------------------------------------
; Input_Modifications SUBROUTINE Starts Here
;------------------------------------------------
\SUB Input_Modifications

Field modifications:

The second step in the domain preprocessor logical flow is to isolate each delimited input field, one at a time, to search for common domain patterns.

Standardization rules that are added to the field modifications subroutine are performed before any other field rules.

The input subroutine section of the pattern-action specification can be found at the beginning of the subroutine section or by searching for the following string:

;----------------------------
; Field_Modifications SUBROUTINE Starts Here
;----------------------------
\SUB Field_Modifications

Domain-specific user subroutines:

Subroutines exist within a domain-specific pattern-action specification.

Add modifications here if you have determined that certain conditions are wholly or partially mishandled or unhandled by the rule set.

Input modifications:

Standardization rules that are added to the input modifications subroutine are performed before any other rules.

The input modification subroutine can be found in the pattern-action specification at the beginning of the subroutine section or by searching for the following string:

;--------------------------
; Input_Modifications
;--------------------------
& ; CALL Input_Modifications SUBROUTINE
CALL Input_Modifications

Unhandled modifications:

Standardization rules that are added to the unhandled modifications subroutine are performed after all other rules.
The unhandled modification subroutine can be found in the pattern-action specification at the beginning of the subroutine section or by searching for the following string:

```
;--------------------------------
; Unhandled Modifications
;--------------------------------
& ; CALL Unhandled_Modifications SUBROUTINE
CALL Unhandled_Modifications
```

**Standardization rule set tester**

The tester lets you quickly test a selected standardize rule set against a one-line test string (a single record) before you run the rule set against an entire input source.

This time-saving option is especially helpful when you plan to use a large input source. When you test the rule set, you can ensure that the resulting input data source performs the way you expect.

You can test domain preprocessor rule sets, domain-specific rule sets, and validation rule sets with the tester.

**Testing a domain preprocessor rule set**

You can test a new or modified domain preprocessor rule set by using the rules management tester.

**About this task**

When you select a domain preprocessor rule set, such as PREP to test, the tester first populates each **Input String** field. Each individual row in a preprocessor rule set test screen has its own history of up to five previously entered input strings. The Designer client maintains a separate history log for each rule set.

You can enter up to six strings of data to test. If you enter more than one input string, the only requirement is that they be in the right order and that the delimiter field next to each input string be set. You can leave blank rows in between.

**Note:** You must set the delimiter for each input string; you cannot leave it at **None**. If you attempt to run the test without specifying a delimiter for each string input, an alert message is displayed.

The input string for preprocessor rule sets is a concatenation of all user-inputted strings that are separated by the delimiter for each string. So the input file contains one long line that is created by a concatenated string of input strings and delimiters.

**Procedure**

1. Right-click the **SET** folder in a rule set and select **Properties** to open the Rules Management window.
2. Click **Test**.
3. If needed, type a new path in the **Locale** field. The standardization tester supports international rules. If no locale is specified, the tester assumes that you want to use the default locale to which your computer is set. By specifying a different locale, you can run data against rule sets that are not designed for the default locale.
4. Enter the input strings that you want to test, or select a previously entered string from the input string's menu.
5. Select a **Delimiter** for each input string that you enter.
6. Click **Test this String** to start the testing.
   - If the test finishes successfully, the results are displayed in the bottom grid, which lists each value (previously called token) and its fields and field descriptions.
7. To stop a test in progress, click **Cancel** to restore the panel to its previous state.
8. To remove all data from all input boxes, reset the delimiters to **None**, and click **Clear Data**.
   - You can then test the same rule set by using a different set of input strings, or you can use the **Rule Set** list to select a different rule set to test.
9. When you finish testing, click **Exit**.

**Testing domain-specific or validation rule sets**
You can test a new or modified domain-specific or validation rule set by using the rules management tester.

**About this task**
When you select a domain-specific or validation rule set, such as ADDR or VTAXID to test, the tester first populates each **Input String** field. Each individual row in a domain-specific or validation rule set test panel has its own history of up to five previously entered input strings. The Designer client maintains a separate history log for each rule set.

For domain-specific or validation rule sets, you can enter only one input string to test.

**Procedure**
1. Right-click the **SET** folder in a rule set and select **Properties** to open the Rules Management window.
2. Click **Test**.
3. Enter the input strings that you want to test, or select a previously entered string from the input string's menu.
4. Click **Test this String** to start the testing.
   - If the test finishes successfully, the results are displayed in the bottom grid, which lists each value (previously called token) and its fields and field descriptions.
5. To stop a test in progress, click **Cancel** to restore the panel to its previous state.
6. To remove the data from the input box and clear the results grid, click **Clear Data**.
   - You can then test the same rule set by using a different set of input strings, or you can use the **Rule Set** list to select a different rule set to test.
7. When you finish testing, click **Exit**.

**Selecting a rule set editor**
You can modify a rule set using an editor installed on your computer.

**Before you begin**
Open the Rules Management window.
Procedure
1. From the Rules Management window, click Options.
2. From the Rules Management Options window, browse for your preferred editor.
3. Select the editor and click OK.

Reference: Rule sets

The reference topics provide more in-depth information about the rule sets that are provided with InfoSphere QualityStage.

You can use these topics to better understand the capabilities of rule sets.

Predefined rule sets
You can use predefined rule sets that are country- or region-specific or that can be applied internationally.

Country or region rule sets

In the Designer client repository view, expand the Standardization Rules folder and select the country or region that you want. The folder for the country or region contains the provided rule sets.

You can install additional rule sets from asset interchange and then import the rule sets into your project.

Table 38. Rule sets for countries or regions

<table>
<thead>
<tr>
<th>Country or region</th>
<th>Rule sets</th>
<th>Location of rule set files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina</td>
<td>ARADDR</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td></td>
<td>ARAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARPREP</td>
<td></td>
</tr>
<tr>
<td>Australia</td>
<td>AUADDR</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>AUAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AUNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AUPREP</td>
<td></td>
</tr>
<tr>
<td>Brazil</td>
<td>BRADDR</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td></td>
<td>BRAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BRPREP</td>
<td></td>
</tr>
<tr>
<td>Canada</td>
<td>CAADDR</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>CAAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CANAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CAPREP</td>
<td></td>
</tr>
<tr>
<td>Chile</td>
<td>CHADDR</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td></td>
<td>CHAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHPREP</td>
<td></td>
</tr>
<tr>
<td>Country or region</td>
<td>Rule sets</td>
<td>Location of rule set files</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>----------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>France</td>
<td>FRADDR, FRAREA, FRNAME, FRPREP</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td>Germany</td>
<td>DEADDR, DEAREA, DENAME, DEPREP</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td>Hong Kong Special Administrative Region of the People's Republic of China - Chinese characters</td>
<td>HKCADDR, HKCNAME</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>Note: The Name domain includes data typically found in the Area domain. The Prep domain is not needed.</td>
<td></td>
</tr>
<tr>
<td>Hong Kong Special Administrative Region of the People's Republic of China - Latin characters</td>
<td>HKADDR, HKNAME, HKPHONE</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>Note: The Name domain includes data typically found in the Area domain. The Prep domain is not needed.</td>
<td></td>
</tr>
</tbody>
</table>
Table 38. Rule sets for countries or regions (continued)

<table>
<thead>
<tr>
<th>Country or region</th>
<th>Rule sets</th>
<th>Location of rule set files</th>
</tr>
</thead>
<tbody>
<tr>
<td>India</td>
<td>INAPAD</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td></td>
<td>INAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INASAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INBHAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INDLAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INGJAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INHPAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INHRAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INJKAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INKAAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INKEAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INMHAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INMPAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INORAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INPBAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INRJAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INTNAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INUPAD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INWBAD</td>
<td></td>
</tr>
<tr>
<td>Note:</td>
<td>The IndiaAddressSharedContainer shared container is imported with the Indian address rule sets. The shared container can be used in a job that standardizes Indian address and area data.</td>
<td></td>
</tr>
<tr>
<td>Ireland</td>
<td>IEADDR</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td></td>
<td>IEAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IENAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IEPREP</td>
<td></td>
</tr>
<tr>
<td>Italy</td>
<td>ITADDR</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>ITAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ITPREP</td>
<td></td>
</tr>
<tr>
<td>Japan</td>
<td>JP1PHN</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>JP2PHN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JPADDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JPAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JPDATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JPNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JPTRIM</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 6. Rule sets applied in the data cleansing process  239
<table>
<thead>
<tr>
<th>Country or region</th>
<th>Rule sets</th>
<th>Location of rule set files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Japan</strong></td>
<td>JPKANA, JPKNAM</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td></td>
<td>JPKANA converts data from Katakana to Kanji.</td>
<td></td>
</tr>
<tr>
<td><strong>Korea</strong></td>
<td>KOADDR, KOAREA, KONAME, KOPREP</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td><strong>Mexico</strong></td>
<td>MXADDR, MXAREA, MXNAME, MXPREP</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td><strong>Netherlands</strong></td>
<td>NLADDR, NLAREA, NLNAME, NLPREP</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td><strong>People’s Republic of China</strong></td>
<td>CNADDR, CNAREA, CNNAME, CNPHONE</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td><strong>Peru</strong></td>
<td>PEADDR, PEAREA, PENAME, PEPREP</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td><strong>Russia</strong></td>
<td>RUADDRL, RUNAMEL</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td></td>
<td>The RUADDRL rule set standardizes address and area information.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>To use these rule sets, you must set the Windows code page to 1251 and the regional settings for your operating system to Russian.</td>
<td></td>
</tr>
<tr>
<td><strong>Spain</strong></td>
<td>ESADDR, ESAREA, ESNAME, ESPREP</td>
<td>Designer client repository view</td>
</tr>
</tbody>
</table>
Table 38. Rule sets for countries or regions (continued)

<table>
<thead>
<tr>
<th>Country or region</th>
<th>Rule sets</th>
<th>Location of rule set files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thailand</td>
<td>THADDRL</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td></td>
<td>THNAMEL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The THADDRL rule set standardizes address and area information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To use these rule sets, you must set the Windows code page to 847 and set the regional settings for your operating system to the Thai language. In the job properties, set the NLS parameter to TIS-620.</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>GBADDR</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>GBAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GBNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GBPREP</td>
<td></td>
</tr>
<tr>
<td>United States</td>
<td>USADDR</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>USAREA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>USNAME</td>
<td></td>
</tr>
<tr>
<td></td>
<td>USPREP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>USTAXID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The USTAXID rules validate United States tax ID format, including the following characteristics:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nine digits</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all numeric</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not consisting of one number, for example 000-00-0000</td>
</tr>
</tbody>
</table>

**Rule sets in the other category**

In the Designer client repository view, expand the **Standardization Rules** folder and select the folder named **Other**. The **Other** folder contains the provided in-country or region-specific rule sets or rule sets that can be applied to more than one country.
<table>
<thead>
<tr>
<th>Scope</th>
<th>Rule sets</th>
<th>Location of rule set files</th>
</tr>
</thead>
<tbody>
<tr>
<td>International</td>
<td>COUNTRY</td>
<td>Designer client repository view</td>
</tr>
</tbody>
</table>
|                                      | Determines the country or region of origin for incoming data. This rule set looks at area information, such as city, state or province, locality, and postal code. The rule set attempts to identify the country or region to which the information belongs and assign the ISO territory code.  
|                                      | Expected input: multinational address information with a default two byte country code delimiter that is enclosed in the characters ZQ, for example: ZQUSZQ. | Designer client repository view |
| United States-specific Expanded Company Name | EXPCOM    | Designer client repository view                                  |
|                                      | Parses company name information into up-to-eight match words (excluding words such as the and and) and extracts distinguishing information such as:  
|                                      | TradeName  
|                                      | StateOrgNum  
|                                      | FranchiseNumber  
|                                      | Division  
|                                      | AccountInfo  
|                                      | CorpDate  
|                                      | Expected input: a valid company name. | Designer client repository view |
| Validation                           | VDATE     | Designer client repository view                                  |
|                                      | Verifies date formats, for example:  
|                                      | mmddyyyy  
|                                      | ddmmyyyy  
|                                      | mmddyy  
|                                      | ddmmyy  
|                                      | yyyyymmdd | Designer client repository view |
| Validation                           | VEMAIL    | Designer client repository view                                  |
|                                      | Verifies email formats, for example:  
|                                      | first.last@domain.org  
|                                      | name@domain.com.ca | Designer client repository view |
Table 39. Rule sets in the other category (continued)

<table>
<thead>
<tr>
<th>Scope</th>
<th>Rule sets</th>
<th>Location of rule set files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation</td>
<td>VPHONE</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>Verifies United States, Canada, and Caribbean phone formats, for example: (nnn) nnn-nnnn nnn nnn nnn nnnnnnnnn Note: A number sequence that leads with 1, is considered invalid.</td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td>VTAXID</td>
<td>Designer client repository view</td>
</tr>
<tr>
<td></td>
<td>Provides rules that validate country-specific tax ID formats. Intended for only the United States.</td>
<td></td>
</tr>
</tbody>
</table>

Rule sets in the sample category

In the Designer client repository view, expand the Standardization Rules folder and select the folder named Sample. The Sample folder contains sample rule sets that can be applied to different types of data.

Table 40. Rule sets in the sample category

<table>
<thead>
<tr>
<th>Rule sets</th>
<th>Location of rule set files</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENPROD</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td>Demonstrates how product description data can be processed through standardization.</td>
<td></td>
</tr>
<tr>
<td>PHPROD</td>
<td>InfoSphere Information Server file directory</td>
</tr>
<tr>
<td>Demonstrates how pharmaceutical data can be processed through standardization.</td>
<td></td>
</tr>
</tbody>
</table>

Rule set development package

You can install a rule set development package from asset interchange, in the InfoSphere Information Server file directory. The package includes the following items:

Standardization rules templates
Includes templates for domains: address, area, and name.

Standardization rules development kit
Consists of a series of jobs that produce reports to assist in custom rules development.

Standardization quality assessment kit
Consists of a series of jobs that produce reports to assist in evaluating standardization results.

Characteristics of Japanese rule sets
Japanese-specific rule sets act upon input and have specific rule set behavior.
**JP1PHN and JP2PHN**

The expected input into Japanese Phone 1 (JP1PHN) and Phone 2 (JP2PHN) rule sets is valid telephone number data for Japan.

The rule sets parse input into a dialing code, area code, exchange, and line number. The rule sets use characters rather than numbers to parse the input. If the input does not contain characters, the input is not parsed and is placed in the UnhandledData column. For example, 0123456789 is not parsed by the JP1PHN or JP2PHN rule set and is placed in the UnhandledData column.

**JPDATE**

The expected input into a Japanese date rule set (JPDATE) is valid date strings that are written in the Gregorian calendar and Japanese era system.

The rule set standardizes input into the Gregorian calendar year. For example, an input year provided in the Japanese era system is standardized into the Gregorian calendar year.

**JPNAME**

The expected input into a Japanese name rule set (JPNAME) is individual names and business names that are written in Japanese characters.

The rule set can handle the data without delimiters, but you can add a literal string to indicate which type of data is given: individual or business.

The rule set handles the input data as an individual name when the input data is followed by the literal ZQINDVZQ. The rule set handles the input data as a business name when the input data is followed by ZQORGNZQ.

**JPTRIM**

The expected input into a Japanese TRIM rule set (JPTRIM) is address data that is written in Japanese characters.

When the input data is written in Katakana, the rule set behavior looks for the literal ZQTRIMZQ placed after the input data. When the data is written in Kanji, the rule set behavior looks for the literal ZQTRIMZQ placed in front of the input data.

**JPAREA**

The expected input into a Japanese Area rule set (JPAREA) is address data that is written in Japanese Kanji characters without blank spaces. You can ensure the input contains no blank spaces by using the output column of JPTRIM, TrimmedValueKanji.

The rule set standardizes the input data into prefecture name, city, JIS5 code, and address domains. The JIS5 code is the number assigned to Japanese cities. The JPADDR rule set uses the JIS5 code to standardize the address domain.
**JPADDR**

The expected input into a Japanese Address rule set (JPADDR) is the street address part of Japanese address data and the JIS5 code for its city. The input can be created by the output column of JPAREA AddressDomain, the literal ZQADDRZQ, and the output column of JPAREA JIS5Code.

The rule set standardizes the input into the Japanese address system, for example, Oaza, Koaza, house numbers, and building name.

**JPKANA**

The expected input into a Japanese Kana rule set (JPKANA) is Japanese address data that is written in Katakana and that is standardized by JPTRIM. The rule set looks for the input to be Trimmed-ValueKanji, which is the output column of JPTRIM. The rule set also looks for the literal ZQKANAZQ, and the output column of JPTRIM "TrimmedValueLookup".

The rule set standardizes the input data into Japanese Kanji addresses.

**JPKNAM**

The expected input into a Japanese Kana Name rule set (JPKNAM) is individual names and business names that are written in Katakana characters. The names must be followed by a literal string that indicates the input type.

The literal string is expected to be ZQINDVZQ when the input is an individual name and ZQORGNZQ when the input is a business name.

The rule set standardizes the input data and provides as output one of the following names:
• Given name and surname when the literal string indicates that the input data is an individual name
• Company name, company prefix, and branch name when the literal string indicates that the input data is a business name

**Indian address rule sets**

Indian address rule sets standardize data based on the conventions of a particular region.

The following table lists the regions for which each Indian address rule set standardizes data.

<table>
<thead>
<tr>
<th>Rule set</th>
<th>States</th>
<th>Territories</th>
</tr>
</thead>
<tbody>
<tr>
<td>INAPAD</td>
<td>Andhra Pradesh</td>
<td></td>
</tr>
<tr>
<td>Rule set</td>
<td>States</td>
<td>Territories</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>INASAD</td>
<td>Arunachal Pradesh</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assam</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manipur</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Meghala</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mizoram</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nagaland</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sikkim and Tripura</td>
<td></td>
</tr>
<tr>
<td>INBHAD</td>
<td>Bihar</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Jharkhand</td>
<td></td>
</tr>
<tr>
<td>INDLAD</td>
<td>Delhi</td>
<td></td>
</tr>
<tr>
<td>INGJAD</td>
<td>Gujarat</td>
<td>Dadra and Nagar Haveli</td>
</tr>
<tr>
<td>INHPAD</td>
<td>Himachal Pradesh</td>
<td></td>
</tr>
<tr>
<td>INHRAD</td>
<td>Haryana</td>
<td></td>
</tr>
<tr>
<td>INJKAD</td>
<td>Jammu and Kashmir</td>
<td></td>
</tr>
<tr>
<td>INKAAD</td>
<td>Karnataka</td>
<td></td>
</tr>
<tr>
<td>INKEAD</td>
<td>Kerala</td>
<td>Lakshwadeep</td>
</tr>
<tr>
<td>INMHAD</td>
<td>Goa</td>
<td>Daman and Diu</td>
</tr>
<tr>
<td></td>
<td>Maharashatra</td>
<td></td>
</tr>
<tr>
<td>INMPAD</td>
<td>Chhattisgarh</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Madhya Pradesh</td>
<td></td>
</tr>
<tr>
<td>INORAD</td>
<td>Orissa</td>
<td></td>
</tr>
<tr>
<td>INPBAD</td>
<td>Punjab</td>
<td>Chandigarh</td>
</tr>
<tr>
<td>INRJAD</td>
<td>Rajasthan</td>
<td></td>
</tr>
<tr>
<td>INTNAD</td>
<td>Tamil Nadu</td>
<td>Pondicherry</td>
</tr>
<tr>
<td>INUPAD</td>
<td>Uttarakhund</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Uttar Pradesh</td>
<td></td>
</tr>
<tr>
<td>INWBAD</td>
<td>West Bengal</td>
<td>Andaman And Nicobar Islands</td>
</tr>
</tbody>
</table>

**Shared container for Indian address rule sets**

You can use the IndiaAddressSharedContainer shared container in a job that standardizes Indian address and area data. The shared container is imported with the Indian address rule sets.

Each Indian address rule set standardizes data based on the conventions of a particular region. The IndiaAddressSharedContainer shared container includes one Standardize stage for each of the Indian address rule sets. The Standardize stages implement standardization processes by applying the Indian address rule sets to the input data.
The shared container takes two columns as input: one column specifies a key for the record, and the other column contains the input address. The Area_Stan Standardize stage standardizes the area data, which includes the pin code, country, state, district and taluk. The Area_Stan Standardize stage also generates the StateIdentifier column, which contains information that indicates the region that the record belongs to. The output from the Area_Stan Standardize stage is used as input for the BifurcateStateWise Switch stage.

The BifurcateStateWise Switch stage sends each record to the Standardize stage for a particular region based on the value of the StateIdentifier column. If a record does not have a valid value in the StateIdentifier column, the address data in that record is not standardized and the record is sent directly to the output file.

