IBM InfoSphere DataStage and QualityStage
Version 9 Release 1

Connectivity Guide for IBM WebSphere MQ Applications

IBM
Note
Before using this information and the product that it supports, read the information in .
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifying dynamic queues for Publish/Subscribe</td>
<td>69</td>
</tr>
<tr>
<td>Configuring the WebSphere MQ connector for publish/subscribe</td>
<td>69</td>
</tr>
<tr>
<td>WebSphere MQ connector properties</td>
<td>70</td>
</tr>
<tr>
<td>Access mode</td>
<td>70</td>
</tr>
<tr>
<td>Accounting token</td>
<td>71</td>
</tr>
<tr>
<td>Alternate security ID</td>
<td>71</td>
</tr>
<tr>
<td>Alternate user ID</td>
<td>71</td>
</tr>
<tr>
<td>Append node number</td>
<td>71</td>
</tr>
<tr>
<td>Application ID data</td>
<td>72</td>
</tr>
<tr>
<td>Application origin data</td>
<td>73</td>
</tr>
<tr>
<td>Backout count</td>
<td>73</td>
</tr>
<tr>
<td>Binding mode</td>
<td>74</td>
</tr>
<tr>
<td>Blocking transaction processing</td>
<td>74</td>
</tr>
<tr>
<td>Channel name</td>
<td>75</td>
</tr>
<tr>
<td>Client channel definition</td>
<td>75</td>
</tr>
<tr>
<td>Cluster queue</td>
<td>75</td>
</tr>
<tr>
<td>Coded character set ID</td>
<td>76</td>
</tr>
<tr>
<td>Coded character set ID for Message conversion</td>
<td>76</td>
</tr>
<tr>
<td>Connection name</td>
<td>77</td>
</tr>
<tr>
<td>Content filter</td>
<td>77</td>
</tr>
<tr>
<td>Context mode</td>
<td>78</td>
</tr>
<tr>
<td>Context mode for Error queue</td>
<td>78</td>
</tr>
<tr>
<td>Context mode for work queue</td>
<td>78</td>
</tr>
<tr>
<td>Correlation ID</td>
<td>79</td>
</tr>
<tr>
<td>Correlation ID for Deregistration</td>
<td>79</td>
</tr>
<tr>
<td>Correlation ID for Registration</td>
<td>79</td>
</tr>
<tr>
<td>Custom value for Feedback</td>
<td>79</td>
</tr>
<tr>
<td>Custom value for Format</td>
<td>80</td>
</tr>
<tr>
<td>Custom value for Message type</td>
<td>80</td>
</tr>
<tr>
<td>Custom value for Publication format</td>
<td>81</td>
</tr>
<tr>
<td>Custom value for Put application type</td>
<td>81</td>
</tr>
<tr>
<td>Deregistration</td>
<td>81</td>
</tr>
<tr>
<td>Dynamic queue</td>
<td>81</td>
</tr>
<tr>
<td>Dynamic reply queue</td>
<td>82</td>
</tr>
<tr>
<td>Enable payload reference</td>
<td>82</td>
</tr>
<tr>
<td>Encoding</td>
<td>82</td>
</tr>
<tr>
<td>Encoding for Message conversion</td>
<td>83</td>
</tr>
<tr>
<td>End of data message type</td>
<td>83</td>
</tr>
<tr>
<td>Error queue</td>
<td>84</td>
</tr>
<tr>
<td>Expiry</td>
<td>84</td>
</tr>
<tr>
<td>Feedback</td>
<td>85</td>
</tr>
<tr>
<td>Filter messages</td>
<td>85</td>
</tr>
<tr>
<td>Format</td>
<td>85</td>
</tr>
<tr>
<td>General options for Deregistration</td>
<td>86</td>
</tr>
<tr>
<td>General options for Registration</td>
<td>87</td>
</tr>
<tr>
<td>Group ID</td>
<td>88</td>
</tr>
<tr>
<td>Header version</td>
<td>88</td>
</tr>
<tr>
<td>Hex for Accounting token</td>
<td>88</td>
</tr>
<tr>
<td>Hex for Alternate security ID</td>
<td>88</td>
</tr>
<tr>
<td>Hex for Correlation ID</td>
<td>88</td>
</tr>
<tr>
<td>Hex for Group ID</td>
<td>88</td>
</tr>
<tr>
<td>Hex for Message ID</td>
<td>89</td>
</tr>
<tr>
<td>Identity options</td>
<td>89</td>
</tr>
<tr>
<td>Match all</td>
<td>90</td>
</tr>
<tr>
<td>Message content descriptor</td>
<td>91</td>
</tr>
<tr>
<td>Message controlled</td>
<td>91</td>
</tr>
<tr>
<td>Message conversion</td>
<td>91</td>
</tr>
<tr>
<td>Message flags</td>
<td>91</td>
</tr>
<tr>
<td>Message ID</td>
<td>92</td>
</tr>
<tr>
<td>Message options</td>
<td>92</td>
</tr>
<tr>
<td>Message order and assembly</td>
<td>92</td>
</tr>
<tr>
<td>Message padding</td>
<td>93</td>
</tr>
<tr>
<td>Message quantity</td>
<td>94</td>
</tr>
<tr>
<td>Message read mode</td>
<td>94</td>
</tr>
<tr>
<td>Message sequence number</td>
<td>94</td>
</tr>
<tr>
<td>Message sequence number for Publication settings</td>
<td>95</td>
</tr>
<tr>
<td>Message service domain</td>
<td>95</td>
</tr>
<tr>
<td>Message set</td>
<td>96</td>
</tr>
<tr>
<td>Message truncation</td>
<td>96</td>
</tr>
<tr>
<td>Message type</td>
<td>96</td>
</tr>
<tr>
<td>Message type for Message content descriptor</td>
<td>97</td>
</tr>
<tr>
<td>Message type for Message conversion</td>
<td>97</td>
</tr>
<tr>
<td>Method name (Blocking transaction processing)</td>
<td>98</td>
</tr>
<tr>
<td>Method name (Message controlled)</td>
<td>99</td>
</tr>
<tr>
<td>Method name (Blocking transaction processing)</td>
<td>100</td>
</tr>
<tr>
<td>Module name</td>
<td>100</td>
</tr>
<tr>
<td>Module name (Blocking transaction processing)</td>
<td>101</td>
</tr>
<tr>
<td>Monitor queue depth</td>
<td>101</td>
</tr>
<tr>
<td>Name</td>
<td>102</td>
</tr>
<tr>
<td>Offset</td>
<td>102</td>
</tr>
<tr>
<td>Original length</td>
<td>103</td>
</tr>
<tr>
<td>Other queue settings</td>
<td>104</td>
</tr>
<tr>
<td>Password</td>
<td>104</td>
</tr>
<tr>
<td>Payload size</td>
<td>104</td>
</tr>
<tr>
<td>Period</td>
<td>105</td>
</tr>
<tr>
<td>Persistence</td>
<td>105</td>
</tr>
<tr>
<td>Persistence options</td>
<td>106</td>
</tr>
<tr>
<td>Physical format</td>
<td>106</td>
</tr>
<tr>
<td>Priority</td>
<td>106</td>
</tr>
<tr>
<td>Publication format</td>
<td>107</td>
</tr>
<tr>
<td>Publication options</td>
<td>107</td>
</tr>
<tr>
<td>Publication settings</td>
<td>108</td>
</tr>
<tr>
<td>Publish/Subscribe</td>
<td>108</td>
</tr>
<tr>
<td>Put application name</td>
<td>108</td>
</tr>
<tr>
<td>Put application type</td>
<td>109</td>
</tr>
<tr>
<td>Put date</td>
<td>110</td>
</tr>
<tr>
<td>Put time</td>
<td>111</td>
</tr>
<tr>
<td>Queue manager for Cluster queue</td>
<td>111</td>
</tr>
<tr>
<td>Queue manager for Connection</td>
<td>112</td>
</tr>
<tr>
<td>Queue manager for Error queue</td>
<td>112</td>
</tr>
<tr>
<td>Queue name</td>
<td>112</td>
</tr>
<tr>
<td>Queue name for Dynamic queue</td>
<td>113</td>
</tr>
<tr>
<td>Queue name for Dynamic reply queue</td>
<td>114</td>
</tr>
<tr>
<td>Queue name for Error queue</td>
<td>115</td>
</tr>
<tr>
<td>Record ordering</td>
<td>115</td>
</tr>
<tr>
<td>Refresh</td>
<td>115</td>
</tr>
<tr>
<td>Registration</td>
<td>116</td>
</tr>
<tr>
<td>Registration options</td>
<td>116</td>
</tr>
<tr>
<td>Remove MQRFH2 header</td>
<td>116</td>
</tr>
<tr>
<td>Reply queue</td>
<td>116</td>
</tr>
<tr>
<td>Reply queue close options</td>
<td>117</td>
</tr>
<tr>
<td>Reply to queue</td>
<td>117</td>
</tr>
<tr>
<td>Reply to queue manager</td>
<td>118</td>
</tr>
<tr>
<td>Report</td>
<td>119</td>
</tr>
<tr>
<td>Row buffer count</td>
<td>119</td>
</tr>
<tr>
<td>Segment size</td>
<td>120</td>
</tr>
<tr>
<td>Segmented message</td>
<td>120</td>
</tr>
<tr>
<td>Service type</td>
<td>120</td>
</tr>
<tr>
<td>Set header fields</td>
<td>120</td>
</tr>
<tr>
<td>Sort order</td>
<td>121</td>
</tr>
</tbody>
</table>
Start value .................................. 121
Stream name .................................. 121
Subscription identity ......................... 121
Subscription name ......................... 122
Subscription point ......................... 122
System value for Feedback ................. 122
System value for Format .................... 122
System value for Message type ........... 123
System value for Publication format ...... 124
System value for Put application type ..... 124
Timeout (Blocking transaction processing) 124
Timestamp .................................. 124
Topic for Publication settings .............. 125
Topic for Registration and Deregistration 125
Transmission queue for Error queue ...... 125
Transmission queue for Other queue settings 126
Transport type ................................ 126
Treat EOL as row terminator .............. 126
Use wildcard ................................ 126
User ID ..................................... 126
Username .................................... 127
Value for Accounting token ............... 127
Value for Alternate security ID .......... 128
Value for Correlation ID ................... 128
Value for Group ID ......................... 129
Value for Message flags ................... 129
Value for Message ID ...................... 130
Value for Report ........................... 130
Wait time .................................. 131
Work queue ................................ 131

Chapter 5. IBM WebSphere MQ stage 133
Functionality of WebSphere MQ Stages ..... 133
Terminology .................................. 134
Client Only and Client/Server Versions ... 135
Publish/Subscribe Communication Model 135
Using the WebSphere MQ Stage .......... 136
IBM WebSphere MQ Stage Message Queues 136
Browsing the Queue ......................... 138
Processing Units of Work .................. 138
Reply and Report Messages ............... 141
Security .................................... 141
Defining the WebSphere MQ Connection 142
Defining an IBM WebSphere MQ Stage ... 143
Connecting to a Queue Manager .......... 143
Defining Character Set Mapping .......... 143
Defining IBM WebSphere MQ Input Data .. 144
About the Input Page ...................... 144
Specifying Transaction Control Information 146
Defining WebSphere MQ Output Data ..... 147
About the Output Page .................... 147
Using Column Data Elements ............. 155

Chapter 6. Environment variables:
WebSphere MQ connector ................. 157
CC_MSG_LEVEL ............................. 157
CC_SE_TIMESTAMP_FF ........................ 157
CC_USE_DEPRECATED_TRANS_PROP ....... 157
CC_WSMQ_FAIL_FOR_ANY_NAMELIST_ERROR 158

Appendix A. Product accessibility ....... 159
Appendix B. Reading command-line syntax .................. 161
Appendix C. How to read syntax diagrams .......... 163
Appendix D. Contacting IBM ............... 165
Appendix E. Accessing and providing feedback on the product documentation .... 167

Index ...................................... 173
Chapter 1. Connector Migration Tool

To take advantage of the additional functionality that connectors offer, use the Connector Migration Tool to migrate jobs to use connectors instead of plug-in and operator stages.

The following table lists the stages that can be migrated to connectors and the corresponding connectors that they are migrated to:

Table 1. List of stages and corresponding connectors

<table>
<thead>
<tr>
<th>Stage</th>
<th>Connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2Z stage</td>
<td>DB2 Connector</td>
</tr>
<tr>
<td>DB2 UDB API stage</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Load stage</td>
<td></td>
</tr>
<tr>
<td>DRS Stage</td>
<td>DRS Connector</td>
</tr>
<tr>
<td>Java Client stage</td>
<td>Java Integration Stage</td>
</tr>
<tr>
<td>Java Transformer stage</td>
<td></td>
</tr>
<tr>
<td>Netezza Enterprise stage</td>
<td>Netezza Connector</td>
</tr>
<tr>
<td>ODBC Enterprise stage</td>
<td>ODBC Connector</td>
</tr>
<tr>
<td>ODBC (Server) stage</td>
<td></td>
</tr>
<tr>
<td>SQLServer Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>Oracle OCI stage</td>
<td>Oracle Connector</td>
</tr>
<tr>
<td>Oracle OCI Load stage</td>
<td></td>
</tr>
<tr>
<td>Oracle Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>Teradata API stage</td>
<td>Teradata Connector</td>
</tr>
<tr>
<td>Teradata Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>Teradata Load stage</td>
<td></td>
</tr>
<tr>
<td>Teradata Multiload stage</td>
<td></td>
</tr>
<tr>
<td>WebSphere™ MQ stage</td>
<td>WebSphere MQ Connector</td>
</tr>
</tbody>
</table>

Migrating jobs to use connectors

To migrate jobs to use the connectors, you need to run the Connector Migration Tool.

To run the Connector Migration Tool, start it from the Microsoft Windows Programs menu or from the command line. If you start the tool from the command line, additional options that are not provided in the user interface are available.

The user interface leads you through the process of evaluating which jobs, shared containers, and stages to migrate. You select the jobs that you want to migrate, and beside each job name, the tool displays an icon that indicates whether or not the job can be fully migrated, partially migrated, or not migrated at all. To refine the list of jobs to evaluate, you can specify that only jobs that contain specific plug-in and operator stages be listed. The tool gives you a chance to make a backup of a job before you migrate it. You can make a backup copy of the job and then migrate the backup, or you can make a backup copy of the job and then migrate the original job. Either way, your original job is never lost. The job is migrated and
placed in the same folder as the original job, and the log file "CCMigration.log", which records the results of the migration, is created in the current directory.

The Connector Migration Tool command line options provide the same functionality as the user interface, as well as a few additional options. Using the command line, you can perform these additional tasks:

- Specify a list of job names to be considered for migration.
- Specify a list of shared container names to be considered for migration.
- Specify a list of stage type names to limit the jobs that are considered for migration.
- Run a practice migration, where the actual migration does not take place but the possible results of the migration are placed in the log file. You can review the results and then refine the migration as necessary before you run the actual migration.
- Produce a report of jobs and their stages and stage types.

Note:

- The Connector Migration Tool does not read environment variables at the operating system level. Environment variables are read only if they are defined within InfoSphere DataStage at the Project level or at the Job level. Project level environment variables are read first, then overwritten by Job environment variables. Environment variables with blank default values are ignored by the Connector Migration Tool. The default values of the environment variables are migrated, but the run-time values are not migrated.
- Throughout this documentation, the term "job" refers to parallel shared containers and server shared containers, as well as IBM® InfoSphere® DataStage® jobs.

Using the user interface to migrate jobs

Use the Connector Migration Tool to view which jobs and stages are eligible for migration and then migrate them to use connectors rather than plug-in and operator stages.

About this task

You use the same project connection details to connect to the Connector Migration Tool as you use to connect to the InfoSphere DataStage and QualityStage® Designer or InfoSphere DataStage and QualityStage Director Client. You must have sufficient user privileges to create and modify the jobs that you are migrating.

Procedure

1. Choose Start > Programs > IBM InfoSphere Information Server > Connector Migration Tool.
2. In the Log on window, complete these fields:
   a. In the Host field, enter the host name of the services tier. You can specify an optional port by separating it from the host name with a colon. The host name that you specify here is the same one that you specify when you start the Designer client, for example, mymachine:9080).
   b. In the User name field, enter your InfoSphere DataStage user name.
   c. In the Password field, enter your InfoSphere DataStage password.
   d. In the Project field, enter the name of the project. To access an InfoSphere DataStage server that is remote from the domain server, specify the project.
name in full as server:[port]/project. As an alternative, you can press the button adjacent to the **Project** field to display a dialog box from which you can select the fully-qualified project name.

e. Click OK. An icon indicates the status of each job. A gray icon indicates that the job cannot be migrated. A gray icon with a question mark indicates that the job might be successfully migrated.

3. Display the jobs and stages to consider for migration:
   - Choose **View > View all jobs** to display all of the jobs in the project. This is the default view.
   - Choose **View > View all migratable jobs** to display all of the jobs that are in the project and that can be migrated to use connectors. Jobs that do not contain any stages that can be migrated are excluded from the job list.
   - Choose **View > View jobs by stage types** to open the Filter by stage type window.

4. Perform the following steps to analyze jobs:
   a. Highlight the job in the job list.
   b. Expand the job in the job list to view the stages in the job.
   c. Select one or more jobs, and click **Analyze**.

   After analysis, the color of the job, stage, or property icon indicates whether or not it can be migrated. A green icon indicates that the job, stage, or property can be migrated. An red icon indicates that the job or stage cannot be migrated. An orange icon indicates that a job or stage can be partially migrated and that a property in a stage has no equivalent in a connector. A gray icon indicates that the job or stage is not eligible for migration.

   **Note:** The Connector Migration Tool displays internal property names, rather than the names that the stages display. To view a table that contains the internal name and the corresponding display name for each property, from the IBM InfoSphere DataStage and QualityStage Designer client, open the Stage Types folder in the repository tree. Double-click the stage icon, and then click the **Properties** tab to view the stage properties.

5. Click **Preferences** and choose how to migrate the job:
   - Choose **Clone and migrate cloned job** to make a copy of the job and then migrate the copy. The original job remains intact.
   - Choose **Back up job and migrate original job** to make a copy of the job and then migrate the original job.
   - Choose **Migrate original job** to migrate the job without making a backup.

6. Select the jobs and stages to migrate, and then click **Migrate**.

   The jobs and stages are migrated and are placed in the same folder as the original job. If logging is enabled, a log file that contains a report of the migration task is created. After a job is successfully migrated, a green checkmark displays beside the job name in the Jobs list to indicate that the job has been migrated.

**Using the command line to migrate jobs**

Run the Connector Migration Tool from the command line to use additional options that are not available in the user interface.
About this task

To run the Connector Migration Tool from the command line, you specify the command `CCMigration`, followed by a series of required and optional parameters. If the Connector Migration Tool is started from the command line, its user interface will be displayed if none of the options `-C`, `-M` or `-B` are specified. If any one of these options is specified, then the migration will proceed without any further interaction with the user. The command line options described below can therefore be used whether or not the user interface is displayed.

After a job is successfully migrated, a green checkmark displays beside the job name in the Jobs list to indicate that the job has been migrated.

Procedure

1. From the IBM InfoSphere DataStage client command line, go to the `<InformationServer>\Clients\CCMigrationTool` directory.
2. Enter the command `CCMigration`, followed by the following required parameters:
   - `-h host:port`, where `host:port` is the host name and port of the InfoSphere DataStage server. If you do not specify a port, the `port` is 9080 by default.
   - `-u user name`, where `user name` is the name of the InfoSphere DataStage user.
   - `-p password`, where `password` is the password of the InfoSphere DataStage user.
   - `-P project`, where `project` is the name of the project to connect to. To specify an InfoSphere DataStage server that is remote from the domain server, specify the fully qualified project name by using the format `server:[port]/project`.
   - One of the following:
     - `-M` If you specify this parameter, the original jobs are migrated, and backup jobs are not created.
     - `-B job name extension`, where `job name extension` is a set of alphanumeric characters and underscores. If you specify this parameter, the Connector Migration Tool creates backup jobs, names the backup jobs `source job name+job name extension`, and then migrates the original jobs. The backup jobs are saved in the same location in the repository as the source jobs.
     - `-C job name extension`, where `job name extension` is a set of alphanumeric characters and underscores. If you specify this parameter, the Connector Migration Tool clones the source jobs, names the cloned jobs `source job name+job name extension`, and then migrates the cloned jobs. The cloned jobs are saved in the same location in the repository as the source jobs.

   If you specify one of these options, the migration proceeds without requiring any additional user input. If you do not specify `-M`, `-B`, or `-C`, the user interface is displayed so that you can make additional choices for how to migrate the jobs.

3. Optional: Enter any of the following optional parameters:
   - `-L log file`, where `log file` is the file name and path for the log file that records the results of the migration.
   - `-S stage types`, where `stage types` is a comma-separated list of stage types. By default, the Connector Migration Tool migrates all stage types. Use this parameter to migrate only jobs that contain the specified stage types. If you specify both the `-S` and `-J` parameters, only the specified stage types within the specified jobs are migrated. If you specify the `-S` parameter and do not specify the `-C`, `-M` or `-B` parameter, only jobs that contain the specified stage
types appear in the job list that is displayed in the user interface. Limiting
the jobs that are displayed can significantly reduce the startup time of the
Connector Migration Tool.

- **-J job names**, where *job names* is a comma-separated list of jobs. By default,
the Connector Migration Tool migrates all eligible jobs in the project. Use this
parameter to migrate only specific jobs. If you specify the `-J` parameter and
do not specify the `-C`, `-M` or `-B` parameter, only the specified jobs appear in
the job list that is displayed in the user interface. Limiting the jobs that are
displayed can significantly reduce the startup time of the Connector
Migration Tool.

- **-c shared container names**, where *shared container names* is a comma-separated
list of shared containers. By default, the Connector Migration Tool migrates
all eligible shared containers in the project. Use this parameter to migrate
only specific shared containers. If you specify the `-c` parameter and do not
specify the `-C`, `-M`, or `-B` parameter, only the specified shared containers
appear in the job list that displays in the user interface. Limiting the shared
containers that display might significantly reduce the startup time of the
Connector Migration Tool.

- **-R** If you specify this parameter, the Connector Migration Tool reports the
details of the migration that would occur if the specified jobs were migrated,
but does not perform an actual migration. The details are reported in the log
file that is specified by using the `-L` parameter.

- **-a auth file**, where *auth file* is the file name that records the user name and
password.

- **-A** If you specify this parameter, the Connector Migration Tool adds an
annotation to the job design. The annotation describes the stages that were
migrated, the job from which the stages were migrated, and the date of the
migration.

- **-d job dump file**, where *job dump file* is the file name and path for a file where
a list of jobs, shared containers, and stages is written. Using a job dump file
is helpful when you want to determine which jobs are suitable for migration.
You can use the `-d` parameter with the `-J`, `-c`, and `-S` parameters to list
particular jobs, shared containers, and stage types, respectively.

- **-V** If you specify this parameter, the Connector Migration Tool specifies the
target connector variant for migrated stages. The format of the list is a
comma-separated list containing `{StageTypeName=Variant}`.

- **-v** If you specify this parameter with the `-d` command, the values of stage
properties will be included in the report. If omitted, the report only contains
stage names and types, but not the stage properties. This option is useful to
identify jobs that have stages with certain property values. If this option is
specified, then `-S` is ignored.

- **-T** If you specify this parameter, the Connector Migration Tool enables the
variant migration mode. All connector stages found in jobs and containers
whose stage type matches those listed by the `-V` command are modified.

- **-U** If you specify this parameter, the Connector Migration Tool enables the
property upgrade migration mode. All connector stages found in jobs and
containers whose properties match the conditions specified in the
`StageUpgrade.xml` file are upgraded.
Example

The following command starts the Connector Migration Tool, connects to the project billsproject on the server dsserver as user billg, and migrates the jobs db2write and db2upsert:

```
CCMigration -h dsserver:9080 -u billg -p padd0ck
-P billsproject -J db2write,db2upsert -M
```

Deprecated stages

Connectors, which offer better functionality and performance, replace some stages, which have been deprecated and removed from the palette. However, you can still use the deprecated stages in jobs, and add them back to the palette.

The following stage types have been removed from palette for the parallel job canvas:

- DB2Z
- DB2® UDB API
- DB2 UDB Load
- DRS
- Dynamic RDBMS
- Java Client
- Java Transformer
- Netezza Enterprise
- ODBC Enterprise
- Oracle 7 Load
- Oracle OCI Load
- Oracle Enterprise
- Teradata API
- Teradata Enterprise
- Teradata Load
- Teradata Multiload
- WebSphere MQ

The following stage type has been removed from the palette for the server job canvas:

- Dynamic RDBMS

When you create new jobs, consider using connectors instead of the deprecated stages. The following table describes which connector to use in place of which deprecated stages:

Table 2. Stages and corresponding connectors

<table>
<thead>
<tr>
<th>Deprecated stages</th>
<th>Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2Z</td>
<td>DB2 Connector</td>
</tr>
<tr>
<td>DB2 UDB API</td>
<td>DB2 Connector</td>
</tr>
<tr>
<td>DB2 UDB Load</td>
<td>DRS connector</td>
</tr>
<tr>
<td>DB2 UDB Enterprise</td>
<td>DRS connector</td>
</tr>
<tr>
<td>DRS</td>
<td>DRS connector</td>
</tr>
</tbody>
</table>
To use any of the deprecated stage types in new jobs, drag the stage type from the repository tree to the canvas or to the palette. From the repository tree, expand **Stage Types**. Under **Stage Types**, expand **Parallel** or **Server** depending on the stage that you want to use. Drag the stage type to the job canvas or to the palette.
Chapter 2. Configuring access to data sources

To configure database connectivity, you must install database client libraries and include the path to these installed libraries in the library path environment variable. Apart from the library path, for certain database types you must set additional environment variables on the engine tier computer.

About this task

You must install the client libraries and include the directory that contains the database client libraries in the library path environment variables. For more information about setting environment variables, see “Setting the library path environment variable” on page 21.

Some 64-bit database client software installations include 32-bit version and 64-bit version of the client libraries. In this case, you must set the library path to point to the libraries that have the bit level that matches the bit level of the engine tier installation. Otherwise, when a job that uses the connector that requires the client libraries runs, an error is reported because the stage library is not able to load the database client libraries.

Procedure

1. Install database client libraries.
2. Configure access to data sources.

Configuring access to DB2 databases

To use the DB2 Connector stage in a job, you must configure DB2 environment variables and set the privileges for DB2 users. The DB2 connector connects to your databases by using the DB2 client on the InfoSphere DataStage nodes.

Before you begin

- Confirm that your system meets the system requirements for InfoSphere Information Server. Make sure that you are using a supported version of IBM DB2. For more information about system requirements, see [http://www.ibm.com/software/data/infosphere/info-server/overview/](http://www.ibm.com/software/data/infosphere/info-server/overview/).
- Install the IBM DB2 client on all InfoSphere DataStage nodes, and make sure that the client is working correctly.
- Use the DB2 Configuration Assistant to test the DB2 client and server connection. If the DB2 client fails to connect to the DB2 server, jobs that use the DB2 Connector stage also fail.
- Catalog each database in the DB2 client.
- InfoSphere DataStage runs many processes for each job. Ensure that DB2 resources, configuration parameters, and manager configuration parameters are configured properly.
- Make sure that the DB2_PMAP_COMPATIBILITY registry variable is set to ON to indicate that the distribution map size remains 4,096 (4-KB) entries. Though DB2® version 9.7 database for Linux, UNIX, and Windows supports distribution map entries up to 32,768 (32 KB), the DB2 connector supports only 4-KB entries in distribution maps.
• If you plan to use the DB2 connector with DB2 for z/OS in jobs with sparsely arriving data (such as jobs that use the Change Data Capture stage), ensure that the idle timeout value set in the DB2 IDTHTION subsystem parameter is longer than the longest expected interval of inactivity for the DB2 Connector stages in the job.

**Procedure**

1. Grant the InfoSphere DataStage users SELECT privileges on the following tables:

   **Table 3. Required SELECT privileges**

<table>
<thead>
<tr>
<th>DB2 product</th>
<th>Tables that require SELECT privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2 Database for Linux, UNIX, and Windows</td>
<td>SYSCAT.COLUMNS</td>
</tr>
<tr>
<td></td>
<td>SYSCAT.KEYCOLUSE</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSDBAUTH</td>
</tr>
<tr>
<td></td>
<td>SYSCAT.TABLES</td>
</tr>
<tr>
<td>DB2 for z/OS</td>
<td>Note: Before the data is loaded to data to</td>
</tr>
<tr>
<td></td>
<td>DB2 for z/OS, make sure the user has the</td>
</tr>
<tr>
<td></td>
<td>GRANT ALL access on SYSIBM.SYSPRINT:</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSCOLUMNS</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSINDEXES</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSKEYCOLUSE</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSKEYS</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSPRINT</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSTABLESPACE</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSTABLES</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSTABLEPART</td>
</tr>
<tr>
<td></td>
<td>SYSIBM.SYSUSERAUTH</td>
</tr>
<tr>
<td>DB2 Database for Linux, UNIX, and Windows</td>
<td>SYSIBM.SYSDUMMY1</td>
</tr>
<tr>
<td>and z/OS</td>
<td>SYSIBM.SYSVIEWS</td>
</tr>
</tbody>
</table>

2. On DB2 for z/OS, ensure that the DBA runs the DSNTIJSG installation job to install the **DSNUTILS** stored procedure. The **DSNUTILS** stored procedure is required to start the bulk loader on DB2 for z/OS. For more information, see [http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z9.doc.inst/src/tpc/db2z_enabledb2suppstdprocs.htm](http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z9.doc.inst/src/tpc/db2z_enabledb2suppstdprocs.htm)

3. Set the **DB2INSTANCE** environment variable to the instance in the DB2 client in which you cataloged the target database.

   You must set the **DB2INSTANCE** environment variable even if the stage accesses the default DB2 instance. The instance that is specified in the **DB2INSTANCE** environment variable becomes the default instance that is used by the connector. If you want to use a DB2 instance other than the default, then enter the name of that instance in the **Instance** property of the DB2 connector in the Properties tab. The DB2 client installs the default instances.

   **Table 4. Default instance installed by the DB2 client**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>DB2 Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux or UNIX</td>
<td>db2inst1</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>DB2</td>
</tr>
</tbody>
</table>
4. Add the path to the directory that contains the client libraries to the library path environment variable. The default path for client libraries is shown in the table.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>DB2 Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux or UNIX</td>
<td>/opt/IBM/db2/V9/lib64</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>C:\IBM\SQLLIB\BIN</td>
</tr>
</tbody>
</table>

5. Optional: If the globalization map name for the DB2 connector job does not match the current system locale on the engine tier, then set the `DB2CODEPAGE` environment variable to a codepage corresponding to the map name. The DB2 code page can also be set by using a DB2 registry variable.

---

**Configuring the DB2 native ODBC driver on AIX**

To configure and use the ODBC driver on AIX operating systems, you must modify the DB2 environment variables and the `ODBCINI` environment variable.

**Before you begin**

- Confirm that your system meets the system requirements and that you are using a supported version of IBM DB2 database systems. For more information, see [System Requirements](#).

**Procedure**

1. Update the 64-bit ODBC driver for DB2 on AIX operating systems so that the DataDirect driver manager can load the ODBC driver:
   a. Open the `db2o.o` driver file, which is in the `$DB2_HOME/sqllib/lib64` directory.
   b. In the `db2o.o` driver file, add a link to the `db2o.o.so` file. For example, you can add the following link:
      ```bash
      ln -s db2o.o db2o.o.so
      ```

2. Add the following entry to the `$ODBCINI` file. The DataDirect driver manager uses the `DriverUnicodeType=1` entry to work with the ODBC driver for DB2.
   ```plaintext
   ****************************************************
   ****************************************************
   Driver=/home/db2inst1/sqllib/lib64/db2o.o.so
   DriverUnicodeType=1
   ****************************************************
   ****************************************************
   Driver=/home/db2inst1/sqllib/lib64/db2o.o.so
   DriverUnicodeType=1
   ```

3. Set the `DB2INSTANCE` environment variable to the instance in the DB2 client in which you cataloged the target database.

4. Add the path to the directory that contains the client libraries to the library path environment variable.

---

**Configuring access to Oracle databases**

You can configure access to an Oracle database from the Oracle client system by setting environment variables and by updating Oracle configuration files such as `tnsnames.ora` and `sqlnet.ora`. For more information, see the Oracle product documentation.

**Before you begin**

- Install client libraries.
• Ensure that your system meets the system requirements and that you have a supported version of the Oracle client and Oracle server. For system requirement information, see [http://www.ibm.com/software/data/infosphere/info-server/overview/](http://www.ibm.com/software/data/infosphere/info-server/overview/).

• Ensure that the Oracle client can access the Oracle database. To test the connectivity between the Oracle client and Oracle database server, you can use the Oracle SQL*Plus utility.

**About this task**

You can use the dsenv script to update the environment variables that are used to configure access to Oracle databases. If you use the script, you must restart the server engine and the ASB Agent after you update the environment variables.

**Procedure**

1. Set either the `ORACLE_HOME` or the `TNS_ADMIN` environment variable so that the Oracle connector is able to access the Oracle configuration file, tnsnames.ora.
   - If the `ORACLE_HOME` environment variable is specified, then the tnsnames.ora file must be in the `$ORACLE_HOME/network/admin` directory.
   - If the `TNS_ADMIN` environment variable is specified, then the tnsnames.ora file must be in the `$TNS_ADMIN` directory.
   - If both environment variables are specified, then the `TNS_ADMIN` environment variable takes precedence.

   Setting these environment variables is not mandatory. However, if one or both environment variables are not specified, then you cannot select a connect descriptor name to define the connection to the Oracle database. Instead, when you define the connection, you must provide the complete connect descriptor definition or specify an Oracle Easy Connect string.

   **Note:** If you use the Oracle Basic Instant Client or the Basic Lite Instant Client, the tnsnames.ora file is not automatically created for you. You must manually create the file and save it to a directory. Then specify the location of the file in the `TNS_ADMIN` environment variable. For information about creating the tnsnames.ora file manually, see the Oracle documentation.

