IBM InfoSphere DataStage and QualityStage
Version 8 Release 7

Java Pack Guide

SC19-3455-00

IBM InfoSphere DataStage and QualityStage
Version 8 Release 7

Java Pack Guide

SC19-3455-00

Note

Before using this information and the product that it supports, read the information in “Notices and trademarks” on page 47.

Contents

Chapter 1. Introduction

Using the Java Pack API, you can create Java programs that interface with IBM® InfoSphere® DataStage® and QualityStage™ Designer server and parallel jobs. Your Java program can:

- Produce (write) rows that are used within the job. In this scenario, the Java program is a source in the job.
- Consume (read) rows that are supplied on an input link. In this scenario, the Java program is a target in the job.
- Process rows from an input link and generate rows on the output link. In this scenario, the Java program acts as a transformer.
- Query column and stage metadata.

Java Pack stages

The Java Pack includes two stages through which you integrate your Java applications in an IBM InfoSphere DataStage job flow:

- Java Transformer stage
- Java Client stage

Note: For more information about active and passive stages, see the *InfoSphere DataStage Designer Client Guide*.

Java Transformer stage

The Java Transformer stage is an active stage. Use it to call a Java application that transforms data from an input link and writes the data to an output link in a job.

Java Client stage

The Java Client stage is a passive stage. Use it in three configurations.

As a source stage

Use the Java Client stage as a source stage if your Java application does not need input data or when the input data can be supplied through a stage property or a job parameter.

As a target stage

Use the Java Client stage as a target stage if your Java application does not produce any output data or if the output is external to InfoSphere DataStage.

As a lookup stage

Your Java application performs lookup functions for a built-in Transformer stage. The Java Client stage is connected to the built-in Transformer stage using a reference link.

Your Java program and Java Pack

Your Java program includes three parts:
- initialization
- processing
- termination

Initialization

When a Java Pack stage starts, the stage instantiates your Java program and calls the logic within the `Stage.initialize()` method.

Typical actions

Action	See
Read column metadata.	"Metadata methods" on page 13
Read stage metadata.	Properties methods

Processing

The stage calls the `Stage.process()` method in three circumstances:
- Each time an input row arrives.
- When there are no rows to process, the logic is called once.
- When there are no more rows to process.

Typical actions

1. Read the incoming row. See readRow() method.
2. Extract the values of its columns, as binary, string, or typed. See Reader methods.
3. Process values, as needed. Use your own code in this step.
4. Create an output row. See createOutputRow() method.
5. Fill its columns with the result of the processing, as binary, string, or typed values. See Writer methods.
6. Write the output row. See writeRow() method.

Termination

The stage calls the `Stage.terminate()` method when the stage ends.

Typical actions

- Any cleanup actions

Class summary

This section lists the major methods within the classes of the `com.ascentialsoftware.jds` package.

Invocation methods

`Stage.initialize()`

`Stage.process()`

`Stage.terminate()`

Column Methods

Management:

`Column.getValueAsRaw()`

`Column.getValueAsString()`

`Column.getValueAsSQLTyped()`

`Column.setValueAsRaw()`

`Column.setValueAsString()`

`Column.setValueAsSQLTyped()`

Metadata:

`Column.getDataElementName()`

`Column.getDescription()`

`Column.getDerivation()`

`Column.getIndex()`

`Column.getName()`

`Column.getSQLDisplayWidth()`

`Column.getSQLPrecision()`

`Column.getSQLScale()`

`Column.getSQLType()`

`Column.getSQLTypeName()`

`Column.isKey()`

`Column.nullAllowed()`

Logging Methods

`Stage.fatal()`

`Stage.info()`

`Stage.isTraceOn()`

`Stage.trace()`

`Stage.warn()`

Row Methods

Management

`Row.getValueAsRaw()`

`Row.getValueAsString()`

`Row.getValueAsSQLTyped()`

`Row.setValueAsRaw()`

`Row.setValueAsString()`

`Row.setValueAsSQLTyped()`

`Stage.createOutputRow()`

`Stage.createRejectRow()`

`Stage.readRow()`

`Stage.rejectRow()`

`Stage.writeRow()`

Metadata

`Row.getColumn()`

`Row.getColumnCount()`

Stage Methods

Application parameters

`Stage.getUserProperties()`

Links

`Stage.hasInputLink()`

`Stage.hasOutputLink()`

`Stage.hasReferenceLink()`

`Stage.hasRejectLink()`

Chapter 2. Java Pack API

The Java Pack API package is: `com.ascentialsoftware.jds`.

The package consists of three public classes:

Public class	For more information
`com.ascentialsoftware.jds.Column`	Column Class
`com.ascentialsoftware.jds.Row`	Row Class
`com.ascentialsoftware.jds.Stage`	Stage Class

Structure of your Java program

Your Java program must implement a subclass of the `Stage` class. The `Stage` class consists of methods for manipulating rows and querying metadata.

The Stage class is the root of all classes that implement a Java Client or Java Transformer stage in your jobs.
- Initialization is achieved by overriding the `Stage.initialize()` method. This is optional.
- Processing is achieved by overriding the `Stage.process()` method.
- Termination is achieved by overriding the `Stage.terminate()` method. This is optional.

The following example shows the skeleton of a simple program. In a single call of the `process()` method, one input row is consumed and processed, and one output row is produced.

```
public class Mytransformer extends Stage
{
   public void initialize()
      {
         //          ...initialize logic
      }
   public void terminate()
      {
         //         ...terminate logic
      }
   public int process()
      {
         Row inputRow = readRow();
         //         ...process input row...
         Row outputRow = createOutputRow();
         //         ...fill output row...
         writeRow(outputRow);
         return OUTPUT_STATUS_READY;
      }
}
```

Stage class

The following sections describes major Stage methods. For descriptions of other Stage methods, see Other Java Pack API methods and the API documentation.

Deployment

Deploy your compiled class or JAR file in a directory that is accessible from your engine tier.

Invocation methods

This section describes the following invocation methods:
- `initialize()`
- `process()`
- `terminate()`

initialize() method

```
public void initialize()
```

The Java Pack API calls the `initialize()` method when a Java Client or Java Transformer stage starts. Override the `initialize()` method in your Stage subclass if you need to perform actions before processing any input or output rows. Examples include setting counters, reading user properties in the Java Pack stages, and opening database connections.

process() method

```
public int process()
```

The `process()` method is the entry point for processing input and output rows. Override the `process()` method in your Stage subclass.