The Standardize stage for each region standardizes the address data in the records that are sent to the stage by the BifurcateStateWise Switch stage. The output from each Standardize stage is then sent to the ConsolidateAllStateStans Funnel stage. The ConsolidateAllStateStans Funnel stage copies the input records to a single output file.

For information about how to use the IndiaAddressSharedContainer shared container in a job, see the annotation for the container.

**ISO territory codes**

InfoSphere QualityStage recognizes two- and three-character ISO territory or region codes.

ISO territory codes are used in the following situations:

- As part of the naming convention for preprocessor and domain specific rule sets
- When you use the COUNTRY rule set and must provide a default delimiter
- As part of the output from the COUNTRY rule set

Input data might include ISO codes as part of a record. Usually, the COUNTRY rule set uses 2-byte territory codes. The MNS stage provides output that contains both 2-byte and 3-byte codes.

<table>
<thead>
<tr>
<th>Territory or region</th>
<th>Two-character</th>
<th>Three-character</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFGHANISTAN</td>
<td>AF</td>
<td>AFG</td>
</tr>
<tr>
<td>ALBANIA</td>
<td>AL</td>
<td>ALB</td>
</tr>
<tr>
<td>ALGERIA</td>
<td>DZ</td>
<td>DZA</td>
</tr>
<tr>
<td>AMERICAN SAMOA</td>
<td>AS</td>
<td>ASM</td>
</tr>
<tr>
<td>ANDORRA</td>
<td>AD</td>
<td>AND</td>
</tr>
<tr>
<td>ANGOLA</td>
<td>AO</td>
<td>AGO</td>
</tr>
<tr>
<td>ANGUILLA</td>
<td>AI</td>
<td>AIA</td>
</tr>
<tr>
<td>ANTARCTICA</td>
<td>AQ</td>
<td>ATA</td>
</tr>
<tr>
<td>ANTIGUA AND BARBUDA</td>
<td>AG</td>
<td>ATG</td>
</tr>
<tr>
<td>ARGENTINA</td>
<td>AR</td>
<td>ARG</td>
</tr>
<tr>
<td>ARMENIA</td>
<td>AM</td>
<td>ARM</td>
</tr>
<tr>
<td>ARUBA</td>
<td>AW</td>
<td>ABW</td>
</tr>
<tr>
<td>AUSTRALIA</td>
<td>AU</td>
<td>AUS</td>
</tr>
<tr>
<td>AUSTRIA</td>
<td>AT</td>
<td>AUT</td>
</tr>
<tr>
<td>Territory or region</td>
<td>Two-character</td>
<td>Three-character</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>AZERBAIJAN</td>
<td>AZ</td>
<td>AZE</td>
</tr>
<tr>
<td>BAHAMAS</td>
<td>BS</td>
<td>BHS</td>
</tr>
<tr>
<td>BAHRAIN</td>
<td>BH</td>
<td>BHR</td>
</tr>
<tr>
<td>BANGLADESH</td>
<td>BD</td>
<td>BGD</td>
</tr>
<tr>
<td>BARBADOS</td>
<td>BB</td>
<td>BRB</td>
</tr>
<tr>
<td>BELARUS</td>
<td>BY</td>
<td>BLR</td>
</tr>
<tr>
<td>BELGIUM</td>
<td>BE</td>
<td>BEL</td>
</tr>
<tr>
<td>BELIZE</td>
<td>BZ</td>
<td>BLZ</td>
</tr>
<tr>
<td>BENIN</td>
<td>BJ</td>
<td>BEN</td>
</tr>
<tr>
<td>BERMUDA</td>
<td>BM</td>
<td>BMU</td>
</tr>
<tr>
<td>BHUTAN</td>
<td>BT</td>
<td>BTN</td>
</tr>
<tr>
<td>BOLIVIA</td>
<td>BO</td>
<td>BOL</td>
</tr>
<tr>
<td>BOSNIA AND HERZEGOWINA</td>
<td>BA</td>
<td>BIH</td>
</tr>
<tr>
<td>BOTSWANA</td>
<td>BW</td>
<td>BWA</td>
</tr>
<tr>
<td>BOUVET ISLAND</td>
<td>BV</td>
<td>BVT</td>
</tr>
<tr>
<td>BRAZIL</td>
<td>BR</td>
<td>BRA</td>
</tr>
<tr>
<td>BRITISH INDIAN OCEAN TERRITORY</td>
<td>IO</td>
<td>IOT</td>
</tr>
<tr>
<td>BRUNEI DARUSSALAM</td>
<td>BN</td>
<td>BRN</td>
</tr>
<tr>
<td>BULGARIA</td>
<td>BG</td>
<td>BGR</td>
</tr>
<tr>
<td>BURKINA FASO</td>
<td>BF</td>
<td>BFA</td>
</tr>
<tr>
<td>BURUNDI</td>
<td>BI</td>
<td>BDI</td>
</tr>
<tr>
<td>CAMBODIA</td>
<td>KH</td>
<td>KHM</td>
</tr>
<tr>
<td>CAMEROON</td>
<td>CM</td>
<td>CMR</td>
</tr>
<tr>
<td>CANADA</td>
<td>CA</td>
<td>CAN</td>
</tr>
<tr>
<td>CAPE VERDE</td>
<td>CV</td>
<td>CPV</td>
</tr>
<tr>
<td>CAYMAN ISLANDS</td>
<td>KY</td>
<td>CYM</td>
</tr>
<tr>
<td>CENTRAL AFRICAN REPUBLIC</td>
<td>CF</td>
<td>CAF</td>
</tr>
<tr>
<td>CHAD</td>
<td>TD</td>
<td>TCD</td>
</tr>
<tr>
<td>CHILE</td>
<td>CL</td>
<td>CHL</td>
</tr>
<tr>
<td>CHINA</td>
<td>CN</td>
<td>CHN</td>
</tr>
<tr>
<td>CHRISTMAS ISLAND</td>
<td>CX</td>
<td>CXR</td>
</tr>
<tr>
<td>Cocos (Keeling) Islands</td>
<td>CC</td>
<td>CCK</td>
</tr>
<tr>
<td>COLOMBIA</td>
<td>CO</td>
<td>COL</td>
</tr>
<tr>
<td>COMOROS</td>
<td>KM</td>
<td>COM</td>
</tr>
<tr>
<td>CONGO</td>
<td>CG</td>
<td>COG</td>
</tr>
<tr>
<td>CONGO, THE DEMOCRATIC REPUBLIC OF THE</td>
<td>CD</td>
<td>COD</td>
</tr>
<tr>
<td>Territory or region</td>
<td>Two-character</td>
<td>Three-character</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>COOK ISLANDS</td>
<td>CK</td>
<td>COK</td>
</tr>
<tr>
<td>COSTA RICA</td>
<td>CR</td>
<td>CRI</td>
</tr>
<tr>
<td>COTE D'IVOIRE</td>
<td>CI</td>
<td>CIV</td>
</tr>
<tr>
<td>CROATIA (local name: Hrvatska)</td>
<td>HR</td>
<td>HRV</td>
</tr>
<tr>
<td>CUBA</td>
<td>CU</td>
<td>CUB</td>
</tr>
<tr>
<td>CYPRUS</td>
<td>CY</td>
<td>CYP</td>
</tr>
<tr>
<td>CZECH REPUBLIC</td>
<td>CZ</td>
<td>CZE</td>
</tr>
<tr>
<td>DENMARK</td>
<td>DK</td>
<td>DNK</td>
</tr>
<tr>
<td>DJIBOUTI</td>
<td>DJ</td>
<td>DJI</td>
</tr>
<tr>
<td>DOMINICA</td>
<td>DM</td>
<td>DMA</td>
</tr>
<tr>
<td>DOMINICAN REPUBLIC</td>
<td>DO</td>
<td>DOM</td>
</tr>
<tr>
<td>EAST TIMOR</td>
<td>TP</td>
<td>TMP</td>
</tr>
<tr>
<td>ECUADOR</td>
<td>EC</td>
<td>ECU</td>
</tr>
<tr>
<td>EGYPT</td>
<td>EG</td>
<td>EGY</td>
</tr>
<tr>
<td>EL SALVADOR</td>
<td>SV</td>
<td>SLV</td>
</tr>
<tr>
<td>EQUATORIAL GUINEA</td>
<td>GQ</td>
<td>GNQ</td>
</tr>
<tr>
<td>ERITREA</td>
<td>ER</td>
<td>ERI</td>
</tr>
<tr>
<td>ESTONIA</td>
<td>EE</td>
<td>EST</td>
</tr>
<tr>
<td>ETHIOPIA</td>
<td>ET</td>
<td>ETH</td>
</tr>
<tr>
<td>FALKLAND ISLANDS (MALVINAS)</td>
<td>FK</td>
<td>FLK</td>
</tr>
<tr>
<td>FAROE ISLANDS</td>
<td>FO</td>
<td>FRO</td>
</tr>
<tr>
<td>FIJI</td>
<td>FJ</td>
<td>FJI</td>
</tr>
<tr>
<td>FINLAND</td>
<td>FI</td>
<td>FIN</td>
</tr>
<tr>
<td>FRANCE</td>
<td>FR</td>
<td>FRA</td>
</tr>
<tr>
<td>FRANCE, METROPOLITAN</td>
<td>FX</td>
<td>FXX</td>
</tr>
<tr>
<td>FRENCH GUIANA</td>
<td>GF</td>
<td>GUF</td>
</tr>
<tr>
<td>FRENCH POLYNESIA</td>
<td>PF</td>
<td>PYF</td>
</tr>
<tr>
<td>FRENCH SOUTHERN TERRITORIES</td>
<td>TF</td>
<td>ATF</td>
</tr>
<tr>
<td>GABON</td>
<td>GA</td>
<td>GAB</td>
</tr>
<tr>
<td>GAMBIA</td>
<td>GM</td>
<td>GMB</td>
</tr>
<tr>
<td>GEORGIA</td>
<td>GE</td>
<td>GEO</td>
</tr>
<tr>
<td>GERMANY</td>
<td>DE</td>
<td>DEU</td>
</tr>
<tr>
<td>GHANA</td>
<td>GH</td>
<td>GHA</td>
</tr>
<tr>
<td>GIBRALTAR</td>
<td>GI</td>
<td>GIB</td>
</tr>
<tr>
<td>GREECE</td>
<td>GR</td>
<td>GRC</td>
</tr>
<tr>
<td>GREENLAND</td>
<td>GL</td>
<td>GRL</td>
</tr>
<tr>
<td>GRENADE</td>
<td>GD</td>
<td>GRD</td>
</tr>
<tr>
<td>GUADELOUPE</td>
<td>GP</td>
<td>GLP</td>
</tr>
<tr>
<td>GUAM</td>
<td>GU</td>
<td>GUM</td>
</tr>
<tr>
<td>Territory or region</td>
<td>Two-character</td>
<td>Three-character</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>GUATEMALA</td>
<td>GT</td>
<td>GTM</td>
</tr>
<tr>
<td>GUINEA</td>
<td>GN</td>
<td>GIN</td>
</tr>
<tr>
<td>GUINEA-BISSAU</td>
<td>GW</td>
<td>GNB</td>
</tr>
<tr>
<td>GUYANA</td>
<td>GY</td>
<td>GUY</td>
</tr>
<tr>
<td>HAITI</td>
<td>HT</td>
<td>HTI</td>
</tr>
<tr>
<td>HEARD AND MC DONALD ISLANDS</td>
<td>HM</td>
<td>HMD</td>
</tr>
<tr>
<td>HOLY SEE (VATICAN CITY STATE)</td>
<td>VA</td>
<td>VAT</td>
</tr>
<tr>
<td>HONDURAS</td>
<td>HN</td>
<td>HND</td>
</tr>
<tr>
<td>HONG KONG S.A.R. OF CHINA</td>
<td>HK</td>
<td>HKG</td>
</tr>
<tr>
<td>HUNGARY</td>
<td>HU</td>
<td>HUN</td>
</tr>
<tr>
<td>ICELAND</td>
<td>IS</td>
<td>ISL</td>
</tr>
<tr>
<td>INDIA</td>
<td>IN</td>
<td>IND</td>
</tr>
<tr>
<td>INDONESIA</td>
<td>ID</td>
<td>IDN</td>
</tr>
<tr>
<td>IRAN (ISLAMIC REPUBLIC OF)</td>
<td>IR</td>
<td>IRN</td>
</tr>
<tr>
<td>IRAQ</td>
<td>IQ</td>
<td>IRQ</td>
</tr>
<tr>
<td>IRELAND</td>
<td>IE</td>
<td>IRL</td>
</tr>
<tr>
<td>ISRAEL</td>
<td>IL</td>
<td>ISR</td>
</tr>
<tr>
<td>ITALY</td>
<td>IT</td>
<td>ITA</td>
</tr>
<tr>
<td>JAMAICA</td>
<td>JM</td>
<td>JAM</td>
</tr>
<tr>
<td>JAPAN</td>
<td>JP</td>
<td>JPN</td>
</tr>
<tr>
<td>JORDAN</td>
<td>JO</td>
<td>JOR</td>
</tr>
<tr>
<td>KAZAKHSTAN</td>
<td>KZ</td>
<td>KAZ</td>
</tr>
<tr>
<td>KENYA</td>
<td>KE</td>
<td>KEN</td>
</tr>
<tr>
<td>KIRIBATI</td>
<td>KI</td>
<td>KIR</td>
</tr>
<tr>
<td>KOREA, DEMOCRATIC PEOPLE’S REPUBLIC OF</td>
<td>KP</td>
<td>PRK</td>
</tr>
<tr>
<td>KOREA, REPUBLIC OF</td>
<td>KR</td>
<td>KOR</td>
</tr>
<tr>
<td>KUWAIT</td>
<td>KW</td>
<td>KWT</td>
</tr>
<tr>
<td>KYRGYZSTAN</td>
<td>KG</td>
<td>KGZ</td>
</tr>
<tr>
<td>LAO PEOPLE’S DEMOCRATIC REPUBLIC</td>
<td>LA</td>
<td>LAO</td>
</tr>
<tr>
<td>LATVIA</td>
<td>LV</td>
<td>LVA</td>
</tr>
<tr>
<td>LEBANON</td>
<td>LB</td>
<td>LBN</td>
</tr>
<tr>
<td>LESOTHO</td>
<td>LS</td>
<td>LSO</td>
</tr>
<tr>
<td>LIBERIA</td>
<td>LR</td>
<td>LBR</td>
</tr>
<tr>
<td>LIBYAN ARAB JAMAHIRIYA</td>
<td>LY</td>
<td>LBY</td>
</tr>
<tr>
<td>LIECHTENSTEIN</td>
<td>LI</td>
<td>LIE</td>
</tr>
<tr>
<td>LITHUANIA</td>
<td>LT</td>
<td>LTU</td>
</tr>
<tr>
<td>Territory or region</td>
<td>Two-character</td>
<td>Three-character</td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>LUXEMBOURG</td>
<td>LU</td>
<td>LUX</td>
</tr>
<tr>
<td>MACAU S.A.R. OF CHINA</td>
<td>MO</td>
<td>MAC</td>
</tr>
<tr>
<td>MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF</td>
<td>MK</td>
<td>MKD</td>
</tr>
<tr>
<td>MADAGASCAR</td>
<td>MG</td>
<td>MDG</td>
</tr>
<tr>
<td>MALAWI</td>
<td>MW</td>
<td>MWI</td>
</tr>
<tr>
<td>MALAYSIA</td>
<td>MY</td>
<td>MYS</td>
</tr>
<tr>
<td>MALDIVES</td>
<td>MV</td>
<td>MDV</td>
</tr>
<tr>
<td>MALI</td>
<td>ML</td>
<td>MLI</td>
</tr>
<tr>
<td>MALTA</td>
<td>MT</td>
<td>MLT</td>
</tr>
<tr>
<td>MARSHALL ISLANDS</td>
<td>MH</td>
<td>MHL</td>
</tr>
<tr>
<td>MARTINIQUE</td>
<td>MQ</td>
<td>MTQ</td>
</tr>
<tr>
<td>MAURITANIA</td>
<td>MR</td>
<td>MRT</td>
</tr>
<tr>
<td>MAURITIUS</td>
<td>MU</td>
<td>MUS</td>
</tr>
<tr>
<td>MAYOTTE</td>
<td>YT</td>
<td>MYT</td>
</tr>
<tr>
<td>MEXICO</td>
<td>MX</td>
<td>MEX</td>
</tr>
<tr>
<td>MICRONESIA, FEDERATED STATES OF</td>
<td>FM</td>
<td>FSM</td>
</tr>
<tr>
<td>MOLDOVA, REPUBLIC OF</td>
<td>MD</td>
<td>MDA</td>
</tr>
<tr>
<td>MONACO</td>
<td>MC</td>
<td>MCO</td>
</tr>
<tr>
<td>MONGOLIA</td>
<td>MN</td>
<td>MNG</td>
</tr>
<tr>
<td>MONTSERRAT</td>
<td>MS</td>
<td>MSR</td>
</tr>
<tr>
<td>MOROCCO</td>
<td>MA</td>
<td>MAR</td>
</tr>
<tr>
<td>MOZAMBIQUE</td>
<td>MZ</td>
<td>MOZ</td>
</tr>
<tr>
<td>MYANMAR</td>
<td>MM</td>
<td>MMR</td>
</tr>
<tr>
<td>NAMIBIA</td>
<td>NA</td>
<td>NAM</td>
</tr>
<tr>
<td>NAURU</td>
<td>NR</td>
<td>NRU</td>
</tr>
<tr>
<td>NEPAL</td>
<td>NP</td>
<td>NPL</td>
</tr>
<tr>
<td>NETHERLANDS</td>
<td>NL</td>
<td>NLD</td>
</tr>
<tr>
<td>NETHERLANDS ANTILLES</td>
<td>AN</td>
<td>ANT</td>
</tr>
<tr>
<td>NEW CALEDONIA</td>
<td>NC</td>
<td>NCL</td>
</tr>
<tr>
<td>NEW ZEALAND</td>
<td>NZ</td>
<td>NZL</td>
</tr>
<tr>
<td>NICARAGUA</td>
<td>NI</td>
<td>NIC</td>
</tr>
<tr>
<td>NIGER</td>
<td>NE</td>
<td>NER</td>
</tr>
<tr>
<td>NIGERIA</td>
<td>NG</td>
<td>NGA</td>
</tr>
<tr>
<td>NIUE</td>
<td>NU</td>
<td>NIU</td>
</tr>
<tr>
<td>NORFOLK ISLAND</td>
<td>NF</td>
<td>NFK</td>
</tr>
<tr>
<td>NORTHERN MARIANA ISLANDS</td>
<td>MP</td>
<td>MNP</td>
</tr>
<tr>
<td>NORWAY</td>
<td>NO</td>
<td>NOR</td>
</tr>
<tr>
<td>OMAN</td>
<td>OM</td>
<td>OMN</td>
</tr>
<tr>
<td>Territory or region</td>
<td>Two-character</td>
<td>Three-character</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>PAKISTAN</td>
<td>PK</td>
<td>PAK</td>
</tr>
<tr>
<td>PALAU</td>
<td>PW</td>
<td>PLW</td>
</tr>
<tr>
<td>PANAMA</td>
<td>PA</td>
<td>PAN</td>
</tr>
<tr>
<td>PAPUA NEW GUINEA</td>
<td>PG</td>
<td>PNG</td>
</tr>
<tr>
<td>PARAGUAY</td>
<td>PY</td>
<td>PRY</td>
</tr>
<tr>
<td>PERU</td>
<td>PE</td>
<td>PER</td>
</tr>
<tr>
<td>PHILIPPINES</td>
<td>PH</td>
<td>PHL</td>
</tr>
<tr>
<td>PITCAIRN</td>
<td>PN</td>
<td>PCN</td>
</tr>
<tr>
<td>POLAND</td>
<td>PL</td>
<td>POL</td>
</tr>
<tr>
<td>PORTUGAL</td>
<td>PT</td>
<td>PRT</td>
</tr>
<tr>
<td>PUERTO RICO</td>
<td>PR</td>
<td>PRI</td>
</tr>
<tr>
<td>QATAR</td>
<td>QA</td>
<td>QAT</td>
</tr>
<tr>
<td>REUNION</td>
<td>RE</td>
<td>REU</td>
</tr>
<tr>
<td>ROMANIA</td>
<td>RO</td>
<td>ROM</td>
</tr>
<tr>
<td>RUSSIAN FEDERATION</td>
<td>RU</td>
<td>RUS</td>
</tr>
<tr>
<td>RWANDA</td>
<td>RW</td>
<td>RWA</td>
</tr>
<tr>
<td>SAINT KITTS AND NEVIS</td>
<td>KN</td>
<td>KNA</td>
</tr>
<tr>
<td>SAINT LUCIA</td>
<td>LC</td>
<td>LCA</td>
</tr>
<tr>
<td>SAINT VINCENT AND THE GRENADES</td>
<td>VC</td>
<td>VCT</td>
</tr>
<tr>
<td>SAMOA</td>
<td>WS</td>
<td>WSM</td>
</tr>
<tr>
<td>SAN MARINO</td>
<td>SM</td>
<td>SMR</td>
</tr>
<tr>
<td>SAO TOME AND PRINCIPE</td>
<td>ST</td>
<td>STP</td>
</tr>
<tr>
<td>SAUDI ARABIA</td>
<td>SA</td>
<td>SAU</td>
</tr>
<tr>
<td>SENEGAL</td>
<td>SN</td>
<td>SEN</td>
</tr>
<tr>
<td>SEYCHELLES</td>
<td>SC</td>
<td>SYC</td>
</tr>
<tr>
<td>SIERRA LEONE</td>
<td>SL</td>
<td>SLE</td>
</tr>
<tr>
<td>SINGAPORE</td>
<td>SG</td>
<td>SGP</td>
</tr>
<tr>
<td>SLOVAKIA (Slovak Republic)</td>
<td>SK</td>
<td>SVK</td>
</tr>
<tr>
<td>SLOVENIA</td>
<td>SI</td>
<td>SVN</td>
</tr>
<tr>
<td>SOLOMON ISLANDS</td>
<td>SB</td>
<td>SLB</td>
</tr>
<tr>
<td>SOMALIA</td>
<td>SO</td>
<td>SOM</td>
</tr>
<tr>
<td>SOUTH AFRICA</td>
<td>ZA</td>
<td>ZAF</td>
</tr>
<tr>
<td>SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS</td>
<td>GS</td>
<td>SGS</td>
</tr>
<tr>
<td>SPAIN</td>
<td>ES</td>
<td>ESP</td>
</tr>
<tr>
<td>SRI LANKA</td>
<td>LK</td>
<td>LKA</td>
</tr>
<tr>
<td>ST. HELENA</td>
<td>SH</td>
<td>SHN</td>
</tr>
<tr>
<td>ST. PIERRE AND MIQUELON</td>
<td>PM</td>
<td>SPM</td>
</tr>
<tr>
<td>SUDAN</td>
<td>SD</td>
<td>SDN</td>
</tr>
<tr>
<td>Territory or region</td>
<td>Two-character</td>
<td>Three-character</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>SURINAME</td>
<td>SR</td>
<td>SUR</td>
</tr>
<tr>
<td>SVALBARD AND JAN MAYEN ISLANDS</td>
<td>SJ</td>
<td>SJM</td>
</tr>
<tr>
<td>SWAZILAND</td>
<td>SZ</td>
<td>SWZ</td>
</tr>
<tr>
<td>SWEDEN</td>
<td>SE</td>
<td>SWE</td>
</tr>
<tr>
<td>SWITZERLAND</td>
<td>CH</td>
<td>CHE</td>
</tr>
<tr>
<td>SYRIAN ARAB REPUBLIC</td>
<td>SY</td>
<td>SYR</td>
</tr>
<tr>
<td>TAIWAN</td>
<td>TW</td>
<td>TWN</td>
</tr>
<tr>
<td>TAJIKISTAN</td>
<td>TJ</td>
<td>TJK</td>
</tr>
<tr>
<td>TANZANIA, UNITED REPUBLIC OF</td>
<td>TZ</td>
<td>TZA</td>
</tr>
<tr>
<td>THAILAND</td>
<td>TH</td>
<td>THA</td>
</tr>
<tr>
<td>TOGO</td>
<td>TG</td>
<td>TGO</td>
</tr>
<tr>
<td>TOKELAU</td>
<td>TK</td>
<td>TKL</td>
</tr>
<tr>
<td>TONGA</td>
<td>TO</td>
<td>TON</td>
</tr>
<tr>
<td>TRINIDAD AND TOBAGO</td>
<td>TT</td>
<td>TTO</td>
</tr>
<tr>
<td>TUNISIA</td>
<td>TN</td>
<td>TUN</td>
</tr>
<tr>
<td>TURKEY</td>
<td>TR</td>
<td>TUR</td>
</tr>
<tr>
<td>TURKMENISTAN</td>
<td>TM</td>
<td>TKM</td>
</tr>
<tr>
<td>TURKS AND CAICOS ISLANDS</td>
<td>TC</td>
<td>TCA</td>
</tr>
<tr>
<td>TUVALU</td>
<td>TV</td>
<td>TUV</td>
</tr>
<tr>
<td>UGANDA</td>
<td>UG</td>
<td>UGA</td>
</tr>
<tr>
<td>UKRAINE</td>
<td>UA</td>
<td>UKR</td>
</tr>
<tr>
<td>UNITED ARAB EMIRATES</td>
<td>AE</td>
<td>ARE</td>
</tr>
<tr>
<td>UNITED KINGDOM</td>
<td>GB</td>
<td>GBR</td>
</tr>
<tr>
<td>UNITED STATES</td>
<td>US</td>
<td>USA</td>
</tr>
<tr>
<td>UNITED STATES MINOR OUTLYING ISLANDS</td>
<td>UM</td>
<td>UMI</td>
</tr>
<tr>
<td>URUGUAY</td>
<td>UY</td>
<td>URY</td>
</tr>
<tr>
<td>UZBEKISTAN</td>
<td>UZ</td>
<td>UZB</td>
</tr>
<tr>
<td>VANUATU</td>
<td>VU</td>
<td>VUT</td>
</tr>
<tr>
<td>VENEZUELA</td>
<td>VE</td>
<td>VEN</td>
</tr>
<tr>
<td>VIET NAM</td>
<td>VN</td>
<td>VNM</td>
</tr>
<tr>
<td>VIRGIN ISLANDS (BRITISH)</td>
<td>VG</td>
<td>VGB</td>
</tr>
<tr>
<td>VIRGIN ISLANDS (U.S.)</td>
<td>VI</td>
<td>VIR</td>
</tr>
<tr>
<td>WALLIS AND FUTUNA ISLANDS</td>
<td>WF</td>
<td>WLF</td>
</tr>
<tr>
<td>WESTERN SAHARA</td>
<td>EH</td>
<td>ESH</td>
</tr>
<tr>
<td>YEMEN</td>
<td>YE</td>
<td>YEM</td>
</tr>
<tr>
<td>YUGOSLAVIA</td>
<td>YU</td>
<td>YUG</td>
</tr>
</tbody>
</table>
Chapter 7. Reference: Stage Editor user interface