2. Optional: Set the library path environment variable to include the directory where the Oracle client libraries are located. The default location for client libraries are as follows:
   - On Windows, `C:\app\username\product\11.2.0\client_1\BIN`, where `username` represents a local operating system user name. If the complete Oracle database product is installed on the InfoSphere Information Server engine computer instead of just the Oracle client product, then the path would be `C:\app\username\product\11.2.0\dbhome_1\BIN`.
   - On Linux or UNIX, `u01/app/oracle/product/11.2.0/client_1`

3. Set the `NLS_LANG` environment variable to a value that is compatible with the NLS map name that is specified for the job. The default value for the `NLS_LANG` environment variable is `AMERICAN_AMERICA.US7ASCII`.

   The Oracle client assumes that the data that is exchanged with the stage is encoded according to the `NLS_LANG` setting. However, the data might be encoded according to the NLS map name setting. If the `NLS_LANG` setting and the NLS map name setting are not compatible, data might be corrupted, and invalid values might be stored in the database or retrieved from the database. Ensure that you synchronize the `NLS_LANG` environment variable and NLS map name values that are used for the job.
On Microsoft Windows installations, if the `NLS_LANG` environment variable is not set, the Oracle client uses the value from the Windows registry.

### Configuring access to ODBC data sources

To use the ODBC data sources in a job, you must configure the ODBC driver and also the ODBC data source name (DSN) definitions.

**Before you begin**

- Install client libraries.
- Test the connection to the ODBC data source.
- On 64-bit Windows computers, make sure that you run the 32-bit version of the Microsoft ODBC Data Source Administrator `C:\Windows\SysWOW64\odbcad32.exe`, as InfoSphere Information Server is a 32-bit application. If you run the 64-bit version of the ODBC administrator application, the Netezza connector cannot locate the specified data source name. If the ODBC administrator application is not accessible through the File menu by default, use the Windows Explorer to access the application.
- For more information about configuring ODBC connectivity in InfoSphere DataStage and configuring ODBC data source name definitions, see the *IBM InfoSphere Information Server Planning, Installation, and Configuration Guide*.

**Procedure**

1. Configure the ODBC DSN definitions.

   **Table 6. Configuring the ODBC data source name definitions**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX or Linux</td>
<td>Set the <code>ODBCINI</code> environment variable to point to the <code>.odbc.ini</code> file. The <code>.odbc.ini</code> file contains the ODBC DSN definitions. <em>Note:</em> The <code>ODBCINI</code> environment variable is set in the <code>dsenv</code> script automatically as part of the InfoSphere Information Server installation process.</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>The DSN definitions are managed by the ODBC driver manager application included with the operating system. The ODBC DSN definitions must be configured as System DSN definitions in the ODBC data source Administrator. The <code>ODBCINI</code> environment variable is not applicable on Microsoft Windows.</td>
</tr>
</tbody>
</table>

2. Add the path to the directory that contains the client libraries to the library path environment variable. The default path for client libraries is as follows:
   - On Windows, `C:\IBM\InformationServer\ODBCDrivers`. On the Microsoft Windows, the ODBC driver manager library is provided by the operating system. The location of the ODBC driver manager is automatically included in the PATH environment variable.
   - On Linux and UNIX, `/opt/IBM/InformationServer/Server/branded_odbc/lib`. The ODBC driver manager is included with InfoSphere Information Server.

3. On UNIX and Linux computers, to restart the server engine and the ASB Agent, enter the following command:
Configuring access to Informix databases

You must configure environment variables to Informix® CLI, Informix Load, Informix XPS Load, and Informix Enterprise stages so that the jobs can access databases.

Configuring the environment for Informix CLI, Informix Load, and Informix XPS Load stages

You must set the INFORMIXDIR, environment variables for Informix CLI, Informix Load, and Informix XPS Load stages so that the IBM InfoSphere DataStage jobs can access the databases.

Before you begin

- Install the IBM Informix server.

Procedure

1. Set the INFORMIXDIR environment variable to point to the installation directory of the IBM Informix server.
2. Make sure that the PATH environment variable contains $INFORMIXDIR/bin.
3. Make sure that the library path environment variable contains $INFORMIXDIR/lib:$INFORMIXDIR/lib/esql. The following is an example of environment variable settings on an AIX® system.

   \[
   \begin{align*}
   \text{INFORMIXDIR} & = \text{/opt/informix/IDS} \\
   \text{LIBPATH} & = \text{/opt/informix/IDS/lib:/opt/informix/IDS/lib/esql:/opt/informix/IDS/lib/cli} \\
   \text{PATH} & = \text{/opt/informix/IDS/bin}
   \end{align*}
   \]
4. For the Informix CLI stage, if the data source uses a translation DLL, you must add $INFORMIXDIR/lib/esql to the shared library search path. If $INFORMIXDIR/lib/esql is not added, a message that is related to loading translation DLL is logged in the IBM InfoSphere DataStage job log.

Example

The following example shows sample DSN entries for AIX, Solaris, HP, and Linux platforms.

[Informix]
Driver=/home/informix/csdk/lib/cli/iclit09b.so
Description=IBM INFORMIX ODBC DRIVER
Database=<stores_demo>
LogonID=<user_id>
Password=<password>
ServerName=<informixserver>
HostName=<informixhost>
Service=<online>
Protocol=ontlitcp EnableInsertCursors=0 GetDBListFromInformix=0
CLIENT_LOCALE=en.us.8859-1
DB_LOCALE=en.us.8859-1
CursorBehavior=0 CancelDetectInterval=0 TrimBlankFromIndexName=1
Every Informix data source to which the InfoSphere DataStage jobs connect must have an entry in the .odbc.ini file. You must specify values for the Database and Server name properties. The CLIENT_LOCALE and DB_LOCALE fields are optional. If you add login ID (UID) or password (PWD) properties, the User Name and Password properties can be left blank. The values that are specified for the login ID (UID) and password (PWD) properties override the values that are specified for the User Name and Password properties in the .odbc.ini file.

**Configuring the environment for Informix Enterprise stages**

You must have the correct privileges and set the environment variables to use the Informix enterprise stage. You must also have a valid account on the databases to which you connect.

**Before you begin**

- Install the client libraries. To run jobs with Informix XPS stages on AIX systems, install the Informix client sdk 3.5.x version with the Informix XPS server.
- Make sure that Informix XPS server is running.

**Procedure**

1. Set the INFORMIXDIR environment variable to point to the installation directory of the IBM Informix server.
2. Set the INFORMIXSERVER environment variable to point to coserver name of coserver 1 in sqlhosts. Make sure that the coserver is accessible from the node on which you invoke the IBM InfoSphere DataStage job.
3. Set the INFORMIXSQLHOSTS environment variable to point to the sql hosts path. For example, /disk6/informix/informix_runtime/etc/sqlhosts.
4. To run jobs with Informix XPS stages on AIX systems, set the LIBPATH environment variable as follows: LIBPATH=\$APT_ORCHHOME/lib;\$INFORMIXDIR/lib;\$dirname \$DHOME\branded_odbc/lib;\$DHOME/lib;\$DHOME/uvdlls;\$DHOME/java/jre/bin/classic;\$DHOME/java/jre/bin;\$INFORMIXDIR/lib;\$INFORMIXDIR/lib/cli;\$INFORMIXDIR/lib/esql

**Configuring access to Sybase databases**

The Sybase Open Client environment is installed and configured on the InfoSphere Information Server engine tier.

**Before you begin**

- Install and configure the SQL Server or Sybase client software. The BCPLoad stage uses the BCP API in the DBLIB/CTLIB and NetLIB client libraries. You must ensure that these components are installed on the InfoSphere DataStage server that set up as a client to the SQL Server DBMS. For more information, see DBMS documentation.
- Make sure that the Sybase database server is accessible from the Sybase client.
- Create table in the database on the SQL Server.
- Make sure that the database is registered on the Sybase client.
Procedure

1. Set the SYBASE environment variable to point to the Sybase installation directory. For example, export SYBASE=/disk3/Sybase.

2. Set the SYBASE_OCS environment variable to point to the Sybase Open Client directory. For example, export SYBASE_OCS=OCS-12.5. This value indicates the version and release of the Open Client product.

3. Specify the database name, host system name or IP address, and port number in the interfaces file (for example, sql.ini) in the $SYBASE directory.

4. Set the DSQUERY environment variable to the name of the Sybase database server to connect to by default when the server name is not specified in the connection request. If the environment variable is not set, the default value SYBASE is used.

5. Set the PATH and library path environment variable to point to the directory that contains the client libraries.
   - On Windows, %SYBASE\%SYBASE_OCS\%bin and %SYBASE\%SYBASE_OCS\%dll directories, where SYBASE and SYBASE_OCS represent the Sybase product installation home directory and Sybase Open Client directory.
   - On Linux and UNIX, $SYBASE/$SYBASE_OCS/lib.

   Note: Make sure that $SYBASE/$SYBASE_OCS/bin is displayed first in the PATH environment variable.

6. For the BCPLoad stages, configure the database for the fast copy (bulk load) option, by using a stored procedure. When this option is used, data is loaded without recording every insert in a log file. For more information about using stored procedures, see Using Stored Procedures.

7. Get login privileges to Sybase by using a valid Sybase user name and corresponding password, server name, and database. These must be recognized by Sybase before you attempt to access it.

8. Use the dsedit utility that is provided with the Sybase Open Client to configure connection to the Sybase database.

9. Test the connectivity to Sybase database outside of InfoSphere DataStage by using tools such as $SYBASE/$SYBASE_OCS/bin/isql tool in Sybase Open Client.

10. Complete the following steps to access Sybase databases with NLS in Sybase enterprise stages.
    a. Create a database and configure the language of your choice. For example, create database <<database path>> COLLATION 932JPN for a Japanese (shift_jis) database.
    b. Install the IBM InfoSphere DataStage server in that particular language, for example, Japanese (shift_jis).
    c. To set the language for InfoSphere DataStage client, use the NLS tab in job properties to select the language.
    d. Make sure that the selected language is set as default in the operating system of the system on which the InfoSphere DataStage client is installed.

Permissions required to access Sybase databases

To complete operations on tables that are hosted by Sybase ASE and Sybase IQ databases, you require specific privileges on the table.
Table 7. Permissions required to access Sybase databases

<table>
<thead>
<tr>
<th>Operation</th>
<th>Options</th>
<th>Syspartition (only for Sybase ASE)</th>
<th>sysobjects privilege on table</th>
<th>SELECT privilege on table</th>
<th>INSERT privilege on table</th>
<th>Delete table</th>
<th>Create table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>Create/Replace</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Write</td>
<td>Append</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Write</td>
<td>Truncate</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Read</td>
<td>All</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Upsert</td>
<td>Update/Insert</td>
<td>To complete an update operation, you require the UPDATE privilege on the table that you want to update. To complete an insert operation, you require the INSERT privilege on the table into which you want to insert records.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lookup</td>
<td>All</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Configuring access to Microsoft SQL Server databases

To configure access to Microsoft SQL Server, you must set ODBCINI environment variable. You must also ensure that the Microsoft SQL Server database is accessible from the Microsoft SQL Server client and test connectivity between Microsoft SQL Server client and the Microsoft SQL Server databases.

**Before you begin**

- Install client libraries.

**About this task**

When you are connecting to a remote database, ensure that the database server is configured to allow remote connections over the TCP/IP protocol.

The Microsoft SQL Server Client cannot be installed on UNIX. Because of this, the InfoSphere DataStage Dynamic RDBMS plug-in stage on UNIX cannot use the Bulk Insert mode operation when the stage is configured for Microsoft SQL Server database type. The DRS plug-in stage on Windows uses the Microsoft OLE DB API for Bulk Load operations and the API is not available on UNIX. When the DRS plug-in stage is configured for Microsoft SQL Server database on UNIX, the database type for the stage is automatically switched to ODBC.

For more information about configuring access to SQL Server InfoSphere DataStage, see the IBM InfoSphere DataStage and QualityStage Connectivity Guide for Microsoft SQL Server and OLE DB Data.

**Procedure**

1. On UNIX or Linux, set the ODBCINI environment variable to point to the .odbc.ini file in which the Microsoft SQL Server connection definitions are created.

2. From the Microsoft SQL Server driver on Windows, test if the Microsoft SQL Server database is ready to receive incoming connections:
   a. In the Microsoft SQL Server DSN configuration window, specify the connection information for the Microsoft SQL Server database.
   b. Click Finish.
   c. Click Test Data Source.
3. On UNIX, to test if the SQL server database is ready to receive incoming connections:
   a. Open the UNIX command-line utility.
   b. Run the DataDirect ODBC example application from the
      installation_directory/example directory, where installation_directory is the
      path to the DataDirect ODBC example application installation directory.
      [root@RH2011 example]# ./example
      ./example DataDirect Technologies, Inc. ODBC Example Application.

      Enter the data source name : mysqlserver
      Enter the user name : username
      Enter the password : password
      Enter SQL statements (Press ENTER to QUIT)
      SQL>
      Exiting from the Example ODBC program
      [root@RH2011 example]# pwd
      installation_directory/example
      [root@RH2011 example]#

Configuring access to Teradata databases

To configure access to Teradata databases, you must install Teradata tools and Teradata transporters and set the environment variables.

Before you begin

Install database client libraries.

Procedure

1. Install Teradata tools and utilities on all nodes that run parallel jobs. For more information, see the installation instructions in the Teradata documentation.
2. Install the Teradata Parallel Transporter. For more information, see the Teradata product documentation.
3. Set the environment variables.

Table 8. Required Environment variables

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Environment variables</th>
</tr>
</thead>
</table>
| AIX              | TWB_ROOT=/usr/tbuild/08.01.00.02
                  | PATH=$TWB_ROOT/bin:$PATH
                  | LIBPATH=$TWB_ROOT/lib:$LIBPATH
                  | NLS_PATH=$TWB_ROOT/msg/%N
                  | export TWB_ROOT PATH LIBPATH NLS_PATH |
| HP-UX            | TWB_ROOT=/usr/tbuild/08.01.00.02
                  | PATH=$TWB_ROOT/bin:$PATH
                  | SHLIB_PATH=$TWB_ROOT/lib:$SHLIB_PATH
                  | NLS_PATH=$TWB_ROOT/msg/%N
                  | export TWB_ROOT PATH SHLIB_PATH NLS_PATH |
| Solaris          | TWB_ROOT=/usr/tbuild/08.01.00.02
                  | PATH=$TWB_ROOT/bin:$PATH
                  | LD_LIBRARY_PATH=$TWB_ROOT/lib:$LD_LIBRARY_PATH
                  | NLS_PATH=$TWB_ROOT/msg/%N
                  | export TWB_ROOT PATH LD_LIBRARY_PATH NLS_PATH |
Configuring access to Netezza databases

To configure access to Netezza database, you must install and configure the Netezza ODBC driver and create the data source.

Configuring access to Netezza databases on Linux and UNIX

To configure access to Netezza database you must specify parameters in the .odbcinst.ini file to configure the Netezza ODBC driver and also modify the .odbc.ini file to configure the data sources.

Before you begin

- Install client libraries.

About this task

If an .odbcinst.ini configuration file exists, you can modify the same file. If there is no existing .odbcinst.ini configuration file, then you can use the odbcinst.ini.sample to create the .odbcinst.ini configuration file. In most scenarios, you can use the contents of the odbcinst.ini.sample file without any changes. However, in the following scenarios, you must change the configuration file:

- If your client system was configured for ODBC drivers other than the Netezza ODBC driver and you want to continue to use those ODBC drivers, do not modify the existing entries in the .odbcinst.ini file. Add an entry for the Netezza ODBC driver at the end of the existing contents of the .odbcinst.ini file.
- If the Netezza client software and a Netezza ODBC driver were installed on your client system, check if the Netezza ODBC driver is configured. If it is not, add an entry to the end of the existing contents of the .odbcinst.ini file.

If the .odbc.ini configuration file exists in your home directory (For example, /home/myname) check if it contains entries for the Netezza appliance data sources to access. If it does not, copy the contents of the odbc.ini.sample file to the end of your existing .odbc.ini configuration file. Do not modify any existing entries in the file.

If you are using the InfoSphere Information Server version of the .odbc.ini configuration file on Linux, create a symbolic link in the folder where the configuration file exists to make sure that the Netezza connector works correctly:

1. Log on as the InfoSphere DataStage administrator.
2. To change to the installation directory of InfoSphere Information Server, enter the command: cd /opt/IBM/InformationServer/Server/DSEngine.
3. To create a symbolic link, enter the command: ln -s .odbc.ini odbc.ini.

Procedure

1. Log in using your user ID and password.
2. Configure the Netezza ODBC driver.
   b. Modify the configuration entries depending on your requirement. Consult your Netezza system administrator to check if you must modify any specific configuration entries for your installation.
   c. Save the file as .odbcinst.ini.
3. Configure the Netezza appliance data sources.
   a. Copy the contents of odbc.ini.sample file into your home directory (For example, /home/myname) and rename it .odbc.ini.
   b. Optional: To add the Netezza data sources to an existing .odbc.ini file, add the lines after [NZSQL] from the sample file to the existing .odbc.ini file.
      In the [ODBC Data Sources] section, add NZSQL = NetezzaSQL to the list of data source names.
   c. Save and close the file.

4. Set the following environment variables:
   
   ```
   export ODBCINI=path_to_odbc.ini_file
   export NZ_ODBC_INI_PATH=location_of_odbc.ini_file
   ```

   **Note:** If the Netezza entries were added to an existing odbc.ini file, set only the NZ_ODBC_INI_PATH variable.

5. To restart the server engine and the ASB Agent, enter the following command.
   
   ```
   cd Install_directory/Server/DSEngine/bin
   ./uv -admin -stop
   ./uv -admin -start
   cd Install_directory/ASBNode/bin
   ./NodeAgents_env_DS.sh
   ./NodeAgents.sh stopAgent
   ./NodeAgents.sh start
   ```

---

### Configuring access to Netezza databases on Microsoft Windows

If InfoSphere Information Server runs on the Microsoft Windows operating system, you must create and configure the data source after you install the Netezza ODBC driver.

**Before you begin**

- Install database client libraries.
- On 64-bit Windows computers, make sure that you run the 32-bit version of the Microsoft ODBC Data Source Administrator C:\Windows\SysWOW64\odbcad32.exe, as InfoSphere Information Server is a 32-bit application. If you run the 64-bit version of the ODBC administrator application, the Netezza connector fails to locate the specified data source name. If the ODBC administrator application is not accessible through the File menu by default, use the Windows Explorer to access the application.
  - On 32-bit Windows, the 32-bit driver is installed in the C:\Windows\System32 directory.
  - On 64-bit Windows, you can install both 32-bit and 64-bit drivers. The 32-bit driver is installed in the C:\Windows\SysWOW64 directory and 64-bit version is installed in the C:\Windows\System32 directory.

**Procedure**

1. To create the data source:
   a. Do one of the following actions depending on your Operating System:
      - On a 32-bit Windows system, click **Start > Control panel > Administrative Tools > Data Sources (ODBC)**.
      - On a 64-bit Windows system, use Explorer to navigate to C:\Windows\SysWOW64\odbcad32.exe.
   b. On the System DSN page, click **Add**.
c. On the Create New Data Source page, select **NetezzaSQL** as the driver to set up the data source for, and then click **Finish**.

2. To configure the ODBC driver:
   
a. On the Netezza ODBC Driver Setup page, specify details about the data source.
   
b. In the **Server** field, specify the host name or the IP address of the Netezza system to which the ODBC driver connects.
   
c. To test the connection, specify the username and password, and then click **Test Connection**.

### Testing database connections by using the ISA Lite tool

After you establish connection to the databases, test the database connection by running the IBM Support Assistant (ISA) Lite for InfoSphere Information Server tool.

For more information about the ISA Lite tool, see the topic about installation verification and troubleshooting.

### Setting the library path environment variable

To apply an environment variable to all jobs in a project, define the environment variable in the InfoSphere DataStage and QualityStage Administrator. The values that are specified for the library path and path environment variables at the project or job level are appended to the existing system values for these variables.

#### About this task

For example, suppose that directory `/opt/branded_odbc/lib` is specified as the value for the library path environment variable at the project level. Directory `/opt/IBM/InformationServer/Server/branded_odbc/lib`, which contains the same libraries but in a different location is already in the library path that is defined at the operating system level or the `dsenv` script. In this case, the libraries from directory `/opt/IBM/InformationServer/Server/branded_odbc/lib` are loaded when the job runs because this directory appears before directory `/opt/branded_odbc/lib` in the values that are defined for the library path environment variable.

The name of the library path environment variable depends on your operating system.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Library path environment variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows</td>
<td>PATH</td>
</tr>
<tr>
<td>HP-UX</td>
<td>SHLIB_PATH</td>
</tr>
<tr>
<td>IBM AIX</td>
<td>LIBPATH</td>
</tr>
<tr>
<td>Other supported Linux and UNIX operating systems, and HP-IA</td>
<td>LD_LIBRARY_PATH</td>
</tr>
</tbody>
</table>

On Linux or UNIX operating systems, the environment variables can be specified in the `dsenv` script. InfoSphere Information Server installations on Windows operating system do not include the `dsenv` script.
Setting the library path environment variable in the dsenv file

On Linux or UNIX operating systems, you can specify the library path environment variables in the dsenv script. When environment variables are specified in the dsenv script, they apply to all InfoSphere DataStage projects that run under the InfoSphere Information Server engine.

Before you begin

Install the client libraries.

Procedure

1. Log in as the root user.
2. Back up the $ISHOME/Server/DSEngine/dsenv script. $ISHOME is the InfoSphere Information Server installation directory (/opt/IBM/InformationServer by default).
3. Open the dsenv script.
4. Add the path to the directory that contains the client libraries to the library path environment variable.
5. To stop and start the InfoSphere Information Server engine, enter the following commands:
   
   bin/uv –admin –stop
   bin/uv –admin –start

6. To change directory to the ASB Agent home directory, enter the following commands:

   cd Install_directory/ASBNode/bin

7. To stop and start the ASB Agent processes, enter the following commands:

   ./NodeAgents.sh stopAgent
   ./NodeAgents.sh start

Results

After you restart the ASB Agent process, the InfoSphere Information Server domain services (WebSphere Application Server) take approximately a minute to register the event.

Setting the library path environment variable in Windows

On the Windows operating system, both the library path and PATH environment variables are represented by the PATH. For InfoSphere Information Server engine and ASB Agent processes to detect changes in the environment variables, the changes must be made at the system level and the InfoSphere Information Server engine must be restarted.

Before you begin

Install the client libraries.

Procedure

1. To edit the PATH system environment variable, click Environment Variable in Advance System Settings, and then select PATH.
2. Click Edit, then specify the path to the directory containing the client libraries.
3. Click OK.
4. Restart the InfoSphere Information Server engine.
5. Restart the ASB Agent processes.
Chapter 3. Configuring authorizations for WebSphere MQ

To enable the WebSphere MQ connector to access the specified queue manager and queue objects, you must grant certain authorizations to the user ID under whose credentials the connector runs.

The setmqaut command and authorization requirements

Use the setmqaut command to grant and revoke authorizations to user IDs and MQ objects.

You use the WebSphere MQ connector in a job to perform these tasks: to establish a connection to a queue manager, to read messages from source queues so that they can later be processed by other stages in the job, and to take data that is produced by the stages in the job and write it as messages to the target queues.

To enable the connector to access the specified queue manager and queue objects, you must grant certain authorizations to the user ID under whose credentials the connector runs. If the user ID does not have the necessary authorizations to the objects, the connector reports MQ error code 2035 (MQRC_NOT_AUTHORIZED).

After you grant authorizations, the authorization service that is associated with the queue manager inspects the list of authorizations that you grant to the user ID to determine whether the user ID is authorized to perform a particular operation on a particular MQ object.

Two factors determine the IDs to which you grant authorizations: whether the connector runs in Server mode or in Client mode and whether the connector is running in a job or is invoked at design time, for example to test the connector. The mode in which the connector runs also determines the objects for which you set authorizations. In general, you must grant authorizations for the queue manager, queue, and namelist objects that the connector accesses.

Depending on the mode in which the connector runs, you use the setmqaut command to grant one or more of the following authorizations to each MQ object:

- altusr – To specify an alternate user ID to use for opening queues
- browse – To browse messages on queues
- connect – To connect to queue managers
- dlt – To automatically delete the dynamic reply queue when closing it
- dsp – To create dynamic queues that are based on a model queue
- get – To get messages from queues
- inq – To inquire about the attributes of MQ objects
- put – To put messages on queues
- setall – To set identity context fields and origin context fields on messages
- setid – To set the identity context fields on messages

After you use the setmqaut command to grant authorizations to MQ objects, issue the REFRESH SECURITY command to refresh the security cache for the queue manager. For example, the refresh the security settings for the queue manager QMNAME, you issue the following command:

```
runmqsc QMNAME
```
When the MQSC editor opens, issue the following MQ command:

```
REFRESH SECURITY
```

The following response message should display to confirm that the security cache was refreshed:

```
AMQ8560: WebSphere MQ security cache refreshed
```

**User IDs**

When you use the `setmqaut` command to grant an authorization, you must specify the MQ object for which to grant the authorization and the principal or group to which to grant the authorization.

The principals and groups correspond to the underlying operating system (OS) users and groups. On UNIX and Linux systems, granting an authorization to a specific user ID automatically grants the same authorization to all of the other users who are members of the same group to which the specific user ID belongs.

When you grant authorizations to MQ objects, you must know which principal to specify. The principal that you specify depends on whether the WebSphere MQ connector runs in Server mode or Client mode and whether you are using the connector during runtime, in a job that is running, or at design time, for example to test the connection or view messages on the queue from the Stage dialog in IBM InfoSphere DataStage and QualityStage Designer.

When you start a job that contains the connector, the connector code runs as part of the job process. The job process runs under the credentials of the user ID that starts the job. To define the mapping between the user IDs in InfoSphere DataStage and user IDs in the operating system, you use the IBM InfoSphere Information Server console. When the connector runs in Server mode, the MQ authorization is performed for the user ID under which the job runs. When the connector runs in Client mode, the user ID that is used for authorization is specified in the client-connection channel definition.

The connector runs at design time when you perform any of the following tasks from the connector stage dialog:

- Click **Test** to test the current connection definition
- Click **View Data** to view messages on the specified queue
- Click **Select** to list and then select queue managers on queues

When the connector runs at design time, it runs within the ASB Agent process, which is a process that runs on the InfoSphere Information Server engine tier. On Microsoft Windows, this process runs as a Windows service under the built-in Local System account. On UNIX and Linux systems, the process runs as a daemon under the privileged user account.

Therefore, if the connector is running in Server mode, the local system or privileged user ID is used for MQ access-control checking. If the connector is running in Client mode, the user ID that is used for MQ access-control checking is the user specified in the client-connection channel definition.

**Server mode**

Using Server mode, the WebSphere MQ connector connects to the queue manager, which must be on the same computer on which the connector runs.
The connection between the processes is established directly by using inter-process communication (IPC) mechanisms, such as shared memory segments and semaphores, and not by using the network protocol stack of the system. In general, the OS user under whose credentials the connector process runs is the user for which the access-control checking is performed.

**Client mode**

Using Client mode, the WebSphere MQ connector connects to the queue manager through the MQI channel.

The MQI channel is a two-way logical communication link between the IBM WebSphere MQ client (on which the connector runs) and the WebSphere MQ server (on which the queue manager runs). You must define the MQI channel on the WebSphere MQ client and on the WebSphere MQ server. The MQI channel definition on the WebSphere MQ server end of the channel is called the server-connection channel definition. The MQI channel definition on the WebSphere MQ client end of the channel is called the client-connection channel definition. Both channel definitions must have the same name.

When the WebSphere MQ connector issues a request to the WebSphere MQ server, for example to establish connection to a queue manager or to put a message on the queue, the connector does not place the request directly on the channel. Instead, the message channel agent (MCA) on the client side prepares and places the request on the MQI channel on behalf of the connector. The security context of the request, which includes the user ID under which the connector is running, is propagated with the request. The request reaches the WebSphere MQ server where it is picked up by the MCA that is associated with the server-connection channel side of the MQI channel. Then the MCA issues the request on the WebSphere MQ server. The authorization service on the WebSphere MQ server performs access-control checking for the user ID that is associated with the MCA on the server-connection channel. In many cases, the user ID is the user ID that was propagated with the request from the client side and corresponds to the user ID under which the connector is running.

However, you can configure the channel so that the user ID that is associated with the MCA on the server-connection channel is not the same user ID as the one that is associated with the MCA on the client-connection channel. To configure an alternate user ID, you use a security exit or set the MCAUSER attribute on the server-connection channel definition.

A security exit is a program that you write. You specify the program in the server-connection channel and client-connection channel definitions. When the MCA is initiated, the security exit is automatically invoked. You can create a security exit to run an additional customized security routine, such as to use the WebSphere MQ server to authenticate the WebSphere MQ client. A security exit can programmatically specify the user ID to associate with the server-connection channel MCA. Then the authorization service uses that ID when it performs access-control checking.

You can specify the MCAUSER attribute of the server-connection channel when you define the server-connection channel. As the value for the attribute, you can specify the user ID to associate with the MCA on the server-connection channel. This user ID is used for performing access-control checking for requests that come from the channel and pass through that MCA.
In summary, when you set authorizations on the MQ objects for a connector that runs in Client mode, you must set the authorizations for the user ID that is associated with the server-connection channel MCA. The user ID might be the user ID that is propagated from the client-connection MCA (the connector side); or it might be the user ID that is provided by the server-connection channel security exit; or it might be the user ID that is specified directly by setting the MCAUSER attribute value in the server-connection channel definition.

**Queues and modes**

The WebSphere MQ connector supports a variety of different queue types and can run in reply/request or in publish/subscribe mode.

See the following topics for information about each queue and mode type:

**Alias queues**

Before you use an alias queue, be sure that you understand how access-control checking is performed for the queue.

An alias queue is an object that references other queues that are called base queues. The WebSphere MQ connector can read messages from an alias queue and write messages to an alias queue. Whether or not an alias queue is being used is completely transparent to the connector. The connector treats an alias queue like any other queue. To configure the connector to access an alias queue, you specify the name of the alias queue wherever the name of the queue is required.

Note that access-control checking is performed only for the alias queue and is not performed for the base queue. Consequently, a user ID might be able to get, put, and browse messages that are on a queue for which the user ID does not have the appropriate authorization. For example, if user ID test is not granted put authorization for queue QUEUE1 but the alias queue AQUEUE1 is defined and the user ID test is granted put authorization for AQUEUE1, user ID test can put messages into the alias queue AQUEUE1, which in turn places the messages into the base queue QUEUE1 even though the user ID test does not have the authorization to put messages on QUEUE1.

To grant an authorization for an alias queue, you use the same format for the setmqaut command that you use when you grant an authorization for any other type of queue. For example, the following command grants put authorization for the alias queue AQUEUE1 to the user ID test:

```bash
setmqaut -m QMNAME -t queue -n AQUEUE1 -p test +put
```

**Remote queues**

The WebSphere MQ connector can put a message on a queue that resides on a remote queue manager that is not the same queue manager to which the connector is connected.

To put a message on a remote queue, the queue manager to which the connector is connected must have a remote queue definition that points to the queue on the remote queue manager. In addition, a sender/receiver channel must be defined between the two queue managers. Then the connector puts the message on the remote queue definition on the queue manager to which it is connected, and IBM WebSphere MQ moves the messages through the channel to the queue that resides on the remote queue manager.
To move a message over the channel to the remote queue manager, the queue manager to which the connector is connected uses an intermediary queue that is called the transmission queue.

To configure the connector to write messages to the remote queue, you specify the name of the remote queue definition the same way that you specify the name of a local queue, either once for all messages by entering the name in the **Queue name** property or for each message separately by creating a WSMQ.QUEUEENAME data element column on the input link.

You grant the same authorizations to a remote queue definition as you grant to a local queue. Therefore, you must grant put authorization for the remote queue definition. For example, the following command grants put authorization to the user ID test for the remote queue definition QUEUE1_REMOTE on the queue manager QM_A, which points to the remote queue QUEUE1, which the remote queue manager QM_B hosts:

```
setmqaut -m QM_A -t queue -n QUEUE1_REMOTE -p test +put
```

Note that you grant the authorizations only for the remote queue definition that is hosted by the queue manager to which the connector connects. You must define and configure the channel between the queue managers; then the communication that takes place between the queue managers to move a message to its destination and the authorization that takes place is completely transparent to the connector.

**Shared cluster queues**

A shared cluster queue is shared across multiple queue managers that are organized into a queue manager cluster.

To configure the WebSphere MQ connector to write messages to a shared cluster queue, you set the **Cluster queue** property to **Yes**. A shared cluster queue is shared across multiple queue managers that are organized into a queue manager cluster. You define the cluster queue instance on only one queue manager in the cluster or on multiple queue managers in the cluster. The connector connects to one queue manager in the cluster and can then be configured to write messages to the shared cluster queue.

You can configure the connector to allow IBM WebSphere MQ to select the shared cluster queue instance to which to write the messages, or you can configure the connector to write the messages to a specific queue instance.

WebSphere MQ can determine the cluster queue instance when it first opens the cluster queue (**On open**) or for each separate message (**Not fixed**). **Not fixed** might be used to support workload balancing. Also, WebSphere MQ can be configured to use the mode specified in the cluster queue definition (**As in queue definition**). You specify the mode to use for selecting the cluster queue instance (**On open**, **Not fixed**, or **As in queue definition**) in the **Usage > Other queue settings > Cluster queue > Binding mode** property.

When it is necessary to write messages to a specific cluster queue instance, you enter the name of the queue manager that hosts the queue instance in the **Usage > Other queue settings > Cluster queue > Queue manager** property.
The connector can send messages across multiple cluster queue instances in the cluster or to one cluster queue instance that WebSphere MQ determines dynamically at runtime or that you specify in the connector properties when you design the job.

If the messages that the connector sends to the cluster queue might be routed to queue instances that are handled by a cluster queue manager that is different from the queue manager to which the connector is connected, you must grant the put authorization for the queue SYSTEM.CLUSTER.TRANSMIT.QUEUE. For example, you must grant the put authorization for the queue SYSTEM.CLUSTER.TRANSMIT.QUEUE when the queue manager to which the connector is connected does not host its own instance of the shared cluster queue.

If the messages that the connector sends to the cluster queue will all be sent to the local queue instance that resides on the queue manager to which the connector is connected, you must grant the put authorization for that local queue.