The Java Pack API calls the `process()` method every time an event occurs:
- Stage initialization is complete.
- An input row arrives.
- An end-of-data or end-of-transmission is received.

A single `process()` call can handle the complete set of input records. However, if you want to monitor, through InfoSphere DataStage Director, the number of records that are processed within a timeframe, it is recommended that `process()` returns frequently.

Reading rows

In your `process()` implementation, you can call the `readRow()` method to read input rows. The effect of not calling the `readRow()` method before returning from the `process()` method depends on which Java Pack stage you use and how it is deployed in your job.

Stage / Deployment	Effect of Omitting a readRow()
Java Transformer	InfoSphere DataStage stops the job and writes the following message to the job log: `Deadlock detected: all input links blocked`
Java Client (target)	The input row is discarded.
Java Client (lookup)	The active stage that is connected to the Java Client stage uses any rows that the Java Client writes on the link.

Default Implementation

The default `process()` implementation reads the next available input row and writes its contents to the job log. These log entries are created only when tracing has been activated for the job through InfoSphere DataStage Director.

Returns

The `process()` method must return one or more of these status fields:

- `OUTPUT_STATUS_END_OF_DATA`

  Indicates that no rows have been written on any output link and that no more rows will be written until the end of the job execution because no more rows are available.
- `OUTPUT_STATUS_NOT_READY`

  Indicates that no rows have been written on any output link. The Java Client stage, when used as a source stage, will call the `process()` method again until `OUTPUT_STATUS_END_OF_DATA` is returned from your `process()` method.
- `OUTPUT_STATUS_READY`

  Indicates that at least one row has been successfully written on an output link.

terminate() method

```
public void terminate()
```

The Java Pack API calls the `terminate()` method when there are no more rows for the Java Client stage to produce or consume. The default implementation performs no processing. Override the `terminate()` method in your Stage subclass if you need to perform cleanup actions, such as closing a `PrintWriter`.

Any exception that might be thrown by this method will stop the job that runs the stage. The message of the exception is recorded in the job log.

Row management methods

This is a list of row management methods:

- `createInputRow()`
- `createOutputRow()`
- `createRejectRow()`
- `readRow()`
- `rejectRow()`
- `writeRow()`

createInputRow() method

```
public Row createInputRow()
```

The `createInputRow()` method creates a row object from which you can access input link information.

Returns

A new empty row associated with the input link or `null`, if no input link is connected to the stage.

createOutputRow() method

```
public Row createOutputRow()
```

The `createOutputRow()` method creates an empty row on an output link, which is populated with the `writeRow()` method.

Returns

A new empty row associated with the output link or `null`, if no output link is connected to the stage.

createRejectRow() method

```
public Row createRejectRow()
```

The `createRejectRow()` method is available for a Java Transformer stage to create a row that is destined for a Reject link.

Returns

A new empty row associated with the Reject link or `null`, if no Reject link is connected to the Java Transformer stage.

readRow() method

```
public Row readRow() throws LinkErrorException,
                            LinkNotReadyException
```

The `readRow()` method reads the next available row from an input link or reference link. Call this method each time the `process()` method is invoked.

Returns

The next available row or `null`. When `readrow()` returns `null`, it signals two possibilities:

- No more rows are available and `process()` will not be called again.
- It is the end of the current transmission in a InfoSphere Information Services Director job. The job will be called again if there are additional service requests.

Throws

Exception	Description
`LinkErrorException`	Thrown when the stage has no input link or reference link, or if there was an unexpected failure while getting a new row. In the latter case, you must return from the `process()` method and the job will be stopped. This runtime exception does not need to be explicitly caught.

Exception	Description
`LinkNotReadyException`	Thrown when the input link is not ready to get new rows. To avoid throwing this exception with a passive stage, do not call `readRow()` more than once in a single invocation of the `process()` method. Although this runtime exception does not need to be explicitly caught, the `process()` method will exit and will be called again when the link is ready.

rejectRow() method

```
public void rejectRow(Row row) throws LinkErrorException,
                                      LinkNotReadyException
```

The `rejectRow()` method writes a row to a Reject link, if it exists. Applies only to the Java Transformer stage.

Parameters

Parameter	Description
`row`	Set of column values to be sent to the Reject link. If `null`, the `rejectRow()` method returns immediately and the presence of a Reject link is not tested.

Throws

Exception	Description
`LinkErrorException`	Thrown when there is no Reject link or if there was an unexpected failure while writing a new row. In the latter case, you must return from the `process()` method and the job will be aborted.
`LinkNotReadyException`	Thrown when the Reject link is not ready to write new rows. In this case, the `process()` method must return. Then InfoSphere DataStage can reset the Reject link.

writeRow() method

```
public void writeRow(Row row) throws LinkErrorException,
                                     LinkNotReadyException
```

The `writeRow()` method writes a row on an output link or to an external target.

Parameters

Parameter	Description
`row`	Set of column values to be sent through the output link. If null, the `writeRow()` method returns immediately; the presence of an output link is not tested.

Throws

Exception	Description
`LinkErrorException`	Thrown when there is no output link or reference link, or if there is an unexpected failure while writing a new row. In the latter case, you must return from the `process()` method and the job will be stopped.
`LinkNotReadyException`	Thrown when the output link is not ready to put new rows. In this case, the `process()` method must return. Then InfoSphere DataStage can reset the output link.

Logging methods

This list names five types of logging methods:

- `fatal()`
- `info()`
- `trace()`
- `warn()`
- `isTraceOn()`

fatal() method

```
public void fatal(String message)
```

The `fatal()` method logs a user-defined error message in the InfoSphere DataStage log. To stop the job, have the `process()` method return immediately after logging the message.

Parameters

Parameter	Description
`message`	The error message that you want to send to the InfoSphere DataStage job log.

info() method

```
public void info(String message)
```

The `info()` method logs an information message in the InfoSphere DataStage log.

Parameters

Parameter	Description
`message`	The information message that you want to send to the InfoSphere DataStage log.

trace() method

```
public void trace(String message)
```

The `trace()` method logs an information message in the InfoSphere DataStage log only if tracing has been activated in InfoSphere DataStage Director. Use this method when you are debugging a Java Pack stage. To avoid building a complex message structure when tracing is not activated, call the `isTraceOn()` method first.

Parameters

Parameter	Description
`message`	The information message that you want to send to the InfoSphere DataStage log.

warn() method

```
public void warn(String message)
```

The `warn()` method logs a warning message in the InfoSphere DataStage log.