The Parallel job stage editors all use a generic user interface.

The exception to that are the Transformer, Lookup, Shared Container, and Complex Flat File stages.

The following table lists the available stage types and gives a quick guide to their function:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set</td>
<td>File</td>
<td>Allows you to read data from or write data to a persistent data set.</td>
</tr>
<tr>
<td>Sequential File</td>
<td>File</td>
<td>Allows you to read data from or write data to one or more flat files.</td>
</tr>
<tr>
<td>File Set</td>
<td>File</td>
<td>Allows you to read data from or write data to a file set. File sets enable you to spread data across a set of files referenced by a single control file.</td>
</tr>
<tr>
<td>Lookup File Set</td>
<td>File</td>
<td>Allows you to create a lookup file set or reference one for a lookup.</td>
</tr>
<tr>
<td>External Source</td>
<td>File</td>
<td>Allows you to read data that is output from one or more source programs.</td>
</tr>
<tr>
<td>External Target</td>
<td>File</td>
<td>Allows you to write data to one or more source programs.</td>
</tr>
<tr>
<td>Complex Flat File</td>
<td>File</td>
<td>Allows you to read or write complex flat files on a mainframe machine.</td>
</tr>
<tr>
<td>SAS Data Set</td>
<td>File</td>
<td>Allows you to read data from or write data to a parallel SAS data set in conjunction with an SAS stage.</td>
</tr>
<tr>
<td>DB2 Enterprise</td>
<td>Database</td>
<td>Allows you to read data from and write data to a DB2 database.</td>
</tr>
<tr>
<td>Oracle Enterprise</td>
<td>Database</td>
<td>Allows you to read data from and write data to an Oracle database.</td>
</tr>
<tr>
<td>Teradata Enterprise</td>
<td>Database</td>
<td>Allows you to read data from and write data to a Teradata database.</td>
</tr>
<tr>
<td>Stage</td>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Informix Enterprise</td>
<td>Database</td>
<td>Allows you to read data from and write data to an Informix database.</td>
</tr>
<tr>
<td>Transformer</td>
<td>Processing</td>
<td>Handles extracted data, performs any conversions required, and passes data to another active stage or a stage that writes data to a target database or file.</td>
</tr>
<tr>
<td>BASIC Transformer</td>
<td>Processing</td>
<td>Same as Transformer stage, but gives access to DataStage BASIC functions.</td>
</tr>
<tr>
<td>Aggregator</td>
<td>Processing</td>
<td>Classifies incoming data into groups, computes totals and other summary functions for each group, and passes them to another stage in the job.</td>
</tr>
<tr>
<td>Join</td>
<td>Processing</td>
<td>Performs join operations on two or more data sets input to the stage and then outputs the resulting data set.</td>
</tr>
<tr>
<td>Merge</td>
<td>Processing</td>
<td>Combines a sorted master data set with one or more sorted update data sets.</td>
</tr>
<tr>
<td>Lookup</td>
<td>Processing</td>
<td>Used to perform lookup operations on a data set read into memory from any other Parallel job stage that can output data or provided by one of the database stages that support reference output links. It can also perform a lookup on a lookup table contained in a Lookup File Set stage.</td>
</tr>
<tr>
<td>Sort</td>
<td>Processing</td>
<td>Sorts input columns.</td>
</tr>
<tr>
<td>Funnel</td>
<td>Processing</td>
<td>Copies multiple input data sets to a single output data set.</td>
</tr>
<tr>
<td>Remove Duplicates</td>
<td>Processing</td>
<td>Takes a single sorted data set as input, removes all duplicate records, and writes the results to an output data set.</td>
</tr>
<tr>
<td>Compress</td>
<td>Processing</td>
<td>Uses the UNIX compress or GZIP utility to compress a data set. It converts a data set from a sequence of records into a stream of raw binary data.</td>
</tr>
<tr>
<td>Stage</td>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Expand</td>
<td>Processing</td>
<td>Uses the UNIX <code>uncompress</code> or <code>GZIP</code> utility to expand a data set. It converts a previously compressed data set back into a sequence of records from a stream of raw binary data.</td>
</tr>
<tr>
<td>Copy</td>
<td>Processing</td>
<td>Copies a single input data set to a number of output data sets.</td>
</tr>
<tr>
<td>Modify</td>
<td>Processing</td>
<td>Alters the record schema of its input data set.</td>
</tr>
<tr>
<td>Filter</td>
<td>Processing</td>
<td>Transfers, unmodified, the records of the input data set which satisfy requirements that you specify and filters out all other records.</td>
</tr>
<tr>
<td>External Filter</td>
<td>Processing</td>
<td>Allows you to specify a UNIX command that acts as a filter on the data you are processing.</td>
</tr>
<tr>
<td>Change Capture</td>
<td>Processing</td>
<td>Takes two input data sets, denoted before and after, and outputs a single data set whose records represent the changes made to the before data set to obtain the after data set.</td>
</tr>
<tr>
<td>Change Apply</td>
<td>Processing</td>
<td>Takes the change data set, that contains the changes in the before and after data sets, from the Change Capture stage and applies the encoded change operations to a before data set to compute an after data set.</td>
</tr>
<tr>
<td>Difference</td>
<td>Processing</td>
<td>Performs a record-by-record comparison of two input data sets, which are different versions of the same data set.</td>
</tr>
<tr>
<td>Compare</td>
<td>Processing</td>
<td>Performs a column-by-column comparison of records in two pre-sorted input data sets.</td>
</tr>
<tr>
<td>Encode</td>
<td>Processing</td>
<td>Encodes a data set using a UNIX encoding command that you supply.</td>
</tr>
<tr>
<td>Decode</td>
<td>Processing</td>
<td>Decodes a data set using a UNIX decoding command that you supply.</td>
</tr>
<tr>
<td>Stage</td>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Switch</td>
<td>Processing</td>
<td>Takes a single data set as input and assigns each input record to an output data set based on the value of a selector field.</td>
</tr>
<tr>
<td>SAS</td>
<td>Processing</td>
<td>Allows you to execute part or all of an SAS application in parallel.</td>
</tr>
<tr>
<td>Generic</td>
<td>Processing</td>
<td>Lets you incorporate an Orchestrate Operator in your job.</td>
</tr>
<tr>
<td>Surrogate Key</td>
<td>Processing</td>
<td>Generates one or more surrogate key columns and adds them to an existing data set.</td>
</tr>
<tr>
<td>Column Import</td>
<td>Restructure</td>
<td>Imports data from a single column and outputs it to one or more columns.</td>
</tr>
<tr>
<td>Column Export</td>
<td>Restructure</td>
<td>Exports data from a number of columns of different data types into a single column of data type string or binary.</td>
</tr>
<tr>
<td>Make Subrecord</td>
<td>Restructure</td>
<td>Combines specified vectors in an input data set into a vector of sub-records whose columns have the names and data types of the original vectors.</td>
</tr>
<tr>
<td>Split Subrecord</td>
<td>Restructure</td>
<td>Creates one new vector column for each element of the original sub-record.</td>
</tr>
<tr>
<td>Combine Records</td>
<td>Restructure</td>
<td>Combines records, in which particular key-column values are identical, into vectors of sub-records.</td>
</tr>
<tr>
<td>Promote Subrecord</td>
<td>Restructure</td>
<td>Promotes the columns of an input sub-record to top-level columns.</td>
</tr>
<tr>
<td>Make Vector</td>
<td>Restructure</td>
<td>Combines specified columns of an input data record into a vector of columns of the same type.</td>
</tr>
<tr>
<td>Split Vector</td>
<td>Restructure</td>
<td>Promotes the elements of a fixed-length vector to a set of similarly named top-level columns.</td>
</tr>
<tr>
<td>Head</td>
<td>Development/ Debug</td>
<td>Selects the first N records from each partition of an input data set and copies the selected records to an output data set.</td>
</tr>
<tr>
<td>Stage</td>
<td>Type</td>
<td>Function</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Tail</td>
<td>Development/ Debug</td>
<td>Selects the last N records from each partition of an input data set and copies the selected records to an output data set.</td>
</tr>
<tr>
<td>Sample</td>
<td>Development/ Debug</td>
<td>Samples an input data set.</td>
</tr>
<tr>
<td>Peek</td>
<td>Development/ Debug</td>
<td>Lets you print record column values either to the job log or to a separate output link as the stage copies records from its input data set to one or more output data sets.</td>
</tr>
<tr>
<td>Row Generator</td>
<td>Development/ Debug</td>
<td>Produces a set of mock data fitting the specified metadata.</td>
</tr>
<tr>
<td>Column Generator</td>
<td>Development/ Debug</td>
<td>Adds columns to incoming data and generates mock data for these columns for each data row processed.</td>
</tr>
<tr>
<td>Write Range Map</td>
<td>Development/ Debug</td>
<td>Allows you to write data to a range map. The stage can have a single input link.</td>
</tr>
</tbody>
</table>

All of the stage types use the same basic stage editor, but the pages that appear when you edit the stage depend on the exact type of stage you are editing. The following sections describe all the page types and subtypes that are available. The individual descriptions of stage editors in the following chapters tell you exactly which features of the generic editor each stage type uses.

### The stage page

The stage page varies according to the stage that it represents.

All stage editors have a stage page. This contains a number of subsidiary tabs depending on the stage type. The only field the stage page itself contains provides the name of the stage being edited.

When naming your stages, use meaningful names, rather than rely on the default names that InfoSphere DataStage allocates. If you rely on default names you may have several different jobs containing stages with the same name.

### General tab

The General tab of Stage Properties provides a text field that lets you type a description of the stage.

From the General tab, you can enter an optional description of the stage. This is valuable information for others who may need to understand your job or its metadata.

### Properties tab

The Properties tab provides settings for general properties.
A Properties tab appears on the Stage page where there are general properties that need setting for the stage you are editing. Properties tabs can also occur under Input and Output pages where there are link-specific properties that need to be set.

In general, processing stages have Properties tabs on the Stage page, while other stages have them on the Input or Output page.

**Working with the properties tree**

The available properties are displayed in a tree structure.

**About this task**

They are divided into categories to help you find your way around them. All the mandatory properties are included in the tree by default and cannot be removed.

**Procedure**

1. Select Tools > Options from the main menu and choose the Transformer item from the tree to open the Stage page.
2. To reset the Invalid column color, click on the color bar and select a new color from the palette.

**Set a property:**

**About this task**

Properties for which you must set a value are shown in red (by default). Properties change to black when you set a value.

To set a property value:

**Procedure**

1. Select File in the list and specify the required property value in the File field. The title of this field and the method for entering a value changes according to the property you have selected.
2. Browse for a property value or insert a job parameter whose value is provided at run time.
3. Click on the arrow and a menu gives access to the Browse Files window, a list of available job parameters, or both. Job parameters are defined in the Job Properties window. See *IBM InfoSphere DataStage and QualityStage Designer Client Guide*.

**Reset the default values in the Properties page:**

You can reset the default values.

**About this task**

The Information field contains details about the property you currently have selected in the tree. Some properties have default values.

**Procedure**

1. Select the property in the tree.
2. Choose Set to default from the shortcut menu.
Results

Some properties are optional. These appear in the Available properties to add field.

Click on an optional property to add it to the tree or choose to add it from the shortcut menu.

Remove the property:

About this task

To select the property in the tree, choose Remove from the shortcut menu.

Add properties:

About this task

Some properties are repeated. The Key property appears in the Available properties to add list when you select the top level Properties node.

Procedure

1. Click on the Key item to add multiple key properties. Where a repeatable property expects a column as an argument, a window is available that lets you specify multiple columns at once.
2. To open the Column Selection window, click the Column button next to the properties tree.
   The Column Selection window opens. The left pane lists all the available columns.
3. Use the right-arrow keys to select some or all of the columns.
4. Use the left-arrow keys to move them back if you change your mind. A separate property appears for each column you have selected.

Results

Some properties have dependents. These are properties which somehow relate to or modify the parent property. They appear under the parent in a tree structure.

For some properties you can supply a job parameter as their value. At runtime the value of this parameter is used for the property. These properties have an arrow next to their Property Value box.

Supply a job parameter:

Procedure

1. Click the drop-down menu.
2. Choose Insert job parameter to get a list of currently defined job parameters.

Use a multiline editor to enter some property values:

Procedure

1. Click on the arrow next to their Property Value box.
2. Choose Switch to multiline editor from the menu.
   The property capabilities are indicated by different icons in the tree.

Configuring the properties tab

Use the Properties tab to specify properties that determine what the stage actually does.
About this task

Two properties are available for the InfoSphere QualityStage stages:

- Alternate locale. Optional. Lets you specify the international locale you want InfoSphere QualityStage to use at the server to process the data.

  This value needs to be set only if you are processing data for a language that is not the default language of the server. For example, the default language for your server is French and the data to be processed is Italian.

  When you change the locale, InfoSphere QualityStage uses the appropriate collating sequence and decimal separators for the alternate language. The value required depends on the type of server and how it is configured.

  If you are using a UNIX server, enter the following command to obtain a list of locales supported by your server:

  `locale -a`

  If you are using a Windows workstation, select your InfoSphere QualityStage server directory and the locale subdirectory. The local subdirectory contains folders that are listed alphabetically by the languages they support.

- Trace Type. This is a debugging property to be used only in conjunction with IBM Software Support.

Procedure

1. In the Available properties to add box, click Alternate locale.

   The property appears under the Options folder and the cursor appears in the Alternate locale field.

2. Enter the locale. If necessary, click the arrow button to access the multiline editor or to insert a job parameter.

3. Click OK to close the stage editor, or click another tab or page to set other stage details.

Advanced tab

The Advanced tab lets you set additional tasks on the Stage Editor.

All stage editors have an Advanced tab. The Advanced tab allows you to do the following tasks:

- Specify the execution mode of the stage. This allows you to choose between Parallel and Sequential operation. If the execution mode for a particular type of stage cannot be changed, then this drop down list is disabled. Selecting Sequential operation forces the stage to be executed on a single node. If you have intermixed sequential and parallel stages this has implications for partitioning and collecting data between the stages. You can also maintain the default setting for the stage (the drop down list tells you whether this is parallel or sequential).

- Set or clear the preserve partitioning flag (this field is not available for all stage types). It indicates whether partitioning is preserved at the next stage of the job. You choose between Set, Clear and Propagate. For some stage types, Propagate is not available. The operation of each option is explained in the following list:
  - Set. Sets the preserve partitioning flag. This flag indicates to the next stage in the job to preserve existing partitioning if possible.
  - Clear. Clears the preserve partitioning flag. Clear does not specify which partitioning method the next stage can use.
- Propagate. Changes the flag to Set or Clear depending on what the previous stage in the job has set (or if that is set to Propagate the stage before that and so on until a preserve partitioning flag setting is encountered).