If there is a chance that the messages will be sent to the local instance or remote instance of the shared cluster queue, you must grant put authorization for both the local queue instance and for the SYSTEM.CLUSTER.TRANSMIT.QUEUE system queue.

**Error queues**

You can configure an error queue to hold messages that were not delivered to the output link or put on the target queue.

When the WebSphere MQ connector reads a message and then deletes it from the source queue but fails to deliver the message to the output link, the message is rolled back on the queue, assuming that the Usage > Message read mode property is set to Delete (under transaction). The message in error and any other messages that were read from the queue in the same transaction are all rolled back. To specify a separate queue, referred to as an error queue, on the queue manager and configure the connector to move the rolled-back messages to the error queue, you set the Usage > Error queue property to Yes and enter the name of the error queue in the Usage > Error queue > Queue name property.

Another scenario for using the error queue is when the connector fails to put a message on the target queue. In this case, if an error queue is specified, the connector tries to put the message on the error queue within the same transaction instead of automatically rolling back the transaction. If the connector successfully puts the message on the error queue, the connector continues processing data and does not roll back the transaction. In other words, the error queue acts as a backup queue that holds messages that cannot be put on the target queue. To configure the connector to put messages on the error queue, you must grant put authorization for the error queue.

In addition, you can configure the connector to preserve the identity context and origin context fields from the original message header; otherwise, the queue manager produces new default values for the fields. To preserve the identity context field values from the original message, you set the Usage > Error queue > Context mode property to Set identity and grant the setid authorization for the error queue and for the queue manager that hosts the error queue.

When the connector is used in Client mode, the error queue does not need to reside under the same queue manager as the main queue to which the connector is
putting messages or from which it is reading messages. You specify the name of
the queue manager for the error queue in the Usage > Error queue > Queue
manager property. It is not possible to specify a client-channel definition for the
connection to this queue manager. Instead, you use the environment variable
MQSERVER, or you use the MQCHLLIB and MQCHLTAB environment variables.

If the error queue is a remote definition queue, WebSphere MQ moves the
messages that the connector places on the remote queue definition in the connected
queue manager to the actual error queue on the remote queue manager.

**Request/Reply mode**

When the WebSphere MQ connector has both input and output links defined, the
connector runs in request/reply mode.

In request/reply mode, the connector reads each record on the input link and puts
it as a message on the request queue. Then for each request message, the connector
waits for the response message to arrive on the reply queue, reads the response
message from the reply queue, and propagates the response message to the output
link.

For the connector to put a message on the request queue, you must grant
authorizations for the request queue. At a minimum, you must grant the put
authorization for the queue. If the request message overrides the default values for
the message header fields that constitute the identity context or origin context of
the message, you must also grant the setid and setall authorizations.

For example, the following command grants the put authorization for the request
queue REQUESTQ, which is hosted by the queue manager QMNAME, to the user
ID test:

```
setmqaut -m QMNAME -t queue -n REQUESTQ -p test +put
```

When the connector puts a message on the request queue, the connector sets the
value of the ReplyToQ message header field to the name of the reply queue on
which the response message will be put. There are two ways to specify this value:

- You can set the value once for all request messages by entering the name of the
  reply queue in the Usage > Set header fields > Reply to queue property.
- You can set the value separately for each request message by defining a column
  with WSMQ.REPLYTOQ data element on the input link. Then the value of this
  field for each input record represents the reply queue name to associate with the
  request message that corresponds to that input record.

If the reply queue name for a request message is specified both ways, the queue
name that is included with the input data is used.

For the connector to read the response message, an external entity, for example
another job, must read the request message from the request queue, possibly
process the message, and then provide the response message on the reply queue.

To configure the connector to get the response message from the reply queue, you
must, at a minimum, grant the get authorization for the reply queue. If the
connector is configured to browse messages from the reply queue instead of
retrieving and then deleting them, you must also grant the browse authorization.
For example, the following command grants the get and browse authorizations for the reply queue REPLYQ, which is hosted by the queue manager QMNAME, to the user ID test:

```bash
setmqaut -m QMNAME -t queue -n REPLYQ -p test +get +browse
```

Note that when the WebSphere MQ connector is configured to run in request/reply mode, the reply queue must be on the same queue manager as the request queue.

To use a dynamic queue as the reply queue, you must specify a model name. At a minimum, you must grant get and browse authorizations for the model queue because the connector opens the dynamic queue and reads the response messages from it, and you must grant dsp authorization for the model queue so that the dynamic queue can be created. For example, the following command grants the get, browse, and dsp authorizations for the model queue MQUEUE1, which is under queue manager QMNAME, to the user ID test:

```bash
setmqaut -m QMNAME -t queue -n MQUEUE1 -p test +get +browse +dsp
```

When the reply queue is a dynamic queue, you can configure the connector to delete the reply queue when it closes the queue. Set the Usage > Other queue settings > Dynamic queue > Reply queue close options property to Delete or Purge and delete. When the property is set to Delete, the connector tries to delete the reply queue. If the queue is not empty, the queue is not deleted and a warning message is logged. When the property is set to Purge and delete, the connector requests that the queue manager clear the queue of any messages and then deletes it.

When a dynamic queue is created on behalf of the connector, it is not necessary to grant dlt authorization for the dynamic queue. However, there is one scenario in which it is necessary to grant dlt authorization for the reply queue. This is the case when the permanent dynamic queue already exists when the connector opens it to use it as the reply queue. In other words, you do not specify a model queue name for the queue manager to create the dynamic queue on behalf of the connector, but instead directly specify the name of the permanent dynamic queue as the reply queue. In this case, you can set the Usage > Other queue settings > Dynamic queue > Reply queue close options property to Delete or Purge and delete, and the connector will try to delete the reply queue when it closes. For the connector to be able to delete the queue, you must grant the dlt authorization for the queue. Otherwise, the connector cannot delete the queue. Note that in this case, only a warning message is logged; the job does not fail.

Deleting a queue when closing it is allowed only for a dynamic queue. Therefore, if you specify a non-dynamic queue as the reply queue and set the Usage > Other queue settings > Dynamic queue > Reply queue close options property to Delete or Purge and delete, the connector tries to delete the reply queue when it closes it; and the MQ error code 2045 (MQRC_OPTION_NOT_VALID_FOR_TYPE) displays.

**Publish/Subscribe mode**

You can configure the WebSphere MQ connector to send and receive publication messages and to automatically register and deregister subscriptions with the broker.

When the Usage > Publish/Subscribe property is set to Yes, the connector is configured for publish/subscribe mode.
If the connector has an input link, the connector works as a publisher and provides publication messages for a specific topic to the message broker, which then distributes the messages to the subscribers who subscribed to that topic. If the connector has an output link, the connector works as a subscriber and reads publication messages from the subscriber queue that it specified when it registered for the specific topic. If the connector has both input and output links, it cannot run in publish/subscribe mode.

The connector supports two message broker products: IBM WebSphere MQ and IBM WebSphere MQ Message Broker.

The connector communicates with the message broker to exchange registrations and publications. When the connector has an input link, the connector acts as a publisher. You can configure the connector to perform these tasks:

- Register itself as a publisher on particular topics before sending any publications on those topics
- Send publications in the form of MQ messages
- Deregister itself

Note that when the connector acts as a publisher, the connector can register and deregister itself only when it is used with WebSphere MQ, not when it is used with WebSphere MQ Message Broker.

When the connector has an output link, the connector acts as a subscriber. You can configure the connector to perform these tasks:

- Register itself as a subscriber on particular topics and specify the queue on which to receive publications
- Read publications
- Deregister itself

Note that when the connector acts as a subscriber, the connector can register and deregister itself with WebSphere MQ and with WebSphere MQ Message Broker.

To register or deregister as a publisher or subscriber and to publish or receive publications, the connector exchanges command messages with the queue manager that runs in the message broker product. When WebSphere MQ is used as the message broker product, you select the queue manager to use and use the strmqbrk command to start the message broker service. When WebSphere MQ Message Broker is used as the message broker product, you specify the queue manager when you configure WebSphere MQ Message Broker.

When the connector and WebSphere MQ are used in publish/subscribe mode, the command messages that they exchange are in MQRFH2 message format. In this case, you must set the Usage > Publish/Subscribe > Service type connector property to MQRFH2.

When the connector is used in publish/subscribe mode, you must grant authorizations for the MQ objects. The message broker that you use, either WebSphere MQ or WebSphere MQ Message broker, and the capacity in which the connector is used, either publisher or subscriber, determine the authorizations that you grant.
The connector connects to the queue manager that acts as the message broker. You must grant connect andinq authorizations for the queue manager so that the connector can connect to it and then check the queue manager’s CCSID attribute value.

The connector sends command messages to the control queue SYSTEM.BROKER.CONTROL.QUEUE. The message broker reads the messages from this queue, processes them, and provides responses for the connector on the reply queue which you specify in the Usage > Publish/Subscribe > Reply queue property.

The connector sends command messages to the message broker to register or deregister as a subscriber. WebSphere MQ can register and deregister as a publisher and send command messages to the control queue. The connector waits for the response messages from the message broker, reads the messages from the queue, and determines if the message broker successfully processed the command.

For the connector to put messages on the SYSTEM.BROKER.CONTROL.QUEUE queue, you must grant the put authorization for this queue. For example, the following command grants put authorization for this queue, which resides under the queue manager QMNAME, to user ID test:

```
setmqaut -m QMNAME -t queue -n SYSTEM.BROKER.CONTROL.QUEUE -p test +put
```

For the message broker to put response messages on the reply queue and for the connector to read messages from the reply queue, you must grant the put and get authorizations for the reply queue. The connector always reads and deletes response messages from the message broker. For example, if the Usage > Publish/Subscribe > Reply queue property is set to REPLYQ, the following command grants get authorization for the reply queue under the queue manager QMNAME to user ID test:

```
setmqaut -m QMNAME -t queue -n REPLYQ -p test +get +put
```

To configure the connector to use a dynamic queue for the reply queue on which to receive response messages from the broker, set the Usage > Publish/Subscribe > Reply queue property to the name of the model queue that the queue manager will use as a template for creating the dynamic queue. Then set the Usage > Publish/Subscribe > Reply queue > Dynamic reply queue property to Yes, and enter the name of the dynamic queue in the Usage > Publish/Subscribe > Reply queue > Dynamic reply queue > Queue name property. To specify the name, use one of the following:

- The full name for the queue.
- The initial portion of the name (fewer than 33 characters), followed by an asterisk (*). In this case, the queue manager uses the initial portion of the queue and adds a suffix to produce a unique queue name.
- An asterisk (*). In this case, the queue manager automatically produces the complete queue name.

You must grant put, get, and dsp authorizations for the specified model queue.

When the connector acts as a publisher with WebSphere MQ, you must grant the put authorization for the message broker's default stream queue SYSTEM.BROKER.DEFAULT.STREAM. The connector publishes publication messages on this stream queue. When the connector acts as a publisher with WebSphere MQ Message Broker, you must grant the put authorization for the queue that is sued as the input queue in the message flow that contains the
publication node. In this case, the WebSphere MQ Message Broker administrator must also grant the authorization for publishing publications on the topic for which the connector is configured to publish.

In addition, when the connector acts as a subscriber with WebSphere MQ, you must grant the browse authorization for the message broker’s stream queue on which the publications are published. You specify the name of the stream queue in the Usage > Publish/Subscribe > Stream name property. The default value for this property is SYSTEM.BROKER.DEFAULT.STREAM., which matches the default stream name for the message broker. This is the stream to which the connector sends publications when it works as a publisher. For example, the following command grants the browse and put authorizations for this queue, which resides under queue manager QMNAME, to the user ID test:

```
setmqaut -m AMNAME -t queue -n SYSTEM.BROKER.DEFAULT.STREAM -p test +browse +put
```

Whether you use WebSphere MQ or WebSphere MQ Message Broker, when the connector acts as a subscriber, you must grant get authorization or grant both browse and get authorizations on the queue that is associated with the subscription on which the connector receives publications. The connector reads publications from this source queue. The authorizations to grant depend on how you configure the connector to read messages from this queue.

### Specifying the client-connection channel definition

When the WebSphere MQ connector runs in Client mode, you must specify the client-connection channel definition.

**About this task**

**Procedure**

The three ways to specify the client-connection channel definition are listed below in the order in which the connector checks for the definition:

- Complete the **Channel name** (specify the name of the client-connection channel through which messages are sent from the connector to the remote queue manager), **Transport type** (select the transport protocol for this client connection), and **Connection name** (specify the name of the client connection for this connector) properties, which are located under the **Connection > Client channel** definition property.

  **Note:** If you use this method to specify the definition, you cannot specify additional client-connection channel settings. For example, you cannot specify a security exit or SSL settings to use on the channel for the data exchange.

- Use the MQSERVER environment variable to specify the client-connection channel definition in the format **ChannelName/TransportType/ConnectionName**.

  **Note:** If you use this method to specify the definition, you cannot specify additional client-connection channel settings. For example, you cannot specify a security exit or SSL settings to use on the channel for the data exchange.

- Use the MQCHLLIB and MQCHLTAB environment variables to point to the location and name of the client channel definition table file. This file, which you define on the WebSphere MQ server machine on which the target queue manager runs, contains client-connection channel definitions for connecting to that queue manager. After you create the file, you copy it from the WebSphere.
MQ server machine to the WebSphere MQ client machine that the WebSphere MQ client uses to connect to the queue manager.

**Note:** You can use this method to specify additional client-connection channel settings. You define the client-connection channel on the target queue manager, copy the generated client channel definition table file from the WebSphere MQ server to the WebSphere MQ client, and then configure the MQCHLLIB and MQCHLTAB environment variables to point to that file.

### Specifying the user name and password

If you use the WebSphere MQ connector in a Microsoft Windows environment, you can specify a user name and password to associate the current process with the alternate OS user.

**About this task**

The connector does not pass the user name and password to IBM WebSphere MQ for security checking. Instead, the connector uses these properties to impersonate the specified user so that the current process is associated with the alternate OS user. The connector uses the LogonUser and ImpersonateLoggedOnUser Windows API methods to perform the impersonation. When the connector attempts to connect to the queue manager, the credentials of the alternate OS user are used instead of the credentials of the OS user for whom the process was originally started.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor and then go to the **Properties** tab.
2. For the **Connection > Username** property, use the format `domain name\user name` to enter the user name.
3. For the **Connection > Password** property, enter the password for the user.

### Reading messages from the queue

When the WebSphere MQ connector has an output link, the connector reads messages from the queue and provides the message data on the output link so that downstream stages in the job can process and consume the message data.

**About this task**

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor and then go to the **Properties** tab.
2. For the **Usage > Queue name** property, enter the name of the queue from which the connector reads messages.
3. For the **Usage > Message read mode** property, choose how to read messages during the current transaction:
   - Choose **Keep** to read messages and then keep them on the queue. Then use setmqaut command to grant the get and browse authorities for the queue.
   - Choose **Delete** or **Delete (under transaction)** to read and then delete messages from the queue, a process that is known as destructively reading messages. Then use the setmqaut command to grant the get authorization for the queue.
• Choose **Move to work queue** to move the messages to the work queue.

**Example**

The following command grants the get and browse authorities for the queue QUEUE1 which is hosted by the queue manager QMNAME to the user ID test:

```bash
setmqaut -m QMNAME -t queue -n QUEUE1 -p test +get +browse
```

### Configuring cursor refreshing

Configure the WebSphere MQ connector to receive messages by using a cursor that can be repositioned to the top of the input queue after a specified amount of time.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor and then go to the **Properties** tab.
2. For the **Usage > Refresh** property, choose **Yes**.
3. For the **Usage > Period** property, enter the number of seconds to read messages from the input queue before repositioning the cursor to the top of that queue. The default value is -1, which specifies that the cursor is repositioned when the end of the queue is reached. To closely monitor high-priority messages, enter 0. Then the cursor is repositioned each time that a new message is read.
4. Use the setmqaut command to grant the get and browse authorizations to the queue.

**Note:** You must grant these authorities, regardless of the value that is specified for the **Message read mode** property.

### Passing message data by reference

To pass message data by reference, the WebSphere MQ connector creates and then passes to the output link a locator string that identifies the message on the source queue.

**About this task**

The downstream stage that consumes the message must be a LOB-aware stage, such as the DB2 connector or the Oracle connector. To retrieve the actual message payload data, the downstream stage invokes the WebSphere MQ connector and passes it the locator string. Then the WebSphere MQ connector uses the locator string to locate and retrieve the corresponding message from the queue and provides the message data to the target stage. In this case, the data exchange is made within the running process of the target stage and does not flow through the job.

If the **Message read mode** property in the connector is set to **Delete** or **Delete (under transaction)**, the WebSphere MQ connector must still peek at the message on the source queue in order to construct the locator string. The WebSphere MQ connector retrieves and then deletes the message from the queue only after the downstream connector stage consumes the actual message data.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor and then go to the **Properties** tab.
2. For the Usage > Enable payload reference property, choose Yes.
3. Use the setmqaut command to grant the get and browse authorizations to the queue.

Filtering messages

When you use the WebSphere MQ connector to read messages from a source queue, you can define a filter condition.

About this task

You can define the following message header field values in the filter: MsgId, CorrelId, GroupId, MsgSeqNum, and Offset. To set the values of these headers in the WebSphere MQ connector, you use the corresponding message header field properties that are under the Usage > Filter messages property.

The connector also supports filtering messages based on the value of additional header fields. For these additional header fields, the connector does not rely on IBM WebSphere MQ to perform the filtering, but instead implements the filtering logic itself. The connector inspects each message that it receives from the queue to make sure that the message meets the conditions that the filter specifies.

For example, if you enter 4 for the Usage > Filter messages > Priority property and choose Delete for the Usage > Message read mode property, the connector browses each message on the queue to determine if the message priority is 4. If the priority is 4, the connector deletes the current message, which is the current message under the cursor, from the queue and provides the message data on the output link. If the priority is not 4, the connector moves the cursor to the next message on the queue and continues browsing and checking messages.

Procedure

1. Double-click the connector with the output link on the job canvas to open the stage editor and go to the Properties tab.
2. For the Usage > Filter messages property, choose Yes.
3. Configure the message header field properties.
4. If you configured the Message ID, Correlation ID, Group ID, Message sequence number, or Offset properties, use the setmqaut command to grant the get authorization to the source queue. If you configured any other property, you must use the setmqaut command to grant the browse and get authorizations to the source queue.

Writing messages to one or more queues

The connector reads the data records that arrive on the input link and writes the records as messages to the specified target queue or queues.

About this task

The WebSphere MQ connector supports writing messages to a single queue or to multiple queues which are grouped together into a namelist object. To write messages to a queue, the connector must have an input link.
Procedure
1. Double-click the connector on the job canvas to open the stage editor and then go to the Properties tab.
2. For the Usage > Queue name property, enter the name of the queue or namelist to which the connector writes messages. As an alternative, the message data itself can specify the queue or namelist name. To configure the message data to include the queue or namelist name, you can define a column with the WSMQ.QUEUENAME data element on the input link. The value of this field in each input record specifies the name of the queue or namelist that stores that record as a message.
3. Use the setmqaut command to grant the user put authorization for each queue. If you use the WSMQ.QUEUENAME column on the input link to specify the queue name or namelist for each input message, you must grant the put authorization for all of the queue names that the data might specify.
4. If you specify a namelist, you must use the stemqaut command to grant the inq authorization for the namelist object.

Example
The following command grants the put authorization for the queue QUEUE1 under the queue manager QMNAME to the user ID test:

```
setmqaut -m QMNAME -t queue -n QUEUE1 -p test +put
```

The following command grants the inq authorization for the namelist NAMELIST1 under queue manager QMNAME1 to user ID test:

```
setmqaut -m QMNAME -t namelist -n NAMELIST1 -p test +inq
```

Writing messages to a dynamic queue

A dynamic queue is one that the queue manager creates by using the specified model queue as a template.

About this task

Procedure
1. Double-click the connector on the job canvas to open the stage editor and then go to Properties tab.
2. Enter the name of the model queue in the Usage > Queue name property.
3. In the Usage > Other queue settings section, set the value of the Dynamic queue property to Yes, and then enter one of the following strings as the value for the Queue name property:
   - A complete dynamic queue name.
   - The initial portion, or stem, of the queue name, followed by an asterisk (*). The queue name cannot exceed 33 characters. The queue manager adds the suffix to the specified stem to produce a unique queue name.
   - A single asterisk (*). In this case, the queue manager produces the complete queue name.
4. Use the setmqaut command to grant the put and dsp authorizations for the model queue. When the connector closes the dynamic queue, the connector does not try to delete the queue. Therefore, you do not grant dlt authorization for the model queue.
Note: If the model queue is used to create a dynamic queue that is used as a reply queue when the connector is configured to run in reply/request mode, you must grant put, get, and dsp authorizations.

Example

The following command grants the put and dsp authorizations for the model queue MQNAME1 under the queue manager QMNAME to the user ID test:
setmqaut -m QMNAME -t queue -n MQNAME1 -p test +put +dsp

Setting the value of message header fields

Each MQ message includes a message header that contains message header fields. You can let WebSphere MQ set the values for these fields, or you can configure the WebSphere MQ connector to set the value for each message or for all messages.

About this task

By default, IBM WebSphere MQ sets the values for the message header fields.

To set the message header field values for each message separately, you define one or more WSMQ data element columns on the input link. Each column corresponds to the message header field for which the value needs to be set. For example, to set Priority and MsgId message header field values for each message, you define columns with the WSMQ.PRIORITY and WSMQ.MSGID data element values. Then the value of these fields in each input record specifies the MsgId and Priority message header field values to set for the corresponding message.

To set the message header field values once for all messages, you enter the values in the corresponding properties that appear under the Set header fields property on the Properties page. For example, to use the value 4 for the Priority message header field for all messages, enter 4 as the value in the Priority property.

When a message header field value is specified both through the connector property and in input data through the data element column on the input link, the value in the input data is used.

Procedure

1. To set values for the identity context of the message, which contains the UserIdentifier, AccountingToken, and ApplIdentityData message header fields, perform these steps:
   a. For the Context mode property, choose Set identity.
   b. Use the setmqaut command to grant the setid authorization for the queue manager that hosts the target queue. For example, the following command grants the setid authorizations for the queue manager QMNAME to the user ID test:
      setmqaut -m QMNAME -t qmgr -p test +setid
   c. Use the setmqaut command to grant the setid authorization for the queue. For example, the following command grants the setid authorization for the queue QUEUE1 under the queue manager QMNAME to the user ID test:
      setmqaut -m QMNAME -t queue -n QUEUE1 -p test +setid
2. To set values for the origin context of the message, which contains the PutAppIdType, PutAppName, PutData, PutTime, and ApplOriginData message header fields, perform these steps:
a. For the **Context mode** property, choose **Set all**.

b. Use the `setmqaut` command to grant the setall authorization for the queue manager that hosts the target queue. For example, the following command grants the setall authorization for the queue manager QMNAME to the user ID test:

   ```bash
   setmqaut -m QMNAME -t qmgr -p test +setall
   ```

c. Use the `setmqaut` command to grant the setall authorization for the queue. For example, the following command grants the setall authorization for the queue QUEUE1 under the queue manager QMNAME to the user ID test:

   ```bash
   setmqaut -m QMNAME -t queue -n QUEUE1 -p test +setall
   ```

---

**Using an alternate user ID to open a queue**

To explicitly specify the user ID to use for access-control checking when opening the queue, configure an alternate user ID.

**About this task**

**Procedure**

1. For the **Other queue settings** property, choose **Yes**.
2. For the **Alternate user ID** property, enter the user ID value to use instead of the current user ID for access-control checking when opening the queue.
3. If IBM WebSphere MQ runs on Microsoft Windows, enter the additional Microsoft Windows security ID (SID) value, which is used to identify a particular user account, in the **Value** property. You can enter up to 40 characters, or you can enter up to 80 characters if you specify arrays of pairs of hexadecimal digits and set the **Hex** property to **Yes**.
4. Use the `setmqaut` command to grant the altusr authorization for the queue manager that host the queue that the connector opens. For example, the following command enables the user ID test to open queues that the queue manager QMNAME hosts by using an alternative user ID for access-control checking:

   ```bash
   setmqaut -m QMNAME -t qmgr -p test +altusr
   ```

---

**Connecting to the queue manager**

The queue manager owns and manages the queues that are used by the WebSphere MQ application. Before the WebSphere MQ connector can open a queue to read and write messages, you must define the connection to the queue manager that hosts that particular queue.

**About this task**

When the connector runs in Server mode or when the connector runs in Client mode and you use a client-channel definition table, you must specify the queue manager to which the connector connects. When the connector runs in Client mode, the queue manager to which the connector connects is implicitly specified through the client-connection channel definition. In this case, you do not need to specify a value for the **Queue manager** property. However, if you do specify a value for the property, the value must match the name of the queue manager for which the client-connection channel is defined.

When the connector is configure to obtain the client-connection channel definition from the client-channel definition table that is referenced by the MQCHLLIB and
MQCHLTAB environment variables and the client-channel definition table contains more than one client-connection channel definition, you use the Queue manager property to reference a specific client-connection channel definition from the table.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor and then go to the Properties tab.
2. For the Connection > Queue manager property, enter or select the name of the queue manager.
3. Use the setmqaut command to grant the connect and inq authorizations to the specified queue manager. The following command grants connect authorization to the user ID test for the queue manager QMNAME:
   ```
   setmqaut -m QMNAME -t qmgr -p test +connect
   ```
   After connecting to the queue manager, the connector opens the queue manager object to inquire about the value of the CCSID queue manager attribute. The connector uses this value to perform data conversions, as necessary. The following command grants the inq authorization to the user ID test for the queue manager QMNAME:
   ```
   setmqaut -m QMNAME -t qmgr -p test +inq
   ```
4. Click Test to test the connection.

**Configuring the display of queue names**

For properties where you enter a queue name, the WebSphere MQ connector can display a list of queues that are available for a particular queue manager.

**Before you begin**

Before you configure the display of queue names, complete these prerequisite tasks:
- Configure a connection to the queue manager. If you do not have a valid connection to the queue manager, error code 2059 (MQRC_Q_MGR_NOT_AVAILABLE) or another similar error code appears.
- Check that the Command Server is running for the specified queue manager.

**Procedure**

1. Use the setmqaut command to grant the put and inq authorizations for the SYSTEM.ADMIN.COMMAND.QUEUE queue so that the connector can send commands.
2. Use the setmqaut command to grant the get, inq, and dsp authorizations for the SYSTEM.DEFAULT.MODEL.QUEUE so that the connector can process responses.
3. Use the setmqaut command to grant the dsp authorization for all queues that reside under the queue manager. The connector evaluates all of the collected queue objects to check their names and usage types. Note that for some queues, such as the SYSTEM.AUTH.DATA.QUEUE queue, access is granted only to users in the mqm group and on Microsoft Windows, to the users in the Administrators group and to the built-in Local System account. On Windows, the ASB Agent service runs under the Local System user ID, so the access is automatically granted. On UNIX and Linux, the ASB Agent daemon runs under the privileged user ID. Therefore, you must add the privileged user ID to the mqm group. If your company's security policies prevent adding the privileged
user to the mqm group, clicking the Select button does not display a list of queue names. You must manually enter the queue name.

### Configuring the display of messages on a queue

You can view the messages that display on a queue.

**About this task**

**Procedure**

1. Use the setmqaut command to grant the get and browse authorizations for the queue to the user under which the ASB Agent service runs (Microsoft Windows) or under which the ASB Agent daemon runs (UNIX and Linux).

2. To view the messages on a queue, click the **View Data** link that displays on the **Usage** bar in the WebSphere MQ connector stage dialog.

### Configuring the display of queue manager names

For properties where you enter a queue manager name, the WebSphere MQ connector can display a list of available queue managers.

**About this task**

How you configure the connector to obtain a list of available queue managers depends on whether the computer is running Microsoft Windows or UNIX or Linux.

**Procedure**

1. To configure the display of queue manager names when you click the Select button beside a property name, perform one of the following tasks:
   - On Microsoft Windows, you must grant Read access for the registry keys to the user under which the ASB Agent service runs. The connector queries the Windows Registry and looks for the keys under the key HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager. The keys that the connector collects correspond to the queue manager names.
   - On UNIX or Linux, you must grant read permission for the /var/mqm/mqs.ini file to the user ID under which the ASB Agent daemon process runs. The connector obtains the names of the available queue managers from this file.

2. To configure the display of cluster queue manager names, perform these additional steps:
   a. Make sure that the Command Server is running for the queue manager to which the connector is connected.
   b. Use the setmqaut command to grant the put and inq authorizations for the SYSTEM.ADMIN.COMMAND.QUEUE queue so that the connector can send the commands that it uses to communicate with the Command Server.
   c. Use the setmqaut command to grant the get, inq, and dsp authorizations for the SYSTEM.DEFAULT.MODEL.QUEUE so that the connector can process the requests.
WebSphere MQ connector and the Distributed Transaction stage

You can design a job that reads messages from the MQ queue, processes the messages in the job, and stores the results in one or more databases, while ensuring that the database operations are all done in a single distributed transaction.

In this type of job design, the WebSphere MQ connector reads messages from a source queue to which an external entity, such as another job, feeds messages that contain business transaction data. You can configure the connector to read the messages from this queue in browse mode and deliver them to the downstream stages for processing or to read the messages in destructive mode, in which case the connector moves the messages in a local transaction from the source queue to another queue called the work queue before delivering the message data to the downstream stage in the job.

The message data that is delivered downstream includes the message identifier of the corresponding source message. When the message data reaches the Distributed Transaction stage, this stage destructively reads the message from the work queue (or the source queue if the work queue is not used) and inserts the data to the target databases, all within the same distributed transaction that is managed by the local MQ queue manager, which is the same queue manager on which the source queue and work queue (if it is used) reside.

To ensure that this job works, you must grant the connect and inq authorizations for the queue manager to which the connector and Distributed Transaction stage connect and that at the same time acts as the distributed transaction manager and grant certain authorizations for the source queue and work queue so that the MQ connector and the Distributed Transaction stage can access them.

If the job includes a source queue that does not use a work queue, you must set the Message read mode property in the connector to Keep. Then you must grant the get and browse authorizations for the source queue. In this case, the connector reads the messages in browse mode, and the Distributed Transaction stage reads them in destructive mode.

If a work queue is used, you must set the Message read mode property in the connector to Move to work queue. You specify the name of the work queue in the Usage > Work queue > Name property. In this case, you must grant get authorization for the source queue because the connector destructively gets messages from the source queue to move them to the work queue. You grant put authorization for the work queue so that the connector can move messages from the source queue to the work queue. In addition, you must grant inq and browse authorizations for the work queue because when the job first starts, the connector checks if there are any messages that remained on the work queue from previous unsuccessful job runs. If the work queue depth is greater than zero, the connector reads the messages on the work queue in browse mode and delivers them as records on the output link.

The inq authorization for the work queue is also required if the connector is configured to monitor the current queue depth of the work queue and to temporarily stop moving messages from the source queue to the work queue if the number of messages on the work queue reaches the specified upper limit. By temporarily stopping the movement of messages, the connector allows the downstream stages to process work queue messages until the specified lower limit for the number of messages on the work queue is reached. Then the connector
resumes moving messages from the source queue to the work queue. To monitor and manage the work queue depth, you set the Monitor queue depth, Minimum depth, and Maximum depth properties.

Another feature supported by the connector for which the inq authorization for the work queue is required is the ability for the connector to stop moving messages from the source queue to the work queue until the job processes a blocking transaction message. To create a blocking transaction message, you write a special method and add it to a module that is specified in the connector. Then for each input message, the connector calls the method to determine if the message should be treated as a blocking transaction message. To configure this feature, you use the Blocking transaction processing, Module name, Method name, and Timeout properties. To use this feature, the connector also requires that the Command Server be running for the specified queue manager. The connector uses the MQAI interface to communicate with the Command Server to obtain information about the current work queue depth. Therefore, you must grant the put and inq authorizations for the SYSTEM.ADMIN.COMMAND.QUEUE queue so that the connector can send the commands, and you must grant the get, inq, and dsp authorizations for the SYSTEM.DEFAULT.MODEL.QUEUE queue so that the connector can process the responses.

For example, if the connector is configured to run in parallel on n nodes, you must grant the browse and inq authorizations to the following queues:

\[
\text{queue}_\text{name}.0, \text{queue}_\text{name}.1, \ldots, \text{queue}_\text{name}.n-1
\]

In this list, the queue_name represents the value of the Usage > Work queue > Name property, and n is the number of nodes on which the connector is configured to run.

To configure the connector to preserve the identity context or both the identity context and the origin context field values of the source messages when the connector moves them to the work queue, set the Usage > Work queue > Context mode property to None, Set id, or Set all. When the property is set to Set all, you must grant the setall authorization for the work queue. Then the connector copies the identity context and origin context fields from the source queue message to the corresponding work queue message.

When the message data reaches the downstream Distributed Transaction stage, the Distributed Transaction stage destructively reads the message from the work queue (or from the original source queue if the work queue is not used) that corresponds to the data and writes the data to the database in a single distributed transaction. Because the corresponding message identifier is carried along with the data, the Distributed Transaction stage knows which message to read. The Distributed Transaction stage refers to the queue from which it reads messages as the work queue, even though the queue might actually be a source queue from which the connector originally read the messages without ever moving them to a separate work queue.

You must grant the connect authorization to the local queue manager that is specified in the Connection > Queue manager property. Note that the Distributed Transaction stage does not support Client mode. Therefore, the queue manager must be local to the connector in the Information Services engine tier.
You must grant the get authorization for the queue that is specified in the **Connection > Work queue** property. However if the **Connection > Append node number** property is set to **Yes**, you must grant the get authorization to the following queues instead:

\[
\text{queue}\_\text{name}.0, \\
\text{queue}\_\text{name}.1 \\
... \\
\text{queue}\_\text{name}.n-1
\]

In this list, the **queue\_name** represents the value of the **Connection > Work queue** property, and \( n \) is the number of nodes on which the Distributed Transaction stage is configured to run.

To configure the Distributed Transaction stage to move to the reject queue the messages for which the distributed transactions fail, you set the **Usage > Reject failing units** property to **Yes**. You enter the name of the reject queue in the **Usage > Reject failing units > Reject queue** property. You must grant the put authorization for the reject queue so that the Distributed Transaction stage can put messages on it.