Parameters

Parameter	Description
`message`	The warning message that you want to send to the InfoSphere DataStage log.

isTraceOn() method

```
public boolean isTraceOn()
```

The `isTraceOn()` method queries the current state of tracing in InfoSphere DataStage Director. Use this method to avoid building complex message structures when tracing is not activated.

Returns

- `true`, if tracing has been activated.
- `false`, if tracing has not been activated.

Properties methods

This list describes the following methods that perform queries on stage properties.

- `getUserProperties()`
- `hasInputLink()`
- `hasOutputLink()`
- `hasReferenceLink()`
- `hasRejectLink()`

getUserProperties() method

```
public String getUserProperties()
```

The `getUserProperties()` method fetches one application parameter string that is defined in a Java Client or Java Transformer stage.

The parameter string has a free format. For example, it can contain job parameters, Java properties, an XML document, or the path name of a file containing properties or a document.

Returns

String containing the user properties. It might be empty or null, if no properties were stored in the stage.

hasInputLink() method

```
public final boolean hasInputLink()
```

The `hasInputLink()` method tests for the presence of an input link.

One input link is supported by the Java Transformer stage and by the Java Client stage when it is deployed as a target stage.

Returns

- `true`, if the stage has an input link.
- `false`, if the stage does not have an input link.

hasOutputLink() method

```
public final boolean hasOutputLink()
```

The `hasOutputLink()` method tests for the presence of an output link.

One output link is supported by the Java Transformer stage and by the Java Client stage when it is deployed as a source stage.

Returns

- `true`, if the stage has an output link.
- `false`, if the stage does not have an output link.

hasReferenceLink() method

```
public final boolean hasReferenceLink()
```

The `hasReferenceLink()` method tests for the presence of a reference link.

A Java Client stage that performs lookups is connected to a built-in Transformer stage through a reference link.

Returns

- `true`, if the stage has a reference link.
- `false`, if the stage does not have a reference link.

hasRejectLink() method

```
public boolean hasRejectLink()
```

The `hasRejectLink()` method tests for the presence of a Reject link.

Only the Java Transformer stage supports a Reject link.

Returns

- `true`, if the stage has a Reject link.
- `false`, if the stage does not have a Reject link.

Column class

About this task

Use the methods of the Column class to get column metadata and to get or set values in specific columns. All the rows of an input or output link share the same column metadata. You can reuse the column object across multiple rows of the same link.

For information about getting or setting values using Column class methods, see Other Java Pack API methods and the API documentation

To get column metadata:

Procedure

1. Do one of the following steps:

   Note: All methods described in this step are in the Stage class.

   a. On an input link, use the row object that is returned by the `readRow()` method or call the `createInputRow()` method.
   b. On an output link, use the row object that is returned by the `createOutputRow()` method.
   c. On a Reject link, use the row object that is returned by the `createRejectRow()` method.
2. Get the column object using the `Row.getColumn()` method.
3. Use one of the methods described in “Metadata methods” on page 16.

Metadata methods

This list describes the following methods:

- `getDataElementName()`
- `getDescription()`
- `getDerivation()`
- `getIndex()`
- `getName()`
- `getSQLDisplayWidth()`
- `getSQLPrecision()`
- `getSQLScale()`
- `getSQLType()`
- `getSQLTypeName()`
- `isKey()`
- `nullAllowed()`

getDataElementName() method

```
public abstract String getDataElementName()
```

The `getDataElementName()` gets the name of the type of data element in a column.

Returns

Name of the type of data element in the column.

getDescription() method

```
public abstract String getDescription()
```

The `getDescription()` method gets the text description of a column.

Returns

Text description of the column.

getDerivation() method

```
public abstract String getDerivation()
```

The `getDerivation()` method gets the expression specifying how the data of a column is aggregated.

Returns

Expression that specifies how the data of this column is aggregated.

getIndex() method

```
public abstract int getIndex()
```

The `getIndex()` method gets the position of a column in a row.

Returns

Position of the column, in the range 0 to `Row.getColumnCount()-1`.

getName() method

```
public abstract String getName()
```

The `getName()` method gets the name of the column.

Returns

Name of the column.

getSQLDisplayWidth() method

```
public abstract int getSQLDisplayWidth()
```

The `getSQLDisplayWidth()` method gets the maximum number of characters required to display a column's data.

Returns

Maximum number of characters required to display the column's data.

getSQLPrecision() method

```
public abstract int getSQLPrecision()
```

The `getSQLPrecision()` method gets the data precision of a column.

- For `SQL_TYPE_CHAR` data, it is the length.
- For `SQL_TYPE_VARCHAR` data, it is the maximum length.

Returns

Data precision of the column.

getSQLScale() method

```
public abstract int getSQLScale()
```

The `getSQLScale()` method gets the data scale factor of a column.

Returns

Data scale factor of the column.

getSQLType() method

```
public abstract int getSQLType()
```

The `getSQLType()` method gets the SQL type of data in the column. Supported data types include:

- `SQL_TYPE_BIGINT`
- `SQL_TYPE_BINARY`
- `SQL_TYPE_BIT`
- `SQL_TYPE_CHAR`
- `SQL_TYPE_DATE`
- `SQL_TYPE_DECIMAL`
- `SQL_TYPE_DOUBLE`
- `SQL_TYPE_FLOAT`
- `SQL_TYPE_INTEGER`
- `SQL_TYPE_LONGVARBINARY`
- `SQL_TYPE_LONGVARCHAR`
- `SQL_TYPE_NUMERIC`
- `SQL_TYPE_REAL`
- `SQL_TYPE_SMALLINT`
- `SQL_TYPE_TIME`
- `SQL_TYPE_TIMESTAMP`
- `SQL_TYPE_TINYINT`
- `SQL_TYPE_UNKNOWN`
- `SQL_TYPE_VARBINARY`
- `SQL_TYPE_VARCHAR`
- `SQL_TYPE_WCHAR`
- `SQL_TYPE_WLONGVARCHAR`
- `SQL_TYPE_WVARCHAR`

Returns

SQL type of the column.

getSQLTypeName() method

```
public static String getSQLTypeName(int sqlType)
```

The `getSQLTypeName()` method gets the InfoSphere DataStage name of a given column's SQL data type. This method is for tracing purposes.

Parameters

Parameter	Description
`sqlType`	One of the values returned by the `getSQLType()` method.

Returns

The InfoSphere DataStage name of the column's SQL type.

isKey() method

```
public abstract boolean isKey()
```

The `isKey()` method indicates whether a column is part of the primary key.