You can keep the default setting for the stage (the drop down list tells you whether this is set, clear, or propagate).

- Specify the combinability mode. Under the covers InfoSphere DataStage can combine the operators that underlie parallel stages so that they run in the same process. This saves a significant amount of data copying and preparation in passing data between operators.

The combinability mode setting tells InfoSphere DataStage your preferences for combining for a particular stage. It has three possible settings:

  - Auto. Use the default combination setting.
  - Combinable. Ignore the operator's default setting and combine if at all possible (some operators are marked as non-combinable by default).
  - Don't Combine. Never combine operators.

  In most cases the setting should be left to Auto.

- Specify node map or node pool or resource pool constraints. The configuration file allows you to set up pools of related nodes or resources. The Advanced tab allows you to limit execution of a stage to a particular node or resource pool.

You can also use the map to specify a group of nodes whose execution is limited just to this stage.

  - Node pool and resource constraints. Specify constraints in the grid. Select Node pool or Resource pool from the Constraint drop-down list. Select a Type for a resource pool and select the name of the pool to which you are limiting execution. You can select multiple node or resource pools. This is only enabled if you have defined multiple pools in the configuration file.

  - Node map constraints. Select the option box and type in the nodes to which execution is limited in the text box. You can also browse through the available nodes to add to the text box. Using this feature conceptually sets up an additional node pool which doesn't appear in the configuration file.

The lists of available nodes, available node pools, and available resource pools are derived from the configuration file.

**Link ordering tab**

Use this tab to order the links for stages that have more than one link and requires you to order links.

The tab allows you to order input links, output links, or both as needed. Where link ordering is not important or is not possible the tab does not appear.

The link label gives further information about the links being ordered. The Merge stage combines data from a sorted master data set with one or more sorted update data sets. The Link Ordering tab determines which items are input links and which items are output links. If you use the arrow keys to reorder the links, the link name changes but not the link label.

The Merge stage handles reject links as well as a stream link and the tab allows you to order these, although you cannot move them to the stream link position. The link labels give the sense of how the links are being used.

The individual stage descriptions tell you whether link ordering is possible and what options are available.
### NLS map tab
If you have NLS enabled on your system, some of your stages will have an NLS Map tab.

This allows you to override the project default character set map for this stage, and in some cases, allows you to enable per-column mapping. When per-column mapping is enabled, you can override the character set map for particular columns (an NLS map field appears on the columns tab allowing you to do this).

Select a map from the list, or click the arrow button next to the list to specify a job parameter.

The following stage types currently support this feature:
- Sequential File
- File Set
- DB2 Enterprise (not per-column mapping)
- Oracle Enterprise (not per-column mapping)

### NLS locale tab
If you have NLS enabled on your system, some of your stages will have an NLS Locale tab.

It lets you view the current default collate convention, and select a different one for the stage if required. You can also use a job parameter to specify the locale, or browse for a file that defines custom collate rules. The collate convention defines the order in which characters are collated, for example, the character Ä follows A in Germany, but follows Z in Sweden.

Select a locale from the list, or click the arrow button next to the list to use a job parameter or browse for a collate file.

The following types of stages have an NLS Locale tab:
- Stages that evaluate expressions, such as the Transformer.
- Stages that need to evaluate the order of key columns.
- The Sort Stage.

### Input page
The Input page gives information about links going into a stage.

In the case of a file or database stage an input link carries data being written to the file or database. In the case of a processing or restructure stage it carries data that the stage will process before outputting to another stage. Where there are no input links, the stage editor has no Input page.

Where it is present, the Input page contains various tabs depending on stage type. The only field the Input page itself contains is Input name, which gives the name of the link being edited. Where a stage has more than one input link, you can select the link you are editing from the Input name drop-down list.

The Input page also has a Columns... button.
Click **Columns** to open a window showing column names from the metadata defined for this link. You can drag these columns to various fields in the Input page tabs as required.

Certain stage types will also have a **View Data...** button.

Click **View Data** to view the actual data associated with the specified data source or data target. The button is available if you have defined metadata for the link. Note the interface allowing you to view the file are slightly different depending on stage and link type.

**Input General tab**

The Input page always has a General tab. Use the General tab to enter an optional description of the link. Specifying a description for each link enhances job maintainability.

**Input Properties tab**

Some types of file and database stages can have properties that are particular to specific input links. In this case the Input page has a Properties tab. This has the same format as the Stage page Properties tab.

**Input Partitioning tab**

Most parallel stages have a default partitioning or collecting method associated with them.

The Partitioning tab can be used depending on the execution mode of the stage (parallel or sequential) and the execution mode of the immediately preceding stage in the job. For example, if the preceding stage is processing data sequentially and the current stage is processing in parallel, the data is partitioned before it enters the current stage. Conversely, if the preceding stage is processing data in parallel and the current stage is sequential, the data is collected as it enters the current stage.

You can override the default partitioning or collecting method on the Partitioning tab. The selected method is applied to the incoming data as it enters the stage on a particular link, and so the Partitioning tab appears on the Input page. You can also use the tab to re-partition data between two parallel stages. If both stages are executing sequentially, you cannot select a partition or collection method and the fields are disabled. The fields are also disabled if the particular stage does not permit selection of partitioning or collection methods. The following table shows what can be set from the Partitioning tab in what circumstances:

<table>
<thead>
<tr>
<th>Preceding Stage</th>
<th>Current Stage</th>
<th>Partition Tab Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel</td>
<td>Parallel</td>
<td>Partition</td>
</tr>
<tr>
<td>Parallel</td>
<td>Sequential</td>
<td>Collect</td>
</tr>
<tr>
<td>Sequential</td>
<td>Parallel</td>
<td>Partition</td>
</tr>
<tr>
<td>Sequential</td>
<td>Sequential</td>
<td>None (disabled)</td>
</tr>
</tbody>
</table>

Use the Partitioning tab to specify whether the data should be sorted as it enters.

The Partitioning tab has the following fields:
Partition type. Choose the partitioning (or collecting) type from the drop-down list. The following partitioning types are available:

- (Auto). InfoSphere DataStage attempts to work out the best partitioning method depending on execution modes of current and preceding stages and how many nodes are specified in the Configuration file. This is the default method for many stages.
- Entire. Every processing node receives the entire data set. No further information is required.
- Hash. The records are hashed into partitions based on the value of a key column or columns selected from the Available list.
- Modulus. The records are partitioned using a modulus function on the key column selected from the Available list. This is commonly used to partition on tag fields.
- Random. The records are partitioned randomly, based on the output of a random number generator. No further information is required.
- Round Robin. The records are partitioned on a round robin basis as they enter the stage. No further information is required.
- Same. Preserves the partitioning already in place. No further information is required.
- DB2. Replicates the DB2 partitioning method of a specific DB2 table. Requires extra properties to be set. Access these properties by clicking the properties button.
- Range. Divides a data set into approximately equal size partitions based on one or more partitioning keys. Range partitioning is often a preprocessing step to performing a total sort on a data set. Requires extra properties to be set. Access these properties by clicking the properties button.

The following collection types are available:

- (Auto). Normally, when you use Auto mode, InfoSphere DataStage eagerly reads any row from any input partition as it becomes available. This fastest collecting method is the default collection method for many stages. In some circumstances InfoSphere DataStage detect further requirements for collected data, for example, it might need to be sorted. Using Auto mode sorts data, if required.
- Ordered. Reads all records from the first partition, then all records from the second partition, and so on. Requires no further information.
- Round Robin. Reads a record from the first input partition, then from the second partition, and so on. After reaching the last partition, the operator starts over.
- Sort Merge. Reads records in an order based on one or more columns of the record. This requires you to select a collecting key column from the Available list.

Available. This lists the input columns for the input link. Key columns are identified by a key icon. For partitioning or collecting methods that require you to select columns, you click on the required column in the list and it appears in the Selected list to the right. This list is also used to select columns on which to sort.

Selected. This list shows which columns have been selected for partitioning, collecting, or sorting and displaying information about them. The available information is whether a sort is being performed (indicated by an arrow), if so the order of the sort (ascending or descending) and collating sequence (sort as EBCDIC), and whether an alphanumeric key is case sensitive or not. Nullable columns are marked to indicate if null columns take first or last position.
can select sort order, case sensitivity, collating sequence, and null positions from the shortcut menu. If applicable, the Usage field indicates whether a particular key column is being used for sorting, partitioning, or both.

- Sorting. The check boxes in the section allow you to specify sort details. The availability of sorting depends on the partitioning method chosen.
  - Perform Sort. Select this to specify that data coming in on the link should be sorted. Select the column or columns to sort on from the Available list.
  - Stable. Select this if you want to preserve previously sorted data sets. The default is stable.
  - Unique. Select this to specify that, if multiple records have identical sorting key values, only one record is retained. If stable sort is also set, the first record is retained.

You can also specify sort direction, case sensitivity, whether sorted as EBCDIC, and whether null columns will appear first or last for each column. Where you are using a keyed partitioning method, you can also specify whether the column is used as a key for sorting, for partitioning, or for both. Select the column in the Selected list and right-click to invoke the shortcut menu. The availability of the sort options depends on the type of data in the column, whether it is nullable or not, and the partitioning method chosen.

If you have NLS enabled, the sorting box has an additional button. Click this to open the NLS Locales tab of the Sort Properties window. This lets you view the current default collate convention, and select a different one for the stage if required. You can also use a job parameter to specify the locale, or browse for a file that defines custom collate rules. The collate convention defines the order in which characters are collated, for example, the character Ä follows A in Germany, but follows Z in Sweden. Select a locale from the list, or click the arrow button next to the list to use a job parameter or browse for a collate file.

If you require a more complex sort operation, you should use the Sort stage.

**DB2 partition properties**

This window appears when you select a Partition type of DB2 and click the properties button. It allows you to specify the DB2 table whose partitioning method is to be replicated.

**Range partition properties**

The Range Partition window appears when you select a Partition type of Range and click the properties button. It allows you to specify the range map that is to be used to determine the partitioning (you create a range map file using the Write Range Map stage). Type in a pathname or browse for a file.

**Input tab Format tab**

Stages that write to certain types of file (for instance the Sequential File stage) also have a Format tab which allows you to specify the format to which a file or files is or are being written.

The Format tab is similar in structure to the Properties tab. A flat file has a number of properties for which you can set different attributes. Select the property in the tree and select the attributes you want to set from the Available properties to add window, it will then appear as a dependent property in the property tree and you can set its value as required. This tab sets the format information for the file at row level. You can override the settings for individual columns using the Edit Column Metadata window.
Click **Load** to load the format information from a table definition in the metadata repository.

The shortcut menu from the property tree gives access to the following functions:

- **Format as.** This applies a predefined template of properties. Choose from the following properties:
  - Delimited/quoted
  - Fixed-width records
  - UNIX line terminator
  - DOS line terminator
  - No terminator (fixed width)
  - Mainframe (COBOL)

- **Add sub-property.** Gives access to a list of dependent properties for the currently selected property (visible only if the property has dependents).

- **Set to default.** Appears if the currently selected property has been set to a non-default value, allowing you to re-select the default.

- **Remove.** Removes the currently selected property. This is disabled if the current property is mandatory.

- **Remove all.** Removes all the non-mandatory properties.

**Record level**

These properties define details about how data records are formatted in the flat file.

The **Input > Record level** location where you can enter a character, this can usually be an ASCII character or a multibyte Unicode character (if you have NLS enabled). The following lists the available properties.

- **Fill char.** Specify an ASCII character or a value in the range 0 to 255. You can also choose Space or Null from a drop-down list. This character is used to fill any gaps in a written record caused by column positioning properties. Set to 0 by default (which is the NULL character). For example, to set it to space you could also type in the space character or enter 32. Note that this value is restricted to one byte, so you cannot specify a multibyte Unicode character.

- **Final delimiter string.** Specify a string to be written after the last column of a record in place of the column delimiter. Enter one or more characters, this precedes the record delimiter if one is used. Mutually exclusive with Final delimiter, which is the default. For example, if you set Delimiter to comma and Final delimiter string to `, ` (comma space - you do not need to enter the inverted commas) all fields are delimited by a comma, except the final field, which is delimited by a comma followed by an ASCII space character.

- **Final delimiter.** Specify a single character to be written after the last column of a record in place of the field delimiter. Type a character or select one of whitespace, end, none, null, tab, or comma. See the following diagram for an illustration.
  - whitespace. The last column of each record will not include any trailing white spaces found at the end of the record.
  - end. The last column of each record does not include the field delimiter. This is the default setting.
  - none. The last column of each record does not have a delimiter; used for fixed-width fields.
  - null. The last column of each record is delimited by the ASCII null character.
- comma. The last column of each record is delimited by the ASCII comma character.
- tab. The last column of each record is delimited by the ASCII tab character.

When writing, a space is now inserted after every field except the last in the record.

- Intact. The intact property specifies an identifier of a partial schema. A partial schema specifies that only the column(s) named in the schema can be modified by the stage. All other columns in the row are passed through unmodified. The file containing the partial schema is specified in the Schema File property on the Properties tab. This property has a dependent property, Check intact, but this is not relevant to input links.

- Record delimiter string. Specify a string to be written at the end of each record. Enter one or more characters. This is mutually exclusive with Record delimiter, which is the default, record type and record prefix.

- Record delimiter. Specify a single character to be written at the end of each record. Type a character or select one of the following:
  - UNIX Newline (the default)
  - null

(To implement a DOS newline, use the Record delimiter string property set to "\r\n" or choose Format as > DOS line terminator from the shortcut menu.)

Note: Record delimiter is mutually exclusive with Record delimiter string, Record prefix, and Record type.

- Record length. Select Fixed where fixed length fields are being written. DataStage calculates the appropriate length for the record. Alternatively specify the length of fixed records as number of bytes. This is not used by default (default files are comma-delimited). The record is padded to the specified length with either zeros or the fill character if one has been specified.

- Record Prefix. Specifies that a variable-length record is prefixed by a 1-, 2-, or 4-byte length prefix. It is set to 1 by default. This is mutually exclusive with Record delimiter, which is the default, and record delimiter string and record type.

- Record type. Specifies that data consists of variable-length blocked records (varying) or implicit records (implicit). If you choose the implicit property, data is written as a stream with no explicit record boundaries. The end of the record is inferred when all of the columns defined by the schema have been parsed. The varying property allows you to specify one of the following IBM blocked or spanned formats: V, VB, VS, VBS, or VR.

This property is mutually exclusive with Record length, Record delimiter, Record delimiter string, and Record prefix and by default is not used.

Field defaults
Defines default properties for columns written to the file or files.

The Field defaults are applied to all columns written, but can be overridden for individual columns from the Columns tab using the Edit Column Metadata window. Where you can enter a character, this can usually be an ASCII character or a multi-byte Unicode character (if you have NLS enabled). The available properties are:
- **Actual field length.** Specifies the number of bytes to fill with the Fill character when a field is identified as null. When DataStage identifies a null field, it will write a field of this length full of Fill characters. This is mutually exclusive with Null field value.

- **Delimiter.** Specifies the trailing delimiter of all fields in the record. Type an ASCII character or select one of whitespace, end, none, null, comma, or tab.
  - **Whitespace.** Characters at the end of a column are ignored, in that they are not treated as part of the column.
  - **End.** The end of a field is taken as the delimiter, in that there is no separate delimiter. This is not the same as a setting of `None' which is used for fields with fixed-width columns.
  - **None.** No delimiter (used for fixed-width).
  - **Null.** ASCII Null character is used.
  - **Comma.** ASCII comma character is used.
  - **Tab.** ASCII tab character is used.

- **Delimiter string.** Specify a string to be written at the end of each field. Enter one or more characters. This is mutually exclusive with Delimiter, which is the default. For example, specifying `", " (comma space - you do not need to enter the inverted commas) specifies each field is delimited by `, ` unless overridden for individual fields.

- **Null field length.** The length in bytes of a variable-length field that contains a null. When a variable-length field is written, InfoSphere DataStage writes a length value of null field length if the field contains a null. This property is mutually exclusive with null field value.

- **Null field value.** Specifies the value written to null field if the source is set to null. Can be a number, string, or C-type literal escape character. For example, you can represent a byte value by `<b>\xooo</b>`, where each `o` is an octal digit 0 - 7 and the first `o` is < 4, or by `<b>\xhh</b>`, where each `h` is a hexadecimal digit 0 - F. You must use this form to encode non-printable byte values.

  This property is mutually exclusive with Null field length and Actual length. For a fixed width data representation, you can use Pad char (from the general section of Type defaults) to specify a repeated trailing character if the value you specify is shorter than the fixed width of the field.

- **Prefix bytes.** Specifies that each column in the data file is prefixed by 1, 2, or 4 bytes containing, as a binary value, either the column's length or the tag value for a tagged field.

  You can use this option with variable-length fields. Variable-length fields can be either delimited by a character or preceded by a 1-, 2-, or 4-byte prefix containing the field length. InfoSphere DataStage inserts the prefix before each field.

  This property is mutually exclusive with the Delimiter, Quote, and Final Delimiter properties, which are used by default.

- **Print field.** This property is not relevant for input links.

- **Quote.** Specifies that variable length fields are enclosed in single quotes, double quotes, or another character or pair of characters. Choose Single or Double, or enter a character. This is set to double quotes by default.

  When writing, InfoSphere DataStage inserts the leading quote character, the data, and a trailing quote character. Quote characters are not counted as part of a field’s length.
• Vector prefix. For fields that are variable length vectors, specifies a 1-, 2-, or 4-byte prefix containing the number of elements in the vector. You can override this default prefix for individual vectors.

Variable-length vectors must use either a prefix on the vector or a link to another field in order to specify the number of elements in the vector. If the variable length vector has a prefix, you use this property to indicate the prefix length. InfoSphere DataStage inserts the element count as a prefix of each variable-length vector field. By default, the prefix length is assumed to be one byte.

**Type defaults**

Data type defaults are properties that apply to all columns of a specific data type unless specifically overridden at the column level.

The Data type defaults are divided into a number of subgroups according to data type.

**General**

The following properties apply to several data types (unless overridden at column level):

• Byte order. Specifies how multiple byte data types (except string and raw data types) are ordered. Choose from the following byte order items:
  – little-endian. The high byte is on the right.
  – big-endian. The high byte is on the left.
  – native-endian. As defined by the native format of the machine. This is the default.

• Data Format. Specifies the data representation format of a field. Applies to fields of all data types except string, ustring, and raw, and to record, subrecord or tagged fields containing at least one field that is neither string nor raw. Choose from the following data formats.

<table>
<thead>
<tr>
<th>Data Format</th>
<th>Description</th>
</tr>
</thead>
</table>
| binary      | A setting of binary has different meanings when applied to different data types:  
  • For decimals, binary means packed.  
  • For other numerical data types, binary means "not text".  
  • For dates, binary is equivalent to specifying the julian property for the date field.  
  • For time, binary is equivalent to midnight_seconds.  
  • For timestamp, binary specifies that the first integer contains a Julian day count for the date portion of the timestamp and the second integer specifies the time portion of the timestamp as the number of seconds from midnight. A binary timestamp specifies that two 32-bit integers are written. |
<table>
<thead>
<tr>
<th>Data Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text (the default)</td>
<td>By default data is formatted as text, as follows:</td>
</tr>
<tr>
<td></td>
<td>• Date data type. Text specifies that the data to be written contains a text-based date in the form %yyyy-%mm-%dd or in the default date format if you have defined a new one on an NLS system.</td>
</tr>
<tr>
<td></td>
<td>• Decimal data type. A field represents a decimal in a string format with a leading space or '-' followed by decimal digits with an embedded decimal point if the scale is not zero.</td>
</tr>
<tr>
<td></td>
<td>The destination string format is: [+-]dd.(ddd] and any precision and scale arguments are ignored.</td>
</tr>
<tr>
<td></td>
<td>• Numeric fields (int8, int16, int32, uint8, uint16, uint32, sfloat, and dfloat). InfoSphere DataStage assumes that numeric fields are represented as text.</td>
</tr>
<tr>
<td></td>
<td>• Time data type. Text specifies that the field represents time in the text-based form %hh:%nn:%ss or in the default date format if you have defined a new one on an NLS system.</td>
</tr>
<tr>
<td></td>
<td>• Timestamp data type. Text specifies a text-based timestamp in the form %yyyy-%mm-%dd %hh:%nn:%ss or in the default date format if you have defined a new one on an NLS system.</td>
</tr>
<tr>
<td></td>
<td>• Field max width. The maximum number of bytes in a column represented as a string. Enter a number. This is useful where you are storing numbers as text. If you are using a fixed-width character set, you can calculate the length exactly. If you are using variable-length character set, calculate an adequate maximum width for your fields. Applies to fields of all data types except date, time, timestamp, and raw; and record, subrecord, or tagged if they contain at least one field of this type.</td>
</tr>
<tr>
<td></td>
<td>• Field width. The number of bytes in a field represented as a string. Enter a number. This is useful where you are storing numbers as text. If you are using a fixed-width character set, you can calculate the number of bytes exactly. If it's a variable length encoding, base your calculation on the width and frequency of your variable-width characters. Applies to fields of all data types except date, time, timestamp, and raw; and record, subrecord, or tagged if they contain at least one field of this type.</td>
</tr>
</tbody>
</table>

If you specify neither field width nor field max width, numeric fields written as text have the following number of bytes as their maximum width:
- 8-bit signed or unsigned integers: 4 bytes
- 16-bit signed or unsigned integers: 6 bytes
- 32-bit signed or unsigned integers: 11 bytes
- 64-bit signed or unsigned integers: 21 bytes
- single-precision float: 14 bytes (sign, digit, decimal point, 7 fraction, "E", sign, 2 exponent)
- double-precision float: 24 bytes (sign, digit, decimal point, 16 fraction, "E", sign, 3 exponent)

- Pad char. Specifies the pad character used when strings or numeric values are written to an external string representation. Enter a character (single byte for strings, can be multi byte for ustrings) or choose null or space. The pad character is used when the external string representation is larger than required to hold the written field. In this case, the external string is filled with the pad character to its full length. Space is the default. Applies to string, ustring, and numeric data types and record, subrecord, or tagged types if they contain at least one field of this type.