To configure the Distributed Transaction stage to preserve the identity context or to preserve both the identity context and origin context field values of the work messages when it moves them to the reject queue, you set the **Usage > Reject failing units > Context mode** property to **None**, **Set id**, or **Set all**. When the value is set to **Set id**, you must grant the setid authorization for the reject queue. Then the connector copies the identity context fields from the work queue message to the corresponding reject queue message. When the property is set to **Set all**, you must grant the setall authorization for the reject queue. Then the connector copies the identity context and origin context fields from the work queue message to the corresponding reject queue message.

If the value that is specified in the **Usage > Reject failing units > Reject queue** property is a namelist, you must grant authorizations to the namelist and to the queues that it contains.

### User IDs to use when granting authorizations to WebSphere MQ objects

This table lists the user IDs, or principals, to specify in the setmqaut command when you grant authorizations for the MQ objects that the WebSphere MQ connector accesses.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Runtime environment</th>
<th>Design time environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>User ID under which the job is running</td>
<td>Built-in Local System user (Microsoft Windows) or root or privileged user (UNIX and Linux)</td>
</tr>
</tbody>
</table>
| Client      | User ID that is associated with the server-connection channel MCA, which is one of the following:  
- The user ID under which the connector is running (propagated from the WebSphere MQ client side)  
- The MCA User ID that is set by the security exit  
- The user ID that is specified through the MCAUSER attribute in the server-connection channel definition | User ID that is associated with the server-connection channel MCA, which is one of the following:  
- The user ID under which the connector is running (propagated from the WebSphere MQ client side)  
- The MCA User ID that is set by the security exit  
- The user ID that is specified through the MCAUSER attribute in the server-connection channel definition |
Required authorizations for WebSphere MQ objects

This table lists the WebSphere MQ objects that require specific authorizations, which you set by using the setmqaut command.

### Table 10. WebSphere MQ objects and required authorizations

<table>
<thead>
<tr>
<th>WebSphere MQ connector property</th>
<th>WebSphere MQ object</th>
<th>Required authorizations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection &gt; Queue manager is set to a certain value</td>
<td>Queue manager that is specified in the property. In Server mode, leaving this property blank specifies the default queue manager.</td>
<td>connect,inq</td>
</tr>
<tr>
<td>Connection &gt; Client channel definition &gt; Channel name property is set to a certain value</td>
<td>Queue manager to which the client-connection channel definition corresponds.</td>
<td>connect,inq</td>
</tr>
</tbody>
</table>

- **Usage > Message read mode property is set to either Delete or Delete under transaction**
  - Queue from which the connector is reading messages. The name of this queue is specified in one of the following locations:
    - In the Usage > Queue name property
    - If the connector is used in request/reply mode, meaning that the connector has both an input link and an output link, specify the queue name in the Usage > Set header fields > Reply to queue property (for all messages) or in the WSMQ.REPLYTOQ data element column on the input link (for individual messages). Note that the queue name can be a model queue name, in which case you must grant the dsp authority for the model queue.

- **Usage > Message read mode property is set to Keep**
  - Queue from which the connector is reading messages. The name of this queue is specified in one of the following locations:
    - In the Usage > Queue name property
    - If the connector is used in request/reply mode, meaning that the connector has both an input link and an output link, specify the queue name in the Usage > Set header fields > Reply to queue property (for all messages) or in the WSMQ.REPLYTOQ data element column on the input link (for individual messages). Note that the queue name can be a model queue name, in which case you must grant the dsp authority for the model queue.

- **Usage > Refresh property is set to Yes**
  - Queue from which the connector is reading messages. The name of this queue is specified in one of the following locations:
    - In the Usage > Queue name property
    - If the connector is used in request/reply mode, meaning that the connector has both an input link and an output link, specify the queue name in the Usage > Set header fields > Reply to queue property (for all messages) or in the WSMQ.REPLYTOQ data element column on the input link (for individual messages). Note that the queue name can be a model queue name, in which case you must grant the dsp authority for the model queue.

- **Usage > Message options > Enable payload reference property is set to Yes**
  - Queue from which the connector is reading messages. The name of this queue is specified in one of the following locations:
    - In the Usage > Queue name property
    - If the connector is used in request/reply mode, meaning that the connector has both an input link and an output link, specify the queue name in the Usage > Set header fields > Reply to queue property (for all messages) or in the WSMQ.REPLYTOQ data element column on the input link (for individual messages). Note that the queue name can be a model queue name, in which case you must grant the dsp authority for the model queue.

- **Message filtering configured through any of the Usage > Filter messages properties other than the following:**
  - Message ID
  - Correlation ID
  - Group ID
  - Message sequence number
  - Offset
  - Queue from which the connector is reading messages. The name of this queue is specified in one of the following locations:
    - In the Usage > Queue name property
    - If the connector is used in request/reply mode, meaning that the connector has both an input link and an output link, specify the queue name in the Usage > Set header fields > Reply to queue property (for all messages) or in the WSMQ.REPLYTOQ data element column on the input link (for individual messages). Note that the queue name can be a model queue name.
| Usage | Message write mode is set to any value in the list of allowed values for this property | Target queue to which the connector is sending messages. The name of the queue is specified in the Usage > Queue name property, or alternatively for each message separately through the WSMQ.QUEUENAME data element column on the input link. If the specified name is a namelist, then the put authority must be granted for each queue in the namelist and theinq authority must be granted for the namelist object itself. | setall |
| Usage | Access mode property is set to Set identity | Target queue to which the connector is sending messages. The name of the queue is specified in the Usage > Queue name property, or alternatively for each message separately through the WSMQ.QUEUENAME data element column on the input link. If the specified name is a namelist, then the setid authority must be granted for each queue in the namelist and theinq authority must be granted for the namelist object itself. | setid |
| Usage | Access mode property is set to Set all | Target queue to which the connector is sending messages. The name of the queue is specified in the Usage > Queue name property, or alternatively for each message separately through the WSMQ.QUEUENAME data element column on the input link. If the specified name is a namelist, then the setall authority must be granted for each queue in the namelist and theinq authority must be granted for the namelist object itself. | setall |
| Usage | Other queue settings > Alternate user ID and Usage > Other queue settings > Alternate security ID are set to some value | Queue manager that hosts the queue that the connector is trying to open. | other |
| Usage | Queue name property is set to the name of a model queue, which the queue manager uses to create a dynamic queue | The model queue that is specified in the property. | put, dep |
| Usage | Other queue settings > Dynamic queue > Reply queue close options property is set to Delete or Purge and delete and a dynamic queue name is specified in the Usage > Set header fields > Reply queue property (for all messages) or in the WSMQ.REPLYTOQ data element column on the input link (for individual messages) | The specified permanent dynamic queue name. | dlt |
| Usage | Queue name property or WSMQ.QUEUENAME data element on the input link contains the name of the shared cluster queue on which messages are put and the messages might be routed to a cluster queue manager other than the queue manager to which the connector is connected | SYSTEM.CLUSTER.TRANSMIT.QUEUE system queue. | put |
| Usage | Error queue > Queue name property contains the name of the queue to use as an error queue | The specified queue name. | put |
| Usage | Error queue > Context mode property is set to Set identity | The specified queue name. | setid |
| Usage | Error queue > Context mode property is set to Set all | The specified queue name. | setall |
| Usage | Error queue > Queue manager property contains the name of the queue manager that hosts the error queue | The specified queue manager name. | connect |
| Usage | Publish/Subscribe > Registration property or Usage > Publish/Subscribe > Deregistration is set to Yes | SYSTEM.BROKER.CONTROL.QUEUE system queue. | put |
| Usage | Publish/Subscribe > Reply queue property is set to the name of the reply queue to use for response messages from the broker | The specified queue name. If you specify a model queue name to use for creating a dynamic reply queue, then you must grant put, get, and dep authorities for the model queue. | put, get |
| Usage | Publish/Subscribe property is set to Yes, IBM WebSphere MQ is the message broker, and the connector acts as a publisher, meaning that an input link has been defined | SYSTEM.BROKER.DEFAULT.STREAM | put |
| Usage | Publish/Subscribe property is set to Yes, IBM WebSphere MQ is the message broker, and the connector acts as a subscriber, meaning that an output link has been defined | SYSTEM.BROKER.DEFAULT.STREAM | put, browse, dlt |
| Usage | Work queue > Name specifies the name of the queue to use as the work queue | The specified queue name value (subject: if the connector is configured to run in parallel on n nodes, the queue names are: queuename 0, queuename 1, queuename.n-1) | put, browse, dep |
| Usage | Work queue > Context mode property is set to Set identity | The specified queue name value (subject: if the connector is configured to run in parallel on n nodes, the queue names are: queuename 0, queuename 1, queuename.n-1) | setid |
| Usage | Work queue > Context mode property is set to Set all | The specified queue name value (subject: if the connector is configured to run in parallel on n nodes, the queue names are: queuename 0, queuename 1, queuename.n-1) | setall |
Troubleshooting

Use these tips to troubleshoot problems that might occur with access-control checking in the MQ authorization service.

When a job that includes the WebSphere MQ connector fails because of a problem with access-control checking, the connector reports MQ error 2035 (MQRC_NOT_AUTHORIZED). If the error is preceded by an informational message indicating that the connector is connecting to the queue manager, then the connector was not granted connect authorization for that queue. If the error is preceded by an informational message indicating that the connector was opening a queue, then the connector was not granted sufficient authorization to open the queue. The connector typically logs information about the queue that it tried to open and logs the open options that it specified when it tried to open the queue. By evaluating the open options, you can often determine which authorizations you need to grant.

For example, the following is an excerpt from the job log that was created when the connector attempted to open queue QUEUE1 to put messages into it:

```
Opening queue QUEUE1 with open options (MQOO_OUTPUT, MQOO_FAIL_IF_QUIESCING)
Open queue failed with reason code: 2035 (MQRC_NOT_AUTHORIZED); CC_WSMQQueue; open(); CC_WSMQQueue.cpp; 372
```

From the MQOO_OUTPUT flag, you can conclude that the connector tried to open the queue for output.

The MQ error 2035 is typically accompanied by additional information in the queue manager's error log. For the job failure shown above, the following is an example of how the queue manager error log information might look:

```
9/13/2009 19:26:58 - Process(7008.130) User(MUSR_MQADMIN) Program(amqzlaa.exe) AMQ8077: Entity 'test' has insufficient authority to access object 'QUEUE1'.
```

EXPLANATION:
The specified entity is not authorized to access the required object. The following requested permissions are unauthorized: put
ACTION:
Ensure that the correct level of authority has been set for this entity against the required object, or ensure that the entity is a member of a privileged group.

The information shows the user ID for which the access-control check failed (test), the MQ object for which the check failed (QUEUE1), and the authorization that was requested by the operation (put).

You can use the MQ command line tool dspmqaut to display the current authorizations that are granted for a particular object to a particular user ID. For example, the following command shows the authorizations that are granted to user ID test for queue QUEUE1 which resides under queue manager QMNAME:

```
dspmqaut -m QMNAME -t queue -n QUEUE1 -p test
Entity test has the following authorities for object QUEUE1:
  get
  browse
  cr
```
Chapter 4. IBM WebSphere MQ connector

You can use the IBM WebSphere MQ connector in your jobs to read messages from and write messages to message queues in IBM WebSphere MQ enterprise applications.

You can use the IBM WebSphere MQ connector in any of the following ways:
- As an intermediary that enables applications to communicate by exchanging messages
- As a path for the transmission of older data to a message queue
- As a message queue reader for transmission to a non-messaging target

Installation and configuration prerequisites for the WebSphere MQ connector

Installation requirements must be met depending upon whether you use the connector in server connection mode or in client connection mode.

This mode is determined by your selection in the Mode property.
- Client connection mode
  - The WebSphere MQ client must be installed on the same node as the connector.
  - The WebSphere MQ server must be installed on a node in the same network in which the connector is installed. The WebSphere MQ server can be installed on the same node as the connector.
  - There must be a network connection between the client node and the server node.
- Server connection mode
  - The WebSphere MQ server must be installed on the same node as the connector.

To use publish/subscribe with the connector, you must meet certain application requirements for each message format.

Queue managers

The queue manager owns and manages the queues that are used by the WebSphere MQ application. Before the connector can open a queue to read and write messages, you must define the connection to the queue manager that hosts that particular queue.

You must meet the following requirements that are determined by the Mode property:
- Server The queue manager must run on the same node as the connector.
- Client The queue manager can run on a remote node in the same network as the connector.

For either mode, the Queue manager property displays a list of queue managers. You can also type a value in the Queue manager property. You can connect to only one queue manager at a time from a connector.
Channels for client connections

If you use the connector as a client application (that is, you set Mode to Client), you must specify a channel definition for client connections.

You can specify the channel for client connections in two ways:

Client channel properties
Specifies only the channel name, transport type, and connection name. You cannot specify other details, such as a security exit or SSL parameters to use with the channel.

Environment variables
Specifies a table for client connection channels. Use either the MQSERVER variable or a combination of MQCHLLIB and MQCHLTAB.

You can also specify a queue manager. The Queue manager property is necessary only when you specify a table for client connection channels by using the environment variables MQCHLLIB and MQCHLTAB. The Queue manager value is used to locate the correct channel in the table.

If you use MQSERVER to specify a table for client connection channels, the Queue manager property is not required. However, if you specify a value for Queue manager, the client connection channel from MQSERVER must point to that queue manager.

Queue manager clustering
You can use queue manager clusters that you define in WebSphere MQ in the WebSphere MQ connector.

When an application connects to one of the queue managers in the cluster and sends messages to its shared cluster queue, two different things can happen:
• The connector makes a request to WebSphere MQ to physically store that message on a specific queue in the cluster.
• Load balancing is used by WebSphere MQ to evenly distribute that message and subsequent messages across all queue instances in the cluster.

Queues and namelists for the WebSphere MQ connector
A queue is an object that stores messages and is managed by a queue manager. A namelist is a list of names of WebSphere MQ objects that includes queues.

In order to run MQ connectors in parallel accessing a single queue, you should configure the queue in SHARE mode. Otherwise, the job fails. To read or write messages, the queue must be opened. For synchronous request and reply messaging, when you specify a queue name, this name is the name of the queue to which the request is sent. The name of the reply queue is included in the request message.

In addition to queues, you can specify namelists for input links. The WebSphere MQ application accesses the namelist at run time and creates a distribution list that contains all of the queue names from the namelist. The application opens the distribution list in the same way that it opens a queue. When a message is sent to the distribution list, the message is sent to each queue in that list.

Type or select the name of the queue or the namelist in the Queue name property.
Note: To view a list of queues on a particular queue manager, that queue manager must be running the Command Server service.

**Configuring authorizations for WebSphere MQ**

To enable the WebSphere MQ connector to access the specified queue manager and queue objects, you must grant certain authorizations to the user ID under whose credentials the connector runs.

**Job design and the WebSphere MQ connector**

Before you can configure the properties of the WebSphere MQ connector, you must first add it to your IBM InfoSphere DataStage job.

The connector can perform several different roles when you add it to your job.

The role of the connector is determined by the links that are attached to it in the job and how the connector is configured. The following links can be used with the WebSphere MQ connector:

- **Input link**
  - The connector sends messages and acts as a publisher if publish/subscribe is enabled.

- **Output link**
  - The connector receives messages and acts as a subscriber if publish/subscribe is enabled.

- **Input link and output link**
  - The connector sends and receives messages in the request and reply scenario.

- **Reject link**
  - The connector passes rows of data that could not be processed and that satisfy specified error criteria to another stage in the job. You can also configure the connector to send rows of data in error to an error queue without a need to define the reject link.

You cannot use reference links with the WebSphere MQ connector.

**Example link configurations**

Here are some example link configurations for the WebSphere MQ connector:

- Input link only
- Output link only
- Input link and output link (request and reply scenario)
- Input link, output link, and reject link (request and reply scenario with reject functionality)
- Input link and reject link

**Transaction processing by the connector**

The connector can read and write messages either inside or outside of the current transaction. Use the Message read mode and Message write mode properties to specify how messages are read or written in the current transaction.
Use the properties in the Transaction group to specify parameters for transactional processing. You can specify how many rows to include per transaction and when to commit messages that are read by the connector. In a request and reply scenario, the connector writes each request message outside of the current transaction so that the message is immediately available for processing. You can specify whether to read reply messages inside or outside of the transaction by using the `Message read mode` property.

**Configuring the connector as a source**

To configure the connector as a source, you must define the connection to a WebSphere MQ queue manager, specify the properties for the output link, and define columns for the data that the connector must read.

**About this task**

In the source context, the connector extracts or reads data from an external WebSphere MQ queue manager.

When the WebSphere MQ connector receives a new message from a queue, certain information about the structure of the message can be determined. For example, the connector can determine whether the message contains only text or whether it contains format headers in the data.

**Restriction:** Only one connection is valid from a single thread. If your transaction involves SEBridge running both the WebSphere MQ plug-in and WebSphere MQ connector on the same thread, a warning message at MQCLOSE and a unrecoverable error at MQCMIT occurs alerting you that the connection is shared and transactions are overlapped between the two MQ stages. To work around these errors, you can add an interprocess stage between the transformer and MQCC to run the plug-in and connector separately.

**Procedure**

1. In the job canvas, add the connector to the job.
2. Add the stage that follows the connector in the job flow.
3. Add the output link from the connector to the next stage. Right-click the connector and then drag it on the next stage. The connector now has an output link that connects it to the next stage in the job flow.
4. Double-click the connector to open the stage editor.
5. On the Properties tab, define the connection properties for the WebSphere MQ queue manager.
6. Optional: On the Advanced tab, specify the custom processing settings.
7. Specify information about the output link:
   a. Select the output link in the navigator.
   b. On the Properties tab, define the usage properties for the link.
   c. On the Columns tab, define the column metadata for the link.
   d. Optional: On the Advanced tab, you can specify custom buffering settings for the link.
8. Click OK to save your changes and to close the stage editor.
Configuring the connector as a target

To configure the connector as a target, you must define the connection to a WebSphere MQ queue manager, specify the properties for the input link, and define columns for the data that the connector will write.

About this task

In the target context, the connector connects to the external WebSphere MQ queue manager and inserts data.

When the WebSphere MQ connector sends a message to the queue, the connector uses the schemas in the job. The connector reads the data from the payload column and uses the data as the payload for the outgoing message.

Procedure

1. In the job canvas, add the connector to the job.
2. Add the input link from the previous stage in the job flow to the connector. Right-click the previous stage and then drag it on the connector. The connector now has an input link that connects it to the previous stage in the job flow.
3. Double-click the connector to open the stage editor.
4. On the Properties tab, define the connection properties for the WebSphere MQ queue manager.
5. Optional: On the Advanced tab, specify custom processing settings.
6. Specify information about the input link:
   a. Select the input link in the navigator.
   b. On the Properties tab, define the usage properties for the link.
   c. On the Columns tab, define the column metadata for the link.
   d. Optional: On the Advanced tab, you can specify custom buffering settings for the link.
7. If the connector has a reject link, specify how to send data to this link:
   a. Select the reject link in the navigator.
   b. On the Reject tab, define the reject conditions for the link.
   c. Optional: On the Advanced tab, specify custom buffering settings for the link.
8. Click OK to save your changes and to close the stage editor.

Reject links and error queues

You can use reject links with the WebSphere MQ connector.

The reject link is an output link to which constraints are added. Reject links and error queues can be configured to work together, depending upon the type of link being used:

Input links

You can use reject links and error queues with input links. If the message cannot be stored on the queue and you specify both the reject link and an error queue, the error queue takes precedence over the reject link. However, if the message cannot be stored on the error queue, the message is rejected and is passed to the reject link if it meets the reject link criteria. In this scenario, the error queue is an alternate destination for the queue.
Output links
You cannot use reject links with output links. However, you can specify an error queue. If you specify an error queue in your job and the transaction in which the message was obtained is rolled back, the connector attempts to move the messages that are in error to the error queue.

Request and reply scenario
You cannot use error queues with requests. However, you can use a reject link for the request part of the request and reply scenario. If you configure a reject link, the connector attempts to reject the request messages that cannot be stored on the queue. If the request messages match the reject link criteria, they are moved to the reject link. Although there is a reply or output link and a reject link in this job, reply messages are never sent to the reject link. When an error queue is defined in the request and reply scenario, and the connector rolls back the transaction in which reply messages were read, the reply messages are moved to the error queue.

Queues and the WebSphere MQ connector
Several different types of queues can be used in the context of a job.

The following types of queues can be used by the WebSphere MQ connector:
- Local queues
- Transmission queues
- Dynamic queues
- Error queues

Local queues
Local source queues and target queues are the standard objects that are used by the WebSphere MQ application. A source queue is a queue from which the connector reads messages. A target queue is a queue to which the connector writes messages. Specify the local queue name in the Queue name property.

Transmission queues
A transmission queue forwards messages to a remote target queue through the queue manager to which it is connected. The WebSphere MQ connector sends messages to a local definition of the remote queue. WebSphere MQ places the messages on the corresponding transmission queue The messages are then forwarded to the remote queue over the transmission channel. The default transmission queue name is the same as the remote queue manager name of the target queue. You can override these settings at run time. Specify the name of the transmission queue in the Transmission queue property.

Dynamic queues
A dynamic queue is created by the WebSphere MQ queue manager at run time to serve a specific message operation for the application that is connected to that queue manager. Dynamic queues can be used in the following contexts:

Target (input link)
The connector sends messages on a queue that can be defined as a dynamic queue.
Publish/subscribe
The connector sends command messages to the broker. The broker replies to these messages, providing a confirmation that the command was processed successfully. The queue that the broker uses to respond to the connector can be a dynamic reply queue.

Request and reply (output link and input link)
In the request and reply scenario, the connector receives the message on the input link and then sends it to the request queue. The connector waits for the reply message on the reply queue. When the connector receives the reply message, the connector sends the message to the output link onto the next stage. You can define the request queue as a dynamic queue or the reply queue as a dynamic reply queue. If you define both queues as dynamic queues, both links must have the same value for the Dynamic queue name property.

Error queues
Error queues are different from reject links.

You can use error queues and reject links together in certain situations. For details, see “Reject links and error queues” on page 55.

An error queue is a local queue to which error messages can be sent by the WebSphere MQ connector when a message operation fails. For target queues, the error queue acts as a backup option for the target queue when messages cannot be written to the target queue. For source queues, the connector retains a list of all retrieved messages within the transaction. If the transaction fails, the WebSphere MQ connector instructs the queue manager to roll back the messages to the source queue. Then, the connector moves the rolled back messages to the error queue in a new transaction.

Note that the usability of error queues for source queues is very limited. Namely the scope of the transaction in which the connector retains the list of retrieved messages does not span all the stages in the job. Instead it only pertains to the connector stage's successful delivery of message records to the output link of the stage. The support for distributed transactions that span all stages in the job requires the use of WebSphere MQ Connector Stage in combination with the Distributed Transaction Stage.

Configuring the connector to connect to the queue manager
Use the properties in the Connection section on the Properties tab to configure the connection of the queue manager for the connector.

About this task
To configure the connector to connect to the queue manager:

Procedure
1. Double-click the connector on the job canvas to open the stage editor.
2. Configure the values for the properties in the Connection section on the Properties tab.
What to do next

Now, you can test the connection.

Testing the connector connection to the queue manager

You can test the connection to your queue manager in the stage editor before you compile and run your job.

Before you begin

You must configure the properties in the Connection section of the stage editor.

About this task

To test the queue manager connection in the stage editor:

Procedure

1. In the Connection section, click Test.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection values are correct</td>
<td>The connection is made, and a confirmation message is displayed. No further action is required.</td>
</tr>
<tr>
<td>Connection values are incorrect</td>
<td>An error message is displayed. Go to the next step.</td>
</tr>
</tbody>
</table>

2. Edit the connection property values.
3. Repeat step 1.

What to do next

Now, you can save this connection information as a data connection object that can be reused. For more information, see the topic about saving connection information as data connection objects.

Specifying error queues

You can specify error queues for the source context (output links) and for the target context (input links). The error queue can be a local queue or a remote queue.

Before you begin

You must create a job in which the WebSphere MQ connector has an input link or an output link. You also must configure your connection properties and other properties as needed for the job at some point before you run the job.

About this task

For output links, the connector makes a list of the messages that the connector retrieves from the queue within the transaction. If the transaction fails, the connector rolls the messages back to the source queue, and then the connector moves the messages to the error queue in a new transaction.
For input links, the error queue acts as a backup queue for the input queue. You can use both an error queue and a reject link in your target context jobs. If a message cannot be stored on the input queue and both the error queue and reject link are specified, the message is sent to the error queue. However, if the message cannot be stored on the error queue, the message is rejected if the message meets the rejection criteria on the Reject tab.

In the request and reply scenario, you can specify an error queue for reply messages. You can specify a reject link for the request messages that cannot be stored on the request queue and that match the reject criteria on the Reject tab.

You can also specify a remote error queue. Instead of moving the error messages to the local error queue, the queue manager moves them to the transmission queue for the specified error queue.

To specify an error queue:

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. Click either the Output tab or the Input tab, then select the output link or the input link.
3. In the Usage section, set Error queue to Yes. The sub-properties of the Error queue group are displayed.
4. If Mode is set to Client, in the Queue manager property, type or select the name of the queue manager for the error queue. If Mode is set to Server, this property is not available.
5. In the Queue name property, type or select the name of your error queue.
6. In the Context mode property, specify how context fields are processed in messages that are sent to the error queue.
7. For remote queues, type or select a value for the transmission queue in the Transmission queue property.

**Specifying dynamic queues in the target context**

On the input link that the connector uses to send messages, you can specify that the queue is a dynamic queue.

**Before you begin**

You must create a job first in which the WebSphere MQ connector has an input link. You also must configure your connection properties and other properties as needed for the job at some point before you run the job.

**About this task**

Although dynamic queues are typically used for reply queues in the request and reply scenario, dynamic queues can also be used when the connector is in the target context. For example, you can configure the connector to use a fixed prefix for the queue name. The queue manager then creates a dynamic queue with that prefix and a unique suffix for each individual job run. The application that receives the messages is aware of the prefix for the queues. This application can process these messages that correspond to separate job runs, and then permanently delete the queues after message processing.
You cannot specify any closing options for the dynamic queue. In the target context, the connector closes the dynamic queue without purging any messages on the queue and without deleting the queue.

To specify a dynamic queue in the connector in a target context (input link) job:

**Procedure**
1. Double-click the connector on the job canvas to open the stage editor.
2. On the Input tab, click the input link.
3. In the Queue name property in the Usage section on the Properties tab, type or select the name of the model queue that is a template for the dynamic queue.
4. In the Usage section on the Properties tab, set Dynamic queue to Yes. The sub-properties of the Dynamic queue group are displayed.
5. In the Queue name property, specify a unique name or part of a name ending with an asterisk (*) for the dynamic queue name. If you type a partial name, the queue manager produces a unique name when it creates the dynamic queue. If the name is not unique for the local queue manager, an error is generated.

---

**Messages and the WebSphere MQ connector**

A message is a string of bytes that transfers information from one application program to another program or to different parts of the same application.

Each message can consist of two or three parts, depending upon its complexity:

- **Message header**
  Contains information about the content and structure of the application data.

- **Format header**
  Contains information about the message format. In certain situations, a format header is separate from the message payload. To ensure that the format header is treated separately from the message payload, the design schema must include the WSMQ.FORMATHEADERS data element column.

  If the WSMQ.FORMATHEADERS data element is specified and the format of the received message matches one of the values in the System value sub-property for Format, the format header is treated separately from the message payload.

- **Message payload**
  Contains the message data in text or binary format. Message data can be structured or unstructured. However, the connector always treats message payload as a single unstructured data value. This means that no more than one text or binary column can be defined on the link to represent message payload data.

The WebSphere MQ connector can use your values to filter source messages and to override target and request message header values.

In request and reply messages, you cannot specify any filter values. The connector searches for the reply message that complies with the report options in the request message. When the default report options are specified, the connector matches the value in the Value sub-property of Correlation ID in the reply message with the value in the Value sub-property of Message ID in the request message.
Message types

The WebSphere MQ connector provides the ability to work with different types of messages. Use the Message Type property to select from a list of predefined message types or type one or more of your own.

For target messages, you can specify multiple values upon which to filter your messages. For source messages, only one value can be selected or entered. Additionally, you cannot set the type for messages in the request and reply scenario. The request message is always the request type. The response message is always the reply type.

The following message types are predefined:

- "Request messages"
- "Reply messages"
- "Report messages"
- "Datagram messages" on page 62

Request messages

A request message is a message that requires a reply. The name of the queue to which the reply message must be sent must be defined in the Reply to queue property. The Report property value indicates the way in which the values for the Message ID property and the Correlation ID property in the reply message correspond to the same properties in the request message.

In most cases, two queues are used. One queue is the queue to which the request message is sent, and another queue is the one from which the reply message is read. This process is synchronous message processing.

Reply messages

A reply message is a message that is sent in response to an earlier request message. This message is sent to the queue as defined in the Reply to queue property of the request message. The Report property indicates the way in which the values for the Message ID and Correlation ID properties in the reply message correspond to the same properties in the request message.

In most cases, two queues are used. One queue is the queue to which the request message is sent, and another queue is the one from which the reply message is read. This process is synchronous message processing.

Report messages

A report message is a message about another message that informs an application about expected or unexpected events that relate to the original message. When a WebSphere MQ application sends a message to a queue, the application can request that the queue manager generate a report message when certain events occur that are related to the message that was sent. The application can define the events that generate the report from the queue manager. You can specify these in the Report property. The queue manager then sends the report to the queue. This queue is specified in the Reply to queue and Reply to queue manager properties in the header of the message that was originally sent.
Use the **Feedback** property to indicate the nature of the report. As with the request messages, the **Report** property of the original message can determine the values for the events that cause the queue manager to send the report.

**Datagram messages**

A *datagram* message is a message that does not require a reply. This process is asynchronous message processing.

Datagram messages are the default type of messages and are not tied to any particular role. These messages carry general information in a format that is understood by the applications that exchange and process them. For example, a datagram message could contain textual data that the receiving application needs to store in a database. Another example is a list of commands that the receiving application needs to invoke or execute in a particular business scenario. Datagram messages are typically used when there is no requirement for the processing application to produce a corresponding response message.

**Message segments**

Messages can be divided into smaller pieces that are called segments, or they can be grouped together. You can work with these message segments in the WebSphere MQ connector.

For example, you can specify segment information in the connector that the queue manager uses to create message segments. Then the queue manager reassembles the segments into logical messages or groups of messages. You can describe a message in several different ways:

**Segments**

The smallest entity in a message. Each segment is one physical message on the queue. You can use segmented messages for input and output messages. However, you cannot create a request message as a segmented message.

**Logical messages**

The ordered association of multiple message segments. Each segment within a logical message has an offset value in bytes that specifies its relative position to the beginning of the logical message. The last segment in a logical message also contains a flag that specifies that the segment is the final segment for this message. Logical messages do not have to be members of a group.

A logical message can also be unsegmented. When it is unsegmented, it is one physical message on the queue.

**Groups of messages**

The grouping together of multiple logical messages. Each logical message has a sequence number that defines the position of that logical message within the group. As with segments in logical messages, the last logical message also contains a flag that specifies that the message is the last logical message for this group.

If a message is too large for a queue, the queue manager, WebSphere MQ application, or connector can split the message into segments. Each segment is then placed on the queue as a separate physical message. The application that retrieves these messages can retrieve them individually, or the application can request that the queue manager reassemble the segments into a single message.
**Message schemas**

A schema defines the structure and the type of contents that each data element within the message segment can contain.

You can define the schema to contain zero, one or more data element columns, and zero or one message payload column.

Data element columns are typically used to represent message header fields. There are also data element columns for the format headers, queue name, and topic name.

The message payload column must be of text or binary type.

**Messages that are sent to the queue or received from the queue**

Use the WebSphere MQ connector in your jobs to put messages on a queue or receive messages from a queue.

When the WebSphere MQ connector puts a message on the queue, the connector uses the schemas in the job. The connector reads the data from the payload column and uses this data as the payload for the outgoing message.

When the WebSphere MQ connector receives a new message from a queue, the connector can determine certain information about the structure of the message. For example, the connector can determine whether the message contains only text or whether it contains format headers in the data.

Additionally, the connector can determine the character set and encoding of this data from the values in the *Coded character set ID* property and the *Encoding* property of the message. The connector then reads the data from the message buffer and assigns this data to the message payload column of the schema as either binary data or text data.

**Converting the character set and encoding for message data**

You can convert the character set and encoding of the original message data. The conversion is for the character set of the text and the encoding of the numbers.

**Before you begin**

You must create a job in which the WebSphere MQ connector is a source or is part of the request and reply scenario. You must have an output link that is attached to the connector and an input link for the other part of the request and reply scenario. However, message conversion is set only on the request link or output link. You also must configure your connection properties and other properties as needed for the job at some point before you run the job.

**About this task**

For example, you need the EBCDIC to ASCII conversion when the connector accesses queues on z/OS® from Windows.

To convert the character set and encoding:
Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. In the navigator, click the output link.
3. In the Usage section of the Properties tab, set Message options to Yes.
4. Set Message conversion to Yes.
5. In the Encoding sub-property, specify the numeric encoding.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert message data to the encoding of the operating system where the connector runs.</td>
<td>Accept the default value, which is -1.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert to a different encoding.</td>
<td>Type a different value that is a decimal representation of a three-digit hexadecimal value. The three digits specify system encodings for floating-point, packed-decimal-integer, and binary-integer numbers. For example, the value 786 corresponds to the hexadecimal value 0x312. This hexadecimal value represents the following combination of values:</td>
</tr>
<tr>
<td>0x300</td>
<td>The zSeries® encoding for floating-point numbers.</td>
</tr>
<tr>
<td>0x010</td>
<td>The normal encoding for packed-decimal-integer numbers.</td>
</tr>
<tr>
<td>0x002</td>
<td>The reversed encoding for binary-integer numbers.</td>
</tr>
</tbody>
</table>

6. In the Coded character set ID property, specify the character set.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert message data to the character set of the operating system where the connector runs.</td>
<td>Accept the default value, which is 0.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert to a different character set.</td>
<td>Type a different value that is the coded character set identifier (CCSID) constant for the character set encoding. For example, the value 1208 corresponds to the UTF-8 encoding for the Unicode character set.</td>
</tr>
</tbody>
</table>

Request and reply scenario and the WebSphere MQ connector

IBM WebSphere MQ connector can be used in synchronous processing, which is also known as the request and reply scenario.