Returns

- `true`, if this column is part of the primary key.
- `false`, if this column is not part of the primary key.

nullAllowed() method

```
public abstract boolean nullAllowed()
```

The `nullAllowed()` method indicates whether this column can contain null values.

Returns

- `true`, if this column can contain null values
- `false`, if this column cannot contain null values.

Row class

Use the methods of the Row class to get column meta data and to get and set column values.

Metadata methods

This section describes the following methods:

- `getColumn()`
- `getColumnCount()`

getColumn() method

```
public abstract Column getColumn(int index)
```

The `getColumn()` method gets metadata for a column. Use the metadata methods of the Column class to query the metadata.

Parameters

Parameter	Description
`index`	The position of the column in the row.

Returns

Object that contains all of the column's metadata.

Throws

Exception	Description
`IndexOutOfBoundsException`	Thrown when the index is not valid. This runtime exception does not need to be explicitly caught.

getColumnCount() method

```
public abstract int getColumnCount()
```

The `getColumnCount()` method gets the number of columns in a row.

Returns

Number of columns in a row.

Reader methods

This section describes the following methods:
- `getValueAsRaw()`
- `getValueAsString()`
- `getValueAsSQLTyped()`

getValueAsRaw() method

```
public abstract byte[] getValueAsRaw(int index)
```

The `getValueAsRaw()` method gets the raw (array of bytes) value corresponding to a given column in a row. Use this method with columns that contain binary values (Binary, LongVarBinary, and VarBinary).

Parameters

Parameters	Description
`index`	The column that contains the value to extract from the row. The first column is 0.

Returns

The raw value found at the given column or `null`, if no value has been assigned.

For performance reasons, the array of bytes returned by this method is not a copy of the original one. Therefore, any modification made to the returned object or the original value (using the `setValueAsRaw()` method) will change the referenced value.

To duplicate the value, use the `java.lang.System.arraycopy()` method.

Throws

Exception	Description
`IndexOutOfBoundsException`	Thrown when the index is not valid. This runtime exception does not need to be explicitly caught.

getValueAsString() method

```
public abstract String getValueAsString(int index)
public abstract String getValueAsString(int index, String charsetName)
```

The `getValueAsString()` method gets the string value corresponding to a given column in a row.

Parameters

Parameter	Description
`index`	The column that contains the value to extract from the row. The first column is 0.
`charsetName`	The name of a supported charset. If `null`, the default charset of the engine tier is used. To find out the list of charsets supported by your Java Virtual Machine (JVM), call the following method: `java.nio.charset.Charset.available Charsets()`

Returns

The string value found at the given column or null, if no value has been assigned.

Throws

Exception	Description
`IndexOutOfBoundsException`	Thrown when the index is not valid. This runtime exception does not need to be explicitly caught.
`UnsupportedEncodingException`	Thrown when the specified charset is not supported.

getValueAsSQLTyped() method

```
public abstract Object getValueAsSQLTyped(int index)
                             throws NumberFormatException,
                              ParseException
```

The `getValueAsSQLTyped()` method gets the value corresponding to a given column in this row, according to its SQL type.

The following table shows the correspondence between the SQL type in the column and the yielded object.

SQL Type	Java Object
Column.SQL_TYPE_BIGINT	Long
Column.SQL_TYPE_BINARY	byte[]
Column.SQL_TYPE_BIT	Boolean
Column.SQL_TYPE_CHAR	String
Column.SQL_TYPE_DATE	java.sql.Date
Column.SQL_TYPE_DECIMAL	BigDecimal
Column.SQL_TYPE_DOUBLE	Double
Column.SQL_TYPE_FLOAT	Float
Column.SQL_TYPE_INTEGER	Integer
Column.SQL_TYPE_LONGVARBINARY	byte[]
Column.SQL_TYPE_LONGVARCHAR	String
Column.SQL_TYPE_NUMERIC	BigDecimal
Column.SQL_TYPE_REAL	Float
Column.SQL_TYPE_SMALLINT	Short
Column.SQL_TYPE_TIME	java.sql.Time
Column.SQL_TYPE_TIMESTAMP	java.sql.Timestamp
Column.SQL_TYPE_TINYINT	Byte
Column.SQL_TYPE_UNKNOWN	String
Column.SQL_TYPE_VARBINARY	byte[]
Column.SQL_TYPE_VARCHAR	String
Column.SQL_TYPE_WCHAR	String
Column.SQL_TYPE_WLONGVARCHAR	String
Column.SQL_TYPE_WVARCHAR	String

Parameters

Parameter	Description
index	The column that contains the value to extract from the row. The first column is 0.

Returns

The object found in this column or `null`, if no value has been assigned.

Throws

Exception	Description
IndexOutOfBoundsException	Thrown when the index is not valid. This runtime exception does not need to be explicitly caught.
NumberFormatException	Thrown when parsing a numeric value fails.

Exception	Description
`ParseException`	Thrown when parsing a date, time, or time stamp value fails.

Notes

- For performance reasons, the array of bytes returned by this method for binary types is not a copy of the original one. Therefore any modification made to either one will change the referenced value.

  If needed, use the `java.lang.System.arraycopy()` method to duplicate the value.
- For Boolean types, string values are parsed in order to get a numeric equivalent. The method will return `false` for 0 values and `true` for any other number.

Writer methods

This list describes the following methods:

- `setValueAsRaw()`
- `setValueAsString()`
- `setValueAsSQLTyped()`

setValueAsRaw() method

```
public abstract void setValueAsRaw(int index, byte[] value)
```

The `setValueAsRaw()` method sets a raw (array of bytes) value at a specified column in a row.

Parameters

Parameter	Description
`index`	The column in the row that will contain the value. The first column is 0.
`value`	The raw value to insert in this row. For performance reasons, this method does not copy the original array of bytes. Therefore any modification to the column or the original array changes the referenced value. To duplicate the value, use the `java.lang.System.arraycopy()` method.

Throws

Exception	Description
`IndexOutOfBoundsException`	Thrown when the index is not valid. This runtime exception does not need to be explicitly caught.

setValueAsString() method

```
public abstract void setValueAsString (int index,
                                       String value)
public abstract void setValueAsString (int index,
                                       String value,
                                       String charset)
```

The `setValueAsString()` method sets a string value at a specified column in a row.