- Character set. Specifies the character set. Choose from ASCII or EBCDIC. The default is ASCII. Applies to all data types except raw and ustring and record, subrecord, or tagged containing no fields other than raw or ustring.

**String**

These properties are applied to columns with a string data type, unless overridden at column level.

- Export EBCDIC as ASCII. Select this to specify that EBCDIC characters are written as ASCII characters. Applies to fields of the string data type and record, subrecord, or tagged fields if they contain at least one field of this type.

- Import ASCII as EBCDIC. Not relevant for input links.

**Decimal**

These properties are applied to columns with a decimal data type unless overridden at column level.

- Allow all zeros. Specifies whether to treat a packed decimal column containing all zeros (which is normally illegal) as a valid representation of zero. Select Yes or No. The default is No.

- Decimal separator. Specify the ASCII character that acts as the decimal separator (period by default).

- Packed. Select an option to specify what the decimal columns contain.
  - Yes. Specifies that the decimal columns contain data in packed decimal format (the default). This has the following sub-properties:

<table>
<thead>
<tr>
<th>Sub-Properties</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check</td>
<td>Yes. Verifies that data is packed.</td>
</tr>
<tr>
<td></td>
<td>No. Does not verify.</td>
</tr>
<tr>
<td>Signed</td>
<td>Yes. To use the existing sign when writing decimal columns.</td>
</tr>
<tr>
<td></td>
<td>No. To write a positive sign (0xf) regardless of the columns' actual sign value.</td>
</tr>
</tbody>
</table>

- No (separate). Specifies that they contain unpacked decimal with a separate sign byte. This has the following sub-property:

<table>
<thead>
<tr>
<th>Sub-Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign Position</td>
<td>Choose leading or trailing as appropriate.</td>
</tr>
</tbody>
</table>

- No (zoned). Specifies that they contain an unpacked decimal in either ASCII or EBCDIC text. This has the following sub-property:
Sub-Property Description

Sign Position
- No (overpunch). Specifies that the field has a leading or end byte that contains a character which specifies both the numeric value of that byte and whether the number as a whole is negatively or positively signed. This has the following sub-property:

<table>
<thead>
<tr>
<th>Sub-Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign Position</td>
<td>Choose leading or trailing as appropriate.</td>
</tr>
</tbody>
</table>

- Precision. Specifies the precision where a decimal column is written in text format. Enter a number. When a decimal is written to a string representation, InfoSphere DataStage uses the precision and scale defined for the source decimal field to determine the length of the destination string. The precision and scale properties override this default. When they are defined, InfoSphere DataStage truncates or pads the source decimal to fit the size of the destination string. If you have also specified the field width property, InfoSphere DataStage truncates or pads the source decimal to fit the size specified by field width.

- Rounding. Specifies how to round a decimal column when writing it. Choose from the following rounding items:
  - up (ceiling). Truncate source column towards positive infinity. This mode corresponds to the IEEE 754 Round Up mode. For example, 1.4 becomes 2, -1.6 becomes -1.
  - down (floor). Truncate source column towards negative infinity. This mode corresponds to the IEEE 754 Round Down mode. For example, 1.6 becomes 1, -1.4 becomes -2.
  - nearest value. Round the source column towards the nearest representable value. This mode corresponds to the COBOL ROUNDED mode. For example, 1.4 becomes 1, 1.5 becomes 2, -1.4 becomes -1, -1.5 becomes -2.
  - truncate towards zero. This is the default. Discard fractional digits to the right of the right-most fractional digit supported by the destination, regardless of sign. For example, if the destination is an integer, all fractional digits are truncated. If the destination is another decimal with a smaller scale, truncate to the scale size of the destination decimal. This mode corresponds to the COBOL INTEGER-PART function. Using this method 1.6 becomes 1, -1.6 becomes -1.

- Scale. Specifies how to round a source decimal when its precision and scale are greater than those of the destination. By default, when the InfoSphere DataStage writes a source decimal to a string representation, it uses the precision and scale defined for the source decimal field to determine the length of the destination string. You can override the default by means of the precision and scale properties. When you do, InfoSphere DataStage truncates or pads the source decimal to fit the size of the destination string. If you have also specified the field width property, InfoSphere DataStage truncates or pads the source decimal to fit the size specified by field width.

 Numeric

These properties apply to integer and float fields unless overridden at column level.

- C_format. Perform non-default conversion of data from integer or floating-point data to a string. This property specifies a C-language format string used for
writing integer or floating point strings. This is passed to `sprintf()`. For example, you specify a C-format of `%x` and a field width of 8, so that integers are written as 8-byte hexadecimal strings.

- **In_format.** This property is not relevant for input links.
- **Out_format.** Format string used for conversion of data from integer or floating-point data to a string. This is passed to `sprintf()`. By default, DataStage invokes the C `sprintf()` function to convert a numeric field formatted as either integer or floating point data to a string. If this function does not output data in a satisfactory format, you can specify the out_format property to pass formatting arguments to `sprintf()`.

**Date**

These properties are applied to columns with a date data type unless overridden at column level. All of these are incompatible with a Data Format setting of Text.

- **Days since.** Dates are written as a signed integer containing the number of days since the specified date. Enter a date in the form `%%yyy-%mm-%dd` or in the default date format if you have defined a new one on an NLS system.
- **Format string.** The string format of a date. By default this is `%%yyy-%mm-%dd`. The Format string can contain one or a combination of the following elements:
  - `%dd`. A two-digit day.
  - `%mm`. A two-digit month.
  - `%year_cutoffyy`. A two-digit year derived from `yy` and the specified four-digit year cutoff, for example `%1970yy`.
  - `%yy`. A two-digit year derived from a year cutoff of 1900.
  - `%yyyy`. A four-digit year.
  - `%ddd`. Day of year in three-digit form (range of 1-366).
  - `%mmm`. Three-character month abbreviation.

The format_string is subject to the following restrictions:

- It cannot have more than one element of the same type, for example it cannot contain two `%dd` elements.
- It cannot have both `%dd` and `%ddd`.
- It cannot have both `%yy` and `%yyyy`.
- It cannot have both `%mm` and `%yyyy`.
- It cannot have both `%mmm` and `%ddd`.
- It cannot have both `%mm` and `%mmm`.
- If it has `%dd`, it must have `%mm` or `%mmm`.
- It must have exactly one of `%yy` or `%yyyy`.

When you specify a date format string, prefix each component with the percent symbol (%). Separate the string’s components with any character except the percent sign (%).

If this format string does not include a day, it is set to the first of the month in the destination field. If the format string does not include the month and day, they default to January 1. Note that the format string must contain a month if it also contains a day; that is, you cannot omit only the month.

The year_cutoff is the year defining the beginning of the century in which all two digit years fall. By default, the year cutoff is 1900; therefore, a two-digit year of 97 represents 1997. You can also set this using the environment variable `APT_DATE_CENTURY_BREAK_YEAR`. See Chapter 7. Reference: Stage Editor user interface 275
"APT_DATE_CENTURY_BREAK_YEAR" in IBM InfoSphere DataStage and QualityStage Parallel Job Advanced Developer’s Guide, but this is overridden by %year_cutoffyy if you have set it.

You can specify any four-digit year as the year cutoff. All two-digit years then specify the next possible year ending in the specified two digits that is the same or greater than the cutoff. For example, if you set the year cutoff to 1930, the two-digit year 30 corresponds to 1930, and the two-digit year 29 corresponds to 2029.

- Is Julian. Select this to specify that dates are written as a numeric value containing the Julian day. A Julian day specifies the date as the number of days from 4713 BCE January 1, 12:00 hours (noon) GMT.

**Time**

These properties are applied to columns with a time data type unless overridden at column level. All of these are incompatible with a Data Format setting of Text.

- Format string. Specifies the format of columns representing time as a string. By default this is %hh-%mm-%ss. The possible components of the time format string are:
  - %hh. A two-digit hours component.
  - %mm. A two-digit minute component (nn represents minutes because mm is used for the month of a date).
  - %ss. A two-digit seconds component.
  - %ss.n. A two-digit seconds plus fractional part, where n is the number of fractional digits with a maximum value of 6. If n is 0, no decimal point is printed as part of the seconds component. Trailing zeros are not suppressed.

  You must prefix each component of the format string with the percent symbol. Separate the string’s components with any character except the percent sign (%).

- Is midnight seconds. Select this to specify that times are written as a binary 32-bit integer containing the number of seconds elapsed from the previous midnight.

**Timestamp**

These properties are applied to columns with a timestamp data type unless overridden at column level.

- Format string. Specifies the format of a column representing a timestamp as a string. Defaults to %yyyy-%mm-%dd %hh:%mm:%ss.
  - %dd. A two-digit day.
  - %mm. A two-digit month.
  - %year_cutoffyy. A two-digit year derived from yy and the specified four-digit year cutoff.
  - %yy. A two-digit year derived from a year cutoff of 1900.
  - %yyyy. A four-digit year.
  - %dd. Day of year in three-digit form (range of 1 - 366)

  The following items explains the format of the hours:
  - %hh. A two-digit hours component.
  - %mm. A two-digit minute component (nn represents minutes because mm is used for the month of a date).
  - %ss. A two-digit seconds component.
- `%ss:n`. A two-digit seconds plus fractional part, where n is the number of fractional digits with a maximum value of 6. If n is 0, no decimal point is printed as part of the seconds component. Trailing zeros are not suppressed.

You must prefix each component of the format string with the percent sign (%). Separate the string’s components with any character except the percent sign (%).

**Input page: Columns tab**

The Input page always has a Columns tab. This displays the column metadata for the selected input link in a grid.

There are various ways of populating the grid:

- If the other end of the link has metadata specified for it, this will be displayed in the Columns tab (metadata is associated with, and travels with, a link).
- You can type the required metadata into the grid. When you are done, click **Save...** to save the metadata as a table definition in the Repository for subsequent reuse.
- You can load an existing table definition from the Repository. Click **Load...** to be offered a choice of table definitions to load. Note that when you load in this way you bring in the columns definitions, not any formatting information associated with them (to load that, go to the Format tab).
- You can drag a table definition from the Repository Window on the Designer onto a link on the canvas. This transfers both the column definitions and the associated format information.

If you select the options in the Grid Properties window (see *IBM InfoSphere DataStage and QualityStage Designer Client Guide*), the Columns tab will also display two extra fields: Table Definition Reference and Column Definition Reference. These show the table definition and individual columns that the columns on the tab were derived from.

If you click in a row and select **Edit Row...** from the shortcut menu, the Edit Column Meta Data window appears, which allows you edit the row details in a window format. It also has a Parallel tab which allows you to specify properties that are peculiar to parallel job column definitions. The window only shows those properties that are relevant for the current link.

The Parallel tab enables you to specify properties that give more detail about each column, and properties that are specific to the data type. Where you are specifying complex data types, you can specify a level number, which causes the Level Number field to appear in the grid on the Columns page.

If you have NLS enabled, and the column has an underlying string type, you can specify that the column contains Unicode data by selecting the Extended (Unicode) check box. Where you can enter a character for any property, this can usually be an ASCII character or a multi-byte Unicode character (if you have NLS enabled).

Some table definitions need format information. This occurs where data is being written to a file where DataStage needs additional information in order to be able to locate columns and rows. Properties for the table definition at row level are set on the Format tab of the relevant stage editor, but you can override the settings for individual columns using the Parallel tab.
Field level

The field level has the following properties:

- **Bytes to Skip.** Skip the specified number of bytes from the end of the previous column to the beginning of this column.

- **Delimiter.** Specifies the trailing delimiter of the column. Type an ASCII character or select one of the following items:
  - Whitespace. The last column of each record will not include any trailing white spaces found at the end of the record.
  - End. The end of a field is taken as the delimiter, that is there is no separate delimiter. This is not the same as a setting of 'None' which is used for fields with fixed-width columns.
  - None. No delimiter (used for fixed-width).
  - Null. ASCII Null character is used.
  - Comma. ASCII comma character used.
  - Tab. ASCII tab character used.

- **Delimiter string.** Specify a string to be written at the end of the column. Enter one or more characters. This is mutually exclusive with Delimiter, which is the default. For example, specifying ``, ` (comma space - you do not need to enter the inverted commas) would have the column delimited by `, `.

- **Drop on input.** Select this property when you must fully define the metadata for a data set, but do not want the column actually read into the data set.

- **Prefix bytes.** Specifies that this column is prefixed by 1, 2, or 4 bytes containing, as a binary value, either the column's length or the tag value for a tagged column. You can use this option with variable-length fields. Variable-length fields can be either delimited by a character or preceded by a 1-, 2-, or 4-byte prefix containing the field length. InfoSphere DataStage inserts the prefix before each field.

  This property is mutually exclusive with the Delimiter, Quote, and Final Delimiter properties, which are used by default.

- **Print field.** This property is intended for use when debugging jobs. Set it to have InfoSphere DataStage produce a message for each of the columns it reads. The message has the following format:

  Importing \textit{N: D}

  Substitute the following for each variable:
  - \textit{N}. The column name.
  - \textit{D}. Imported column data. Non-printable characters contained in \textit{D} are prefixed with an escape character and written as C string literals. If the column contains binary data, it is output in octal format.

- **Quote.** Specifies that variable length columns are enclosed in single quotes, double quotes, or another ASCII character or pair of ASCII characters. Choose Single or Double, or enter a character.

- **Start position.** Specifies the starting position of a column in the record. The starting position can be either an absolute byte offset from the first record position (0) or the starting position of another column.

- **Tag case value.** Explicitly specifies the tag value corresponding to a subfield in a tagged subrecord. By default the fields are numbered 0 to \textit{N}-1, where \textit{N} is the number of fields. (A tagged subrecord is a column whose type can vary. The sub fields of the tagged subrecord are the possible types. The tag case value of the tagged subrecord selects which of those types is used to interpret the column's value for the record.)
String type
This has the following properties:

- Character Set. Choose from ASCII or EBCDIC (not available for ustring type (Unicode)).
- Default. The default value for a column. This is used for data written by a Generate stage. It also supplies the value to substitute for a column that causes an error (whether written or read).
- Export EBCDIC as ASCII. Select this to specify that EBCDIC characters are written as ASCII characters (not available for ustring type (Unicode)).
- Is link field. Selected to indicate that a column holds the length of another, variable-length column of the record or of the tag value of a tagged record field.
- Import ASCII as EBCDIC. Select this to specify that ASCII characters are read as EBCDIC characters (not available for ustring type (Unicode)).
- Field max width. The maximum number of bytes in a column represented as a string. Enter a number. This is useful where you are storing numbers as text. If you are using a fixed-width character set, you can calculate the length exactly. If you are using a variable-length character set, calculate an adequate maximum width for your fields. The property applies to fields of all data types except date, time, timestamp, and raw; and record, subrecord, or tagged if they contain at least one field of this type.
- Field width. The number of bytes in a column represented as a string. Enter a number. This is useful where you are storing numbers as text. If you are using a fixed-width character set, you can calculate the number of bytes exactly. If it’s a variable length encoding, base your calculation on the width and frequency of your variable-width characters. Applies to fields of all data types except date, time, timestamp, and raw; and record, subrecord, or tagged if they contain at least one field of this type.
- Pad char. Specifies the pad character used when strings or numeric values are written to an external string representation. Enter a character (single-byte for strings, can be multi-byte for ustrings) or choose null or space. The pad character is used when the external string representation is larger than required to hold the written field. In this case, the external string is filled with the pad character to its full length. Space is the default. Applies to string, ustring, and numeric data types and record, subrecord, or tagged types if they contain at least one field of this type.

Date type
Data type has the following properties:

- Byte order. Specifies how multiple byte data types are ordered. Choose from:
  - little-endian. The high byte is on the right.
  - big-endian. The high byte is on the left.
  - native-endian. As defined by the native format of the machine.
- Character Set. Choose from ASCII or EBCDIC.
- Days since. Dates are written as a signed integer containing the number of days since the specified date. Enter a date in the form %yyyy-%mm-%dd or in the default date format if you have defined a new one on an NLS system.
- Data Format. Specifies the data representation format of a column. Choose from:
  - binary
  - text
For dates, binary is equivalent to specifying the julian property for the date field, text specifies that the data to be written contains a text-based date in the form %yy-%mm-%dd or in the default date format if you have defined a new one on an NLS system.

- Default. The default value for a column. This is used for data written by a Generate stage. It also supplies the value to substitute for a column that causes an error (whether written or read).
- Format string. The string format of a date. By default this is %yy-%mm-%dd. The Format string can contain one or a combination of the following elements:
  - %dd. A two-digit day.
  - %mm. A two-digit month.
  - %year_cutoffyy. A two-digit year derived from yy and the specified four-digit year cutoff, for example %1970yy.
  - %yy. A two-digit year derived from a year cutoff of 1900.
  - %yyyy. A four-digit year.
  - %ddd. Day of year in three-digit form (range of 1-366).
  - %mmm. Three-character month abbreviation.

The format string is subject to the following restrictions:
- It cannot have more than one element of the same type, for example it cannot contain two %dd elements.
- It cannot have both %dd and %ddd.
- It cannot have both %yy and %yyyy.
- It cannot have both %mm and %yyyy.
- It cannot have both %mmm and %ddd.
- It cannot have both %mm and %mmm.
- If it has %dd, it must have %mm or %mmm.
- It must have exactly one of %yy or %yyyy.

When you specify a date format string, prefix each component with the percent symbol (%). Separate the string's components with any character except the percent sign (%).

If this format string does not include a day, it is set to the first of the month in the destination field. If the format string does not include the month and day, they default to January 1. Note that the format string must contain a month if it also contains a day; that is, you cannot omit only the month.

The year_cutoff is the year defining the beginning of the century in which all two digit years fall. By default, the year cutoff is 1900; therefore, a two-digit year of 97 represents 1997. You can also set this using the environment variable APT_DATE_CENTURY_BREAK_YEAR but this is overridden by %year_cutoffyy if you have set it.

You can specify any four-digit year as the year cutoff. All two-digit years then specify the next possible year ending in the specified two digits that is the same or greater than the cutoff. For example, if you set the year cutoff to 1930, the two-digit year 30 corresponds to 1930, and the two-digit year 29 corresponds to 2029.

- Is Julian. Select this to specify that dates are written as a numeric value containing the Julian day. A Julian day specifies the date as the number of days from 4713 BCE January 1, 12:00 hours (noon) GMT.

**Time type**

Time type has the following properties:
• Byte order. Specifies how multiple byte data types are ordered. Choose from the following byte order:
  – little-endian. The high byte is on the right.
  – big-endian. The high byte is on the left.
  – native-endian. As defined by the native format of the machine.
• Character Set. Choose from ASCII or EBCDIC.
• Default. The default value for a column. This is used for data written by a Generate stage. It also supplies the value to substitute for a column that causes an error (whether written or read).
• Data Format. Specifies the data representation format of a column. Choose from the following data format:
  – binary
  – text
    For time, binary is equivalent to midnight_seconds, text specifies that the field represents time in the text-based form %hh:%nn:%ss or in the default date format if you have defined a new one on an NLS system.
• Format string. Specifies the format of columns representing time as a string. By default this is %hh-%nn-%ss. The following are the possible components of the time format string:
  – %hh: A two-digit hours component.
  – %nn: A two-digit minute component (nn represents minutes because mm is used for the month of a date).
  – %ss: A two-digit seconds component.
  – %ss.n: A two-digit seconds plus fractional part, where n is the number of fractional digits with a maximum value of 6. If n is 0, no decimal point is printed as part of the seconds component. Trailing zeros are not suppressed.
    You must prefix each component of the format string with the percent symbol. Separate the string’s components with any character except the percent sign (%).
• Is midnight seconds. Select this to specify that times are written as a binary 32-bit integer containing the number of seconds elapsed from the previous midnight.