In this scenario, the connector sends a message from the input link to the request queue, and then blocks and waits for the corresponding reply message on the reply queue. The connector repeats this procedure for all messages that arrive on the input link when the job runs.

In its role as consumer, the connector performs the following high-level tasks:

1. Creates a request message.
2. Sends the request message to the target queue. The message identifier and the correlation identifier of the request message are noted.
In its role as producer, the connector performs the following high-level tasks:

1. Opens the queue in the **Reply to queue** property.
2. Issues a call to receive the reply messages when the message identifier of the request message matches the correlation identifier of the reply message.

The request context and the reply context each have their own data set definition (schema) and data set (data). The same queue manager must host both the request and reply queues. You cannot specify a value for the **Reply to queue manager** property.

The type of queue manager depends on your connection mode:

**Server mode**

The queue manager is the local queue manager. You can define a remote queue on the local queue manager. The connector sends the request message to the remote queue. However, the application that processes the request message from the remote queue manager must send the reply message back to the local queue manager. Then the connector can read the reply message from the local queue.

**Client mode**

The queue manager can be a local queue manager or a remote queue manager.

Request messages are read, processed, and replied to outside of the job in which the connector operates in the request and reply scenario. This independent message processing can occur in another IBM InfoSphere DataStage job that has a WebSphere MQ connector, or in a separate, stand-alone MQ application.

### Configuring the WebSphere MQ connector for the request and reply scenario

You can configure the connector to participate in request and reply processing.

**Before you begin**

The connector must have an input link and an output link. You must also configure your connection properties and other properties as needed for the job before you run the job.

**About this task**

To configure the connector for the request and reply scenario:

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. To specify the queue that sends request messages, use one of these methods:

<table>
<thead>
<tr>
<th>Job context</th>
<th>Property settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design-time</td>
<td>Queue name property on the Properties tab</td>
</tr>
<tr>
<td>Runtime</td>
<td>WSMQ.QUEUE/NAME data element on the input link schema on the Columns tab</td>
</tr>
</tbody>
</table>

   If you specify the value at design time, the value applies to all request messages for the job. If you specify the value at run time, each request message
has its own value. When both design-time and runtime values are provided, the runtime values take precedence.

3. To specify the queue that reads reply messages, use one of these methods:

<table>
<thead>
<tr>
<th>Job context</th>
<th>Property settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design-time</td>
<td>Reply to queue property on the Properties tab</td>
</tr>
<tr>
<td>Runtime</td>
<td>WSMQ.REPLYTOQ data element on the input link schema on the Columns tab</td>
</tr>
</tbody>
</table>

If you specify the value at design time, the value applies to all request messages for the job. If you specify the value at run time, each request message has its own value. When both design-time and runtime values are provided, the runtime values take precedence.

4. Optional: Define the request queue or the reply queue as a dynamic queue. For details, see “Specifying dynamic queues for the request links and reply links.”

5. Optional: To specify the message identifier to match against the correlation identifiers in the reply message, use one of these methods:

<table>
<thead>
<tr>
<th>Job context</th>
<th>Property settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design-time</td>
<td>Value sub-property for the Message ID property on the Properties tab</td>
</tr>
<tr>
<td>Runtime</td>
<td>WSMQ.MSGID data element on the input link schema on the Columns tab</td>
</tr>
</tbody>
</table>

If you specify the value at design time, the value applies to all request messages for the job. If you specify the value at run time, each request message has its own value. When both design-time and runtime values are provided, the runtime values take precedence. If you skip this step, the queue manager assigns the unique message identifier to each request message.

### Specifying dynamic queues for the request links and reply links

You can configure a request queue and the reply queue to be dynamic queues.

#### Before you begin

You must create a job in which the WebSphere MQ connector is part of the request and reply scenario. Attach an input link and an output link to the connector. You also must configure your connection properties and other properties as needed for the job at some point before you run the job.

#### About this task

You can control whether the request queue or the reply queue is opened as a dynamic queue. To use the request queue as a dynamic queue, you must specify values in the following properties:

- **Queue name** value in the **Dynamic queue** group
- **Queue name** property in the **Usage** section that is pointing to a model queue instead of a local queue

To use the reply queue as a dynamic queue, you must specify values in the following properties:

- **Queue name** value in the **Dynamic queue** group
• **Reply to queue** property in the **Set header fields** group that is pointing to a model queue instead of a local queue

To specify a dynamic queue for the request or reply links:

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. On the Output tab, select the output link for requests or select the input link on the Input tab for replies.
3. Specify the templates to create the dynamic queues:
   - For a dynamic request queue, in the **Usage** section on the Properties tab, type or select the name of the model queue in the **Queue name** property.
   - For a dynamic reply queue, in the **Set header** section on the Properties tab, type or select the name of the model queue in the **Reply to queue** property.
4. For either the request or reply queue, in the **Other queue settings** section on the Properties tab, set **Dynamic queue** to **Yes**.
5. Specify the name of the dynamic queue in the **Queue name** property. For request queues, you can specify the full name or a partial name for the dynamic queue. A partial name is specified with an asterisk (*) at the end of the name. If you type a partial name, the queue manager produces a unique queue name when it creates the dynamic queue. If the name is not unique for the local queue manager, an error is generated.
6. For reply queues only, specify how the connector closes the queue in the **Reply queue close options** property. The connector can close the queue, purge the messages from the queue before closing it, or the connector can delete the queue after closing it.

---

**Information reuse and the WebSphere MQ connector**

The WebSphere MQ provides ways for you to save and reuse information such as connection specifications and property values in a data connection object, the metadata, and the job parameters.

**Metadata and the WebSphere MQ connector**

When the connector has access to the runtime server in a design-time environment, you can work with metadata at the column level.

You can load metadata from your local repository or a remote repository if the remote repository is configured to be shared in the administrator application. You can also save metadata for later reuse by the same type of connector.

**Publish/subscribe and the WebSphere MQ connector**

IBM WebSphere MQ connector supports the publish/subscribe messaging model.

You must install WebSphere MQ software to enable the publish/subscribe broker to use the queue manager. Also, you cannot use the request and reply scenario with publish/subscribe.

In publish/subscribe, WebSphere MQ applications or subscribers subscribe to certain topics and then receive messages related to those topics. Applications that publish messages are publishers. The broker is the process that manages the
messages from the publishers to the subscribers. The broker must verify that the subscribers receive the publication messages to which they subscribed.

The connector acts as either a publisher or a subscriber. As a publisher, the connector sends publication messages from the input link to the publisher queue. The broker reads the messages on the publisher queue and sends the messages to the subscriber queues. As a subscriber, the connector reads messages from the subscriber queue and delivers them on the output link for further processing in the job.

The links that are attached to the connector determine how the queue is used by the queue manager in publish/subscribe:

**Input link**
Name of the publisher queue. The publisher stage uses this queue to send messages to the broker. For the MQRFH service type, this name is the stream name. For the MQRFH2 service type, this name is the queue name that is associated with the input node. The broker collects the publication messages in this node and passes them to the publication node in the same message flow.

**Output link**
Name of the subscriber queue. The broker sends publication messages for the subscriber queue in this stage.

**Publish/subscribe installation activities**

When you install the SupportPac that is required for publish/subscribe, you can define a queue manager as a publish/subscribe broker. When the broker is started, the publish/subscribe infrastructure is established on the queue manager. The infrastructure includes various system queues for the internal management of subscription and publication messages by the broker.

**Publish/subscribe prerequisites**
To use publish/subscribe with the WebSphere MQ connector, you must install certain applications, SupportPacs, or fix packs, depending upon whether you want to use the MQRFH or MQRFH2 message format.

When you install the SupportPac, you can define a queue manager as a publish/subscribe broker. When the broker is started, the necessary publish/subscribe infrastructure is established on the queue manager. The infrastructure includes various system queues for the internal management of subscription and publication messages by the broker.

**MQRFH message format requirements**
You must install one of the following version combinations:
- IBM WebSphere MQ, version 5.3 with the MA0C SupportPac with a fix pack less than 8 or with no fix pack
- IBM WebSphere MQ, version 5.3 without the MA0C SupportPac, but with fix pack 8 or later
- IBM WebSphere MQ, version 6.0

**MQRFH2 message format requirements**
You must use the IBM WebSphere Message Broker 6.0.
Specifying dynamic queues for Publish/Subscribe

In publish/subscribe, you can specify that the reply queue is a dynamic queue.

Before you begin

You must create a job in which the WebSphere MQ connector is configured for publish/subscribe. If the connector is configured as a publisher, it has an input link. If the connector is configured as a subscriber, it has an output link. You also must configure your connection properties and other properties as needed for the job before you run the job.

About this task

In publish/subscribe, the connector sends command messages to the broker. The broker replies to these command messages by acknowledging that the messages are successfully processed. You can specify that the queue for these messages between the connector and the broker is a dynamic queue. You cannot specify closing options for the connector. If the reply queue is a dynamic reply queue, the connector purges the messages from the queue, closes the queue, and then deletes the queue at the end of the job.

To specify a dynamic queue in your publish/subscribe job:

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. In the navigator, click the input link for the publisher or the output link for the subscriber.
3. In the Reply queue property in the Publish/Subscribe group on the Properties tab, type or select the name of the model queue that is the template for the dynamic queue.
4. In the Publish/Subscribe group, set Dynamic reply queue to Yes.
5. In the Queue name property, specify the full name or partial name for the dynamic queue. A partial name is specified with an asterisk (*) at the end of the name. If you type a partial name, the queue manager produces a unique queue name when it creates the dynamic queue. If the name is not unique for the local queue manager, an error is generated.

Configuring the WebSphere MQ connector for publish/subscribe

You can configure the WebSphere MQ connector to use the publish/subscribe model.

Before you begin

You must add an input link or an output link to the connector, depending upon whether you want the connector to act as a publisher or a subscriber. Also, the messages must be in either the MQHRF or MQHRF2 format.

About this task

The connector performs the following high-level tasks in publish/subscribe:

1. Assembles messages in MQHRF or MQHRF2 format.
2. Provides values for the commands in the message format header fields of publication command messages.
3. Sends the registration and deregistration command messages to the correct control queue on the broker.

To configure the connector for publish/subscribe:

**Procedure**
1. Double-click the connector on the job canvas to open the stage editor.
2. Set **Publish/Subscribe** to **Yes**.
3. Select the **Service type** of the command messages for registration, deregistration, and publication.
4. To register a subscriber or publisher for a specific topic, type the topic name in the **Topic** sub-property of the **Registration** property.
5. To deregister a subscriber or publisher for a specific topic, type the topic name in the **Topic** sub-property of the **Deregistration** property.
6. To send a publication message on a specific topic, use one of the following options:
   - Type the topic name in the **Topic** sub-property of the **Publication settings** property
   - Provide the topic name value to the WSMQ.TOPIC data element column when this column is defined on the input link
7. For input links (publishers) only, configure the properties in the **Publication settings** group to instruct the connector to prepare the publication command message.
8. Configure the properties in the **Registration** group to instruct the connector to prepare the registration command message. The connector then sends this message to the broker control queue at the beginning of the job.
9. Configure the properties in the **Deregistration** group to instruct the connector to prepare the command deregistration message. The connector then sends this message to the broker control queue at the end of the job.
10. For output links (subscribers), specify the subscription values that apply to registration or deregistration.
    a. Specify the **Subscription name**, **Subscription identity**, and **Stream name** properties.
    b. For the MQRFH2 service type only, specify the **Content filter** property and the **Subscription point** property.
11. Before you run your publish/subscribe job, start the queue manager that is the broker with the following command: `strmqbrk -m qmname`.

**WebSphere MQ connector properties**

Properties define how the connector operates in a job. Different properties are available depending upon the context in which you use the connector: source context, target context, or request context for request and reply scenarios.

All of the properties that display in the **Connection** section or the **Usage** section of the Properties tab are listed in alphabetical order.

**Access mode**

Use this property to specify how the source queue is opened.
For output links, this access mode is for the queue upon which you are reading messages.

For reply links, this access mode is for the queue from which the replies are received.

The default value is **As in queue definition**.

The following values are available:

**As in queue definition**
- The queue is opened by using the default access as defined for that queue.

**Shared**
- The queue can be accessed simultaneously by multiple applications or by the same application.

**Exclusive**
- The queue can be accessed by only one application at a time. No application can open this queue until it is closed by the application that has opened it.

**Exclusive if granted**
- The attempt to open the queue initially is in exclusive mode. If the attempt fails (because another application already this queue open), another attempt is made to open the queue with shared access.

**Accounting token**
- Use the properties in this group to specify the accounting token in messages.
- This token is constructed with application-specific and environment-specific information.
- For input links, this property is available only if you set **Context mode** to **Set identity** or **Set all**.

**Alternate security ID**
- Use the properties in this group to specify the Windows security identifier (SID) that uniquely identifies the alternate user.
- The queue manager uses this identifier when the manager opens this queue.

**Alternate user ID**
- Use this property to specify the identifier of the alternate user that the security service of the queue manager uses to open this queue.
- Valid values are up to 12 characters.

**Append node number**
- Use this property to append the node number to the work queue name.
- This property specifies whether the node number is appended to the work queue name. If you set this property to Yes, you can use separate work queues for each parallel node. The work queue that the connector opens on each node is named...
work_queue.node_number where work_queue is the value specified in the Name property of the Work queue property and node_number is the zero-based index of the node.

This property is available only if you set the Message read mode property to Move to work queue.

The default is No.

Related reference:
- “Name” on page 102
  Use this property to specify the work queue that messages will be moved to.
- “Work queue” on page 131
  Use the properties in this group to define a work queue.
- “Context mode for work queue” on page 78
  Use this property to specify the context mode when the work queue is opened.
- “Monitor queue depth” on page 101
  Use this property to control the work queue depth during job execution.
- “Message read mode” on page 94
  Use this property to specify how messages are read in the current transaction.

**Application ID data**

Use this property to specify values that represent the application that originally put the message on the queue.

There is no default value. If this value is blank for an input link, the default value for the source message is used. The default source message value is generated by the queue manager; and it is a blank value. If this value is blank for an output link, this property is not used for filtering messages.

For input links and request links, valid values are up to 32 characters.

For input links and request links, this property is available only if you set Context mode to Set identity or Set all.

For output links, you can type one or more character values that are separated by spaces or commas, up to a total of 256 characters. When you type multiple values, the connector uses all of these values to filter messages.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.
The data element for this property is WSMQ.APPLIDENTITYDATA.

**Application origin data**

Use this property to specify customized, application-specific information about the origin of the message.

There is no default value. If this value is blank for an input link, the default value for the output message is used. The default output message value is generated by the queue manager; and it is a blank value. If this value is blank for an output link, this property is not used for filtering messages.

For input and request links, valid values are up to 4 characters.

For input links and request links, this property is available only if you set **Context mode** to **Set all**.

For output links, you can type one or more characters that are separated by spaces or commas, up to a total of 256 characters. When you type multiple values, the connector uses all of these values to filter messages.

You can use different ways to specify this value for input links and output links:

- **Specify the value at the job level.** You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- **For input links, you can specify the value at the message level.** Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the **Data element** value on the **Columns** tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the **Data element** column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.APPLORIGINDATA.

**Backout count**

For output links only, use this property to specify the specific value or a range of values that represents the number of times that a message was rolled back for message filtering.

The queue manager increases this number each time that a message that was read from the queue is rolled back. The one exception to this computation is that the browse operation is not included in this count.

There is no default value. If this value is blank, this property is not used for filtering messages.

Use the value in this property to specify filtering conditions for messages. You can provide the value in different ways:

- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: a-b
• A combination of lists and ranges that are comma-delimited or space-delimited
Type no more than 256 characters.

The data element for this property is WSMQ.BACKOUTCOUNT.

**Binding mode**
Select the binding mode to use when the queue manager selects a physical queue instance from the cluster.

The default value is *As in queue definition*.

The following values are available:

**As in queue definition**
The default binding for the shared cluster queue can be used. The WebSphere MQ administration tools specify this binding for the cluster queue. This setting is resolved to one of the other two values for this property.

**On open**
The queue manager determines the queue instance in the cluster to which all messages are sent. When the cluster queue is initially opened by the queue manager, the queue manager selects a queue instance. This is the queue to which all subsequent messages are sent.

**Not fixed**
The queue manager selects the queue instance for each individually sent message. As a result, many different physical queue instances might be used. Select this option to provide load balancing.

**Case sensitive**
Use this property to specify whether or not text comparisons are case-sensitive.

If the connector uses multiple input links and you choose *Ordered* in the *Record ordering* field, use the *Case sensitive* field to specify whether or not text comparisons in a sort operation are case-sensitive.

**Blocking transaction processing**
Use this property to indicate whether the connector should call an external method, which is created by the user and provided in a shared library, for each input message to determine if it is a blocking transaction message.

A blocking transaction message is the source queue message that the connector must write to the work queue, release down the output link, and wait for the job to process it and remove it from the work queue before fetching the next message from the source queue.

The default value is No.

This property is available only if you set the *Message read mode property* to *Move to work queue*.
Related reference:

"Method name (Blocking transaction processing)" on page 98
Use this property to specify the external (user-defined) method that the connector calls for each input message to determine if that message represents a blocking transaction message.

"Module name (Blocking transaction processing)" on page 101
Use this property to specify the name of the shared (dynamic) library which implements the method that the connector calls after each message to determine whether the message represents a blocking transaction message.

"Timeout (Blocking transaction processing)" on page 124
Use this property to specify the time, in seconds, that the connector will wait for a job to process the blocking transaction message before logging error messages and stopping the job.

Channel name
Use this property to specify the name of the client connection channel through which messages are sent from the connector to the remote queue manager.

As a part of the Client channel definition group of properties, this value contributes to the replacement of any values that are specified in the MQSERVER, MQCHLLIB, or MQCHLTAB environment variables.

Valid values for this property are up to 20 characters.

Client channel definition
Use the properties in this group to define the connection channel that connects to the remote queue manager from the connector.

This group of properties is available only if you set the Mode property to Client.

The values for the properties in this group replace any values that are specified in the MQSERVER, MQCHLLIB, or MQCHLTAB environment variables. Alternatively, you can use these environment variables instead of specifying the sub-properties in this group.

Only the channel name, transport type and connection name are included in the channel definition when you use these properties. The resulting channel definition is similar to the definition that is specified by the MQSERVER environment variable. You cannot specify other channel settings, such as a specific security exit or SSL settings. If you need to control these additional channel attributes, specify the channel definition through the MQCHLLIB and MQCHLTAB environment variables.

Cluster queue
Use this property to specify whether the target queue is accessed as a shared cluster queue.

If you set this value to Yes and you also specify a value in the Queue manager property for the cluster queue, the message is sent to that queue manager. If you set the value to Yes and no queue manager name is specified, the queue manager is selected dynamically from the cluster.

The default value is No.
Coded character set ID

Use this property to specify the identifier for the coded character set for messages.

For input and request links, you can specify only one integer between -2 and 999999999. The following values are available:

-2 Specifies that the EMBEDDED coded character set identifier special value is used.
-1 Specifies that the INHERIT coded character set identifier special value is used.
0 Specifies that the DEFAULT or the Q_MGR coded character set identifier value for the current queue manager connection is used.
1-999999999 Specifies that the specific coded character set identifier value is used. This identifier must be recognized by the queue manager.

The default value for input links and request links is 0.

For output links, you can type one or more numeric values up to a total of 256 characters. There is no default value for output links. When this value is blank, messages are not filtered based upon their coded character set ID value.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding column (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.CODEDCHARSETID.

Coded character set ID for Message conversion

Use this property to specify the identifier for the coded character set used when the queue manager converts character data in source messages.

Use this property, along with the Encoding property, to determine the text and numeric conversions when you set the Message conversion property to Yes.

This property is available for output links and reply links only. If you do not select or type a value (that is, 0 remains the value), the default coded character set identifier of the running platform is assumed. The default value is 0. Valid values are integers between -2 and 999999999. The following values are available:

-2 Specifies that the EMBEDDED coded character set identifier special value is used.
-1 Specifies that the INHERIT coded character set identifier special value is used.

0 Specifies that the DEFAULT or the Q_MGR coded character set identifier value for the current queue manager connection is used.

1-999999999 Specifies that the specific coded character set identifier value is used. This identifier must be recognized by the queue manager.

**Connection name**

Use this property to specify the name of the client connection for this connector.

As a part of the Client channel definition group of properties, this value contributes to the replacement of any values that are specified in the MQSERVER, MQCHLLIB, or MQCHLTAB environment variables.

Valid values are up to 264 characters.

You must type a value that complies to the format of the transport type that you have specified in the Transport type property. For examples of different types and formats, refer to the following table.

<table>
<thead>
<tr>
<th>Type</th>
<th>Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU62 or NetBIOS</td>
<td>ModeName/TpName</td>
<td>80X99</td>
</tr>
<tr>
<td>TCP or UDP</td>
<td>server-address(PortNumber)</td>
<td>9.20.4.2(2005)</td>
</tr>
<tr>
<td>SPX</td>
<td>ConnectionName(SocketNumber)</td>
<td>000001.00005A7161E5(5E88)</td>
</tr>
<tr>
<td>DECnet</td>
<td>nodename(objectnumber)</td>
<td>node(task)</td>
</tr>
</tbody>
</table>

**Content filter**

Use this property to specify the content filter when the subscriber registers or deregisters.

This property is available only if you set the value of the Service type property to MQRFH2.

The filter contains an expression that the broker uses when the broker determines whether to forward publications to the subscriber. You can specify this value in addition to the Topic property. If you specify both properties, the broker forwards the publications to the subscriber:

- When the topic of the publication is one of the topics to which the subscriber has registered
- When the content of the publications matches the content of the filter expression provided in this Content filter property

If you set the values for both the Registration property and the Deregistration property to No, the value of this property is ignored.

Valid values are up to 512 characters. There is no default value.
Context mode

Use this property to specify whether any identity context or origin context values are included in all messages when the target queue is opened.

When a queue manager opens a queue to send messages to it, the queue manager can open the queue with different context mode options. The context mode is determined partially by the identity fields that are specified on the messages that are sent to this queue. You select this value depending upon the security preferences you want to define and upon the message origin data elements in the schema that is used for target messages.

The default value is None.

The following values are listed in order by the level at which message context information is overridden.

None  No context fields are included in the target messages.

Set identity

Identity context fields can be included in the target messages with the following properties:

• User ID
• Accounting token
• Application ID data

Set all

Both identity and origin context fields can be included in the target messages. In addition to the properties for the Set identity value, the following origin context properties are available for the Set all value:

• Put application name
• Put application type
• Put date
• Put time
• Application origin data

Context mode for Error queue

Use this property to specify the context mode when the error queue is opened.

The default value is None.

For messages that are sent to the error queue, the following modes are available:

None  Does not preserve any context fields

Set identity

Preserves the identity context fields

Set all  Preserves the identity and origin context fields

Context mode for work queue

Use this property to specify the context mode when the work queue is opened.

The default value is None.

For messages that are moved from the source queue to the work queue, the following modes are available:
None  Does not preserve any context fields of the source message header

Set identity
Preserves the identity context fields of the source message header

Set all  Preserves the identity and origin context fields of the source message header

Related reference:
“Work queue” on page 131
Use the properties in this group to define a work queue.
“Name” on page 102
Use this property to specify the work queue that messages will be moved to.
“Append node number” on page 71
Use this property to append the node number to the work queue name.
“Monitor queue depth” on page 101
Use this property to control the work queue depth during job execution.
“Message read mode” on page 94
Use this property to specify how messages are read in the current transaction.

Correlation ID
Use the properties in this group to specify the correlation identifier for messages.

Use the correlation identifier to relate one message to another or one message to other work that is performed by a specific application.

Correlation ID for Deregistration
Use this property to specify the correlation identifier for deregistration command messages.

The connector uses this value for the correlation identifier field in the deregistration command message header. If you also specify Correlation ID as identity in the General options property, the value in the Correlation ID property for the Deregistration group identifies the publisher for input links or the subscriber for output links.

Valid values are up to 48 characters. There is no default value.

Correlation ID for Registration
Use this property to specify the correlation identifier for registration command messages.

The connector uses this ID for the correlation identifier field in the registration command message header. If you also specify Correlation ID as identity as the value for the General options property, the value in this property identifies the publisher for input links or the subscriber for output links.

Valid values for this property are up to 48 characters. There is no default value.

Custom value for Feedback
Use this property to specify a user-defined feedback code or reason code for the report message.

You can type feedback codes or reason codes.
For input links and reject links, if you already selected a value in the **System value** property, you cannot type a custom value in this property. If you did not select a **System value** property value, type a single integer between 0 and 999999999.

For output links, you can type the custom value in different ways:
- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: a-b
- A combination of lists and ranges that are comma-delimited or space-delimited

For output links, each integer must be between 0 and 999999999.

The total length of the combined values in this property must not exceed 256 characters. There is no default value. If this value is blank for an output link, this property is not used for filtering messages.

**Custom value for Format**

Use this property to specify a user-defined value for the message format.

For input links, if you already selected a value in the **System value** property, you cannot type a value in this property. If you did not select a value in the **System value** property, type a custom value in this property up to 8 characters. There is no default value.

For output links, you can type one or more values in this property, in addition to the specified value in the **System value** property. Valid values for output links are strings.

You can type up to eight characters, not including special characters in this total, for each value. You can also type multiple values. Each value must be separated with a comma or space. Insert a backslash (\) as an escape character before any comma, space, or backslash (\) in any of the values. For example, if your value is FMT \1\2, type the following characters:

FMT \ 1\2

The total length of the value can be up to 256 characters. There is no default value. If this value is blank for an output link, no filtering is performed on custom format values for source messages.

**Custom value for Message type**

Use this property to specify a user-defined value for the message type (input links) or a range of values for source messages (output links).

For input links, if you already selected a value in the **System value** property, you cannot type a value in this property. If you did not select a value in the **System value** property, type a single integer between 1 and 999999999. There is no default value.

For output links, you can type the value of this property in different ways:
- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: a-b
- A combination of lists and ranges that are comma-delimited or space-delimited
For output links, each integer must be between 1 and 999999999. The total length of the combined values in this property must not exceed 256 characters. There is no default value. If this value is blank for an output link, messages are not filtered based upon their message type value.

**Custom value for Publication format**

For input links only, use this property to specify a user-defined format for the publication payload.

If you already selected a value in the **System value** property, you cannot type a custom value. If you did not select a **System value** property, type the custom value in this property up to 8 characters. There is no default value.

**Custom value for Put application type**

Use this property to specify a user-defined value for the application type upon which the message is put.

For input links, if you already selected a value in the **System value** property, you cannot type a custom value. If you did not select a value in the **System value** property, type an integer between -1 and 999999999. There is no default value.

For output links, you can type the value of this property in different ways:

- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: a-b
- A combination of lists and ranges that are comma-delimited or space-delimited

For output links, each integer must be between -1 and 999999999, where -1 specifies UNKNOWN. The total length of the combined values in this property must not exceed 256 characters. There is no default value. If this value is blank for an output link, this property is not used for filtering messages.

**Deregistration**

Use this property to specify whether the broker deregisters this publisher or subscriber from the specified topic. This deregistration occurs before job completion, but after all of the messages in the job are produced or consumed.

For input links (as a publisher), this property is available only if you set **Service type** to MQRFH. For output links (as a subscriber), you can set **Service type** to MQRFH or MQRFH2.

The default value is No.

Use the sub-properties in this group to specify additional deregistration information.

**Dynamic queue**

Use this property to specify whether the queue manager creates a dynamic queue at run time to serve only the message operation of a specific application. This queue name is based upon the name of a model queue.

For input links, if you set **Dynamic queue** to Yes and **Queue name** to a model queue name, the target dynamic queue is created by the queue manager and messages are sent to the dynamic queue. Although you can specify the queue name, you
cannot specify the close options because the connector has no way of knowing whether the sent messages were processed by their consumers. Therefore, although the connector can close the dynamic queue, the connector cannot destroy the dynamic queue.

For request links, if you set **Queue name** to point to a model queue, the queue manager creates the request dynamic queue. The queue manager sends request messages to the request dynamic queue. If the **Reply to queue** property points to a model queue, the queue manager creates the reply dynamic queue. The name of this dynamic reply queue is specified in the request messages. The connector waits for reply messages to appear on this queue.

The default value is **No**.

**Dynamic reply queue**

Use this property to specify whether the reply queue is accessed as a dynamic queue. The broker uses this reply queue to send replies to the command messages that the connector sends to the broker.

This property is available only if you set **Publish/Subscribe** to **Yes**.

If you set **Dynamic reply queue** to **Yes**, a dynamic queue is created for the connector by the queue manager. The connector then waits for the broker to send response messages to it. A value of **Yes** also specifies that the queue name in the **Reply queue** property is used as the model queue name.

The default value is **No**.

**Enable payload reference**

Use this property to specify whether to pass a reference to the message payload rather than passing the actual payload data.

This property is available only if you set **Message order and assembly** to **Individual (unordered)**, **Individual (ordered)**, or **Assemble logical messages**.

If you set **Enable payload reference** to **Yes**, a reference string (locator) is passed instead of the inline message payload. The message is then read at the point in the job where the message needs to be consumed. If a connector that supports payload references, such as an ODBC connector or a WebSphere MQ connector, is used as a destination stage, the actual payload data is read by the connector. If another stage, such as a Sequential file stage is at this point in the job where the message needs to be consumed, the message reference string is written to the file and not the actual payload data.

If you use a reference string instead of the actual payload data, you can move larger messages through the job that are otherwise not allowed. However, you also cannot manipulate the payload data as it is passed to its destination because you pass the reference string, not the actual data.

The default value is **No**.

**Encoding**

Use this property to specify the encoding value or values for numeric data in a message.
For input and request links, you can specify only one integer that is between -1 and 999999999. A value of -1 specifies that the native number encoding for the current queue manager connection is used. The default value is -1.

For output links, you can type the custom value in different ways:

- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: a-b
- A combination of lists and ranges that are comma-delimited or space-delimited

The total length of the combined values in this property must not exceed 256 characters. There is no default value. If this value is blank, this property is not used for filtering messages.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.ENCODING.

**Encoding for Message conversion**

Use this property to specify the encoding when the queue manager converts numeric data in source messages.

This encoding property defines how values such as binary integers, packed decimal integers, and floating point numbers are processed in the message format headers. Use this property with the Coded character set ID property to determine the numeric and text conversions when the Message conversion property is set to Yes.

If you do not specify a value or set it to -1, the native numeric encoding for the current queue manager connection is assumed. The default value is -1. Valid values are integers between -1 and 999999999.

**End of data message type**

Use this property to specify the message type in the message header that specifies the end of the message reading process.

The connector stops reading messages when it receives a message that matches the type that is specified in this property. The exception to this is when there are
additional messages in a group of messages and the message group must be assembled. In the group assembly scenario, the rest of the messages in the group are also read.

You can specify an integer between 0 and 999999999.

There is no default value.

**Error queue**

Use the properties in this group to define a local or remote error queue.

For source queues (output links), the connector uses the error queue to prevent messages from remaining on the source queue if the connector rolls back the transaction. For target queues (input links), the connector can use the error queue as a backup target queue for messages that fail to be sent to the target queue.

For output links, this property is available only if you set Message read mode to Delete (under transaction). This value specifies that the failed messages are removed from the source queue after they have been rolled back by the connector. Then the failed messages are moved to the error queue. If you set Message read mode to Delete, the messages are automatically removed from the queue. No rollback is possible. If you set Message read mode to Keep, the messages remain on the source queue after the job has failed.

You can also use the reject link to handle failed messages for input links only.

The default value for this property is No.

**Expiry**

Use this property to specify the value that defines the lifetime of a message.

The lifetime of a message is the amount of time, in tenths of seconds, that expires between the time at which the message is sent by one application and the time at which another application reads the message. If this amount of time is greater than the expiry time, the message expires and the message is discarded by the queue manager. However, the message is not discarded by the queue manager if another application attempts to read it.

For input and request links, specify only one value, which is an integer between -1 and 999999999. A value of -1 specifies an unlimited number of seconds, which means that the message does not expire. The default value is -1.

For output links, you can type the custom value in different ways:

- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: a-b
- A combination of lists and ranges that are comma-delimited or space-delimited

The total length of the combined values in this property must not exceed 256 characters. There is no default value. If this value is blank, this property is not used for filtering messages.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also
specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.EXPIRY.

**Feedback**

Use the properties in this group to specify acceptable feedback and reason codes for target messages (input links and request links) or source messages (output links).

For source messages (output links), you can specify multiple values for filtering. You can include any combination of system feedback codes and custom feedback codes.

For target messages (input links and request links), you can specify only one value: system or custom. This value is then set in the message header.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.FEEDBACK.

**Filter messages**

Use the properties in this group to specify filter criteria for source messages (output links).

The default value is No.

**Format**

Use the properties in this group to specify the format for the message.
For source messages (output links), you can specify multiple values for filtering. You can include any combination of system formats and custom formats.

For target messages (input links and request links), you can specify only one value: system or custom. This value is then set in the message header.

You can use different ways to specify this value for input links and output links:
- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.FORMAT.

**General options for Deregistration**

Use this property to specify whether the broker deregisters the publisher for input links or deregisters the subscriber for output links from the specified topic upon job completion.

This property is available only if you set Deregistration to Yes.

You can select multiple values from this list by clicking the property and then selecting the check boxes. When you finish, click outside of this property. There is no default value.

The following values are available:

**Correlation ID as identity**
For input links, the topics are deregistered for this publisher if the correlation identifier is part of the identity of the publisher. For output links, the topics are deregistered for this subscriber if the Correlation ID property has been specified as part of the identity of the subscriber.

**Deregister all**
For input links, all topics that are registered to this publisher are deregistered. For output links, all topics that are registered to this subscriber are deregistered.

**Leave only**
For output links only, the broker removes the identity set that is associated with the subscription. The identity set is specified in the Identity options property in the Registration group. However, the subscription is not removed, even if its identity set becomes empty as a result of this operation.

**Variable user ID**
For output links only, the broker does not use the user ID of the current
user to identify the subscription to be deregistered. You must select this value to deregister a subscription that was registered by another user. You must also specify this same Variable user ID value in the General options property in the Registration group.

**Note:** If you provide a value for the Subscription name property, the Variable user ID value is overridden by the Subscription name value.

**General options for Registration**

Use this property to specify how the publisher (for input links) or subscriber (for output links) is identified to a broker.