Parameters

Parameter	Description
`index`	The column in the row that will contain the value. The first column is 0.
`value`	The string value to insert in this row.
`charset`	The name of a supported charset. If `null`, the default charset of the engine tier is used. To find out the list of charsets supported by your JVM, call the following method: `java.nio.charset.Charset.availableCharsets()`

Throws

Exception	Description
`IndexOutOfBoundsException`	Thrown when the index is not valid. This runtime exception does not need to be explicitly caught.
`UnsupportedEncodingException`	Thrown when the specified charset is not supported.

setValueAsSQLTyped() method

```
public abstract void setValueAsSQLTyped (int index,
                                         Object value)
```

The `setValueAsSQLTyped()` method sets the value in a given column in a row, according to its SQL type.

The `java.lang.Object.toString()` method is called on the value before assigning it to the column unless it is an array of bytes (`byte[]`). In this case, it is either passed as is when the column's SQL type is Binary, LongVarBinary, or VarBinary, or it is converted to a string, using the `java.lang.String(byte[])` constructor.

Parameters

Parameter	Description
`index`	The column in the row that will contain the value. The first column is 0.
`value`	The value to insert in this row. For performance reasons, arrays of bytes are not copied by this method, when the type is binary. Therefore, any further modification to the initial value might be reflected in the column until the row is actually written. To duplicate the value, use the `java.lang.System.arraycopy()` method.

Throws

Exception	Description
`IndexOutOfBoundsException`	Thrown when the index is not valid. This runtime exception does not need to be explicitly caught.

Chapter 3. Using the Java Client stage

The role of a Java Client stage determines which links it supports:

- As a source stage, a Java Client stage can support one output link for writing data.
- As a target stage, a Java Client stage can support one input link for incoming data.
- As a lookup stage, a Java Client stage can support one reference link.

Adding a Java Client stage to a job

About this task

You configure the Java Client stage while building a job in InfoSphere DataStage Designer Client.

To add a Java Client stage to the canvas:

Procedure

1. From the Real Time group in the Palette pane, drag the Java Client stage icon onto the canvas.
2. Connect links to the Java Client stage, as needed.

Results

In the following sections, all steps are carried out in the Java Client stage.

Stage instances

Two or more instances of a Java Pack stage in a job flow use different classloaders. Therefore, they do not share resources, such as static variables.

Identifying your Java application

Use the package name and the Stage subclass of your Java application to identify your Java application.

```
package com.mycompany.examples;
...
public class CurrencyFinder extends Stage
```

Creating Stage properties

About this task

To identify your Java application:

Procedure

1. On the Stage page, click the **General** tab.
2. In the Transformer Class Name field, enter the fully-qualified name of your Stage subclass.

3. In the User's Classpath grid, enter the classpath of your Java application. You have two choices for specifying paths:
   - Use a separate line for each path.
   - If you want to enter multiple paths on a single line, use the separator required on your engine tier host. For UNIX, use the colon (:). For Windows, use the semi-colon (;).
4. In the Description field, optionally enter a description of the transformation.

Specifying application parameters

You can save parameter strings as stage properties and use the `Stage.getuserProperties()` method to fetch them. The parameter string has a free format. For example, it can contain job parameters, Java properties, an XML document, or the path name of a file containing properties or a document.

Creating Stage properties

About this task

To specify the parameter string:

Procedure

1. On the Stage page, click the **Properties** tab.
2. Perform one of the following steps:
   - In the **User's Properties** box, specify the parameter string. A path can contain forward slashes or backslashes.
   - Click **Load** to load a parameter string from a file. In the Open dialog box, locate the file, and click **OK**. The contents of the file appear in the User's Properties box.

Specifying JVM options

You can save JVM options as stage properties, which are used when your Java application runs. Options vary by JVM provider.

JVM instances

For server jobs, a job with multiple Java Pack stages (Java Client and Java Transformer) can load as many JVMs as the number of Java Pack stages in the job. The number of JVMs will vary. For parallel jobs, each Java Pack stage runs a separate JVM.

Creating stage properties

About this task

To save JVM options:

Procedure

1. On the Stage page, click the **Options** tab.
2. In the Java Virtual Machine Options grid, perform one of the following steps:
   - Specify JVM options on separate lines.
   - Specify JVM options on the same line, separated by spaces.

Specifying column definitions

Use the standard InfoSphere DataStage grid to specify the input and output columns involved in the transformation.

Define input columns

Procedure

1. On the Input page, click the **Columns** tab.
2. Specify the column metadata. For more information about the Columns page, see the *InfoSphere DataStage Designer Client Guide*.

Define output columns

Procedure

1. On the Output page, click the **Columns** tab.
2. Specify the column metadata. For more information about the Columns page, see the *InfoSphere DataStage Designer Client Guide*.

Chapter 4. Using the Java Transformer stage

A Java Transformer stage can support three links:
- One input link for incoming data.
- One output link for writing transformation results.
- One optional Reject link for writing rejected rows and erroneous data.

Adding a Java Transformer stage to a job

About this task

You configure the Java Transformer stage while building a job in InfoSphere DataStage Designer Client.

To add a Java Transformer stage to the canvas:

Procedure

1. From the Real Time group in the Palette pane, drag the Java Transformer stage icon onto the canvas.
2. Connect input and output links to the Java Transformer stage.

Results

In the following sections, all steps are carried out in the Java Transformer stage.

Stage instances

Two or more instances of a Java Pack stage in a job flow use different classloaders. Therefore, they do not share resources, such as static variables.

Identifying your Java application

Use the package name and the Stage subclass of your Java application to identify your Java application.

```
package com.mycompany.examples;
...
public class CurrencyFinder extends Stage
```

Creating Stage properties

About this task

To identify your Java application:

Procedure

1. On the Stage page, click the **General** tab.
2. In the Transformer Class Name field, enter the fully-qualified name of your Stage subclass.
3. In the User's Classpath grid, enter the classpath of your Java application. You have two choices for specifying paths:

- Use a separate line for each path.
- If you want to enter multiple paths on a single line, use the separator required on your engine tier host. For UNIX, use the colon (:). For Windows, use the semi-colon (;).

4. In the Description field, optionally enter a description of the transformation.

Specifying application parameters

You can save parameter strings as stage properties and use the `Stage.getuserProperties()` method to fetch them. The parameter string has a free format. For example, it can contain job parameters, Java properties, an XML document, or the path name of a file containing properties or a document.

Creating Stage properties

About this task

To specify the parameter string:

Procedure

1. On the Stage page, click the **Properties** tab.
2. Perform one of the following steps:
   - In the **User's Properties** box, specify the parameter string. A path can contain forward slashes or backslashes.
   - Click **Load** to load a parameter string from a file. In the Open dialog box, locate the file, and click **OK**. The contents of the file appear in the User's Properties box.