**Timestamp type**
The following describes the Timestamp type properties:
• Byte order. Specifies how multiple byte data types are ordered. Choose from the following byte order:
  – little-endian. The high byte is on the right.
  – big-endian. The high byte is on the left.
  – native-endian. As defined by the native format of the machine.
• Character Set. Choose from ASCII or EBCDIC.
• Data Format. Specifies the data representation format of a column. Choose from the following data format:
  – binary
  – text
    For timestamp, binary specifies that the first integer contains a Julian day count for the date portion of the timestamp and the second integer specifies the time portion of the timestamp as the number of seconds from midnight. A binary timestamp specifies that two 32-but integers are written. Text specifies
a text-based timestamp in the form %yy-%mm-%dd %hh:%nn:%ss or in the
default date format if you have defined a new one on an NLS system.

- Default. The default value for a column. This is used for data written by a
  Generate stage. It also supplies the value to substitute for a column that causes
  an error (whether written or read).

- Format string. Specifies the format of a column representing a timestamp as a
  string. Defaults to %yy-%mm-%dd %hh:%nn:%ss. Specify the format as follows:
  For the date:
  - %dd: A two-digit day.
  - %mm: A two-digit month.
  - %year_cutoffyy: A two-digit year derived from yy and the specified four-digit
    year cutoff.
  - %yy: A two-digit year derived from a year cutoff of 1900.
  - %yyyy: A four-digit year.
  - %ddd: Day of year in three-digit form (range of 1 - 366)
  For the time:
  - %hh: A two-digit hours component.
  - %nn: A two-digit minute component (nn represents minutes because mm is
    used for the month of a date).
  - %ss: A two-digit seconds component.
  - %ss.n: A two-digit seconds plus fractional part, where n is the number of
    fractional digits with a maximum value of 6. If n is 0, no decimal point is
    printed as part of the seconds component. Trailing zeros are not suppressed.
    You must prefix each component of the format string with the percent symbol
    (%). Separate the string’s components with any character except the percent
    sign (%).

**Integer type**

The following explains the integer type properties:

- Byte order. Specifies how multiple byte data types are ordered. Choose from the
  following byte order:
  - little-endian. The high byte is on the right.
  - big-endian. The high byte is on the left.
  - native-endian. As defined by the native format of the machine.

- Character Set. Choose from ASCII or EBCDIC.

- C_format. Perform non-default conversion of data from a string to integer data.
  This property specifies a C-language format string used for reading/writing
  integer strings. This is passed to sscanf() or sprintf().

- Default. The default value for a column. This is used for data written by a
  Generate stage. It also supplies the value to substitute for a column that causes
  an error (whether written or read).

- Data Format. Specifies the data representation format of a column. Choose from
  the following data format:
  - binary
  - text

- Field max width. The maximum number of bytes in a column represented as a
  string. Enter a number. This is useful where you are storing numbers as text. If
  you are using a fixed-width character set, you can calculate the length exactly. If
  you are using variable-length character set, calculate an adequate maximum
width for your fields. Applies to fields of all data types except date, time, timestamp, and raw; and record, subrecord, or tagged if they contain at least one field of this type.

- **Field width.** The number of bytes in a column represented as a string. Enter a number. This is useful where you are storing numbers as text. If you are using a fixed-width character set, you can calculate the number of bytes exactly. If it is a variable length encoding, base your calculation on the width and frequency of your variable-width characters. Applies to fields of all data types except date, time, timestamp, and raw; and record, subrecord, or tagged if they contain at least one field of this type.

- **In_format.** Format string used for conversion of data from string to integer. This is passed to `sscanf()`. By default, DataStage invokes the C `sscanf()` function to convert a numeric field formatted as a string to either integer or floating point data. If this function does not output data in a satisfactory format, you can specify the `in_format` property to pass formatting arguments to `sscanf()`.

- **Is link field.** Selected to indicate that a column holds the length of another, variable-length column of the record or of the tag value of a tagged record field.

- **Out_format.** Format string used for conversion of data from integer to a string. This is passed to `sprintf()`. By default, DataStage invokes the C `sprintf()` function to convert a numeric field formatted as integer data to a string. If this function does not output data in a satisfactory format, you can specify the `out_format` property to pass formatting arguments to `sprintf()`.

- **Pad char.** Specifies the pad character used when the integer is written to an external string representation. Enter a character (single-byte for strings, can be multi-byte for ustrings) or choose null or space. The pad character is used when the external string representation is larger than required to hold the written field. In this case, the external string is filled with the pad character to its full length. Space is the default.

### Decimal type

The decimal type has the following properties:

- **Allow all zeros.** Specifies whether to treat a packed decimal column containing all zeros (which is normally illegal) as a valid representation of zero. Select Yes or No.

- **Character Set.** Choose from ASCII or EBCDIC.

- **Decimal separator.** Specify the character that acts as the decimal separator (period by default).

- **Default.** The default value for a column. This is used for data written by a Generate stage. It also supplies the value to substitute for a column that causes an error (whether written or read).

- **Data Format.** Specifies the data representation format of a column. Choose from the following data format:
  - **binary**
  - **text**

  For decimals, binary means packed. Text represents a decimal in a string format with a leading space or `-` followed by decimal digits with an embedded decimal point if the scale is not zero. The destination string format is: [+ -][d][d][d] and any precision and scale arguments are ignored.

- **Field max width.** The maximum number of bytes in a column represented as a string. Enter a number. This is useful where you are storing numbers as text. If you are using a fixed-width character set, you can calculate the length exactly. If you are using variable-length character set, calculate an
adequate maximum width for your fields. Applies to fields of all data types
except date, time, timestamp, and raw; and record, subrecord, or tagged if they
contain at least one field of this type.

- Field width. The number of bytes in a column represented as a string. Enter a
  number. This is useful where you are storing numbers as text. If you are using a
  fixed-width character set, you can calculate the number of bytes exactly. If it's a
  variable length encoding, base your calculation on the width and frequency of
  your variable-width characters. Applies to fields of all data types except date,
  time, timestamp, and raw; and record, subrecord, or tagged if they contain at
  least one field of this type.

- Packed. Select an option to specify what the decimal columns contain, choose from:
  - Yes to specify that the decimal columns contain data in packed decimal
    format (the default). This has the following sub-properties:
      Check. Select Yes to verify that data is packed, or No to not verify.
      Signed. Select Yes to use the existing sign when writing decimal columns.
      Select No to write a positive sign (0xf) regardless of the columns' actual sign
      value.
  - No (separate) to specify that they contain unpacked decimal with a separate
    sign byte. This has the following sub-property:
    Sign Position. Choose leading or trailing as appropriate.
  - No (zoned) to specify that they contain an unpacked decimal in either ASCII
    or EBCDIC text. This has the following sub-property:
    Sign Position. Choose leading or trailing as appropriate.
  - No (overpunch) to specify that the field has a leading or end byte that
    contains a character which specifies both the numeric value of that byte and
    whether the number as a whole is negatively or positively signed. This has
    the following sub-property:
    Sign Position. Choose leading or trailing as appropriate.

- Precision. Specifies the precision where a decimal column is represented in text
  format. Enter a number. When a decimal is written to a string representation,
  InfoSphere DataStage uses the precision and scale defined for the source decimal
  field to determine the length of the destination string. The precision and scale
  properties override this default. When they are defined, InfoSphere DataStage
  truncates or pads the source decimal to fit the size of the destination string. If
  you have also specified the field width property, InfoSphere DataStage truncates
  or pads the source decimal to fit the size specified by field width.

- Rounding. Specifies how to round the source field to fit into the destination
decimal when reading a source field to a decimal. Choose one of the following
rounding types:
  - up (ceiling). Truncate source column towards positive infinity. This mode
    corresponds to the IEEE 754 Round Up mode. For example, 1.4 becomes 2,
    -1.6 becomes -1.
  - down (floor). Truncate source column towards negative infinity. This mode
    corresponds to the IEEE 754 Round Down mode. For example, 1.6 becomes 1,
    -1.4 becomes -2.
  - nearest value. Round the source column towards the nearest representable
    value. This mode corresponds to the COBOL ROUNDED mode. For example,
    1.4 becomes 1, 1.5 becomes 2, -1.4 becomes -1, -1.5 becomes -2.
  - truncate towards zero. This is the default. Discard fractional digits to the right
    of the right-most fractional digit supported by the destination, regardless of
    sign. For example, if the destination is an integer, all fractional digits are
truncated. If the destination is another decimal with a smaller scale, truncate
to the scale size of the destination decimal. This mode corresponds to the
COBOL INTEGER-PART function. Using this method 1.6 becomes 1, -1.6
becomes -1.

- Scale. Specifies how to round a source decimal when its precision and scale are
greater than those of the destination. By default, when the InfoSphere DataStage
writes a source decimal to a string representation, it uses the precision and scale
derived for the source decimal field to determine the length of the destination
string. You can override the default by means of the precision and scale
properties. When you do, InfoSphere DataStage truncates or pads the source
decimal to fit the size of the destination string. If you have also specified the
field width property, InfoSphere DataStage truncates or pads the source
decimal to fit the size specified by field width. Specifies how to round a source decimal
when its precision and scale are greater than those of the destination.

**Float type**

The following explains the float type properties:

- C_format. Perform non-default conversion of data from a string to floating-point
data. This property specifies a C-language format string used for reading
floating point strings. This is passed to sscanf().

- Character Set. Choose from ASCII or EBCDIC.

- Default. The default value for a column. This is used for data written by a
Generate stage. It also supplies the value to substitute for a column that causes
an error (whether written or read).

- Data Format. Specifies the data representation format of a column. Choose from
the following data format:
  - binary
  - text

- Field max width. The maximum number of bytes in a column represented as a
string. Enter a number. This is useful where you are storing numbers as text. If
you are using a fixed-width character set, you can calculate the length exactly. If
you are using variable-length character set, calculate an adequate maximum
width for your fields. Applies to fields of all data types except date, time,
timestamp, and raw; and record, subrecord, or tagged if they contain at least one
field of this type.

- Field width. The number of bytes in a column represented as a string. Enter a
number. This is useful where you are storing numbers as text. If you are using a
fixed-width character set, you can calculate the number of bytes exactly. If it's a
variable length encoding, base your calculation on the width and frequency of
your variable-width characters. Applies to fields of all data types except date,
time, timestamp, and raw; and record, subrecord, or tagged if they contain at least one
field of this type.

- In_format. Format string used for conversion of data from string to floating
point. This is passed to sscanf(). By default, InfoSphere DataStage invokes the C
sscanf() function to convert a numeric field formatted as a string to floating
point data. If this function does not output data in a satisfactory format, you can
specify the in_format property to pass formatting arguments to sscanf().

- Is link field. Selected to indicate that a column holds the length of a another,
variable-length column of the record or of the tag value of a tagged record field.

- Out_format. Format string used for conversion of data from floating point to a
string. This is passed to sprintf(). By default, InfoSphere DataStage invokes the C
sprintf() function to convert a numeric field formatted as floating point data
to a string. If this function does not output data in a satisfactory format, you can specify the out_format property to pass formatting arguments to sprintf().

- **Pad char.** Specifies the pad character used when the floating point number is written to an external string representation. Enter a character (single-byte for strings, can be multi-byte for ustrings) or choose null or space. The pad character is used when the external string representation is larger than required to hold the written field. In this case, the external string is filled with the pad character to its full length. Space is the default.

**Nullable**
This appears for nullable fields.

- **Actual field length.** Specifies the number of bytes to fill with the Fill character when a field is identified as null. When InfoSphere DataStage identifies a null field, it will write a field of this length full of Fill characters. This is mutually exclusive with Null field value.
- **Null field length.** The length in bytes of a variable-length field that contains a null. When a variable-length field is read, a length of null field length in the source field indicates that it contains a null. When a variable-length field is written, InfoSphere DataStage writes a length value of null field length if the field contains a null. This property is mutually exclusive with null field value.
- **Null field value.** Specifies the value given to a null field if the source is set to null. Can be a number, string, or C-type literal escape character.

For example, you can represent a byte value by »000, where each o is an octal digit 0 - 7 and the first o is < 4, or by »\xhh, where each h is a hexadecimal digit 0 - F. You must use this form to encode non-printable byte values.

This property is mutually exclusive with Null field length and Actual length. For a fixed width data representation, you can use Pad char (from the general section of Type defaults) to specify a repeated trailing character if the value you specify is shorter than the fixed width of the field. On reading, specifies the value given to a field containing a null. On writing, specifies the value given to a field if the source is set to null. Can be a number, string, or C-type literal escape character.

**Generator**
If the column is being used in a Row Generator or Column Generator stage, this allows you to specify extra details about the mock data being generated. The exact fields that appear depend on the data type of the column being generated. They allow you to specify features of the data being generated.

For example, integers allow you to specify if values are random or whether they cycle. If they cycle, you can specify an initial value, an increment, and a limit. If they are random, you can specify a seed value for the random number generator, whether to include negative numbers and a limit.

The following diagram shows the generate options available for the different data types.
All data types

All data types other than string have two types of operation, cycle and random:

- **Cycle.** The cycle option generates a repeating pattern of values for a column. It has the following optional dependent properties:
  - Increment. The increment value added to produce the field value in the next output record. The default value is 1 (integer) or 1.0 (float).
  - Initial value. is the initial field value (value of the first output record). The default value is 0.
- Limit. The maximum field value. When the generated field value is greater than Limit, it wraps back to Initial value. The default value of Limit is the maximum allowable value for the field's data type.

You can set these to `part' to use the partition number (for example, 0, 1, 2, 3 on a four node system), or `partcount' to use the total number of executing partitions (for example, 4 on a four node system).

- Random. The random option generates random values for a field. It has the following optional dependent properties:
  - Limit. Maximum generated field value. The default value of limit is the maximum allowable value for the field's data type.
  - Seed. The seed value for the random number generator used by the stage for the field. You do not have to specify seed. By default, the stage uses the same seed value for all fields containing the random option.
  - Signed. Specifies that signed values are generated for the field (values between -limit and +limit). Otherwise, the operator creates values between 0 and +limit.

You can limit and seed to `part' to use the partition number (e.g., 0, 1, 2, 3 on a four node system), or `partcount' to use the total number of executing partitions (for example, 4 on a four node system).

Strings

By default the generator stages initialize all bytes of a string field to the same alphanumeric character. The stages use the following characters, in the following order:

abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ

For example, the following string with a length of 5 would produce successive string fields with the values:

aaaaa
bbbbb
cccccc
ddddd
...

After the last character, capital Z, values wrap back to lowercase a and the cycle repeats.

You can also use the algorithm property to determine how string values are generated, this has two possible values: cycle and alphabet.

- Cycle. Values are assigned to a generated string field as a set of discrete string values to cycle through. This has the following dependent property.
  - Values. Repeat this property to specify the string values that the generated data cycles through.

- Alphabet. Values are assigned to a generated string field as a character string each of whose characters is taken in turn. This is like the default mode of operation except that you can specify the string cycled through using the dependent property String.

Decimal

As well as the Type property, decimal columns have the following properties:

- Percent invalid. The percentage of generated columns that contain a invalid values. Set to 10% by default.
• Percent zero. The percentage of generated decimal columns where all bytes of the decimal are set to binary zero (0x00). Set to 10% by default.

**Date**

As well as the Type property, date columns have the following properties:
• Epoch. Use this to specify the earliest generated date value, in the format `yyyy-mm-dd` (leading zeros must be supplied for all parts). The default is 1960-01-01.
• Percent invalid. The percentage of generated columns that will contain invalid values. Set to 10% by default.
• Use current date. Set this to generate today’s date in this column for every row generated. If you set this all other properties are ignored.

**Time**

As well as the Type property, time columns have the following properties:
• Percent invalid. The percentage of generated columns that will contain invalid values. Set to 10% by default.
• Scale factor. Specifies a multiplier to the increment value for time. For example, a scale factor of 60 and an increment of 1 means the field increments by 60 seconds.

**Timestamp**

As well as the Type property, time columns have the following properties:
• Epoch. Use this to specify the earliest generated date value, in the format `yyyy-mm-dd` (leading zeros must be supplied for all parts). The default is 1960-01-01.
• Use current date. Set this to generate today’s date in this column for every row generated. If you set this all other properties are ignored.
• Percent invalid. The percentage of generated columns that will contain invalid values. Set to 10% by default.
• Scale factor. Specifies a multiplier to the increment value for time. For example, a scale factor of 60 and an increment of 1 means the field increments by 60 seconds.

**Vectors**

If the row you are editing represents a column which is a variable length vector, tick the Variable check box. The Vector properties appear, these give the size of the vector in one of two ways:
• Link Field Reference. The name of a column containing the number of elements in the variable length vector. This should have an integer or float type, and have its Is Link field property set.
• Vector prefix. Specifies 1-, 2-, or 4-byte prefix containing the number of elements in the vector.

If the row you are editing represents a column which is a vector of known length, enter the number of elements in the Vector Occurs box.
Subrecords
If the row you are editing represents a column which is part of a subrecord the Level Number column indicates the level of the column within the subrecord structure.

If you specify Level numbers for columns, the column immediately preceding will be identified as a subrecord. Subrecords can be nested, so can contain further subrecords with higher level numbers (that is, level 06 is nested within level 05). Subrecord fields have a Tagged check box to indicate that this is a tagged subrecord.

Extended
For certain data types the Extended check box appears to allow you to modify the data type as follows:
- Char, VarChar, LongVarChar. Select to specify that the underlying data type is a ustring.
- Time. Select to indicate that the time field includes microseconds.
- Timestamp. Select to indicate that the timestamp field includes microseconds.
- TinyInt, SmallInt, Integer, BigInt types. Select to indicate that the underlying data type is the equivalent uint field.

Advanced tab
The Advanced tab allows you to specify how InfoSphere DataStage buffers data being input this stage. By default InfoSphere DataStage buffers data in such a way that no deadlocks can arise. A deadlock is defined as a number of mutually dependent stages that are waiting for input from another stage and cannot output until they receive it.

The size and operation of the buffer are usually the same for all links on all stages. The default values that the settings take can be set using environment variables.

Use the Advanced tab to specify buffer settings on a per-link basis. You should only change the settings if you fully understand the consequences of your actions. Otherwise you could cause deadlock situations to occur.

Any changes you make on this tab are reflected in the Output page Advanced tab of the stage at the other end of this link.

The settings are as follows:
- Buffering mode. Select one of the following from the drop-down list.
  - (Default). This takes whatever the default settings are as specified by the environment variables known as Auto-buffer unless you have explicitly changed the value of the APT_BUFFERING_POLICY environment variable.
  - Auto buffer. Buffer output data only if necessary to prevent a dataflow deadlock situation.
  - Buffer. This unconditionally buffers all data output from this stage.
  - No buffer. Do not buffer output data under any circumstances. This could potentially lead to deadlock situations if not used carefully.

If you choose the Auto buffer or Buffer options, you can also set the values of the various buffering parameters:
- Maximum memory buffer size (bytes). Specifies the maximum amount of virtual memory, in bytes, used per buffer. The default size is 3145728 (3 MB).
• Buffer free run (percent). Specifies how much of the available in-memory buffer to consume before the buffer resists. This is expressed as a percentage of Maximum memory buffer size. When the amount of data in the buffer is less than this value, new data is accepted automatically. When the data exceeds it, the buffer first tries to write some of the data it contains before accepting more. The default value is 50% of the Maximum memory buffer size. You can set it to greater than 100%, in which case the buffer continues to store data up to the indicated multiple of Maximum memory buffer size before writing to disk.

• Queue upper bound size (bytes). Specifies the maximum amount of data buffered at any time using both memory and disk. The default value is zero, meaning that the buffer size is limited only by the available disk space as specified in the configuration file (resource scratchdisk). If you set Queue upper bound size (bytes) to a non-zero value, the amount of data stored in the buffer will not exceed this value (in bytes) plus one block (where the data stored in a block cannot exceed 32 KB).

If you set Queue upper bound size to a value equal to or slightly less than Maximum memory buffer size, and set Buffer free run to 1.0, you will create a finite capacity buffer that does not write to disk. However, the size of the buffer is limited by the virtual memory of your system and you can create deadlock if the buffer becomes full.

• Disk write increment (bytes). Sets the size, in bytes, of blocks of data being moved to/from disk by the buffering operator. The default is 1048576 (1 MB). Adjusting this value trades amount of disk access against throughput for small amounts of data. Increasing the block size reduces disk access, but may decrease performance when data is being read/written in smaller units. Decreasing the block size increases throughput, but may increase the amount of disk access.

---

**Output page**

The Output page gives information about links going out of a stage. In the case of a file or database stage an input link carries data being read from the file or database. In the case of a processing or restructure stage it carries data that the stage has processed. Where there are no output links the stage editor has no Output page.

Where it is present, the Output page contains various tabs depending on stage type. The only field the Output page itself contains is Output name, which gives the name of the link being edited. Where a stage has more than one output link, you can select the link you are editing from the Output name drop-down list.