You can select multiple values from this list by clicking the property and then selecting the check boxes. When you finish, click outside of this property. There is no default value.

The following values are available:

**Correlation ID as identity**

For input links, the correlation identifier that is specified in the Correlation ID property in the Registration group is part of the identity of the publisher. The correlation identifier is especially important when multiple publishers access the same queue because each publisher is uniquely identified.

For output links, the correlation identifier that is specified in the Correlation ID property is part of the identity of the subscriber.

**Anonymous**

For input links, Anonymous specifies to the broker that the identity of the publisher is not to be revealed except to those subscribers with more authority.

For output links, Anonymous specifies to the broker that the identity of the subscriber is not to be revealed except to those publishers with more authority.

**Note:** For output links, this value is ignored unless you set Service type to MQRFH.

**Local**

For input links, Local specifies to the broker that the publications for this publisher are sent only to those subscribers who are registered as local on this same broker. The subscriber must also register for the published topics.

For output links, Local specifies to the broker that this subscription is local. Only publications that are made on this broker and on this subscription are distributed to this subscriber.

**New publications only**

For output links, only those new publications that are published after the subscriber registers are sent to the subscriber. The publications that are retained are not sent to the subscriber, even if they are published on a topic for which the subscriber registered.

**Duplicates OK**

For output links, Duplicates OK specifies to the broker that it can send the same publication more than once to the subscriber. As a result, the broker does not have to check for duplicate publications, thus potentially improving performance.
Group ID

Use this property to specify the group identifier for messages. The group identifier defines the messages that belong to a specified group.

For input links and request links, this property is available only if you set Header version to 2. For output links, this property is always available.

If you specify a value in the Value property, you must also specify the correct values for the Message flags property and the Message sequence number property. Then the queue manager knows that this is a group message. If this property is not correctly specified, the Group ID property values are ignored in the source message.

Header version

Use this property to specify the WebSphere MQ application version number for the header structure of the source message.

The default value is 2.

The following values are available:

1 This version is supported in all environments.
2 This version is supported in specific environments, including AIX, Solaris, Linux, and Windows. The queue manager performs additional checks on any header structures that appear at the beginning of the application message data.

Hex for Accounting token

Use this property to specify whether the value in the Value property is treated as an array of pairs of hexadecimal digits or as text.

The default value is No, which specifies that the value is treated as text.

Hex for Alternate security ID

Use this property to specify whether the value in the Value property is treated as an array of pairs of hexadecimal digits or as text.

Set this property to Yes so that you can type a 40-byte security identifier in the Value property.

The default value is Yes, which specifies that the value is treated as an array of pairs of hexadecimal digits.

Hex for Correlation ID

Use this property to specify whether the value in the Value property is treated as an array of pairs of hexadecimal digits or as text.

The default value is No, which specifies that the value is treated as text.

Hex for Group ID

Use this property to specify whether the value in the Value property is treated as an array of pairs of hexadecimal digits or as text.

This property is available only if you set Header version to 2.
The default value is No, which specifies that the value is treated as text.

**Hex for Message ID**

Use this property to specify whether the value in the **Value** property is treated as an array of pairs of hexadecimal digits or as text.

The default value is No, which specifies that the value is treated as text.

**Identity options**

Use this property to specify identity registration options for this subscriber.

Each application that requires subscriptions is represented by an identity. The broker maintains a set of subscriber identities for each subscription.

If no identity values are specified, the registration continues, regardless of any potential set of identities that are present.

You can select one or more values from this list. There is no default value.

The following values are available:

**Add name**

This subscriber name is added to the subscription if it is missing. If a subscription already exists for this name or if a matching subscription with a different name exists, the registration fails.

**No alteration**

Existing matching subscription attributes are not modified when the subscription is created. However, for subsequent subscriptions that match the identity of existing subscriptions, modifiable attributes of the original subscription are overwritten.

If either Join shared or Join exclusive is specified as the **Identity options** value, along with the **Subscription identity** value, the **Identity options** value is ignored. The Join shared value or the Join exclusive value specifies that the broker adds the **Subscription identity** value to the current set of identities for the subscription, regardless of whether No alteration is selected.

**Join shared**

The identity is added to the identity set for the subscription. The following statements must all be true:

- None of the current members of the identity set for this subscription match the new one.
- The subscription is not exclusively locked.

If the identity already has a shared entry for the subscription, the command succeeds. But it returns an already joined warning. If the subscription is exclusively locked, a subscription locked message is returned unless the entry with the locked subscription has a shared, preexisting identity. If both entries have the same identity, the lock is automatically modified to become a shared lock.

If the subscription contains a user ID that is different from the one in the identity, the registration fails unless **Variable user ID** is set on the original subscription. If this value is set, the user ID of the command message is checked to determine whether this ID has the authority to browse the
stream queue and to put to the queue of the subscriber. If this ID does not have sufficient authority, the registration fails again.

**Join exclusive**

This identity is added as the exclusive member of the identity set for the subscription. No other identities can be added to this set.

If the subscription is exclusively locked, the registration fails if the identity with the exclusive lock is not this one. If it is the same identity, the subscription succeeds with an already joined warning.

If this identity has joined with the value of Join shared and this identity is the only entry in this identity set, the set is changed to an exclusive lock that is held by this identity. Otherwise, if the subscription currently has other identities in the identity set with shared access, the registration fails.

If an application attempts to register by using an identity with a user ID that is different from the currently registered user ID, the registration fails unless **Variable user ID** is set on the original subscription. Additionally, if it is set, there is sufficient authority to browse the stream queue and to put to the queue of the subscriber.

**Variable user ID**

The subscriber identity is not restricted to a single user ID. Any user can modify or deregister a subscription if this user has sufficient authority.

To add this value to an existing subscription, the command must come from the same user ID as the original subscription.

If this value refers to an existing subscription with this value selected, the subscription succeeds only if the user ID of the new subscription has the authority to both browse the stream queue and to put to the subscriber queue of the modified subscription. After successful completion, future publications to this subscriber are put to the queue of the subscriber by using the new user ID.

If this subscription does not have this value enabled and the original subscription does have it enabled, this value is removed from this subscription; and the user ID of this subscription is now fixed. If, after this has been implemented, a subscriber already exists who has the same identity, but with a different user ID that is associated to the subscription, the registration fails.

**Key column**

Use this property to specify the name of the column to use as the sort key.

If the connector uses multiple input links and you choose **Ordered** in the **Record ordering** field, use the **Key column** field to specify the field to use as the sort key.

**Match all**

Use this property to determine how filtering based on report values or message flags is handled.

If you set **Match all** to Yes, source messages must contain all of the specified values. If you set **Match all** to No, source messages can contain any of the specified values. The default value is No.
Message content descriptor

Use this property to specify whether the connector includes the service folder (<mcd>) for the message content descriptor in the publication messages.

This property is available only if you set Service type to MQRFH2.

The service folder of the message content descriptor provides information about the publication message structure in addition to the fields in the publication message header and other data in the MQRFH2 format header.

The default value is No.

Message controlled

Use this property to indicate whether the connector should make an external method call after each message to determine whether the transaction should be committed after that message. If the End of wave property is set to a value other than None, committing the transaction also implies the end of the current transaction wave.

The default value is No.

Related reference:

- “Method name (Message controlled)” on page 99
- “Module name” on page 100

Message conversion

Use the properties in this group to specify whether the connector requests that the queue manager performs a message conversion.

The conversion request is for numeric and character data in source messages.

If you set Message conversion to Yes, you must specify the sub-properties in this group.

The default value is No.

Message flags

For input links, this property specifies the flags that are associated with this target message. For output links, use the properties in this group to specify the flags that are associated with this source message.

For source messages (output links), use the Value sub-property to specify message flags for filtering.

For input links only, this property is available only if you set Header version to 2.
There is no default value. If this value is blank for output links, this property is not used for filtering messages.

**Note:** If you set *Segmented message* to Yes, the connector automatically sets the offset values on the generated message segments. If you select any of the following values, they are ignored. The other values for group messages are used in the target messages.

- Segmentation allowed
- Segment
- Last segment

For target messages (input links and request links), you can specify multiple values. These values are then set in the message header.

You can use different ways to specify this value for input links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the *Data element* value on the *Columns* tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

The data element for this property is WSMQ.MSGFLAGS.

**Message ID**

Use the properties in this group to specify the message identifier for messages.

For output links only, you can define this value using the asterisk (*) as a wildcard character. You must also set *Hex* to No.

If you do not specify a value for the *Value* property for target messages (input links), the queue manager assigns a unique value to the message.

**Message options**

Use the properties in this group to specify whether you are going to define the order, the structure, and the access mode of the messages.

You see different sub-properties, depending upon whether an input link or an output link is used with the connector. The default value is No.

**Message order and assembly**

Use this property to specify how segments, logical messages, and groups of messages are retrieved from the source message queue.

When you define the order and assembly options for messages, note the following guidelines:

- A group can consist of one or more logical messages.
- A logical message can belong to a group. But the logical message does not have to belong to a group.
• A logical message can be segmented. The segments are physical messages on the queue.
• A logical message might not be segmented. The entire message is one physical message on the queue.

Valid values

The default value is Individual (ordered).

The following values are available:

Individual (unordered)
   Each physical message on the queue, whether it is a segment or a logical message, is treated as a single message unit; and it is retrieved as such.

Individual (ordered)
   Each physical message on the queue, whether it is a segment or a logical message, is treated as a single message unit; and it is returned as such. All message segments and non-segmented, logical messages are returned in their logical order. The offset values determine the order of message segments; and message sequence numbers determine the order of logical messages.

Assemble logical messages

   Note: When the connector retrieves reply messages as part of a request, the connector always treats these messages as described below.

   Message segments are never returned as individual messages. Only complete logical messages are returned. If a logical message is segmented, the queue manager first reassembles the message before passing the message to the connector. Then the connector returns the message as a single data unit. The logical order of returned messages is followed. The assembled logical messages are returned only after all other logical messages from the same group with a lower sequence number are already returned.

Assemble groups
   Messages that belong to a group of messages are never returned individually from the connector. Only complete groups of messages or logical messages that do not belong to any group are returned. Messages that belong to a group are concatenated by the connector and are returned as a single record.

Message padding

Use this property to specify whether padding is added to the message payload column. The message payload column is the message body minus any format headers.

If you set Message padding to Yes, the following padding is implemented in the message payload:

• If the message payload contains text data, the column for the message is padded with space characters by using the correct character set.
• If the message payload contains binary characters, the column for this message is padded with NULL bytes.
• In the request and reply scenario, padding applies to the payload of the reply messages.
The default value is No.

**Message quantity**

Use this property to specify the number of messages (not rows) to retrieve from the input queue.

After the connector retrieves the number of messages that are specified by this property, the connector stops. When the connector runs in parallel, each process stops when the process retrieves the number of messages that are specified by the property.

The message quantity is the number of queue messages, not the number of rows. When you group messages, each group counts as one message. You can specify integers between -1 and 999999999. A value of -1, which is the default value, specifies an indefinite number of messages. A value of 0 specifies no messages.

**Message read mode**

Use this property to specify how messages are read in the current transaction.

The default value is Delete (under transaction).

The following values are available:

- **Keep** The message remains on the queue. The message is read in browse mode outside of the current transaction. Regardless of what happens to the job, the message always remains on the queue.
  
  If the connector is part of a parallel job and you have configured the job to run on multiple nodes, setting this value produces duplicate messages because each instance of the stage independently browses the messages. To avoid this scenario, configure this stage to run on a single node by setting Execution mode on the Advanced tab to Sequential.

- **Delete** The message is read outside of the current transaction and is subsequently deleted. Regardless of what happens to the job, the message is always deleted from the queue.

- **Delete (under transaction)** The message is read from the queue within the current transaction. If the transaction is committed, the message is removed from the queue. If the transaction is rolled back, the message remains on the queue.

- **Move to work queue** The message is read from the source queue in the local transaction and is moved to the work queue within the same local transaction.

**Message sequence number**

Use this property to specify the sequential number of the logical message within a message group.

Each logical message in a message group is assigned a number beginning with 1 for the first message. This number is increased at a rate of one for each new logical message in this group. Physical messages that are not part of a group are all assigned a sequence number of 1.

For input links and request links, this property is available only if you set **Header version** to 2.
For input links and request links, the default value is 1. You can type a single integer between 1 and 999999999.

For output links, there is no default value. If this value is blank, this property is not used for filtering messages. When you want to filter messages, you can type multiple values in this property.

For output links, you can type the value of this property in different ways:
- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: a-b
- A combination of lists and ranges that are comma-delimited or space-delimited

Each integer must be between 1 and 999999999. The total length of the combined values in this property must not exceed 256 characters. There is no default value. If this value is blank for an output link, the default value for the source message is used.

You can use different ways to specify this value for input links and output links:
- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.MSGSEQNUMBER.

**Message sequence number for Publication settings**
Use this property to specify whether to update the message sequence number in the published messages.

The default value is No. If you select Yes, you must specify the value in the Start value property.

**Message service domain**
Select the service domain for the publication messages.

This property is available only if you set Message content descriptor to Yes.

This value corresponds to the value of the <Msd> element in the folder element of the <mcd> message content descriptor service. This element defines the service domain, which is known to the message broker.

Select one of the following values:
- mrm The message is managed by the Message Repository Manager.
xml  The message is a self-described XML message.
xmllns The message is a self-described XML message with namespaces.
idoc The message is an SAP IDoc message.
none The publication payload is treated as raw bytes of data. The payload is delivered in this format.

The default value is mrm.

**Message set**

Use this property to specify the name of the message set for publication messages.

This property is available only if you set *Message service domain* property to mrm or idoc.

This property represents the value of the `<Set>` element of the `<mcd>` message content descriptor service folder. You must define the message set for the message broker.

Valid values are up to 128 characters. There is no default value.

**Message truncation**

Use this property to specify whether truncation is performed on messages with source message payloads that are larger than the column size for that payload.

Truncation applies only to message payload columns. Other columns are represented as data elements in the schema. These other, non-payload columns must be the correct type and size. Otherwise, the connector determines that the schema contains errors.

If you set *Message truncation* to Yes, the message payload is truncated to the requested column size in characters for text or bytes for binary.

If you set *Message truncation* to No and the message payload in the received message is larger than the payload column in the schema, the connector produces as many records for this message as are required to transfer the entire message payload.

In the request and reply scenario, truncation is applied to the reply message payload.

The default value is Yes.

**Message type**

Use the properties in this group to define the message types that filter source messages (output links). Alternatively, select the message type to set on the target messages (input links).

For source messages (output links), you can specify multiple values for filtering. You can include any combination of system message types and custom message types.

For target messages (input links), you can specify only one value: system or custom. This value is then set in the message header.
You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.MSGTYPE.

**Message type for Message content descriptor**

Use this property to specify the name of the message type for the publication messages.

This property is available only if you set Message service domain to mrm or idoc.

This property corresponds to the value of the <Type> element in the <mcd> element in the message content descriptor service folder. This value represents the message type that must be defined in the message set specified in the Message set property.

Valid values are up to 128 characters. There is no default value.

**Message write mode**

Use this property to specify how messages are written in the current transaction.

The following values are available:

- **Create** The message is created on the target queue outside of the current transaction. Regardless of what happens to the job, the message always remains on the target queue.

- **Create (under transaction)** The message is created on the target queue within the current transaction. If the transaction is committed, the message remains on the target queue. If the transaction is rolled back, the message does not remain on the target queue.

- **Create on content** The message is created on the target queue only if its content is not empty except for the header. The creation takes place outside of the current transaction. If the message content is not empty, the message remains on the target queue, regardless of subsequent job processing.

- **Create on content (under transaction)** The message is created on the target queue only if its content is not empty except for the header. This creation takes place within the current transaction.
transaction. If the transaction is committed, the message remains on the
target queue. If the transaction is rolled back, the message does not remain
on the target queue.

The default value is Create (under transaction).

Method name (Blocking transaction processing)

Use this property to specify the external (user-defined) method that the connector
calls for each input message to determine if that message represents a blocking
transaction message.

Enter the name of the C method that is used to determine whether a source queue
message represents a blocking transaction. The signature of this method is as
follows:

```c
extern "C" int methodName(void* pMessageHeader,
                          void* pMessageBody,
                          int iMessageBodyLength,
                          char** pszLogMessage);
```

`methodName`

the name of the method. This value must be specified in the **Method name**
property.

`pMessageHeader`

pointer to the message header structure (MQMD*) for the source message.
The method might consult the message header field values to determine if
the message is a blocking transaction.

`pMessageBody`

pointer to the memory buffer that contains the full message body of the
source queue message. This includes format headers (if any) and the
message payload.

`iMessageBodyLength`

the length in bytes of the memory buffer pointed to by the `pMessageBody`
argument.

`pszLogMessage`

Pointer to a char* buffer in which the method stores any text message that
it wants to be logged in the job log. The text message will be logged with
the Info severity. When the connector invokes the method, the value
`*pszLogMessage` will be set to NULL. When the method completes, the
connector will check the value of `*pszLogMessage` and if it is not NULL it
will log it as an informational message and will then call:

```c
free(*pszLogMessage);
```

This means that the method needs to allocate the `*pszLogMessage` buffer by
using the `malloc()` method.

`return value`

This integer return value has the following states:

- 0       the message does not signify a blocking transaction
- 1       the message signifies a blocking transaction

*Any other value*

An error occurred. The connector displays an exception and the job
is aborted.
Related reference:

"Module name (Blocking transaction processing)" on page 101

Use this property to specify the name of the shared (dynamic) library which implements the method that the connector calls after each message to determine whether the message represents a blocking transaction message.

"Timeout (Blocking transaction processing)" on page 124

Use this property to specify the time, in seconds, that the connector will wait for a job to process the blocking transaction message before logging error messages and stopping the job.

**Method name (Message controlled)**

Use this property to specify the external, user-defined, method that the connector should call after each message to determine whether the transaction needs to be committed after that message. If the End of wave property is set to a value other than None, committing the transaction also implies the end of the current transaction wave.

The signature of this method is as follows:

```c
extern "C" int methodName(void* pMessageHeader,
                         void* pMessageBody,
                         int iMessageBodyLength,
                         char** pszLogMessage);
```

**methodName**

the name of the method. This value must be specified in the Method name property.

**pMessageHeader**

pointer to the message header structure (MQMD*) for the source message. The method might consult the message header field values to determine if the transaction should be committed after that message.

**pMessageBody**

pointer to the memory buffer that contains the full message body of the source queue message. This includes format headers (if any) and the message payload.

**iMessageBodyLength**

the length in bytes of the memory buffer pointed to by the `pMessageBody` argument.

**pszLogMessage**

Pointer to a `char*` buffer in which the method stores any text message that it wants to be logged in the job log. The text message will be logged with the Info severity. When the connector invokes the method, the value `*pszLogMessage` will be set to NULL. When the method completes, the connector will check the value of `*pszLogMessage` and if it is not NULL it will log it as an informational message and will then call:

```c
free(*pszLogMessage);
```

This means that the method needs to allocate the `*pszLogMessage` buffer by using the `malloc()` method.

**return value**

This integer return value has the following states:

0 the message does not signify a blocking transaction

1 the message signifies a blocking transaction
Any other value
An error occurred. The connector displays an exception and the job is aborted.

Related reference:

"Module name"
Use this property to specify the name of the shared (dynamic) library that implements the method that the connector calls after each message to determine whether it should commit the transaction after that message.

"Message controlled" on page 91
Use this property to indicate whether the connector should make an external method call after each message to determine whether the transaction should be committed after that message. If the End of wave property is set to a value other than None, committing the transaction also implies the end of the current transaction wave.

Mode
Use this property to specify whether the connector functions as a server or client application.

The default value is Server.

This property is the first one in the Connection section. The value that you select for this property determines whether server-specific or client-specific properties are displayed on this tab.

Null order
Use this property to specify where to place null values in the sort order.

If the connector uses multiple input links and you choose Ordered in the Record ordering field, use the Null order field to specify where to put null values in relation to non-null values. The choices are Before and After.

Module name
Use this property to specify the name of the shared (dynamic) library that implements the method that the connector calls after each message to determine whether it should commit the transaction after that message.

Enter the name of the shared library that exports the method specified in the Method name (Message controlled) property. This value may be a full path to the library, or just the name of the library.

If only the name of the library is specified, the environment must be configured so that the connector can open the library at runtime. On Windows this means that the library must reside in a directory listed in the PATH environment variable which is in effect for the job process. For AIX the same applies, but the environment variable is LIBPATH. For HP-UX it is SHLIB_PATH. For other UNIX platforms, such as Linux and Solaris, it is LD_LIBRARY_PATH.
Related reference:

"Method name (Message controlled)" on page 99
Use this property to specify the external, user-defined, method that the connector should call after each message to determine whether the transaction needs to be committed after that message. If the End of wave property is set to a value other than None, committing the transaction also implies the end of the current transaction wave.

"Message controlled" on page 91
Use this property to indicate whether the connector should make an external method call after each message to determine whether the transaction should be committed after that message. If the End of wave property is set to a value other than None, committing the transaction also implies the end of the current transaction wave.

Module name (Blocking transaction processing)
Use this property to specify the name of the shared (dynamic) library which implements the method that the connector calls after each message to determine whether the message represents a blocking transaction message.

Enter the name of the shared library that exports the method specified in the Method name (Blocking transaction processing) property. This value may be a full path to the library, or just the name of the library.

If only the name of the library is specified, the environment must be configured so that the connector can open the library at runtime. On Windows this means that the library must reside in a directory listed in the PATH environment variable which is in effect for the job process. For AIX the same applies, but the environment variable is LIBPATH. For HP-UX it is $SHLIB_PATH. For other UNIX platforms, such as Linux and Solaris, it is $LD_LIBRARY_PATH.

Related reference:

"Method name (Blocking transaction processing)" on page 98
Use this property to specify the external (user-defined) method that the connector calls for each input message to determine if that message represents a blocking transaction message.

"Timeout (Blocking transaction processing)" on page 124
Use this property to specify the time, in seconds, that the connector will wait for a job to process the blocking transaction message before logging error messages and stopping the job.

Monitor queue depth
Use this property to control the work queue depth during job execution.

This property is available only if you set the Message read mode property to Move to work queue.

The following values are available:

Minimum depth
Defines the lower limit allowed for the work queue depth. When it is reached the connector resumes moving messages to the work queue. The value must be a non-negative whole number that is not greater than the Maximum depth.

Maximum depth
Defines the maximum depth allowed for the work queue. If the message
depth on work queue exceeds this limit, the connector stops moving
messages to the work queue and waits for the depth to be less than the
Minimum depth property value. The Maximum depth must be a positive
whole number that is not greater than 999,999,999. To specify unlimited
maximum depth, use the value -1.

Related reference:
“Name”
Use this property to specify the work queue that messages will be moved to.
“Append node number” on page 71
Use this property to append the node number to the work queue name.
“Context mode for work queue” on page 78
Use this property to specify the context mode when the work queue is opened.
“Message read mode” on page 94
Use this property to specify how messages are read in the current transaction.
“Work queue” on page 131
Use the properties in this group to define a work queue.

Name
Use this property to specify the work queue that messages will be moved to.

This property is available only if you set the Message read mode property to
Move to work queue.

Click Select to choose an existing queue on the currently configured queue
manager connection.

Related reference:
“Work queue” on page 131
Use the properties in this group to define a work queue.
“Append node number” on page 71
Use this property to append the node number to the work queue name.
“Context mode for work queue” on page 78
Use this property to specify the context mode when the work queue is opened.
“Monitor queue depth” on page 101
Use this property to control the work queue depth during job execution.
“Message read mode” on page 94
Use this property to specify how messages are read in the current transaction.

Offset
Use this property to specify the location, in bytes, of a physical message segment
relative to the start of the logical message to which the segment belongs.

This property applies to physical messages that are segments of a logical message.
For physical messages that are not segments of a logical message, this value is
always set to 0.

For report messages that are part of a logical group, the Original length property
also determines the offset.

Input and request links
This property is available only if you set Header version to 2.
For input links and request links, this property is the offset value for target messages. The default value is 0. Valid values are integers between 0 and 999999999.

If you set Segmented message to Yes, the connector automatically sets the offset values on the generated message segments. Any value that you type in this property is ignored.

**Output links**

For output links, you can type a single integer or a range of values. There is no default value. If you do not type a value for an output link, this property is not used for filtering messages.

For output links, you can type the value of this property in different ways:
- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: a-b
- A combination of lists and ranges that are comma-delimited or space delimited

Each value must be between 0 and 999999999. The total length of the combined values must not exceed 256 characters.

You can use different ways to specify this value for input links and output links:
- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.OFFSET.

**Original length**

For output links only, use this property to specify the length of the message segment to which the report message relates. This property is not the length of the entire logical message, nor is it the length of the data in the report message itself. The report message thus becomes another segment in the logical message.

This property is applicable only to report messages that are segments. This length is measured differently than the Offset property.

The value in this property is used to filter all messages in this job at design time.

For output links, you can type the custom value in different ways:
- A list of comma-delimited or space-delimited integers
• A range of integers in the following format: a-b
• A combination of lists and ranges that are comma-delimited or space-delimited

The total length of the combined values in this property must not exceed 256 characters. There is no default value. If this value is blank, this property is not used for filtering messages.

The data element for this property is WSMQ.ORIGINALLENGTH.

**Other queue settings**

Use the properties in this group to specify whether you want to specify additional settings for the queue to be accessed.

If you set *Other queue settings* to Yes, you see different sub-properties in this group, depending upon the type of link that you use with this connector and the link that you have selected in the navigator.

For input links, these properties are for the output queue or namelist to which you are sending messages.

For output links, these properties are for the input queue from which you are receiving messages.

For requests, these properties are for the request queue to which you are sending request messages.

The default value is No.

**Password**

Use this property to specify the password of the user specified by the value typed in the *Username* property. The connector impersonates this user on the local computer before connecting to the queue manager (Windows only).

Valid values are up to 256 characters.

**Payload size**

For output links only, use this property to specify acceptable payload sizes and ranges of payload sizes for source messages.

The message payload is the portion of the message that follows and that does not include the message header. This value specifies the size of that payload. If the column is a text message payload column, the payload size is measured in characters. If the column is a binary message payload column, the payload size is measured in bytes.

If the format header data element (WSMQ.FORMATHEADERS) is present in the schema and format headers are present in the message, the format headers are not counted as part of the payload size. Otherwise, the format headers are included in the payload size total.

For output links, you can type the custom value in different ways:
• A list of comma-delimited or space-delimited integers
• A range of integers in the following format: a-b
• A combination of lists and ranges that are comma-delimited or space-delimited. The total length of the combined values in this property must not exceed 256 characters. There is no default value. If this value is blank, this property is not used for filtering messages.

**Period**

Use this property to specify the number of seconds to read messages from the input queue before repositioning the queue cursor at the beginning of that queue.

This property is available only if you set Refresh to Yes.

The default value is -1, which specifies that the cursor is repositioned when the end of the queue is reached.

If you want to more closely monitor high-priority messages, specify 0 for this value. This value specifies that the cursor is repositioned each time that a new message is read.

You can specify integers between -1 and 999999999.

**Persistence**

Use this property to specify whether the message survives after the queue manager is restarted. Restarts might occur because of a system failure or because the queue manager has been manually restarted.

Persistent and non-persistent messages can reside on the same queue.

For input and request links, select one value from the list. This value is then set in the message header. The default value is As in queue definition.

For output links, select one or more values by clicking within the property and then select the appropriate check boxes. When you finish, click outside of the property. There is no default value. If this value is blank for an output link, this property is not used for filtering messages.

You can use different ways to specify this value for input links and output links:
- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.PERSISTENCE.
Persistence options

Use this property to specify the persistence registration value for the subscriber.

This property is available only if you set Registration to Yes.

The following values are available:

Non persistent
Publications that are sent from a broker to the subscriber are sent as a non-persistent message, regardless of the persistence setting in the publication message that is received by the broker.

Persistent
Publications that are sent from a broker to the subscriber are sent as a persistent message, regardless of the persistence setting in the publication message that is received by the broker.

Persistent as publish
Publications that are sent from a broker to the subscriber are sent with the persistence of the original publication.

Persistent as queue
Publications that are sent from a broker to the subscriber are sent with the persistence specified on the subscriber queue. This persistence is derived from the DEFPSIST setting of the subscriber queue definition that is local to the broker.

The default value is Persistent as publish.

Physical format

Use this property to specify the name of the Message Repository Manager (MRM) physical format in the specified message set that is used for the publication messages.

This property is available only if you set Message service domain to mrm or idoc.

This property represents the value of the <Fmt> element in the <mcd> message content descriptor service folder. You must define the message set for the message broker.

Valid values are up to 128 characters. There is no default value.

Priority

Use this property to specify the importance of the message in comparison with other messages on the queue.

For input and request links, specify an integer between -1 and 999999999, where a value of -1 specifies that the default value for the queue is used. The default value is -1.

For output links, you can type the custom value in different ways:
- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: a-b
- A combination of lists and ranges that are comma-delimited or space-delimited
The total length of the combined values in this property in an output link must not exceed 256 characters. There is no default value. If this value is blank, this property is not used for filtering messages.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.PRIORITY.

Publication format

Use the properties in this group to specify the format of the publication payload.

For target messages (input links and request links), you can specify only one value: system or custom. This value is then set in the MQRFH or MQRFH2 format header of the publication message, depending upon the format header that you use.

Publication options

Use this property to specify the publication options when you publish messages.

These options refer specifically to the publication messages rather than to the publisher.

Select one or more values in this list by clicking the value and then select the check boxes. When you finish, click outside of this property. There is no default value.

The following values are available:

Correlation ID as identity

The correlation ID is part of the publisher identity.

Retain publication

The broker keeps a copy of the publication. If this value is not selected, the publication is deleted as soon as the broker sends the publication to all of its current subscribers.

No registration

If the publisher is not already registered with the broker as a publisher for this stream and topic, the broker cannot perform an implicit registration. If the publisher is already registered, this value has no effect on this publication.

If you set Service mode to MQRFH2, this value is ignored.
Publication settings

Use the properties in this group to specify the options to apply when you prepare publication messages.

This property is available only if you set Publish/Subscribe to Yes.

Because the connector is acting as a publisher, this functionality is available only for input links.

Publish/Subscribe

Use the properties in this group to specify whether the connector is used in the Publish/Subscribe mode of operation.

The default value is No.

When you set Publish/Subscribe to Yes, the connector runs in Publish/Subscribe mode. For input links, the connector acts as a publisher. For output links, the connector acts as a subscriber. You cannot use Publish/Subscribe mode with request and reply mode.

Note: You must satisfy certain software requirements to be able to use this functionality.

Publish/subscribe prerequisites

To use publish/subscribe with the WebSphere MQ connector, you must install certain applications, SupportPacs, or fix packs, depending upon whether you want to use the MQRFH or MQRFH2 message format.

When you install the SupportPac, you can define a queue manager as a publish/subscribe broker. When the broker is started, the necessary publish/subscribe infrastructure is established on the queue manager. The infrastructure includes various system queues for the internal management of subscription and publication messages by the broker.

MQRFH message format requirements

You must install one of the following version combinations:

- IBM WebSphere MQ, version 5.3 with the MA0C SupportPac with a fix pack less than 8 or with no fix pack
- IBM WebSphere MQ, version 5.3 without the MA0C SupportPac, but with fix pack 8 or later
- IBM WebSphere MQ, version 6.0

MQRFH2 message format requirements

You must use the IBM WebSphere Message Broker 6.0.

Put application name

Use this property to specify the name of the application for target messages (for input links) or source messages (for output links).

The format of the application name is dependent upon the type of application that is specified in the Put application type property.
For input links and request links, this property is available only if you set **Context mode** to **Set all**.

For input and request links, valid values are up to 28 characters.

For output links, you can type multiple values separated by spaces or commas up to a total length of 256 characters. This property provides filtering on multiple values.

There is no default value. If this value is blank for an input link, the default value for the target message is used. The queue manager constructs the name based upon the name of the application process that sends the message. If this value is blank for an output link, this property is not used for filtering messages.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the **Data element** value on the **Columns** tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the **Data element** column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is **WSMQ.PUTAPPLNAME**.

### Put application type

Use the properties in this group to specify the put application values for target messages (for input links) or source messages (for output links).

For input links, this property is available only if you set **Context mode** to **Set all**.

For input links, only one value either in the **System value** property or in the **Custom value** property can be specified. This value is then set in the message header.

For output links, you can specify multiple values for filtering. These values can include any combination of values in the **System value** property or **Custom value** property.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the **Data element** value on the **Columns** tab.
The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.PUTAPPLTYPE.

**Put date**

Use this property to specify the put date for target messages (for input links) or source messages (for output links).

Use the YYYYMMDD format and Greenwich Mean Time (GMT) as the time zone reference.

For input links only, this property is available only if you set Context mode to Set all.

**Valid values**

For input links, you can type one eight-character date value by using the YYYYMMDD format. There is no default value. If this value is blank for an input link, the default value for the target message is used, which is the current date.

For output links, you can type any combination of specific dates and ranges of dates in different ways:

- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: YYYYMMDD-YYYYMMDD
- A combination of lists and ranges that are comma-delimited or space-delimited

The total length of the combined values in this property must be between 8 and 256 characters. Each date must follow the eight-character YYYYMMDD format. There is no default value. If this value is blank for an output link, this property is not used for filtering messages.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.PUTDATE.
Put time

Use this property to specify the put time for target messages (for input links) or source messages (for output links).

Use the HHMMSSSTH format and Greenwich Mean Time (GMT) as the time zone reference.

For input links only, this property is available only if you set Context mode to Set all.

Valid values

For input links, you can type one eight-character time value by using the HHMMSSSTH format. There is no default value. If this value is blank, the default value for the target message is used, which is the current date.

For output links, you can type any combination of specific times and ranges of times in different ways:

- A list of comma-delimited or space-delimited integers
- A range of integers in the following format: HHMMSSSTH-HHMMSSTH
- A combination of lists and ranges that are comma-delimited or space-delimited

The total length of the combined values in this property must be between 8 and 256 characters. Each time value must follow the eight-character HHMMSSSTH format.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.PUTTIME.

Queue manager for Cluster queue

Use this property to select the cluster queue manager name to which the message is sent.

If you do not select a name and you set Cluster queue to Yes, the queue manager is dynamically selected from the cluster.