Specifying JVM options

You can save JVM options as stage properties, which are used when your Java application runs. Options vary by JVM provider.

JVM instances

For server jobs, a job with multiple Java Pack stages (Java Client and Java Transformer) can load as many JVMs as the number of Java Pack stages in the job. The number of JVMs will vary. For parallel jobs, each Java Pack stage runs a separate JVM.

Creating stage properties

About this task

To save JVM options:

Procedure

1. On the Stage page, click the **Options** tab.
2. In the Java Virtual Machine Options grid, perform one of the following steps:
   - Specify JVM options on separate lines.
   - Specify JVM options on the same line, separated by spaces.

Specifying a Reject link

The Java Transformer stage supports one Reject link to which you can send rejected input rows.

If the input and output links use the same table definition, you can call the `Stage.rejectRow()` method to pass rejected input rows to a Reject link. If the table definitions are not identical, call the `Stage.createRejectRow()` method to create an empty row and populate it, as needed.

Creating Stage properties

About this task

To define an output link as a Reject link:

Procedure

1. On the Output page, click the **General** tab.
2. Select the **Is Reject Link** box.
3. In the Description box, optionally enter a description of the Reject link.

Specifying column definitions

Use the standard InfoSphere DataStage grid to specify the input and output columns involved in the transformation.

Define input columns

Procedure

1. On the Input page, click the **Columns** tab.
2. Specify the column metadata. For more information about the Columns page, see the *InfoSphere DataStage Designer Client Guide*.

Define output columns

Procedure

1. On the Output page, click the **Columns** tab.
2. Specify the column metadata. For more information about the Columns page, see the *InfoSphere DataStage Designer Client Guide*.

Chapter 5. Other Java Pack API methods

The following methods are described. These methods are not discussed in Java Pack API. For more information about these methods, see the API documentation.

Stage class methods

Method	Description
`abort()`	Terminates a Java Client or Java Transformer stage after a job failure.
`isJobStopped()`	Checks whether the InfoSphere DataStage job has stopped or aborted.

Column class methods

Method	Description
`getValueAsRaw()`	Gets the raw (array of bytes) value corresponding to a column in the specified row.
`getValueAsSQLTyped`	Gets the value corresponding to a column in the specified row, according to its SQL type.
`getValueAsString()`	Gets the string value corresponding to a column in the specified row, with or without a charset.
`setValueAsRaw()`	Sets a raw (array of bytes) value corresponding to a column in the specified row.
`setValueAsSQLTyped()`	Sets the value corresponding to a column in a specified row, according to its SQL type.
`setValueAsString()`	Sets a string value corresponding to a column in the specified row, with or without a charset.

Chapter 6. Sample programs

The following sample programs are described. These programs illustrate the `Stage`, `Column`, and `Row` classes of the `com.ascentialsoftware.jds` package.

Programs for the Java Client stage

This section describes sample Java programs that can be called from the Java Client stage.

Methods and status fields of the `com.ascentialsoftware.jds` package are highlighted as links. Click them to access descriptions in Java Pack API.

Source stage example

In this example, the Java Client stage is deployed as a source stage and writes rows to an output link. The number of rows to generate, 20, is specified as the `User's Properties` value in the Java Client stage and is fetched using the `Stage.getUserProperties()` method.

```
package com.ascentialsoftware.jds.test;

import com.ascentialsoftware.jds.Row;
import com.ascentialsoftware.jds.Stage;

public class TableSource extends Stage
{
    public void initialize()    {
        trace("TableSource.initialize");
        _rowCount  = 0;
        _rowNumber = 0;

        String userProperties = getUserProperties();

        try {
            _rowCount = Integer.parseInt(userProperties);
        } catch (NumberFormatException eNumberFormat) {
            fatal("TableSource.process: row count '" + userProperties + "': " +
                eNumberFormat.getMessage());
        }
    }

    public void terminate()    {
        trace("TableSource.terminate");
    }

    public int process()    {
        // Generate a row where each column has the format "r:c",
        // where "r" is the row number and "c" the column number.
        // The total number of rows to generate is given
        // in the user's properties of the stage.

        _rowNumber++;

        if (_rowNumber > _rowCount) {
            return OUTPUT_STATUS_END_OF_DATA;
        }

        Row outputRow   = createOutputRow();
        int columnCount = outputRow.getColumnCount();

        for (int columnNumber = 0; columnNumber < columnCount; columnNumber++) {
            String value = Integer.toString(_rowNumber) + ":" +
            Integer.toString(columnNumber + 1);

            outputRow.setValueAsString(columnNumber, value);
        }

        writeRow(outputRow);

        return OUTPUT_STATUS_READY;
    }

    private int _rowCount;
    private int _rowNumber;
}
```