The Output page also has a Columns... button. Click **Columns...** to open a window showing column names from the metadata defined for this link. You can drag these columns to various fields in the Output page tabs as required.

Certain stage types will also have a View Data... button. Click **View Data** to view the actual data associated with the specified data source or data target. The button is available if you have defined metadata for the link.

The Sequential File stage has a Show File... button, rather than View Data... . This shows the flat file as it has been created on disk.
General tab
The Output page always has a General tab. This allows you to enter an optional description of the link. Specifying a description for each link enhances job maintainability.

Properties tab
Some types of file and database stages can have properties that are particular to specific output links. In this case the Output page has a Properties tab. This has the same format as the Stage page Properties tab.

Format tab
Stages that read from certain types of file (for example, the Sequential File stage) also have a Format tab which allows you to specify the format of the file or files being read from.

The Format page is similar in structure to the Properties page. A flat file has a number of properties that you can set different attributes for. Select the property in the tree and select the attributes you want to set from the Available properties to add window. It then appears as a dependent property in the property tree and you can set its value as required. This tab sets the format information for the file at row level. You can override the settings for individual columns using the Edit Column Metadata window.

Format details are also stored with table definitions, and you can use the Load... button to load a format from a table definition stored in the metadata repository.

The short-cut menu from the property tree gives access to the following functions:
- Format as. This applies a predefined template of properties. Choose from the following properties:
  - Delimited/quoted
  - Fixed-width records
  - UNIX line terminator
  - DOS line terminator
  - No terminator (fixed width)
  - Mainframe (COBOL)
- Add sub-property. Gives access to a list of dependent properties for the currently selected property (visible only if the property has dependents).
- Set to default. Appears if the currently selected property has been set to a non-default value, allowing you to re-select the default.
- Remove. Removes the currently selected property. This is disabled if the current property is mandatory.
- Remove all. Removes all the non-mandatory properties.

Record level
These properties define details about how data records are formatted in the flat file.

Where you can enter a character, this can usually be an ASCII character or a multi-byte Unicode character (if you have NLS enabled). The following are the available properties:
- Fill char. Does not apply to output links
• Final delimiter string. Specify the string written after the last column of a record in place of the column delimiter. Enter one or more characters, this precedes the record delimiter if one is used. Mutually exclusive with Final delimiter, which is the default.

For example, if you set Delimiter to comma (see under “Field Defaults” for Delimiter) and Final delimiter string to `, ` (comma space - you do not need to enter the inverted commas) all fields are delimited by a comma, except the final field, which is delimited by a comma followed by an ASCII space character. InfoSphere DataStage skips the specified delimiter string when reading the file.

• Final delimiter. Specify the single character written after the last column of a record in place of the field delimiter. Type a character or select one of whitespace, end, none, null, tab, or comma. InfoSphere DataStage skips the specified delimiter string when reading the file.
  – whitespace. The last column of each record will not include any trailing white spaces found at the end of the record.
  – end. The last column of each record does not include the field delimiter. This is the default setting.
  – none. The last column of each record does not have a delimiter, used for fixed-width fields.
  – null. The last column of each record is delimited by the ASCII null character.
  – comma. The last column of each record is delimited by the ASCII comma character.
  – tab. The last column of each record is delimited by the ASCII tab character.

When writing, a space is now inserted after every field except the last record.

• Intact. The intact property specifies an identifier of a partial schema. A partial schema specifies that only the column or columns named in the schema can be modified by the stage. All other columns in the row are passed through unmodified. The file containing the partial schema is specified in the Schema File property on Outputs tab. This property has the following dependent property:
  Check intact. Select this to force validation of the partial schema as the file or files are imported. Note that this can degrade performance.

• Record delimiter string. Specify the string at the end of each record. Enter one or more characters. This is mutually exclusive with Record delimiter, which is the default, and record type and record prefix.

• Record delimiter. Specify the single character at the end of each record. Type a character or select one of the following:
  – UNIX Newline (the default)
  – null
To specify a DOS newline, use the Record delimiter string property set to "\R\N" or choose **Format as > DOS line terminator** from the shortcut menu.

Record delimiter is mutually exclusive with Record delimiter string, Record prefix, and record type.

- **Record length.** Select Fixed where fixed length fields are being read. InfoSphere DataStage calculates the appropriate length for the record. Alternatively, specify the length of fixed records as number of bytes. This is not used by default (default files are comma-delimited).

- **Record Prefix.** Specifies that a variable-length record is prefixed by a 1-, 2-, or 4-byte length prefix. It is set to 1 by default. This is mutually exclusive with Record delimiter, which is the default, and record delimiter string and record type.

- **Record type.** Specifies that data consists of variable-length blocked records (varying) or implicit records (implicit). If you choose the implicit property, data is written as a stream with no explicit record boundaries. The end of the record is inferred when all of the columns defined by the schema have been parsed. The varying property allows you to specify one of the following IBM blocked or spanned formats: V, VB, VS, VBS, or VR.

This property is mutually exclusive with Record length, Record delimiter, Record delimiter string, and Record prefix and by default is not used.

**Field defaults**

The Field defaults are applied to all columns read from the file or files but can be overridden for individual columns from the Columns tab using the Edit Column Metadata window. Where you can enter a character, this can usually be an ASCII character or a multi-byte Unicode character (if you have NLS enabled). The available properties are:

- **Actual field length.** Specifies the number of bytes to fill with the Fill character when a field is identified as null. When DataStage identifies a null field, it will write a field of this length full of Fill characters. This is mutually exclusive with Null field value.

- **Delimiter.** Specifies the trailing delimiter of all fields in the record. Type an ASCII character or select one of whitespace, end, none, null, comma, or tab.
  - **Whitespace.** Characters at the end of a column are ignored, in that they are not treated as part of the column.
  - **End.** The end of a field is taken as the delimiter, in that there is no separate delimiter. This is not the same as a setting of `None' which is used for fields with fixed-width columns.
  - **None.** No delimiter (used for fixed-width).
  - **Null.** ASCII Null character is used.
  - **Comma.** ASCII comma character is used.
  - **Tab.** ASCII tab character is used.

- **Delimiter string.** Specify the string at the end of each field. Enter one or more characters. This is mutually exclusive with Delimiter, which is the default. For example, specifying ` `, ` (comma space - you do not need to enter the inverted commas) specifies each field is delimited by `, ` unless overridden for individual fields.

- **Null field length.** The length in bytes of a variable-length field that contains a null. When a variable-length field is read, a length of null field length in the source field indicates that it contains a null. This property is mutually exclusive with null field value.
Null field value. Specifies the value given to a null field if the source is set to null. Can be a number, string, or C-type literal escape character. For example, you can represent a byte value by `<i>`000, where each o is an octal digit 0-7 and the first o is < 4, or by `\xhh`, where each h is a hexadecimal digit 0-F. You must use this form to encode non-printable byte values.

This property is mutually exclusive with Null field length and Actual length. For a fixed width data representation, you can use Pad char (from the general section of Type defaults) to specify a repeated trailing character if the value you specify is shorter than the fixed width of the field.

Prefix bytes. Specifies that each column in the data file is prefixed.

You can use this option with variable-length fields. Variable-length fields can be either delimited by a character or preceded by a 1-, 2-, or 4-byte prefix containing the field length. InfoSphere DataStage reads the length prefix but does not include the prefix as a separate field in the data set it reads from the file.

This property is mutually exclusive with the Delimiter, Quote, and Final Delimiter properties, which are used by default.

Print field. This property is intended for use when debugging jobs. Set it to have InfoSphere DataStage produce a message for every field it reads. The message has the following format:

Importing N: D

The variables are interpreted as follows.
- N is the field name.
- D is the imported data of the field. Non-printable characters contained in D are prefixed with an escape character and written as C string literals; if the field contains binary data, it is output in octal format.

Quote. Specifies that variable length fields are enclosed in single quotes, double quotes, or another character or pair of characters. Choose Single or Double, or enter a character. This is set to double quotes by default.

When reading, InfoSphere DataStage ignores the leading quote character and reads all bytes up to but not including the trailing quote character.

Vector prefix. For fields that are variable length vectors, specifies that a 1-, 2-, or 4-byte prefix contains the number of elements in the vector. You can override this default prefix for individual vectors. Variable-length vectors must use either a prefix on the vector or a link to another field in order to specify the number of elements in the vector. If the variable length vector has a prefix, you use this property to indicate the prefix length. InfoSphere DataStage reads the length prefix but does not include it as a separate field in the data set. By default, the prefix length is assumed to be one byte.

**Type defaults**

These are properties that apply to all columns of a specific data type unless specifically overridden at the column level. They are divided into a number of subgroups according to data type.

**General**

These properties apply to several data types (unless overridden at column level):
- Byte order. Specifies how multiple byte data types (except string and raw data types) are ordered. Choose from one of the following byte order types:
  - little-endian. The high byte is on the right.
  - big-endian. The high byte is on the left.
- native-endian. As defined by the native format of the machine. This is the default.

- Data Format. Specifies the data representation format of a field. Applies to fields of all data types except string, ustring, and raw and to record, subrecord or tagged fields containing at least one field that is neither string nor raw. Choose from one of the following data formats:
  - binary
  - text (the default)

  A setting of binary has different meanings when applied to different data types:
  - For decimals, binary means packed.
  - For other numerical data types, binary means "not text".
  - For dates, binary is equivalent to specifying the julian property for the date field.
  - For time, binary is equivalent to midnight_seconds.
  - For timestamp, binary specifies that the first integer contains a Julian day count for the date portion of the timestamp and the second integer specifies the time portion of the timestamp as the number of seconds from midnight. A binary timestamp specifies that two 32-bit integers are written.

By default data is formatted as text, as follows:
  - For the date data type. Text specifies that the data read contains a text-based date in the form %Y%Y-%MM-%DD or in the default date format if you have defined a new one on an NLS system.
  - For the decimal data type. A field represents a decimal in a string format with a leading space or '-' followed by decimal digits with an embedded decimal point if the scale is not zero. The destination string format is: [+ | -]ddd.[ddd] and any precision and scale arguments are ignored.
  - For numeric fields (int8, int16, int32, uint8, uint16, uint32, sfloat, and dfloat). InfoSphere DataStage Assumes that numeric fields are represented as text.
  - For the time data type. Text specifies that the field represents time in the text-based form %hh:mm:ss or in the default date format if you have defined a new one on an NLS system.
  - For the timestamp data type. Text specifies a text-based timestamp in the form %Y%Y-mm-dd %hh:mm:ss or in the default date format if you have defined a new one on an NLS system.

- Field max width. The maximum number of bytes in a column represented as a string. Enter a number. This is useful where you are storing numbers as text. If you are using a fixed-width character set, you can calculate the length exactly. If you are using variable-length character set, calculate an adequate maximum width for your fields. Applies to fields of all data types except date, time, timestamp, and raw; and record, subrecord, or tagged if they contain at least one field of this type.

- Field width. The number of bytes in a field represented as a string. Enter a number. This is useful where you are storing numbers as text. If you are using a fixed-width character set, you can calculate the number of bytes exactly. If it's a variable length encoding, base your calculation on the width and frequency of your variable-width characters. Applies to fields of all data types except date, time, timestamp, and raw; and record, subrecord, or tagged if they contain at least one field of this type.

  If you specify neither field width nor field max width, numeric fields written as text have the following number of bytes as their maximum width:
  - 8-bit signed or unsigned integers: 4 bytes
- 16-bit signed or unsigned integers: 6 bytes
- 32-bit signed or unsigned integers: 11 bytes
- 64-bit signed or unsigned integers: 21 bytes
- single-precision float: 14 bytes (sign, digit, decimal point, 7 fraction, "E", sign, 2 exponent)
- double-precision float: 24 bytes (sign, digit, decimal point, 16 fraction, "E", sign, 3 exponent)

- Pad char. This property is ignored for output links.
- Character set. Specifies the character set. Choose from ASCII or EBCDIC. The default is ASCII. Applies to all data types except raw and ustring and record, subrecord, or tagged containing no fields other than raw or ustring.

**String**

These properties are applied to columns with a string data type, unless overridden at column level.
- Export EBCDIC as ASCII. Not relevant for output links.
- Import ASCII as EBCDIC. Select this to specify that ASCII characters are read as EBCDIC characters.

For ASCII-EBCDIC and EBCDIC-ASCII conversion tables.

**Decimal**

These properties are applied to columns with a decimal data type unless overridden at column level.
- Allow all zeros. Specifies whether to treat a packed decimal column containing all zeros (which is normally illegal) as a valid representation of zero. Select Yes or No. The default is No.
- Decimal separator. Specify the ASCII character that acts as the decimal separator (period by default).
- Packed. Select an option to specify what the decimal columns contain, choose from:
  - Yes. Specifies that the decimal fields contain data in packed decimal format (the default). This has the following sub-properties:
    - Check. Select Yes to verify that data is packed, or No to not verify.
    - Signed. Select Yes to use the existing sign when reading decimal fields. Select No to write a positive sign (0xf) regardless of the fields’ actual sign value.
  - No (separate). Specifies that they contain unpacked decimal with a separate sign byte. This has the following sub-property:
    - Sign Position. Choose leading or trailing as appropriate.
  - No (zoned). Specifies that they contain an unpacked decimal in either ASCII or EBCDIC text. This has the following sub-property:
    - Sign Position. Choose leading or trailing as appropriate.
  - No (overpunch). Specifies that the field has a leading or end byte that contains a character which specifies both the numeric value of that byte and whether the number as a whole is negatively or positively signed. This has the following sub-property:
    - Sign Position. Choose leading or trailing as appropriate.
- Precision. Specifies the precision of a packed decimal. Enter a number.
• Rounding. Specifies how to round the source field to fit into the destination decimal when reading a source field to a decimal. Choose from the following methods of rounding:
  – up (ceiling). Truncate source column towards positive infinity. This mode corresponds to the IEEE 754 Round Up mode. For example, 1.4 becomes 2, -1.6 becomes -1.
  – down (floor). Truncate source column towards negative infinity. This mode corresponds to the IEEE 754 Round Down mode. For example, 1.6 becomes 1, -1.4 becomes -2.
  – nearest value. Round the source column towards the nearest representable value. This mode corresponds to the COBOL ROUNDED mode. For example, 1.4 becomes 1, 1.5 becomes 2, -1.4 becomes -1, -1.5 becomes -2.
  – truncate towards zero. This is the default. Discard fractional digits to the right of the right-most fractional digit supported by the destination, regardless of sign. For example, if the destination is an integer, all fractional digits are truncated. If the destination is another decimal with a smaller scale, truncate to the scale size of the destination decimal. This mode corresponds to the COBOL INTEGER-PART function. Using this method 1.6 becomes 1, -1.6 becomes -1.
• Scale. Specifies the scale of a source packed decimal.

**Numeric**

These properties apply to integer and float fields unless overridden at column level.

• C_format. Perform non-default conversion of data from string data to a integer or floating-point. This property specifies a C-language format string used for reading integer or floating point strings. This is passed to `sscanf()`. For example, you specify a C-format of `%x` and a field width of 8, so that a 32-bit integer is formatted as an 8-byte hexadecimal string.

• In_format. Format string used for conversion of data from string to integer or floating-point data. This is passed to `sscanf()`. By default, InfoSphere DataStage invokes the C `sscanf()` function to convert a numeric field formatted as a string to either integer or floating point data. If this function does not output data in a satisfactory format, you can specify the in_format property to pass formatting arguments to `sscanf()`.

• Out_format. This property is not relevant for output links.

**Date**

These properties are applied to columns with a date data type unless overridden at column level. All of these are incompatible with a Data Format setting of Text.

Days since. Dates are written as a signed integer containing the number of days since the specified date. Enter a date in the form `%yyyy-%mm-%dd` or in the default date format if you have defined a new one on an NLS system.

Format string. The string format of a date. By default this is `%yyyy-%mm-%dd`. The Format string can contain one or a combination of the following elements:

• `%dd`: A two-digit day.
• `%mm`: A two-digit month.
• `%year_cutoffyy`: A two-digit year derived from `yy` and the specified four-digit year cutoff, for example `%1970yy`. 
- %yy: A two-digit year derived from a year cutoff of 1900.
- %yyyy: A four-digit year.
- %ddd: Day of year in three-digit form (range of 1-366).
- %mmm: Three-character month abbreviation.

The format_string is subject to the following restrictions:
- It cannot have more than one element of the same type, for example it cannot contain two %dd elements.
- It cannot have both %dd and %ddd.
- It cannot have both %yy and %yyyy.
- It cannot have both %mm and %ddd.
- It cannot have both %mmm and %ddd.
- It cannot have both %mm and %mmm.
- If it has %dd, it must have %mm or %mmm.
- It must have exactly one of %yy or %yyyy.

When you specify a date format string, prefix each component with the percent symbol (%). Separate the string’s components with any character except the percent sign (%).

If this format string does not include a day, it is set to the first of the month in the destination field. If the format string does not include the month and day, they default to January 1. Note that the format string must contain a month if it also contains a day; that is, you cannot omit only the month.

The year_cutoff is the year defining the beginning of the century in which all two-digit years fall. By default, the year cutoff is 1900; therefore, a two-digit year of 97 represents 1997. You can also set this using the environment variable APT_DATE_CENTURY_BREAK_YEAR, but this is overridden by %year_cutoffyy if you have set it.

You can specify any four-digit year as the year cutoff. All two-digit years then specify the next possible year ending in the specified two digits that is the same or greater than the cutoff. For example, if you set the year cutoff to 1930, the two-digit year 30 corresponds to 1930, and the two-digit year 29 corresponds to 2029.

Is Julian. Select this to specify that dates are written as a numeric value containing the Julian day. A Julian day specifies the date as the number of days from 4713 BCE January 1, 12:00 hours (noon) GMT.

**Time**

These properties are applied to columns with a time data type unless overridden at column level. All of these are incompatible with a Data Format setting of Text.

Format string. Specifies the format of columns representing time as a string. By default this is %hh-%mm-%ss. The possible components of the time format string are:
- %hh: A two-digit hours component.
- %mm: A two-digit minute component (mm represents minutes because mm is used for the month of a date).
• %ss: A two-digit seconds component.
• %ss.n: A two-digit seconds plus fractional part, where \( n \) is the number of fractional digits with a maximum value of 6. If \( n \) is 0, no decimal point is printed as part of the seconds component. Trailing zeros are not suppressed.

You must prefix each component of the format string with the percent symbol. Separate the string’s components with any character except the percent sign (%).

Is midnight seconds. Select this to specify that times are written as a binary 32-bit integer containing the number of seconds elapsed from the previous midnight.

**Timestamp**

These properties are applied to columns with a timestamp data type unless overridden at column level.

Format string. Specifies the format of a column representing a timestamp as a string. Defaults to %yyyy-%mm-%dd %hh:%nn:%ss. Specify the format as follows:

**Date:**
• %add: A two-digit day.
• %amm: A two-digit month.
• %year_cutoffyy: A two-digit year derived from yy and the specified four-digit year cutoff.
• %yy: A two-digit year derived from a year cutoff of 1900.
• %yyyy: A four-digit year.
• %ddd: Day of year in three-digit form (range of 1 - 366).

**Time:**
• %ahl: A two-digit hours component.
• %amm: A two-digit minute component (mm represents minutes because mm is used for the month of a date).
• %ss: A two-digit seconds component.
• %ss.n: A two-digit seconds plus fractional part, where \( n \) is the number of fractional digits with a maximum value of 6. If \( n \) is 0, no decimal point is printed as part of the seconds component. Trailing zeros are not suppressed.

You must prefix each component of the format string with the percent symbol (%). Separate the string’s components with any character except the percent sign (%).

**Output page columns tab**

The Output page always has a Columns tab. The Column page displays the column metadata for the selected output link in a grid.

There are the following various ways of populating the grid:
• If the other end of the link has metadata specified for it, the information is displayed in the Columns tab (metadata is associated with and travels with a link).
• You can type the required metadata into the grid. After you have done this, click **Save**... to save the metadata as a table definition in the metadata repository for subsequent reuse.
You can load an existing table definition from the metadata repository. Click Load... and you will see a choice of table definitions to load.

If the stage you are editing is a general or restructure stage with a Mapping tab, you can drag data from the left pane to the right pane. This automatically populates the right pane and the Columns tab.

If runtime column propagation is enabled in the DataStage Administrator, you can select the Runtime column propagation to specify that columns encountered by the stage can be used even if they are not explicitly defined in the metadata. There are some special considerations when using runtime column propagation with certain stage types:

- Sequential File
- File Set
- External Source
- External Target

See the individual stage descriptions for details of these.

If the selected output link is a reject link, the column metadata grid is read only and cannot be modified.

If you select the options in the Grid Properties window, the Columns tab will also display two extra fields: Table Definition Reference and Column Definition Reference. These show the table definition and individual columns from which the columns on the tab were derived.