There is no default value. If this value is blank for an input link, the default value for the target message is used. Valid values are up to 48 characters.
Queue manager for Connection

Use this property to select the name of the queue manager to access for the connection.

If you set Mode to Server and you leave this value blank, the default queue manager is used (if it exists).

If you set Mode to Client and you want to type a value rather than select it, the value must exactly match the queue manager name for which the client channel is defined.

Instead of using this property to define the queue manager when the connector is in client mode, you can define the value in the following ways:

- Define the connection to the queue manager by using the Client channel definition properties.
- You can use the MQSERVER environment variable or the combination of the MQCHLLIB and MQCHLTAB environment variables to specify the client connection definition.

Valid values are up to 48 characters.

Queue manager for Error queue

Use this property to select the name of the queue manager that hosts the error queue.

This property is available only if you set Mode to Client.

If you do not specify a name in this property, the connector assumes that the error queue resides on the currently connected queue manager. There is no default value.

Valid values are up to 48 characters.

Queue name

Use this property to select the name of the queue from which you want to receive messages (for output links), to which you want to send messages (for input links), or to which you want to send request messages (for request links).

In Publish/Subscribe mode, this value is the subscriber queue name.

Note: You can view a list of queue names only if the queue manager is running the Command Server service. Otherwise, you must type the value.

Input links

For input links, this name is for the destination queue or queue namelist to which you want to send a message. For namelists, a distribution list is generated at run time that contains the queue names from the list. In Publish/Subscribe mode, this value is used as the stream queue.

If you set Dynamic queue to Yes, this name is for the model queue to use as the template for the creation of the dynamic queue.
Output links

For output links, this name is for the source queue from which you want to receive messages. In Publish/Subscribe mode, the value is used as the subscriber queue.

Request links

For request links, this name is for the request queue to which request messages are sent. The name of the reply queue is included in the request message.

If you set Dynamic queue to Yes, this property value represents the name of the model queue to use as a template for the creation of the dynamic queue.

Valid values

There is no default value. If this value is blank for an input link, the queue name must be provided in the input data by using the WSMQ.QUEUENAME data element column. Otherwise, the job fails.

Valid values are up to 48 characters.

You can use different ways to specify this value for input links and output links:

• Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

• For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.QUEUENAME.

Queue name for Dynamic queue

Use this property to specify the name or portion of the name of the dynamic queue.

Input links

For input links, this name is for the destination dynamic queue. The Queue name property specifies the name of the model queue that acts as a template to create the dynamic queue. If the queue specified in the Queue name property is a regular queue as opposed to a model queue, the regular queue is opened, and the name specified in the Queue name sub-property for the Dynamic queue property is ignored.

Request links

For request messages, this name is for the request dynamic queue and the reply dynamic queue. Whether the request queue is opened as a dynamic queue
depends upon the name in the **Queue name** property. If this name points to a model queue, the request queue is opened as a dynamic queue. If this name points to a regular queue, the regular queue is opened.

Whether the reply queue is opened as a dynamic queue depends upon the name in the **Reply to queue** property. If this name points to a model queue, the reply queue is opened as a dynamic queue. If this name points to a regular queue, the regular queue is used for the reply queue.

**Valid values**

The asterisk (*) is the default value, and it is processed as a wildcard in this property. If you want to use the wildcard, adhere to the following restrictions:

- Only one asterisk (*) can be used in the value.
- The asterisk must be used at the end of your name. This position must be no later than the thirty-third character of the 48-character length limit of the name.

There are several different ways to define the name:

- Allow the queue manager to generate the entire name.
  Leave the default asterisk (*) as the value. The queue manager creates the dynamic reply queue name by replacing the asterisk with a 48-character queue name.
- Use the asterisk (*) wildcard with your stem and let the queue manager generate the rest of the name.
  Provide the stem and use the asterisk is the last character. Make sure that you adhere to the asterisk rules previously mentioned.
- Type your own name (up to 48 characters).
  The queue manager uses this name to open the dynamic queue.

Valid values are up to 48 characters.

**Queue name for Dynamic reply queue**

Use this property to specify the name of the dynamic reply queue.

This property is available only if you set **Dynamic reply queue** to Yes.

The asterisk (*) is the default value and is treated as a wildcard with the following restrictions:

- Only one asterisk (*) can be used in the name.
- If you use the asterisk, it must appear as the last character of your name, and you can use a maximum of 33 out of the 48 total available characters.

There are several different ways to define the name:

- Allow the queue manager to generate the name.
  Leave the default asterisk (*) as the value. The queue manager creates the dynamic reply queue name by replacing the asterisk with a 48-character queue name.
- Use the asterisk (*) wildcard as part of your name and let the queue manager generate the reminder of the name.
  Make sure that the asterisk is the last character in the part of the value that you provide and that you follow the asterisk rules previously mentioned.
- Type your own name (up to 48 characters).
The queue manager uses this name to open the dynamic queue. Use this way to define the name for output links.

There is no default value. If the value is blank for an input link, the default value for the output message is used. If this value is blank for an output link, this property is not used for filtering messages.

Valid values are up to 48 characters.

**Queue name for Error queue**

Use this property to select or type the name of the error queue.

For input links, this property specifies the name of the queue that serves as a backup for the target queue.

For output links, this property specifies the name of the queue that stores messages that were read from the source queue during a transaction if the transaction fails. If the transaction fails and is rolled back, these messages are moved from the source queue to the source error queue.

You can use error queues on different queue managers by using the local definitions of remote queues. If a remote queue is named, the queue manager moves messages to the transmission queue for the remote error queue. You must specify the remote queue name in the *Transmission queue* property.

**Note**: Namelists are not used. If the queue manager cannot open the error queue, the connector does not attempt to open the namelist.

There is no default value. If you set *Error queue* to *Yes*, you must specify a value for this property. Valid values are up to 48 characters.

You can provide a value for this property in either of the following ways:
- If you have a connection to the queue manager, click *Select*. Select the name you want from the list of existing queues on that queue manager.
- If you want to use a remote queue manager or for any other reason, type the name of the error queue.

**Record ordering**

Use this property to specify how to process records across multiple links.

Specify how to process records from multiple links. Choose one of the following:
- **All records** - All records from the first link are processed; then all records from the second link are processed; and so on.
- **First record** - One record from each link is processed until all records from all links have been processed.
- **Ordered** - Records are selected from the input links based on the order that you specify by using the *Key column*, *Sort order*, and *Null order* fields.

**Refresh**

Use this property to specify whether the queue cursor is repositioned to the beginning of the input queue.
The default value is No. If you select Yes, use the **Period** property to specify the timeframe within which this repositioning occurs.

**Registration**

Use this property to specify whether the publisher or subscriber registers with the broker after the job starts and before messages are either produced or consumed.

This property is available only if you set **Publish/Subscribe** to Yes. Additionally, for input links only, the **Service type** property must be set to MQRFH.

The default value is No.

Use the sub-properties in this group to specify additional registration information.

**Registration options**

Use this property to select the registration options for the publication of messages.

If you set **Service mode** to MQRFH2, the **Anonymous** value is ignored, if selected. In MQRFH2 service mode, any of the other **Registration options** values that you select are used as publication options.

You can select multiple values by clicking the property and then click the check boxes. When you finish, click outside of the property. There is no default value.

The following values are available:

- **Correlation ID as identity**
  - The correlation ID is used as part of the publisher identity.

- **Anonymous**
  - This value specifies to the broker that the identity of the publisher is not to be revealed except to those subscribers with more authority.

- **Local**
  - This value specifies to the broker that publications with this value are sent to the subscriber.

**Remove MQRFH2 header**

Use this property to specify whether MQRFH2 header needs to be removed from the source message.

The default value is No. Both the MQRFH2 header and the message body are passed to the next stage.

If you set Yes for this property, only the message body is passed to the next stage.

**Reply queue**

Use this property to select the name of the reply queue that appears in the registration and deregistration command messages that the connector sends as part of the job.

This property is available only if you set **Publish/Subscribe** to Yes. For input links only, you must also set **Service type** to MQRFH2. For output links, either service type is available.
The connector specifies the reply queue in the registration and deregistration command messages that are sent as part of the job. The broker sends a response message to the queue to inform the connector if the broker successfully processed the command message. The connector reads the reply message. If there is an error, the connector logs that error and stops the processing of the job.

If you specify the Dynamic reply queue property, that name is used as the name of the model queue.

Valid values are up to 48 characters. There is no default value.

**Reply queue close options**

For request links only, use this property to specify how the dynamic reply queue is closed.

This property is available only if you set Dynamic queue to Yes.

The default value is None.

The following values are available:

- **None**  
  No close options are specified.

- **Delete**  
  Select this value only if the dynamic queue does not contain any messages upon closing. If there are messages on the queue, an error message is generated, and the queue is not deleted.

  The dynamic queue can be deleted as part of the closing process.

- **Purge and delete**  
  The remaining messages on the queue are purged before deletion. If you are not sure whether there will be messages at closing time and you do not need to preserve those messages, select this value rather than the Delete value.

**Reply to queue**

Specify the name of the message queue upon which reply messages or report messages are returned for the messages that are sent by the connector.

This local name of a queue is defined on the queue manager that is specified in the Reply to queue manager property. In request and reply mode with a dynamic reply queue, the local name specifies the name of the model queue from which to build the dynamic reply queue.

This property requires additional configuration. You must select or type a queue name for this property in the following conditions:

- If the Message type property is Request
- If the Report property specifies that report messages are requested

There is no default value. If this value is blank for an input link, the default value for the target message is used. If this value is blank for an output link, this property is not used for filtering messages.

For input and request links, valid names are up to 48 characters.
For output links, you can type multiple names, separated by spaces or commas, up to 512 characters. This property provides filtering on multiple names separated by spaces or commas.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.REPLYTOQ.

**Reply to queue manager**

Use this property to specify the name of the reply queue manager upon which the reply queue resides.

There is no default value. If this value is blank for an input link, the default value for the target message is used. If this value is blank for an output link, this property is not used for filtering messages.

For input links, valid names are up to 48 characters. This property is not available for request links. The reply queue must reside on the queue manager to which the connector is connected.

For output links, you can select or type one name or type multiple names that are separated by spaces or commas up to a total of 512 characters. This property is used for filtering on multiple names.

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.
The data element for this property is WSMQ.REPLYTOQMGR.

**Report**

For input and request links, use this property to specify the report information that the connector sends in the message header. For output links, use the **Value** sub-property to specify the value used to filter report messages.

For input links, this property specifies the value that is set in the message header. You can select multiple values from this list by clicking the property and then click one or more check boxes. When you finish, click outside of this property.

For request links, the **Report** value defines how the message identifier and the correlation identifier of the reply message correspond to the message identifier and the correlation identifier of the request message. By default, the correlation identifier of the reply message must match the message identifier of the request message.

There is no default value. If this value is blank for an input link, the default value for the target message is used.

You can use different ways to specify this value for input links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.
- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the **Data element** value on the **Columns** tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

The data element for this property is WSMQ.REPORT.

**Row buffer count**

Use this property to specify the number of rows (records) that create one composite message to be sent to the target queue.

The message payload from each row is concatenated into the composite message. The final composite message uses the message header and format headers that are established for the first row in the buffer.

If the end of the input data is reached and there are fewer rows that are buffered than the specified value (except for a value of -1), these rows are sent together as the last composite message.

**Note:** If you want to specify the **Record count** property for the end of wave processing, that number must be a multiple of the **Row buffer count** value.

You can select a single number. The default value is 1.
Valid values are integers between -1 and 999999999. If you set Row buffer count to -1, the connector buffers all available rows and sends them as a single message at the end of the job. If you set Row buffer count to 0, the connector sends only the last message at the end of the job.

**Segment size**

Use this property to specify the size of each segment in bytes.

This property is available only if you set Segmented message to Yes.

The size of the last segment of the message might be smaller than the specified size.

The default value is 1024. You can specify integers between 1 and 999999999.

**Segmented message**

Use this property to specify whether target messages are separated into segments that are then sent to the target queue as opposed to sending a single message.

If an error occurs when some of the segments are sent, the entire source message is sent to the error queue if one is defined or to the reject link if one is defined. The entire message is sent as opposed to individual segments. A segment that is successfully stored on the target queue will not be removed from the target queue.

Request messages are always sent as single, physical messages. Therefore, this property is not available for the request and reply scenario.

The default value is No. If you set Segmented message to Yes, you must specify the size of the segment in the Segment size property.

**Service type**

Use this property to specify the message format of command messages that the connector sends to the broker.

Command messages are registration and deregistration messages for publishers and subscribers, as well as publication messages for publishers. This property determines the WebSphere MQ format of these command messages. This property also defines the format of the response messages for the command messages. Response messages are sent by the broker to the connector about the outcome of the command messages that are sent by the connector.

Select the value that is most compatible with your broker. For the IBM WebSphere MQ Publish/Subscribe broker, you must select MQRFH as the Service type value. For the IBM WebSphere Message Broker, either of the service types can be used. However, select MQRFH2 because MQRFH is supported only for compatibility with existing WebSphere MQ Publish/Subscribe applications.

The default value is MQRFH.

**Set header fields**

For input links and request links only, use the properties in this group to specify whether message header data is overwritten by the values in this group.

The default value is No.
Sort order
Use this property to specify whether to sort values in ascending or descending order.

If the connector uses multiple input links and you choose Ordered in the Record ordering field, use the Sort order field to specify whether to sort values in ascending or descending order.

Start value
Use this property to specify the initial message sequence number for the first published message.

This property is available only if you set Message sequence number in the Publication settings property group to Yes.

This sequence number is increased by a count of one for subsequently published messages. The default value is 1. You can specify an integer between 1 and 999999999.

Stream name
Use this property to specify the name of the stream for publications.

This property is available only when you set Service type to MQRFH.

Each stream represents a set of queues. There is one name for each broker that supports this stream. Specify the stream name when you register or deregister the subscriber.

Unless you select Yes for either the Registration property or for the Deregistration property, this value is ignored.

The default value that encompasses all brokers in a network is the following text:

SYSTEM.BROKER.DEFAULT.STREAM

Valid names are up to 48 characters.

Subscription identity
Use this property to specify the subscriber application identity for the subscription that is registered, deregistered, or both registered and deregistered.

In this scenario, the connector is the subscriber application that acts as a subscriber in the job. The broker manages a set of identities for each subscription. The broker uses the identities to manage subscriber requests according to their subscription attributes and duration.

If you set either Registration or Deregistration to No, the subscription identity is ignored.

Value identities are up to 128 characters. There is no default value.
Subscription name

Use this property to define the name of the subscription for registration, deregistration, or both, depending upon the publish/subscribe values.

The subscription name takes precedence over the traditional subscription identity, which is a combination of the Queue name property, the Queue manager name property, and the Correlation ID property for Registration. Unless you specified Variable user ID for Identity options in a previous command message, the user ID of subsequent register and deregister command messages for this subscription must match.

If you set either Registration or Deregistration to No, the subscription name is ignored.

Valid names are up to 64 characters. There is no default value.

Subscription point

Use this property to specify the subscription point when you register or deregister the subscriber.

The subscription point is the point of the publication node in the message flow from which publications are forwarded by the broker to this subscriber.

This property is available only if you set Service type to MQRFH2.

If you set both Registration and Deregistration to No, the subscription point is ignored.

Valid subscription points are up to 64 characters. There is no default value.

System value for Feedback

For input links, select one feedback code or reason code (MQRC) for target messages. For output links, select one or more feedback codes or reason codes (MQRC) for source messages.

For input links, you can provide only one feedback or reason code by using this property or the Custom Value property. To specify a custom value, leave this property blank. Otherwise, select one code from the list. There is no default value for input links. If this value is blank for an input link, the default value for the target message is used.

For output links, you can select multiple codes from the list, and you can also add a code from the Custom value property. Select multiple codes by clicking the property and then select the check boxes. When you finish, click outside of this property. The default value for output links is None. If this value is blank for an output link, this property is not used for filtering messages.

System value for Format

For input links, select one format for target messages. For output links, select one or more formats for source messages.
For input links, you can provide one value for Format by using this property or the Custom Value property. To specify a custom value, leave this property blank. Otherwise, select one format from the list. The default value for input links is MQSTR.

For output links, you can select multiple values from the list, and you can also add a format from the Custom value property. Select multiple formats by clicking the property and then select the check boxes. When you finish, click outside of this property. There is no default value for output links. If this value is blank for an output link, this property is not used for filtering messages.

**System value for Message type**

For input links, select one message type for target messages. For output links, select one or more message types for source messages.

For input links, you can provide one message type for Message type by using this property or the Custom Value property. To specify a custom value, leave this property blank. Otherwise, select one message type from the list. The default value for input links is Datagram.

For output links, you can select multiple message types from the list, and you can also add a message type from the Custom value property. Select multiple message types by clicking the property and then select the check boxes. When you finish, click outside of this property. There is no default value for output links. If this value is blank for an output link, this property is not used for filtering messages.

For request links, you cannot select a value. It is always Request.

For input and output links, the following values are available:

**Datagram**

This message does not require a reply message. Datagram messages are used in asynchronous contexts.

**Request**

This message does require a reply message. If you select this value, you must define the name of the queue to which the reply message must be sent in the Reply to queue property. Use the Report property to specify how the Message ID and Correlation ID properties in the request message correspond to the same properties in the reply message.

Two queues are used. One is the queue to which the request message is sent and the other queue is the one from which the reply message is read. Request messages are used in synchronous contexts.

**Reply**

This message is the reply to an earlier request message. The message is sent to the queue specified in the Reply to queue property of the request message. Use the Report property to specify how the Message ID and Correlation ID properties in the reply message correspond to the same properties in the request message.

**Report**

This message is a report message, which is a message about another message. Use the report message to inform an application about expected or unexpected events that relate to the original message. When sending a message to a queue, a WebSphere MQ application can request that the queue manager generates a report message when certain events occur that are related to the message that was sent. You can define the events that
generate the report from the queue manager in the Report property. The queue manager then sends the report to the queue that was specified in the Reply to queue property and in the Reply to queue manager property in the header of the message that was originally sent.

**System value for Publication format**
For input links, select one publication payload format for target messages.

For input links, you can provide one value for Publication format by using this property or the Custom Value property. To specify a custom value, leave this property blank. Otherwise, select one publication format from the list. The default value for input links is MQSTR.

**System value for Put application type**
For input links, select one put application type for target messages. For output links, select one or more put application types for source messages.

For input links, you can provide one put application type for Put application type by using this property or the Custom Value property. To specify a custom value, leave this property blank. Otherwise, select one put application type from the list. The default value is NO CONTEXT.

For output links, you can select multiple put application types from the list, and you can also add a put application type from the Custom value property. Select multiple put application types by clicking the property and then select the check boxes. When you finish, click outside of this property. There is no default value for output links. If this value is blank for an output link, this property is not used for filtering messages.

**Timeout (Blocking transaction processing)**
Use this property to specify the time, in seconds, that the connector will wait for a job to process the blocking transaction message before logging error messages and stopping the job.

The default value is -1, meaning unlimited time. The allowed values are -1 to 999,999,999.

Related reference:

"Method name (Blocking transaction processing)" on page 98
Use this property to specify the external (user-defined) method that the connector calls for each input message to determine if that message represents a blocking transaction message.

"Module name (Blocking transaction processing)" on page 101
Use this property to specify the name of the shared (dynamic) library which implements the method that the connector calls after each message to determine whether the message represents a blocking transaction message.

**Timestamp**
Use this property to specify whether timestamp information is included in published messages.

The default value is No.
**Topic for Publication settings**

Use this property to specify the name of the topic upon which the publications are sent.

The topic name is applied to all publications that are sent during the processing of the job. To send publications on different topics as part of the same job process, define a WSMQ.TOPIC data element column in the input link schema. The publication topic can then be specified as part of the data at the individual publication message level.

Valid topic names are up to 256 characters. There is no default value.

**Topic for Registration and Deregistration**

Use this property to specify the one or more topics for which to register or deregister the publisher or subscriber.

Your selection of publisher or subscriber depends upon the current usage context and whether this topic is for the **Registration** property or for the **Deregistration** property.

To register or deregister multiple topics at the same time, you can type one topic value, then type a comma or a space character as a separator, and then type another topic value. You can repeat these steps up to a total of 512 characters.

There is no default value.

If any topic name in a group of topics that you want to register or deregister contains any of the following characters, you must type a backslash character before the special character:

- Space
- Comma
- Backslash

Refer to the following example.

**Example of multiple topics where one topic contains a space**

You have two different topics that you want to register or deregister: Sport Results and Weather. You must type the following value for this property:

```
Sport\ Results,Weather
```

**Transmission queue for Error queue**

Use this property to specify the name of the transmission queue when you want to use a remote queue for the error queue.

The transmission queue must always reside on the connected queue manager.

You can type the name or press **Select** to select the queue from a list of queues. There is no default value. If this value is blank for an input link, the default transmission queue is used.

Valid values are up to 48 characters.
**Transmission queue for Other queue settings**

Specify the name of the transmission queue if the destination queue is a remote queue.

The transmission queue must always reside on the connected queue manager.

Use this property to override transmission queue settings at run time when the local definition of the remote queue is opened.

There is no default value.

Valid names are up to 48 characters.

**Transport type**

Select the transport protocol for this client connection.

Make sure that the protocol is supported on the running platform.

The default value is Local.

As a part of the **Client channel definition** group of properties, this value contributes to the replacement of any values in the MQSERVER, MQCHLLIB, or MQCHLTAB environment variables.

**Treat EOL as row terminator**

Use this property to specify whether the end-of-line character in the message payload is processed as the end of a row.

If you set **Treat EOL as row terminator** to Yes, the following statements are true:

- One output message can result in several rows of data.
- On each platform, the end-of-line terminators for that platform determine the end of the row.

The default value is No.

**Use wildcard**

Use this property to determine whether an asterisk (*) in the property value is interpreted as a wildcard character that represents a contiguous group of characters.

For output links only, this property is a sub-property for several properties when you set Hex to No.

The default value is No.

**User ID**

Use this property to specify the user identifier for the application that receives the source messages or that sends the target messages.

For input links, this property is available only if you set **Context mode** to Set identity or Set all.
This property can override the default value on target messages. Thus, the
property provides an alternate user ID.

There is no default value. If this value is blank for an input link, the default value
for the target message is used. If this value is blank for an output link, this
property is not used for filtering messages.

For input and request links, valid values are up to 12 characters.

For output links, you can type multiple values separated by commas or spaces, up
to a total of 256 characters. This property provides filtering on multiple values.

**Username**

Use this property to specify the user name of the user that the connector
impersonates on the local computer before the connector connects to the queue
manager (Windows only).

Use the following format to specify the domain or server name:

```
username@DNS_domain_name
```

The **Username** property is different from other user ID properties, such as the **User
ID** property that specifies the message originator or the **Alternate user ID**
property that specifies the ID that is used to open the queue.

In client mode, the connector does not include the values specified by the **Username**
and **Password** properties in the client channel definition.

Valid values are up to 256 characters. There is no default value.

**Value for Accounting token**

Use this property to specify the accounting token in source messages (for output
links) or target messages (for input links and request links).

There is no default value. If this value is blank for an input link, the default value
for the target message is used. If this value is blank for an output link, this
property is not used for filtering messages.

Valid values are up to 32 characters (or 64 characters if pairs of hexadecimal digits
are used and you have set **Hex** to **Yes**).

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets
  the value at design time for all messages in this job. If, for some reason, you also
  specify the data element as a column in the schema (see below), the data
element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this
  property, you specify a column in the schema for this value by selecting the
  corresponding value (see below) as the **Data element** value on the **Columns** tab.
The value that is stored in each incoming message is then used during runtime
processing when it is put on the queue. The value in the input link schema
overrides the design-time value of this property at runtime.
For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.ACCOUNTINGTOKEN.

For output links only, you can use an asterisk (*) as a wildcard character to represent contiguous characters if you set Use wildcard to Yes.

You can use only one wildcard character per property value. If you use more than one wildcard character, only the first usage is interpreted as a wildcard. All subsequent usages are interpreted as actual values, not as wildcard values.

**Value for Alternate security ID**

Use this property to specify the alternate security identifier to open a queue.

There is no default value. If this value is blank for an input link, the queue manager does not perform authorization based upon this value.

Specify a 40-byte value by typing an array of pairs of hexadecimal digits. To do this, you must set Hex to Yes. You must use the IBM WebSphere MQ format for this value.

Valid values are up to 40 characters (or 80 characters if arrays of pairs of hexadecimal digits are used and you set Hex to Yes).

**Value for Correlation ID**

Use this property to specify the correlation identifier in source messages (for output links) or target messages (for input links and request links).

There is no default value. If this value is blank for an input link, the default value for the target message is used. If this value is blank for an output link, this property is not used for filtering messages.

Valid values are up to 24 characters (or 48 characters if pairs of hexadecimal digits are used and you set Hex to Yes).

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.
The data element for this property is WSMQ.CORRELID.

For output links only, you can use an asterisk (*) as a wildcard character to represent contiguous characters if you set Use wildcard to Yes.

You can use only one wildcard character per property value. If you use more than one wildcard character, only the first usage is interpreted as a wildcard. All subsequent usages are interpreted as actual values, not as wildcard values.

**Value for Group ID**

Use this property to specify the group identifier in source messages (for output and request links) or target messages (for input links).

This property is available only if you set Header version to 2.

There is no default value. If this value is blank for an input link, the default value for the target message is used. If this value is blank for an output link, this property is not used for filtering messages.

Valid values are up to 24 characters (or 48 characters if pairs of hexadecimal digits are used and you set Hex to Yes).

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the Data element value on the Columns tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

For output links, you can view only the value in the Data element column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.GROUPID.

For output links only, you can use an asterisk (*) as a wildcard character to represent contiguous characters if you have set Use wildcard to Yes.

You can use only one wildcard character per property value. If you use more than one wildcard character, only the first usage is interpreted as a wildcard. All subsequent usages are interpreted as actual values, not as wildcard values.

**Value for Message flags**

For output links (source messages) only, use this property to specify flags that are associated with the message.
You can select one or more values from this list. If you select multiple values and you set **Match all** to *No*, any one of these message flag values is accepted. To specify that all of the values must match to be accepted, you must set **Match all** to *Yes*.

There is no default value.

For output links, you can view only the value in the **Data element** column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.MSGFLAGS.

**Value for Message ID**

Use this property to specify the message identifier in source messages (for output and request links) or target messages (for input links).

There is no default value. If this value is blank for an input link, the queue manager assigns a unique value to the message. If this value is blank for an output link, this property is not used for filtering messages.

Valid values are up to 24 characters (or 48 characters if pairs of hexadecimal digits are used and you set *Hex* to *Yes*).

You can use different ways to specify this value for input links and output links:

- Specify the value at the job level. You type the value in this property, which sets the value at design time for all messages in this job. If, for some reason, you also specify the data element as a column in the schema (see below), the data element takes precedence over the property.

- For input links, you can specify the value at the message level. Instead of this property, you specify a column in the schema for this value by selecting the corresponding value (see below) as the **Data element** value on the **Columns** tab. The value that is stored in each incoming message is then used during runtime processing when it is put on the queue. The value in the input link schema overrides the design-time value of this property at runtime.

  For output links, you can view only the value in the **Data element** column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

  The data element for this property is WSMQ.MSGID.

  For output links only, you can use an asterisk (*) as a wildcard character to represent contiguous characters if you set **Use wildcard** to *Yes*.

  You can use only one wildcard character per property value. If you use more than one wildcard character, only the first usage is interpreted as a wildcard. All subsequent usages are interpreted as actual values, not as wildcard values.

**Value for Report**

For output links only, select the report flag values that are used to filter source messages. Only messages with the specified report message header field values are accepted by the connector.
When certain events occur, as defined by the application, the queue manager sends a report to the queue. This queue and queue manager are specified in the **Reply to queue** property and in the **Reply to queue manager** property.

Select multiple values by clicking the property and then select the check boxes. When you finish, click outside of this property. If you select multiple values and you set **Match all** to No, any one of these report values is accepted. To specify that all of the values must match to be accepted, you must set **Match all** to Yes.

The value in this property is used to filter all messages in this job at design time. There is no default value. If this value is blank for an output link, this property is not used for filtering messages.

For output links, you can view only the value in the **Data element** column that was retrieved when the message was read from the queue. Generally speaking, you use the data element on an output link when you plan to use this value later on in your job processing.

The data element for this property is WSMQ.REPORT.

**Wait time**

Use this property to specify the maximum number of seconds to wait for a new message to arrive upon the input queue.

The default value is -1, which specifies an indefinite amount of time.

Select or type an integer between -1 and 999999999.

**Work queue**

Use the properties in this group to define a work queue.

The connector moves messages from the source queue to the specified work queue prior to releasing the message data on the output link. When the connector stage is configured to use a work queue and the job runs, the connector first reads and processes any messages that are present on the work queue. Then it resumes reading and processing messages from the source queue.

This property is available only if you set **Message read mode** to **Move to work queue**. This value specifies that the message is moved from the source queue to the work queue in a local MQ transaction.

Note that reading the messages from the work queue when the job starts is performed in browse mode and that the messages are left on the work queue after they are fetched. Other mechanisms must be used to remove the messages from the work queue. Typically, this is done by the Distributed Transaction stage that is running in the same job and is removing messages from the work queue and updating target relational databases within global (distributed) transactions.
Related reference:

“Name” on page 102
Use this property to specify the work queue that messages will be moved to.

“Append node number” on page 71
Use this property to append the node number to the work queue name.

“Context mode for work queue” on page 78
Use this property to specify the context mode when the work queue is opened.

“Monitor queue depth” on page 101
Use this property to control the work queue depth during job execution.

“Message read mode” on page 94
Use this property to specify how messages are read in the current transaction.
Chapter 5. IBM WebSphere MQ stage

IBM WebSphere MQ stage is a passive stage that offers a message-based solution to customers where messaging represents another form of source and target data. The WebSphere MQ stage lets the InfoSphere DataStage and QualityStage Designer read from and write to WebSphere MQ message queues.

When you use IBM InfoSphere DataStage to access WebSphere MQ data, you can choose from a collection of connectivity options. For most new jobs, use the WebSphere Connector stage, which offers better functionality and performance than the WebSphere MQ stage.

The WebSphere MQ stage was deprecated and removed from the palette. For the WebSphere MQ stage to be available on the palette, you must choose to install the stage when you install IBM InfoSphere Information Server.

If you have jobs that use the WebSphere MQ stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

You can use the WebSphere MQ stage in the following ways:
- An intermediary between applications, transforming messages as they are sent between programs
- A conduit for the transmission of legacy data to a message queue
- A message queue reader for transmission to a non-messaging target

You can use the WebSphere MQ stage as a source or a target in any InfoSphere DataStage data flow diagram. It handles data in standard row and column format. As a message writer, the stage writes only datagram messages. As a message reader, the stage accepts all message types.
- If the reads are browse reads, the message remains on the queue.
- If transaction control for a unit of work applies, the message is removed from the queue but only after the commit following a successful write.

If a request message is read, you must ensure that another application reads and responds to the request message.

WebSphere MQ has two versions:
- Client only
- Client/Server

The difference is described in "Client Only and Client/Server Versions."

Functionality of WebSphere MQ Stages

The WebSphere MQ stage has the following functionality:
- Reads from and writes to IBM WebSphere MQ message queues.
- Connects to a single queue manager, but can open several queues. You can associate each link with a different queue.
- Processes string-formatted messages.
• Provides read options to maintain the message on the queue, delete the message immediately after it is read, or delete the message when the job completes successfully.
• Provides termination conditions by processing a user-defined message type, a user-specified timeout period, or message count. (for message reading)
• Specifies the message descriptor fields that get sent on the input link. (for message writing)
• Controls how a message is to be delivered, such as its priority, persistence, and expiry data. (for message writing)
• Supports many-to-one and one-to-many row formats.
• Supports local units of work.
• Provides a custom GUI to facilitate the defining of metadata associated with the choice of message descriptor fields.
• Defines new data elements that specify transformations on message descriptor data.
• Supports the publish/subscribe communication model.
• Supports NLS (National Language Support).

The following functionality is not supported:
• Reference links. These are unsupported since message data cannot be guaranteed to be persistent, and lookups are not key-based.
• Processing nonstring-based messages, such as triggers and other event-related messages.
• Global units of work.
• Metadata import.
• Data browsing, which is the ability to use the custom GUI to view sample native data.
• Request messages. All queue puts are datagram messages.
• Single-row, repeating-group row data.
• Complex message formats.
• MQ Series Client-only option.

Terminology

The following table lists the IBM WebSphere MQ stage terms used in this document:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datagram message</td>
<td>A message containing information for which no response is expected.</td>
</tr>
<tr>
<td>Message Queue</td>
<td>A synonym for queue.</td>
</tr>
<tr>
<td>Namelist</td>
<td>An WebSphere MQ object that contains a list of queue names.</td>
</tr>
<tr>
<td>Object authority manager (OAM)</td>
<td>The default authorization service for command and object management for WebSphere MQ on UNIX and Windows systems.</td>
</tr>
</tbody>
</table>
### Client Only and Client/Server Versions

There are two versions of WebSphere MQ:
- **Client only** is designed for workstation and remote wireless support. It has a smaller footprint and does not require a local WebSphere MQ server. IBM InfoSphere DataStage does not have to be installed on the MQ server.
- **Client/Server** requires a WebSphere MQ server be present. InfoSphere DataStage must be installed on the MQ server.

**Note:** You cannot use both versions on a single InfoSphere DataStage computer.

The user interface for both versions is the same.

### Publish/Subscribe Communication Model

With the publish/subscribe communication model, applications are not tied to particular partners. Publish/subscribe systems deal with data and have no specific requirements for the recipients or the sources of the messages. Publish/subscribe decouples the provider of information from the consumers of that information.

The provider of the information is called a publisher. Publishers supply information about a subject. The consumer of the information is called a subscriber. A broker acts as an intermediary between the two.

**Note:** A subscriber must register and deregister with a broker in order to receive publications. This can be done outside of IBM InfoSphere DataStage or through the IBM WebSphere MQ stage.

Information is sent in an MQ message, and the subject of the information is identified by a topic. The publisher specifies the topic when the information is published. The subscriber specifies the topics about which information is desired. The subscriber is sent only the information to which it subscribes.
The WebSphere MQ stage can register and unregister a subscription, read the message and strip off header information from the message, and, if desired, store the actual topic in a column.