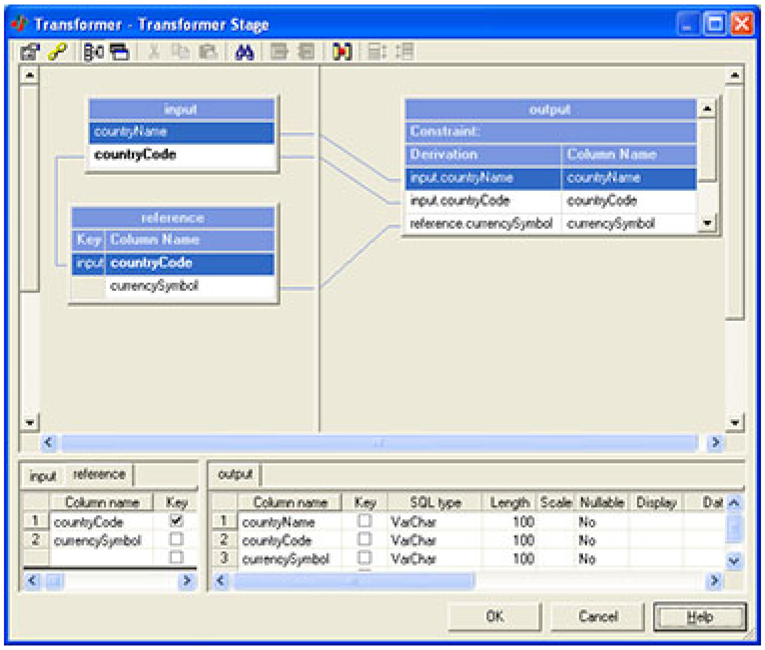
Target stage example

In this example, the Java Client stage is deployed as target stage and reads rows from an input link. The data are converted to uppercase, enclosed in quotes, set in a comma-delimited format, and written to a file whose path is specified as the `User's Property` value in the Java Client stage. The file path is fetched using the `Stage.getUserProperties()` method.

```
package com.ascentialsoftware.jds.test;

import com.ascentialsoftware.jds.Row;
import com.ascentialsoftware.jds.Stage;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class UpperCaseTarget extends Stage
{
    public void initialize()    {
        trace("UpperCaseTarget.initialize");
        _rowCount     = 0;
        _resultWriter = null;

        String userProperties = getUserProperties();

        try {
            _resultWriter = new PrintWriter(new FileWriter(userProperties), true);
        } catch (IOException eIO) {
            fatal("Cannot open '" + userProperties + "': " + eIO.getMessage());
        }
    }

    public void terminate()    {
        trace("UpperCaseTarget.terminate");
    }

    public int process()    {
        // Read a row, convert all its columns to upper case
        // surrounded by double-quotes and delimited by commas,
        // and write the result to a file. The file path is given
        // in the user's properties of the stage.

        _rowCount++;

        Row inputRow    = readRow();

        if (inputRow == null) {
            // No row currently available or end of data.
            // The function must return but it could be called again later on.
            // The stage actually ends when "terminate" is called.
            return OUTPUT_STATUS_END_OF_DATA;
        }

        int columnCount = inputRow.getColumnCount();

        for (int columnNumber = 0; columnNumber < columnCount; columnNumber++) {
            String value = inputRow.getValueAsString(columnNumber);

            _resultWriter.print("\"" + value.toUpperCase() + "\"");
            if (columnNumber < columnCount - 1) {
                _resultWriter.print(",");
            } else {
                _resultWriter.println();
            }
        }

        // No rows were written since there is no output link.
        return OUTPUT_STATUS_NOT_READY;
    }

    private int         _rowCount;
    private PrintWriter _resultWriter;
}
```

Transformer stage

The following Transformer Stage example shows the relationships between input, reference, and output columns. The input column `countryName` is passed through to the output link.



Key	Column Name
input	countryCode
	currencySymbol

Derivation	Column Name
input.countryName	countryName
input.countryCode	countryCode
reference.currencySymbol	currencySymbol

input | reference

	Column name	Key
1	countryCode	✓
2	currencySymbol	☐
		☐

output

	Column name	Key	SQL type	Length	Scale	Nullable	Display	Dat
1	countryName	☐	VarChar	100		No		
2	countryCode	☐	VarChar	100		No		
3	currencySymbol	☐	VarChar	100		No		

Sample program

The sample Java program, using the base classes `Currency` and `Locale` of the `java.util` package, finds currency symbols for a set of country codes.

In the `initialize()` method, the reference link is inspected to find the indexes of the country code and the currency symbol columns. As demonstrated, use the `createInputRow()` method when you want to inspect the input column metadata in an `initialize()` method.

```
package com.ascentialsoftware.jds.test;

import com.ascentialsoftware.jds.Column;
import com.ascentialsoftware.jds.Row;
import com.ascentialsoftware.jds.Stage;

import java.util.Currency;
import java.util.Locale;

public class CurrencyFinderLookup extends Stage
{
    /**
     * Initializes some variables.
     */
    public void initialize()    {
      _countryCodeColumnIndex    = -1;
      _currencySymbolColumnIndex = -1;

      Row inputRow = createInputRow();
      for (int columnNumber = 0; columnNumber < columnCount; columnNumber++) {
         Column column = inputRow.getColumn(columnNumber);

                if (column.getName().equals("countryCode")) {
                    _countryCodeColumnIndex = columnNumber;
                }

                if (column.getName().equals("currencySymbol")) {
                    _currencySymbolColumnIndex = columnNumber;
                }
            }

            if (_countryCodeColumnIndex < 0) {
                fatal("column \"countryCode\" not found.");
                return;
            }

            if (_currencySymbolColumnIndex < 0) {
                fatal("column \"currencySymbol\" not found.");
                return;
            }
     }
   }

    /**
     * Does nothing but log a message if the traces are on.
     */
    public void terminate()    {
        trace("CurrencyFinderLookup.terminate");
    }

    /**
     * Reads country codes from an input row, converts them to currency symbols,
     * and writes the latter on an output row. If the currency code is invalid,
     * the row is written instead on the reject link.
     *
     * @return {@link #OUTPUT_STATUS_READY} every time a row is written,
     *         {@link #OUTPUT_STATUS_END_OF_DATA} otherwise.
     */
    public int process()    {
        Row inputRow = readRow();

        if (inputRow == null) {
            // No row currently available or end of data.
            return OUTPUT_STATUS_END_OF_DATA;
        }

        Row      outputRow = createOutputRow();
        Currency currency  = null;
        String   value     = inputRow.getValueAsString(_countryCodeColumnIndex);

        if (value != null) {
            Locale locale = new Locale("", value.toLowerCase(), "");

            try {
                currency = Currency.getInstance(locale);
            }
            catch (IllegalArgumentException eIllegalArgument) {
                warn("unknown country code: " + value);
            }
        }

        outputRow.setValueAsString(_currencySymbolColumnIndex,
                                   currency == null ? null : currency.getSymbol());
        writeRow(outputRow);

        return OUTPUT_STATUS_READY;
    }

    private int _countryCodeColumnIndex;
    private int _currencySymbolColumnIndex;
}
```

Programs for the Java Transformer stage

This section describes sample Java programs that perform transformations and can be called from the Java Transformer stage.

Methods and status fields of the `com.ascentialsoftware.jds` package are highlighted in blue. Click them to access descriptions in Java Pack API.