If you click in a row and select Edit Row... from the shortcut menu, the Edit Column metadata window opens, which allows you to edit the row details. It also has a Parallel tab which allows you to specify properties that are particular to parallel job column definitions. The properties you can specify here are the same as those specified for input links.

**Mapping tab**

For processing and restructure stages, the Mapping tab specifies how the output columns are derived; that is, what input columns map onto them or how they are generated.

The left pane lists the input columns and the generated columns. These are read only and cannot be modified on this tab. These columns represent the data that the stage has produced after it has processed the input data.

The right pane lists the output columns for each link. You populate it by dragging input columns over, or by using the Auto-match facility. If you have not yet defined any output column definitions, populating the columns defines them for you. If you have already defined output column definitions, InfoSphere DataStage performs the mapping for you. You can do this explicitly using the auto-match facility, or implicitly by just visiting the Mapping tab and click OK (which is the equivalent of auto-matching on name).

There is also a shortcut menu which gives access to a range of column selection and editing functions, including the facilities for selecting multiple columns and editing multiple derivations.
You can choose not to map all the left hand columns, if your output data is a subset of your input data. Be aware that, if you have Runtime Column Propagation turned on for that link, the data you have not mapped will appear on the output link anyway.

You can also perform mapping without actually opening the stage editor. Select the stage in the Designer canvas and choose Auto-map from the shortcut menu.

More details about mapping operations for the different stages are given in the individual stage descriptions.

A shortcut menu can be invoked from the right pane that allows you to:
- Find and replace column names.
- Validate a derivation you have entered.
- Clear an existing derivation.
- Append a new column.
- Select all columns.
- Insert a new column at the current position.
- Delete the selected column or columns.
- Cut and copy columns.
- Paste a whole column.
- Paste just the derivation from a column.

The Find button opens a window that allows you to search for particular output columns.

The Auto-Match button opens a window that automatically maps left pane columns onto right pane columns according to the specified criteria.

Select Location match to map input columns onto the output ones occupying the equivalent position. Select Name match to match by names. You can specify that all columns are to be mapped by name, or only the ones you have selected. You can also specify that prefixes and suffixes are ignored for input and output columns, and that case can be ignored.

**Configure the Mapping tab**
The Mapping tab allows you to map input columns to the output.

**About this task**

It is not necessary to load column definitions into the output stage of your One-source matching job in advance. Instead, you use the Mapping tab to specify which columns are output.

You can specify which input columns to include in your output. Generally you should retain the columns added by the stage.

To map the output columns from the One-source Match stage, follow these steps.

**Procedure**

1. On the Output page, select an output link from the Output Name drop-down list.
2. Select the Mapping tab.
3. Drag each column you want to output from the Columns list on the left to the output link list on the right. You can also select a group of columns and drag them at once, or right-click and choose Select All.

   **Note:** When you drag columns in this manner to create output table definitions, the table definitions are not automatically saved in the metadata repository. For best metadata management, on the Columns tab of the Output page, click Save and save the newly created table definition.

4. Repeat this process for each output.

**Results**

Once you have set the stage properties, you are ready to save, compile and run the job.

**Advanced tab**

The Advanced tab allows you to specify how InfoSphere DataStage buffers data being output from this stage.

The InfoSphere DataStage buffers data in such a way that no deadlocks can arise. A deadlock being the situation where a number of stages are mutually dependent and are waiting for input from another stage, and cannot output until they have received it.

The size and operation of the buffer are usually the same for all links on all stages. The default values that the settings take can be set using environment variables.

The Advanced tab allows you to specify buffer settings on a per-link basis. You should only change the settings if you fully understand the consequences of your actions, otherwise you might cause deadlock situations to arise.

Any changes you make on this tab will automatically be reflected in the Input page, Advanced tab of the stage at the other end of this link.

The settings are as follows:

- **Buffering mode.** Select one of the following from the drop-down list.
  - (Default). This will take whatever the default settings are as specified by the environment variables (this will be Auto-buffer unless you have explicitly changed the value of the APT_BUFFERING _POLICY environment variable).
  - Auto buffer. Buffer output data only if necessary to prevent a dataflow deadlock situation.
  - Buffer. This will unconditionally buffer all data output from this stage.
  - No buffer. Do not buffer output data under any circumstances. This could potentially lead to deadlock situations if not used carefully.

If you choose the Auto buffer or Buffer options, you can also set the values of the various buffering parameters:

- **Maximum memory buffer size (bytes).** Specifies the maximum amount of virtual memory, in bytes, used per buffer. The default size is 3145728 (3 MB).
- **Buffer free run (percent).** Specifies how much of the available in-memory buffer to consume before the buffer resists. This is expressed as a percentage of Maximum memory buffer size. When the amount of data in the buffer is less
than this value, new data is accepted automatically. When the data exceeds it, the buffer first tries to write some of the data it contains before accepting more. The default value is 50% of the Maximum memory buffer size. You can set it to greater than 100%, in which case the buffer continues to store data up to the indicated multiple of Maximum memory buffer size before writing to disk.

- **Queue upper bound size (bytes).** Specifies the maximum amount of data buffered at any time using both memory and disk. The default value is zero, meaning that the buffer size is limited only by the available disk space as specified in the configuration file (resource scratchdisk). If you set Queue upper bound size (bytes) to a non-zero value, the amount of data stored in the buffer does not exceed this value (in bytes) plus one block (where the data stored in a block cannot exceed 32 KB).

If you set Queue upper bound size to a value equal to or slightly less than Maximum memory buffer size and set Buffer free run to 1.0. You will create a finite capacity buffer that will not write to disk. However, the size of the buffer is limited by the virtual memory of your system and you can create deadlock if the buffer becomes full.

- **Disk write increment (bytes).** Sets the size, in bytes, of blocks of data being moved to/from disk by the buffering operator. The default is 1048576 (1 MB). Adjusting this value trades amount of disk access against throughput for small amounts of data. Increasing the block size reduces disk access, but may decrease performance when data is being read/written in smaller units. Decreasing the block size increases throughput, but may increase the amount of disk access.
Appendix A. Product accessibility

You can get information about the accessibility status of IBM products.

The IBM InfoSphere Information Server product modules and user interfaces are not fully accessible. The installation program installs the following product modules and components:
- IBM InfoSphere Blueprint Director
- IBM InfoSphere Discovery
- IBM InfoSphere Metadata Workbench
- IBM InfoSphere Business Glossary
- IBM InfoSphere Business Glossary Anywhere
- IBM InfoSphere Information Analyzer
- IBM InfoSphere QualityStage
- IBM InfoSphere Information Services Director
- IBM InfoSphere DataStage
- IBM InfoSphere DataStage and QualityStage Designer
- IBM InfoSphere Data Click
- IBM InfoSphere FastTrack
- IBM InfoSphere Data Replication

For information about the accessibility status of IBM products, see the IBM product accessibility information at [http://www.ibm.com/able/product_accessibility/index.html](http://www.ibm.com/able/product_accessibility/index.html).

Accessible documentation

Accessible documentation for InfoSphere Information Server products is provided in an information center. The information center presents the documentation in XHTML 1.0 format, which is viewable in most web browsers. Because the information center uses XHTML, you can set display preferences in your browser. This also allows you to use screen readers and other assistive technologies to access the documentation.

The documentation that is in the information center is also provided in PDF files, which are not fully accessible.

IBM and accessibility

See the [IBM Human Ability and Accessibility Center](http://www.ibm.com/able) for more information about the commitment that IBM has to accessibility.
Appendix B. Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

Table 42. IBM resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description and location</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Support Portal</td>
<td>You can customize support information by choosing the products and the topics that interest you at <a href="http://www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server">www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server</a></td>
</tr>
<tr>
<td>Software services</td>
<td>You can find information about software, IT, and business consulting services, on the solutions site at <a href="http://www.ibm.com/businesssolutions/">www.ibm.com/businesssolutions/</a></td>
</tr>
<tr>
<td>My IBM</td>
<td>You can manage links to IBM Web sites and information that meet your specific technical support needs by creating an account on the My IBM site at <a href="http://www.ibm.com/account/">www.ibm.com/account/</a></td>
</tr>
<tr>
<td>Training and certification</td>
<td>You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at <a href="http://www.ibm.com/training">http://www.ibm.com/training</a></td>
</tr>
</tbody>
</table>
Appendix C. Accessing and providing feedback on the product documentation

Documentation is provided in a variety of locations and formats, including in help that is opened directly from the product client interfaces, in a suite-wide information center, and in PDF file books.

The information center is installed as a common service with InfoSphere Information Server information center. The information center contains help for most of the product interfaces, as well as complete documentation for all the product modules in the suite. You can open the information center from the installed product or from a web browser.

Accessing the information center

You can use the following methods to open the installed information center.

- Click the Help link in the upper right of the client interface.

  Note: From IBM InfoSphere FastTrack and IBM InfoSphere Information Server Manager, the main Help menu item opens a local help system. Choose Help > Open Info Center to open the full InfoSphere Information Server information center.

- Press the F1 key. The F1 key typically opens the topic that describes the current context of the client interface.

  Note: The F1 key does not work in web clients.

- Use a web browser to access the installed information center even when you are not logged in to the product. Enter the following address in a web browser:
  http://host_name:port_number/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.home.doc/topics/ic_homepage_IS.html

  where host_name is the name of the services tier computer where the information center is installed, and port_number is the port number for InfoSphere Information Server. The default port number is 9080. For example, on a Microsoft Windows Server computer named server1, that uses the default port, the web address is in the following format:

A subset of the information center is also available on the IBM website and periodically refreshed at http://pic.dhe.ibm.com/infocenter/iisinfsv/v9r1/index.jsp. This information center is the most up-to-date version and might include corrections, provided as comments.

Obtaining PDF and hardcopy documentation

- The PDF file books are available online and can be accessed from this support document: https://www.ibm.com/support/docview.wss?uid=swg27008803&wv=1
• You can also order IBM publications in hardcopy format online or through your local IBM representative. To order publications online, go to the IBM Publications Center at [http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss](http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss).

Providing comments on the documentation

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

• To provide a comment about the information center that is hosted on the IBM website, sign in and add a comment. Comments submitted this way are viewable by the public. See for more information.

• To send a comment about the information to IBM that is not viewable by anyone else, click the Feedback link on the top right side of any topic in the information center. You can do this from an information center that is installed with InfoSphere Information Server or from the information center that is available on the IBM website.

• Send your comments by using the online readers’ comment form at [www.ibm.com/software/awdtools/rcf/](http://www.ibm.com/software/awdtools/rcf/)

• Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the information (if applicable). If you are commenting on specific text, include the location of the text (for example, a title, a table number, or a page number).
Notices and trademarks

This information was developed for products and services offered in the U.S.A.

Notices

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web
sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to
IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**Privacy policy considerations**

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session or persistent cookies. If a product or component is not listed, that product or component does not use cookies.

**Table 43. Use of cookies by InfoSphere Information Server products and components**

<table>
<thead>
<tr>
<th>Product module</th>
<th>Component or feature</th>
<th>Type of cookie that is used</th>
<th>Collect this data</th>
<th>Purpose of data</th>
<th>Disabling the cookies</th>
</tr>
</thead>
</table>
| Any (part of InfoSphere Information Server installation) | InfoSphere Information Server web console | • Session  
• Persistent | User name                     | • Session management  
• Authentication                          | Cannot be disabled                        |
| Any (part of InfoSphere Information Server installation) | InfoSphere Metadata Asset Manager | • Session  
• Persistent | No personally identifiable information | • Session management  
• Authentication  
• Enhanced user usability  
• Single sign-on configuration | Cannot be disabled                        |
| InfoSphere DataStage                                | Big Data File stage             | • Session  
• Persistent | • User name  
• Digital signature  
• Session ID                        | • Session management  
• Authentication  
• Single sign-on configuration      | Cannot be disabled                        |
<table>
<thead>
<tr>
<th>Product module</th>
<th>Component or feature</th>
<th>Type of cookie that is used</th>
<th>Collect this data</th>
<th>Purpose of data</th>
<th>Disabling the cookies</th>
</tr>
</thead>
<tbody>
<tr>
<td>InfoSphere DataStage</td>
<td>XML stage</td>
<td>Session</td>
<td>Internal identifiers</td>
<td>• Session management</td>
<td>Cannot be disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Authentication</td>
<td></td>
</tr>
<tr>
<td>InfoSphere DataStage</td>
<td>IBM InfoSphere DataStage and QualityStage Operations Console</td>
<td>Session</td>
<td>No personally identifiable information</td>
<td>• Session management</td>
<td>Cannot be disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Authentication</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Single sign-on configuration</td>
<td></td>
</tr>
<tr>
<td>InfoSphere Data Quality Console</td>
<td></td>
<td>Session</td>
<td>No personally identifiable information</td>
<td>• Session management</td>
<td>Cannot be disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Authentication</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Single sign-on configuration</td>
<td></td>
</tr>
<tr>
<td>Information Governance Catalog</td>
<td>InfoSphere blueprint Director, InfoSphere Business Glossary, InfoSphere Metadata Workbench</td>
<td>• Session, Persistent</td>
<td>• Internal identifiers, State of the tree</td>
<td>• Session management</td>
<td>Cannot be disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Authentication</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Enhanced user usability</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Single sign-on configuration</td>
<td></td>
</tr>
<tr>
<td>InfoSphere Information Analyzer</td>
<td>Data Rules stage in the InfoSphere DataStage and QualityStage Designer client</td>
<td>Session</td>
<td>Session ID</td>
<td>Session management</td>
<td>Cannot be disabled</td>
</tr>
</tbody>
</table>

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.


**Trademarks**

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

The following terms are trademarks or registered trademarks of other companies:
Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The United States Postal Service owns the following trademarks: CASS, CASS Certified, DPV, LACSLink, ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS and United States Postal Service. IBM Corporation is a non-exclusive DPV and LACSLink licensee of the United States Postal Service.

Other company, product or service names may be trademarks or service marks of others.
Index

A
ABS_DIFF comparison 83
actions overview 173
Standardization Rules Designer 230
adding domain preprocessor literals 29
ADDR domain 188
address matching examples 55
Advanced tab Input page 290
Output page 303
Stage page 262
agreement weights 63
AN_DINT comparison 84
AN_INT comparison 84
AREA domain 188
asset interchange commands 202
B
baseline test runs match passes 128
best record 157
blocking defined 59
examples 62
strategies 59
blocking columns choosing columns 62
overview 60
specifying 76
business intelligence fields domain-specific rule sets 191
VDATE rule sets 195
VERMAIL rule sets 196
VPHONE rule sets 199, 200
C
CHAR comparison 85
character concatenate investigate 7
corresponding investigate 7
Character investigate 12
Character investigation column masks 14
setting up 12
classifications .CLS file 167
adding in the Standardization Rules
Designer 225
affect on pattern rules 224
custom classes 165
default classes 165
definitions 166
file format for Standardization Rules
Designer 226
importing in the Standardization
Rules Designer 225
literals 167
classifications (continued)
overview 164
Standardization Rules Designer 223
strategies for classification 211
cleansing data introduction 1
CNT_DIFF comparison 86
Column Mask Selection window 12
column masks 14
changing 13
Columns tab 300
columns tab, Input page 277
columns comparisons 84
ABS_DIFF 83
AN_INTERVAL 84
CHAR 85
CNT_DIFF 86
D_INT 86
D_USPS 87
DATES 88
DELTA_PERCENT 89
DISTANCE 90
INT_TO_INT 91
INTERVAL_NOPAR 92
INTERVAL_PARITY 93
LR_CHAR 93
MULT_ALIGN 95
MULT_EXACT 98
MULT_RANGE 98
NAME_UNCERT 100
NUMERIC 101
PREFIX 101
PRORATED 102
TIME 103
UNCERT 104, 105
USPS 105
USPS_DINT 106
USPS_INT 108
composite weights 63
about 63
histogram 125
conditions overview 173
Standardization Rules Designer 229
configure
Investigate stage 12
Survive stage 133
Console for InfoSphere Information
Server running reports 21
country identifier rule set delimiter 181
country or region identifier listing 237
literal 180
rule set 180
unknown 180
COUNTRY rule set description 180
COUNTRY rule sets output data 181
creating simple survive rules 154
current record 157
customer support contacting 307
cutoffs adjusting 125
assigning 114
clerical 65
match 65
D
D_INT comparison 86
D_USPS comparison 87
data types date 271
decimal 271
default 271
string 271
date formats, input 194
DATE8 comparison 88
DB2 partition properties 267
default field values
columns read from files 294
columns written to files 269
default parsing parameters
VDATE rule set 194
VERMAIL rule sets 196
VPHONE rule sets 198
VTAXID rule sets 200
defining simple survive rules 154
delimiters
domain preprocessor rule sets 182
DELTA_PERCENT comparison 89
dependent match type
One-source Match stage 139
dictionaries
business intelligence fields 191
domain-specific 191
matching columns 192
overview 170
reporting columns 192
disagreement weights 63
DISTANCE comparison 90
domain preprocessor (PREP) rule
sets 182
add a pattern override 215
add text override 216
addition of literals 29
assigning tokens to a domain
mask 187
change a token override code 215
dictionary content 185
domain fields and columns 186
field pattern override 215
field text override 216
input pattern override 215
naming conventions 184
preparing input data 182
user flag descriptions 187
value 184

© Copyright IBM Corp. 2004, 2013

317
domain-preprocessor (PREP) rule sets 186
  domain-specific rule sets 188
country or region list 237
data flags 192
description 188
expected input 190
file names 189
input text override 218
overrides 189
test 236
unhandled text override 218

e
environment variables
  QSM_MAT_UNCERT_VERSION 104, 105
error reporting fields
  VDATE rule sets 195
  VEMAIL rule sets 197
  VPHONE rule sets 199
  VTAXID rule sets 201
evaluate records 157
examples
  blocking columns 62
  computation of weight 65
deduplication 57
  individual matching 54
  many-to-one matching 55
  match passes 59

F
field defaults
  Input page 269
  Output page 294
file names
  domain specific 189
Format tab
  Input page 267, 292
frequency cutoff 15
frequency data
  u probabilities 66
frequency distributions
  Match Frequency stage 132
frequency information
  viewing 112
Frequency/Weight histogram using 125
fuzzy matches 63

G
General tab 259, 265, 292
Byte order 295
Data format 295
Input page 265
Output page 292
Stage page 259

H
holding area
  Match Designer 115

I
independent match type
  One-source Match stage 139
Indian address rule sets
  regions 245
  shared container 246
InfoSphere DataStage
  stage page 259
InfoSphere Information Server for Web console
  Investigation reports 19
InfoSphere QualityStage
  Investigate stage 7
  Investigation reports 19
  Rules Management window 201
  source data preparation 23
  standardization requirements 210
  Standardize stage 28
  Survive stage 152
InfoSphere QualityStage jobs
  set up
  Investigate stage 10
InfoSphere QualityStage reports 19
Input page 264
  Advanced tab 290
  columns tab 277
  field defaults 269
  format tab 267
  Format tab 292
  General tab 265
  partitioning tab 265
  Properties tab 265
  Record level 268
  Stage Editor 264
input pattern override 217
input text and unhandled text override add 218
INT_TO_INT comparison 91
  INTERVAL_NOPAR comparison 92
  INTERVAL_PARTY comparison 93
introduction
  data cleansing 1
  Investigate Advanced Options window 12, 14
Investigate stage
  advanced options for Word investigation 15
  changing column mask 13
  character concatenate investigate 7
  character discrete investigate 7
  Character investigation, setting up 12
  configure 12
  create a token or pattern report 9
  creating reports 20
  domain-specific rules sets 188
  frequency cutoff 15
  Investigate Advanced Options window 12, 14
  Investigation reports 19
  ordering links 16
  overview 7
  pattern report 19
  removing column from investigation 13
  reports 19
  rule sets 8
Investigate stage (continued)
  set up Word investigation 14
  setting stage properties 17
  setting up job 10
  stage properties 16, 18
  token report 19
  values 8
  Word investigation using 8
Investigate Stage
  Column Mask Selection window 12
columns masks 14
pattern reports 15
Investigation reports
  pattern 19
  running reports 21
token 19
Investigation reports, types of 19
investigation reports creating reports 20
ISO territory or region codes 247
istool import 202

J
Japanese rule sets
  expected input and characteristics 244
jobs
  predefined
  matching 147
  standardization 32

L
legal notices 311
Level Number field 277
link ordering 16
link ordering tab
  Stage page 263
link ordering tab, stage page 263
literals
  adding with domain preprocessor rule sets 29
  literals in input records 27
lookup tables
  adding in the Standardization Rules Designer 227
definitions 169
file format for Standardization Rules Designer 228
importing in the Standardization Rules Designer 228
overview 168
Standardization Rules Designer 227
LR_CHAR comparison 93
LR_UNCERT 94
LR_UNCERT comparison 94
match comparisons 94

M
m probabilities
  about 64
many-to-one duplicate match type
  overview 146