Uppercase conversion

The sample program reads rows from an input link, converts column values to uppercase and writes the results to an output link. Rows that contain an asterisk (*) are sent to a Reject link.

```
package com.ascentialsoftware.jds.test;

import com.ascentialsoftware.jds.Row;
import com.ascentialsoftware.jds.Stage;

public class UpperCase extends Stage
{
    public int process()    {
        // Read a row, convert all its columns to upper case,
        // and write the result. If one column of the input row
        // contains the character '*', the row is rejected.

        Row      inputRow    = readRow();

        if (inputRow == null) {
            return OUTPUT_STATUS_END_OF_DATA;
        }

        boolean  reject      = false;
        int      columnCount = inputRow.getColumnCount();
```

```
            Row      outputRow   = createOutputRow();

            for (int columnNumber = 0; columnNumber < columnCount; columnNumber++) {
                String value = inputRow.getValueAsString(columnNumber);

                if ((value == null) || (value.indexOf('*') >= 0)) {
                    reject = true;
                    outputRow.setValueAsString(columnNumber, value);
                } else {
                    outputRow.setValueAsString(columnNumber, value.toUpperCase());
                }
            }

            if (reject) {
                rejectRow(outputRow);
            } else {
                writeRow(outputRow);
            }
        return OUTPUT_STATUS_READY;
        }

}
```

Sorting rows

The sample program sorts all input rows based on two properties that are fetched from the Java Transformer stage using the `Stage.getUserProperties()` method. The `User's Properties` provide the sort column and order:

- `descending = false`
- `nKeyIndex = 5`

Looping on the `readRow()` method is demonstrated, which might exhaust the available rows on an input link and generate a `LinkNotReadyException`. Returning from the `process()` method between two calls of `readRow()` allows the system to fetch the next row.

```
package com.ascentialsoftware.jds.test;

import com.ascentialsoftware.jds.Column;
import com.ascentialsoftware.jds.Row;
import com.ascentialsoftware.jds.Stage;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Properties;

/**
 * Example of transformer that sorts all incoming rows by the value
 * of a column whose index is stored in the stage's user properties.
 */
public class Sorter extends Stage
{
    /**
     *  Reads the user properties of the stage, returned as a single string.
     *  In this case, the string contains a set of actual properties, in the
     *  form of a list of key-value pairs, easy to process using Java's
     *  Properties class.
     *
     *  Example of such a string: "descending = false\nkeyIndex = 5"
     */
```

```
public void initialize()    {
    byte[]      userProperties = getUserProperties().getBytes();
    InputStream propertyStream = new ByteArrayInputStream(userProperties);
    Properties  properties     = new Properties();
    String      propertyValue;

    try {
        properties.load(propertyStream);
    } catch (IOException eIO) {
        //  Should never happen.
    }

    //  This property tells whether the rows must be sorted
    //  in an ascending or a descending order.
    propertyValue = properties.getProperty("descending");
    _descending   = (propertyValue == null)
                     ? false
                     : propertyValue.equals("true");

    //  This property gives the index of the column that contains
    //  the key by which each row will be sorted.
    propertyValue = properties.getProperty("keyIndex");
    _keyIndex     = (propertyValue == null)
                     ? 0
                     : Integer.parseInt(propertyValue);

    _keys = new ArrayList();
    _rows = new HashMap();
}

/**
 *  Loops on reading input rows as long as new rows are available.
 *  Rows are stored in an hash map. When no more rows are available,
 *  the stored rows are sorted and written on the output link,
 *  in the right order.
 *
 *  Note: the same thing could be achieved by storing only one row
 *  every time this method is called. Looping on all rows within
 *  one call is more efficient, but it is recommended to give
 *  the hand back to the system from time to time.
 */
public int process()    {
    Row inputRow = readRow();
    if (inputRow == null) {
        return OUTPUT_STATUS_END_OF_DATA;
    }

    do {
        String key = inputRow.getValueAsString(_keyIndex);
        _keys.add(key);
        _rows.put(key, inputRow);
        inputRow = readRow();
    }
    while (inputRow != null);

    Collections.sort(_keys);

    int keyCount = _keys.size();
    if (_descending) {
        for (int keyNumber = keyCount - 1; keyNumber >= 0; keyNumber--) {
            Row outputRow = (Row)_rows.get(_keys.get(keyNumber));
            writeRow(outputRow);
        }
    } else {
        for (int keyNumber = 0; keyNumber < keyCount; keyNumber++) {
            Row outputRow = (Row)_rows.get(_keys.get(keyNumber));
            writeRow(outputRow);
            }
        }

        return OUTPUT_STATUS_READY;
    }

    private boolean   _descending;
    private int       _keyIndex;
    private ArrayList _keys;
    private HashMap   _rows;
}
```

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 1. IBM resources

Resource	Description and location
IBM Support Portal	You can customize support information by choosing the products and the topics that interest you at www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server
Software services	You can find information about software, IT, and business consulting services, on the solutions site at www.ibm.com/businesssolutions/
My IBM	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 www.ibm.com/account/
Training and certification	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 http://www.ibm.com/software/sw-training/
IBM representatives	You can contact an IBM representative to learn about solutions at www.ibm.com/connect/ibm/us/en/

Providing feedback

The following table describes how to provide feedback to IBM about products and product documentation.

Table 2. Providing feedback to IBM

Type of feedback	Action
Product feedback	You can provide general product feedback through the Consumability Survey at www.ibm.com/software/data/info/consumability-survey

Table 2. Providing feedback to IBM (continued)

Type of feedback	**Action**
Documentation feedback	To comment on the information center, click the Feedback link on the top right side of any topic in the information center. You can also send comments about PDF file books, the information center, or any other documentation in the following ways: • Online reader comment form: www.ibm.com/software/data/rcf/ • E-mail: comments@us.ibm.com

Accessing 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 IBM InfoSphere Information Server. 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 item opens a local help system. Choose **Help > Open Info Center** to open the full suite 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/ic-homepage.html. The 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 iisdocs2, the Web address is in the following format: http://iisdocs2:9080/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.nav.doc/dochome/iisinfsrv_home.html.

A subset of the information center is also available on the IBM Web site and periodically refreshed at http://publib.boulder.ibm.com/infocenter/iisinfsv/v8r7/index.jsp.

Obtaining PDF and hardcopy documentation

- A subset of the PDF file books are available through the InfoSphere Information Server software installer and the distribution media. The other 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.

Providing feedback about the documentation

You can send your comments about documentation in the following ways:

- Online reader comment form: www.ibm.com/software/data/rcf/
- E-mail: comments@us.ibm.com

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 Business Glossary
- IBM InfoSphere Business Glossary Anywhere
- IBM InfoSphere DataStage
- IBM InfoSphere FastTrack
- IBM InfoSphere Information Analyzer
- IBM InfoSphere Information Services Director
- IBM InfoSphere Metadata Workbench
- IBM InfoSphere QualityStage

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.

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. XHTML allows you to set display preferences in your browser. It also allows you to use screen readers and other assistive technologies to access the documentation.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.

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.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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

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.

Trademarks

IBM, the IBM logo, and ibm.com are 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.

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.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium 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, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office

UNIX is a registered trademark of The Open Group in the United States and other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Index

Printed in USA

SC19-3455-00