**Note:** The WebSphere MQ stage supports only the XML standard convention of self-defining messages.

For a description of the Subscription tab, see “Subscription Tab” on page 152.

---

**Using the WebSphere MQ Stage**

**About this task**

The primary purpose of the IBM WebSphere MQ stage is to read from or write to WebSphere MQ message queues. See “WebSphere MQ Stage Message Queues” for an explanation of message queues.

To read data from a WebSphere MQ queue, you can use one of two approaches:

- Browse the queue (see “Browsing the Queue”), leaving the message intact or destroying the message
- Use transaction control under unit of work to read a message from a queue and write it successfully before a commit destroys the original message (see “Processing Units of Work”)

---

**IBM WebSphere MQ Stage Message Queues**

The following sections describe queue managers, queue and message details, and rows in messages as well as formats and security.

**Queue Manager**

The queue manager controls one or more queues. When you design a job, you specify appropriate stage properties to connect to the queue manager. Each stage instance can connect to only a single queue manager.

Since you can open any number of queues, however, the queue name is a link property. This required property lets a stage instance open a single queue per link for reading or writing. You can use the IBM InfoSphere DataStage and QualityStage Director to validate the connection by using the supplied values for the queue and queue manager names.

Namelists are supported for input links. A namelist is a WebSphere MQ object that contains a list of queue names. If you specify a namelist instead of a queue name, the stage writes messages to each queue defined in the namelist by dynamically creating a distribution list from these queues. Distribution lists let you use a single write to send a message to multiple destinations. You can use a single open to open multiple queues and use a single write to send the message to each of these queues.

**Note:** If you encounter problems when creating a queue manager with WebSphere MQ 6, then apply FixPack 6.0.2 for WebSphere MQ.

For more information about the input and output links, see “Defining WebSphere MQ Input Data” and “Defining WebSphere MQ Output Data”.

---
Queue and Message Details
You can specify link properties to define additional queue and message details. The column metadata describes how to parse the message content into one or more columns. Since complex message formats are not supported, the IBM WebSphere MQ stage parses all messages as fixed-length records. The metadata defines the length of each field. The WebSphere MQ stage supports messages that are equal to or shorter than the field lengths defined in the metadata. For more information about column widths and data lengths, see “Rules for Column Length” on page 155.

Because there is currently no standard for describing and retrieving the structure and arrangement of a WebSphere MQ message, you must handle this in your application design. The WebSphere MQ stage cannot obtain the structure of the message dynamically.

Actual column definitions are retrieved directly from the application programs, their supported design tools or entered manually. For messages that use XML content, the InfoSphere DataStage support for Document Type Definitions (DTD) and XML document metadata defines message details.

Rows in a Message
In simple terms, IBM InfoSphere DataStage reads and processes a row for every queue get operation or executes a queue put for every row that the stage receives from another part of the job.

However, for some applications, there is no one-to-one correlation between a relational row and a single transaction contained within a message. Furthermore, the WebSphere MQ message can have its own hierarchical structure. There can be one physical row per message, but the row itself can contain multiple repeating groups.

WebSphere MQ messages can be very large. A single message can be an entire answer set or the contents of a file. You define the length of a row when you design the link of the InfoSphere DataStage job in one of the following ways:

- **Number of columns.** Multiple rows can be enclosed in a single message. The total number of columns, including their byte lengths, equals the length of the row. The stage releases rows to the InfoSphere DataStage engine based on this length as it reads messages. See “Columns Tab” for more information about the columns.

- **CRLF.** If CRLF is available in ASCII messages, it optionally indicates the end of a row when reading messages. This means that for one queue get, there can be many rows released down a link for processing. The **Ignore end-of-record** box on the Output page specifies whether to treat CRLF as a row terminator.

- **Rows per Message.** When writing WebSphere MQ messages, the **Rows per Message** box on the Input page specifies when to execute a queue put.

For more information about the Input and Output pages, see “About the Input Page” and “About the Output Page”.

XML Format
XML is a popular format for messages in data integration applications. The hierarchical nature of XML implies the possibility that a single queue get equals a single row.
It also implies that the row has a complex internal structure that includes repeating groups. This can be true for other message formats as well. In these cases, IBM WebSphere MQ stage simply processes these messages as a single row and lets the row be transformed by existing technologies within InfoSphere DataStage, such as the XML reader stage. The WebSphere MQ stage does not support complex message parsing.

**Browsing the Queue**

**About this task**

One method of reading messages from a queue is to browse the queue. With a browse mode read, you choose whether to remove the message from the queue. A nondestructive read leaves the message intact. A destructive read destroys the message under specific circumstances.

After IBM WebSphere MQ stage reads the message, it releases the row or message on its output link to:

- Another WebSphere MQ stage
- A Transformer stage
- Any other active or passive stage

The row or message can then be processed as appropriate for the application.

**Processing Units of Work**

An alternative method of handling queues includes not only reading a message from a queue but removing that message from the queue after the message is successfully processed. To do this, the IBM WebSphere MQ stage incorporates transaction control into queue management by using units of work. The WebSphere MQ stage supports the local definition of a unit of work that is exclusive to and within the context of the connection to a single queue manager. This is useful when InfoSphere DataStage reads from a queue, translates message contents, then delivers the information to another queue. Transaction control requires the use of a transformer stage, and transaction control must be turned on (see [Enable transaction grouping](#)).

**Note:** Global units of work cross MQ and RDBMS boundaries, such as reading a message, then performing an SQL insert. These global units of work are not currently supported. Only local units of work are currently supported.

InfoSphere DataStage processes a unit of work as follows:

1. Reads the queue without destruction as a queue-browse operation.
   Messages remain on the source queue after the read. This is done because it is not prudent to remove a message from the queue until the message has been processed successfully. The reader WebSphere MQ stage has no way of determining if processing is successful.
2. Processes the message.
3. Executes a second read of the message.
4. Writes the message to the target queue.
5. Executes a queue commit.

   If the job fails before the commit, the original message is still available in the source queue. This functionality is supported by the syncpoint control options.
for queue get and queue put and the commit and rollback calls. If the job is successful, the original message is removed as part of the commit.

Job Scenario
About this task

The following job shows how the WebSphere MQ stage handles a local unit of work. For more information about input and output links, see "General tab" for the Input page and "General Tab" for the Output page.

A Transformer stage splits the physical message into logical rows, which can be sent to a WebSphere MQ writer stage on separate input links. You can coordinate these message writes in a local unit of work because the input link rereads the original message.

The Transaction Handling tab on the Input page defines the role of each link in the transaction control group. For more information about the Transaction Handling tab on the Input page, see Transaction Handling Tab.

The first link, DSLink4, defines the controlling link, which carries the message identifier. Unconventionally, it does a read of the message identified by the message identifier rather than a write.

A successful write on DSLink5 commits the unit work, removing it from the source queue. A failure rolls it back, which causes the message to remain on the source queue.

If more than two queues receive the original message or a transformation of the message, it implies that there are more than two input links to the WebSphere MQ stage. In this case, only the last link causes a commit on success. A failure on any input link, including the controlling link, causes a rollback, leaving the original message intact.

These next sections give detailed information about how the WebSphere MQ stage incorporates unit of work and transaction control while doing the following:

- Reading queues in browse mode
- Writing messages to destination queues by using message identifiers
- Executing a queue commit for a local unit of work

Figure 1. Job Flow Showing Message Split into Logical Rows

A Transformer stage splits the physical message into logical rows, which can be sent to a WebSphere MQ writer stage on separate input links. You can coordinate these message writes in a local unit of work because the input link rereads the original message.
Reading Queues
IBM WebSphere MQ stage reads messages from a queue in browse mode and releases the row or message on its output link to a Transformer stage. A browse mode read is nondestructive.

The row contains a unique message identifier that moves along the data flow to the Transformer stage that coordinates transaction control. This Transformer stage branches into two or more output links that enter the same WebSphere MQ stage for writing.

Writing Messages to Target Queues
The Transformer stage defines the set of output links as a transactional group. One of the links in the transactional group is defined as the controlling link. The only column that must be defined on the controlling link is the message identifier.

Executing a Queue Commit
About this task
In IBM WebSphere MQ stage with multiple input links, the controlling link uses the unique message identifier provided on the link to reread the original message in the source queue.

The queue read is then executed under syncpoint control, which implies the start of a transaction. The stage writes the message to its target queues for each of its subsequent links in the transaction group.

If any queue writes fail, the WebSphere MQ stage executes a queue rollback based on the reread message, and the original message remains on the queue. However, if all the queue writes succeed, the stage executes a queue commit, and the original message is removed from the queue.

Transactional control is unsupported when there is a one-to-many relationship between a message and InfoSphere DataStage rows. Avoid this situation by doing the following:
• Use the Ignore end of record field on the General tab of the Output page to ignore CRLF as a logical row terminator (see "General Tab").
• Set the total combined length of the columns designated by Number of columns to be at least the size of the message.

Troubleshooting
This section describes possible problems you might encounter and ways to resolve them.

Troubleshooting Infinite loops
About this task
If you read from and write to the same queue in the same process, the messages are continuously written into the queue until the maximum number as specified by Message Limit is reached. An infinite loop can result. To resolve this problem, do one of the following:
• Message Limit. Specify a positive value to cause the stage to stop reading messages. For more information about the parameters on the General tab of the Output page, see "General Tab".
• Staging area. Use a Sequential File stage as a staging area for temporary storage. Then add another process to write messages back to the queue.
Troubleshooting Queue Manager Configuration

Only one connection to one queue manager is allowed for one process. To resolve this limitation, do one of the following:

- **Remote queue definition.** Configure a queue belonging to a second queue manager as a remote queue definition for the primary queue manager.

- **Cluster.** Configure the primary and remote queue managers as a cluster. This lets a single local queue manager access queues belonging to different queue managers.

- **Staging area.** Use a Sequential File stage as a staging area for temporary storage. Then add another process to write messages back to the queue. This way, each process has its own connection to its own queue manager.

Reply and Report Messages

When a queue read occurs, some messages read by the stage can request a reply or report message in return. Since these requests are usually specific to the application, IBM WebSphere MQ stage does not return any reply or report messages requested by the sending application.

However, in the job scenario described in [Job Scenario] messages within a job retain the message descriptor information that is defined by the message originator. When the message reaches its intended destination, the destination application services the replies and reports requested by the originating application.

Security

The object authority manager (OAM) is the default authorization service for command and object management for IBM WebSphere MQ on UNIX and Microsoft Windows systems. It authorizes access to queue managers and queues by using access control groups, which correspond to user groups for operating systems. The user identity of the application determines the access to WebSphere MQ objects. You can replace the OAM or run it with your own security service.

When a message is put on a queue, the queue manager supplies a user name in the message descriptor. The default OAM then authorizes access based on this user name. The queue manager gets the user name from the operating system by default. However, the application can supply its own user id, which must be a valid operating system user.

You can use the stage properties **User Name** and **Password** to change the identity of the user running the job. Once authenticated, the stage uses the user name to connect to the queue manager as follows:

- **For message writes.** The identity context of the message is modified to reflect the specified user.

- **For message reads.** The rights associated with the specified user are used by the queue manager to determine message availability.

**Windows.** For Windows, the IBM InfoSphere DataStage user that executes a job must have the 'create a token object' policy granted by a Windows administrator. Without this user policy, the user cannot impersonate the user specified in the user name property of the WebSphere MQ stage.
**Administration groups.** By default, users belonging to an administration group have complete access to a queue manager and local queues for a system. Examples of an administration group are:

- Administrator on Windows
- root on UNIX
- The WebSphere MQ mqm group on both operating systems

Other users must be given explicit access to the various WebSphere MQ objects.

**setmqaut utility.** If you are using the OAM, use the setmqaut utility to grant the required permissions to other operating system users. The following minimal permissions are required for input and output links. These permissions are required for a non-administrative user to run jobs that contain a WebSphere MQ stage instance:

- Output links (read)
  - Connect on the queue manager
  - Get and browse on the queue
- Input links (write)
  - Connect on the queue manager
  - Put and passall on the queue
- Input links with a namelist (distribution list write)
  - Connect on the queue manager
  - Inq on the namelist
  - Put and passall on all queues names in the namelist
- Input links within local units of work (for details on units of work, see “Processing Units of Work”)
  - See the previous permissions, depending on the destination (queue or namelist)
  - Get on the queue for the syncpoint read

---

**Defining the WebSphere MQ Connection**

When you use the stage GUI to edit IBM WebSphere MQ stage, the MQSeries Stage dialog box opens. This dialog box has the **Stage**, **Input**, and **Output** pages (depending on whether there are inputs to and outputs from the stage). In this case, there are no outputs from the stage.

- **Stage.** This page displays the name of the stage you are editing. The **General** tab defines the WebSphere MQ connection. For details, see [Connecting to a Queue Manager](#).

  The **NLS** tab defines a character set map to use with the stage. This tab appears only if you have installed NLS for InfoSphere DataStage. For details, see [Defining Character Set Mapping](#).

- **Input.** This page is displayed only if you have an input link to this stage. It specifies the queue or namelist to which messages are written. It also specifies priorities, types of messages, the persistence of messages, and how messages are written to a queue.

- **Output.** This page is displayed only if you have an output link to this stage. It specifies the conditions to be met for the stage to stop reading messages. It also specifies how to handle the end of logical InfoSphere DataStage rows and message tracking options.
Defining an IBM WebSphere MQ Stage

About this task

The main steps in defining an IBM WebSphere MQ stage from the MQSeries Stage dialog box are as follows:

Procedure

1. Connect to a queue manager (see "Connecting to a Queue Manager").
2. Optional: Define a character set map (see "Defining Character Set Mapping").
3. Define the data on the input links if a target stage (see "Defining WebSphere MQ Input Data"). Or define the data on the output links if a source stage (see "Defining WebSphere MQ Output Data").
4. Click OK to close this dialog box. Changes are saved when you save the job design.

Connecting to a Queue Manager

IBM WebSphere MQ connection parameters are set on the General tab on the Stage page.

To connect to a WebSphere MQ queue manager, specify the following information:

- **Queue Manager.** The name of the queue manager to which the stage connects. This is a required field.
- **User Name.** The alternate user name you can use to connect to the queue manager. If this field is left empty, the InfoSphere DataStage connection information is used. User Name is not active on UNIX platforms.
- **Password.** A password for the specified user name. It is ignored if User Name is omitted. Password is not active on UNIX platforms.
- **Description.** Optional. A description of the purpose of the stage.

Defining Character Set Mapping

About this task

You can define a character set map for a Plug-in stage. Do this from the NLS tab that appears on the Stage page. The NLS tab appears only if you have installed NLS.

Specify the information by using the following button and fields:

- **Map name to use with stage.** Defines the default character set map for the project or the job. You can change the map by selecting a map name from the list.
- **Use Job Parameter...** Specifies parameter values for the job. Use the format #Param#, where Param is the name of the job parameter. The string #Param# is replaced by the job parameter when the job is run.
- **Show all maps.** Lists all the maps that are shipped with IBM InfoSphere DataStage.
- **Loaded maps only.** Lists only the maps that are currently loaded.
Defining IBM WebSphere MQ Input Data

When you write messages to a message queue, the IBM WebSphere MQ stage has an input link. Define the properties of this link and the column definitions of the data on the Input page in the MQSeries Stage dialog box of the stage GUI.

About the Input Page

The Input page has an Input name field, the General, Options, Columns, and Transaction Handling tabs, and the Columns... button:

- **Input name.** The name of the input link. Choose the link you want to edit from the Input name list box. More than one input name exists when more than two queues receive the original message or a transformation of it. The first link can define the controlling link, which does the queue read (the reread of the original message). But the order of link execution can be modified in the transformer. To determine the order of execution:
  - Pass the cursor on a link in the job flow diagram. A Tooltip shows the link's order of execution.
  - Open the Transformer Editor. The sequence of the output links in the right pane indicates the order of execution.
  - To change the execution sequence, click Output Link Execution Order. The Transformer Stage Properties dialog box appears open to the Link Ordering tab on the Stage page.
  - Select the Transaction Handling tab on the Input page. Select Enable Transaction Grouping. The links are listed in the order of execution.
- **Columns...** Click the button to display a brief list of the columns designated on the input link. As you enter detailed metadata in the Columns tab, you can leave this list displayed.

Note: View Data is not supported on input links.

General tab

This tab is displayed by default. It contains the following fields:

- **Queue Name or Namelist.** The name of a queue or namelist to which messages are written. If you specify a namelist, messages are written to each queue in the namelist. If in transactional mode, the message writes are not committed unless the writes to all the queues in the namelist are successful.
- **Rows per Message.** The number of rows the stage buffers before executing a write. The default is one write per row arriving on the input link.
- **Message Priority.** The priority of the message that is written to the queue. A value of -1 causes the message to be written with the default priority of the queue manager. Zero is the lowest priority. If the specified priority exceeds the maximum priority supported by the queue manager, the queue manager accepts the message, placing it on the queue at the maximum priority for the queue manager. The queue manager returns a warning in this case.
- **Message Type.** The type of message to be written. The list box displays the IBM WebSphere MQ system-defined message types: Request, Reply, Report, and Datagram. You can also enter a numeric value representing a user-defined message type. The default is a datagram message.
- **Message Expiry.** The length of time in tenths of a second that a message remains on the queue. If the message remains on the queue longer, it is discarded. A value of -1 represents an indefinite amount of time. A value of 0 is not allowed.
• **Message Persistence.** The persistence of messages written to the queue. Choose one of the following values:
  - **Default.** A message is placed on the queue with the default persistence of the queue manager.
  - **Persistent.** A message survives restarts of the queue manager.
  - **Non-persistent.** A message does not survive restarts of the queue manager.

• **Message Format.** The format of messages written to the queue. Choose one of the following values:
  - **MQSTR**
  - **MQRFH2**
    The default is **MQSTR**.

• **Reply-To Queue.** The name of the queue. If **Message Type** is **Request**, you have the option to provide the name of a queue.

• **Reply-To QManager.** The name of the queue manager. If **Message Type** is **Request**, you have the option to provide the name of a queue manager.

• **CCSID.** The character set identifier of the character data in the message. Choose one of the four default values described below or provide a value of your choice.
  - **Queue Manager.** The character set identifier of the queue manager.
  - **Default.** The default coded character set identifier.
  - **Inherit.** The inherited character set identifier of this structure.
  - **Embedded.** The embedded character set identifier.
    The default is **Queue Manager**. See IBM WebSphere MQ documentation for additional information.

**Options Tab**
The Options tab contains options that control how a message is written.

This tab contains the following fields:

• **Reject row on transaction failure.** The handling of a message when an attempt to write the message fails. If selected and a put message fails, a job continues to run except in the following cases:
  - **MQRC_CONNECTION_BROKEN**
  - **MQRC_MISSING_REPLY_TO_Q**
  - **MQRC_NOT_OPEN_FOR_INPUT**
  - **MQRC_PUT_INHIBITED**
  - **MQRC_Q_DELETED**
  - **MQRC_Q_FULL**
  - **MQRC_Q_MGR_NOT_AVAILABLE**
  - **MQRC_Q_MGR_STOPPING**
  - **MQRC_CONNECTION_STOPPING**

• **Do not pad spaces onVarChar columns.** The treatment of a message when its length is less than the maximum length of the VarChar column. If selected, the column is not padded with spaces at the end of the message.

**Columns Tab**
This tab contains the column definitions for the data written to the message queues. The **Columns** tab behaves the same way as the **Columns** tab in the ODBC stage.
Transaction Handling Tab
This tab contains the parameters that define the role of each link in the transaction control group, letting you view the transaction handling features of the stage as it writes to the data source.

Using transaction handling depends on whether Enable transaction grouping is selected. You can specify the number of rows written before each commit.

For information about specifying transaction control information, see “Specifying Transaction Control Information”.

The Transaction Handling tab contains the following fields:
- **Enable transaction grouping.** A check box, which if selected, displays the grid with details of the transaction group to which the currently selected input link belongs. Enable transaction grouping is available if you have at least two input links. The check box is cleared by default.
- **Rows per transaction.** The number of rows written before the data is committed to the data table. Set the value explicitly to 1 to avoid causing the local units of work configuration to fail.
- **On Skip.** The action taken after a successful write. It specifies whether to continue or to roll back if a link is skipped because of an unsatisfied constraint in the preceding Transformer stage. Choose Continue or Rollback from the list. On Skip is available if Enable transaction grouping is selected.
- **On Fail.** The action taken after a failed write. It specifies that a transaction is rolled back. A rollback causes the message to remain on the source queue. On Fail is available if Enable transaction grouping is selected.

Specifying Transaction Control Information

About this task
As previously discussed, you can associate multiple input links writing to a single data source together as a transaction group. The transaction grouping feature is turned on and off by using the Enable transaction grouping check box on the Transaction Handling tab (it is cleared by default).

If you clear Enable transaction grouping, you can enter a suitable value in the Rows per transaction field on the Transaction Handling tab. This is the number of rows written before the data is committed to the data table. The default value is 0, that is, all the rows are written before being committed to the data table.

If transaction grouping is enabled, the following rules govern the grouping of links:
- All the input links in the transaction group must originate from the same Transformer stage.
- The ordering of the links within the transaction group is determined in the preceding Transformer stage.
- A transaction group cannot use a Rows per transaction value other than 1.

Note the following facts about transaction groups:
A transaction starts at the beginning of each iteration of the Transformer stage preceding the IBM WebSphere MQ stage. Any uncommitted changes left over from a previous transaction are rolled back.

The links in the transaction group are processed in the order specified by the Transformer stage. Individual links are skipped depending on the constraints specified in the preceding Transformer stage.

Each link in the transaction group can specify whether to rollback on failure. A rollback on any link causes the transaction to be abandoned, and any subsequent links in the group are skipped.

Each link in the transaction group can be set to rollback if a constraint on that link is not met. Again, such a rollback causes the transaction to be abandoned, and any subsequent links in the group are skipped.

The row counter for each link is incremented only if the message write associated with the link executes successfully and the transaction is successfully committed.

The transaction ends after the last link in the transaction group is processed unless a preceding link performs a rollback. In this case, the transaction ends there.

For information about using links in a transaction group, see “Processing Units of Work”.

**Defining WebSphere MQ Output Data**

When you read data from an IBM WebSphere MQ queue, the WebSphere MQ stage has an output link. The properties of this link and the column definitions are defined on the Output page in the MQSeries Stage dialog box.

**About the Output Page**

The Output page has an **Output name** field, the **General**, **Options**, **Message Type Options**, **Message Filtering Options**, **Message Tracking Options**, **Subscription**, and **Columns** tabs, and the **Columns**... and **View Data**... buttons.

- **Output name**. The name of the output link. Choose the link you want to edit from the **Output name** list box. This list box displays all the output links.
- **Columns**... . Click the button to display a brief list of the columns designated on the output link. As you enter detailed metadata in the **Columns** tab, you can leave this list displayed.
- **View Data**. Click the button to browse an IBM WebSphere queue. All browsing uses nondestructive reads. If you select **Destructive Read** (see “Options Tab”), the option is ignored.

**View Data** allows you to select which columns to display by using the **Display** button and selecting the columns of interest. If a field contains binary data, it is replaced with a question mark (?) to insure it displays properly on the screen if the data has no special meaning. If the binary data has semantic meaning, the stage attempts to convert the data into a field that can be displayed and conveys the semantic meaning.

Message filters are allowed with browsing. See “Message Filtering Options Tab”.

A combination of **Wait Time**, **Message Limit**, and **End of Data Message** (see “General Tab”) terminates browsing the queue. **Wait Time** is set to 0 internally to prevent extended wait times.

- A value of 0 in **Message Limit** causes all the messages in the queue to display.
- A value of \( n \) in **Message Limit** causes \( n \) messages in the queue to display.
- If \( n \) is greater than the number of messages, the value in **End of Data Message** signals the end of the messages.

In the following examples, the browse queue contains ten messages.
- If **Message Limit** is set to 0, ten messages are displayed.
- If **Message Limit** is set to 5, five messages are displayed.
- If **Message Limit** is set to 100, ten messages are displayed with no wait.
- If **Message Limit** is set to 10, ten messages are displayed.

**Note:** **View Data** is not active unless you provide **Queue Name** on the Stage page.

**General Tab**

This tab provides the interface for entering queue names, the job controls necessary to stop reading messages, and options to track messages.

IBM WebSphere MQ queue stage must know when to stop reading messages from the queue. Since messages can be delivered predictably or randomly, it is difficult to identify the logical end of the data set.

Output link properties control when the stage stops reading messages from the queue. These properties are not mutually exclusive. The stage returns an end of data message when the first of these conditions is met. You can define the following:
- The time to wait for a message to arrive
- The message sent when the end of the data is reached
- The number of messages read from the queue

Output link properties also provide read options to
- Maintain messages on the queue
- Delete each message immediately after it is read
- Delete messages when the job completes successfully

The **General** tab contains the following fields:
- **Queue Name.** The name of the queue from which messages are read.
- **Wait Time.** The number of seconds to wait for a message to be read from the queue. If this period elapses and no messages are available, the job finishes. Choose one of the following values:
  - 0 specifies no wait
  - -1 specifies an indefinite wait period (default)
  - Positive integers specify to wait \( n \) seconds
    If you specify a wait time of more than 5 minutes, the time is increased by intervals of 5 minutes, for example, 7 minutes is increased to a wait time of 10 minutes.
- **Message Limit.** The number of messages to be read from the queue. A positive integer (a value greater than zero) for this field causes the stage to stop reading messages after the specified number has been read, even if messages remain on the queue. The default value 0 indicates that **Message Limit** does not indicate the end of the data.
Note: A warning occurs that the job has no termination conditions if all the following conditions are met:
- Wait Time is -1
- Message Limit is <= 0
- End of Data Message is 0

- **End of Data Message.** An integer, which is stored in the message descriptor, that represents the user-defined message type. A default value of 0 indicates that no special messages should be expected to signal the termination of message reading. Typically, the stage processes datagram or request messages. If the stage receives a message whose type matches this integer, it ends the processing. The following MQ constants are the upper and lower limit values for user-defined message type and define the range expected for the End of Data Message value.
  - MQMT_APPL_FIRST
  - MQMT_APPL_LAST
You can specify integers between MQMT_APPL_FIRST and MQMT_APPL_LAST inclusively.

- **Description.** Optional. A description of the output link.

Note: A job can run indefinitely if it does not reach **Message Limit** or if it does not receive **End of Data Message**. To avoid this possibility, specify a suitable value for **Wait Time** on the **General** tab of the Output page.

**Options Tab**
This tab provides options determining how messages are read. The **Options** tab contains the following check boxes:

- **Destructive Read.** The type of read to be executed. If not selected (the default), the message is left on the queue after it is read. If selected, each message is deleted from the queue immediately after it is read.

  **Note:** Do not use **Destructive Read** in a job with local unit of work (see "Processing Units of Work").

- **Commit/Backout only once at end of job.** A further refinement of the type of read to be executed. This check box is active only if **Destructive Read** is selected. If **Commit/Backout only once at end of job** is selected, messages are deleted only when the job finishes successfully. If the job does not finish successfully, messages remain on the queue. If not selected (the default), each message is deleted from the queue immediately after it is read.

  Do not use **Commit/Backout only once at end of job** with any job consisting of multiple processes. In the following example, the job writes to a table or a file and reads from that table or file, IBM InfoSphere DataStage introduces multiple processes in this configuration.
In the next example, there are two target files directly linked to the WebSphere MQ stage, and again InfoSphere DataStage introduces multiple processes.

**Note**: If an After SQL statement fails and the job is aborted, messages are removed from the queue even if **Commit/Backout only once at end of job** is selected, because the message has already been successfully moved to the target.

- **Ignore end-of-record**. A check box specifying that the stage does not treat carriage returns or new lines in the message stream as the end of a logical InfoSphere DataStage row. If cleared, a carriage return or new line indicates the end of a logical InfoSphere DataStage row. Therefore, a single WebSphere MQ message can result in many InfoSphere DataStage rows.

- **Truncate on Buffer Mismatch**. A check box specifying whether to truncate a message. If selected and the buffer size is smaller than an incoming message, the stage truncates the message. If selected and the buffer size is greater than an incoming message, the stage pads the message with blanks or zeroes. If not selected (the default) and the buffer size is smaller than an incoming message, the stage splits the message into several rows.
• **Do not pad spaces on VarChar columns.** The treatment of a message when its length is less than the maximum length of the VarChar column. If selected, the column is not padded with spaces at the end of the message. **Do not pad spaces on VarChar columns** is not active unless **Truncate on Buffer Mismatch** is selected.

• **Do data Conversion on the MQGET call.** A check box specifying automatic conversion is to take place. If the check box is selected, the stage sets the MQGMO_CONVERT option in the Message Get Options structure, and automatic conversion occurs. If the check box is not selected (the default), the stage does not set the MQGMO_CONVERT option, and conversion does not occur.

**Message Type Options Tab**
This tab provides options determining how messages are read.

The **Message Type Options** tab contains a set of check boxes and a place to specify additional values. Only messages with a message type matching the selected message filter options are extracted from the queue and passed as columns on the output link. **System First** and **System Last** together make up a range for system-specific values. **Application First** and **Application Last** together make up a range for application-specific values.

Use **Additional values** to provide specific values. You can provide numbers and ranges of numbers separated by commas.

The values indicated by the check boxes and the values provided in **Additional values** are concatenated into one comma-delimited string value for processing.

**Message Filtering Options Tab**
This tab provides options to filter or validate messages. Every input value must meet the specified condition.

The **Message Filtering Options** tab contains the following fields:

• **Apply Filtering.** A check box specifying whether filtering is active. If not selected (the default), filtering is not active.

• The following table contains a list of properties that can be filtered and the condition against which each is validated:

  **Table 13. Properties and conditions**

<table>
<thead>
<tr>
<th>Property</th>
<th>Condition against which Validated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Value from 0 to 999,999,999</td>
</tr>
<tr>
<td>Correlation ID</td>
<td>Character string limited to 24 characters for each element that is not a job parameter</td>
</tr>
<tr>
<td>Group ID</td>
<td>Character string limited to 24 characters for each element that is not a job parameter</td>
</tr>
<tr>
<td>Offset</td>
<td>Value from 0 to 999,999,999</td>
</tr>
<tr>
<td>Sequence</td>
<td>Value from -1 to 999,999,999</td>
</tr>
<tr>
<td>Put Date</td>
<td>YYYMMDD (limited to 8 characters for each element if not a job parameter)</td>
</tr>
<tr>
<td>Put Time</td>
<td>HHMMSSSTH (limited to 8 characters for each element if not a job parameter)</td>
</tr>
<tr>
<td>Message Format</td>
<td>Character string limited to 8 characters for each element that is not a job parameter</td>
</tr>
</tbody>
</table>
Table 13. Properties and conditions (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Condition against which Validated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Buffer Length</td>
<td>Value from 0 to 4194304</td>
</tr>
<tr>
<td>Originating User ID</td>
<td>Character string limited to 12 characters for each element that is not a job parameter</td>
</tr>
<tr>
<td>Original Application Name</td>
<td>Character string limited to 28 characters for each element that is not a job parameter</td>
</tr>
<tr>
<td>Original Application Type</td>
<td>Value from 1 to 999999999</td>
</tr>
</tbody>
</table>

**Message Tracking Options Tab**

This tab provides options to track messages.

The **Message Tracking Options** tab contains a subset of the message tracking data that can be extracted from the message descriptor and passed as columns on the output link. The message tracking data include:

- **Priority**. The message priority
- **Correlation ID**. The message correlation identifier
- **Group ID**. The message group identifier
- **Offset**. The message offset for segmented messages
- **Sequence**. The message sequence number
- **Put Date**. The unchanged message put date
- **Put Time**. The unchanged put time
- **Message Type**. The message type
- **Message Format**. The format name of the message data
- **Message Buffer Length**. The length of the message returned from the read
- **Originating User ID**. The user identifier
- **Original Application Name**. The name of the application that put the message
- **Original Application Type**. The type of application that put the message

The column metadata for these message tracking items is automatically maintained by the user interface.

**Subscription Tab**

The **Subscription** tab, found on the Output page, supports the publisher/subscriber communication model.

The **Subscription** tab contains the following fields:

- **Register/Deregister subscriber**. A check box specifying whether the subscriber is registering outside of IBM InfoSphere DataStage or through the WebSphere MQ stage. If selected, the stage submits a Register Subscriber command.
- **Topics/Topics**. The topics of interest to the subscriber. Enter the topic of interest. The stages reads messages that match the specified topic. Topics can be multilevel. Use a slash (/) to separate levels. The maximum length of a topic name is 126 bytes. The stage supports wild cards. Use
  - The asterisk (*) to match multilevel topics. (The pound sign (#) has special meaning in InfoSphere DataStage and cannot be used.)
  - The plus sign (+) to match single-level topics.
• **Local subscription.** A check box specifying to the broker that the subscription is local and should not be distributed to other brokers in the network. Only publications at this node by a publisher specifying Local are sent to this subscriber.

• **New publications only.** A check box specifying that only new publications, and not those retained at the time the subscription is registered, are sent.

• **Reply-To queue is model queue name.** A check box specifying that the broker sends the subscription response to the reply queue, which is a model queue. The stage creates the model queue at runtime.

• **Correl ID as subscriber's ID.** A check box specifying that the Correlation ID in the message descriptor is part of the subscriber's identity.

• **Reply-To Queue.** The name of the queue to which the broker sends the subscription response.

• **Subscription Name.** The subscription name. If specified, the subscription name is the single field used to identify a subscription, overriding the traditional identity.

  **Note:** Traditional identity is the queue manager, queue, and optional correlation id used to refer to a subscription.

• **Correl ID.** The correlation id value that is used for the subscriber's identity.

**Reading a Mix of Message Formats**

**About this task**

IBM WebSphere MQ stage allows you to design a job that reads a mix of message formats, for example, string, MQRFH2, and NONE formats. To read a message in the MQRFH2 format, you must select **Apply Filtering.** See **Message Filtering Options Tab.** If Apply Filtering is not selected, the stage reads only messages in string format.

**Reading Selectively from Multiple Topics**

**About this task**

Where multiple topics are subscribed to the same queue, you have the option to read certain topics. In the **Message Filtering Options** tab, enter the desired name in **Topic.** This is a case-sensitive entry. You can filter multiple topics. Use a comma (,) to separate the topics. Topic filtering applies only to MQRFH2-format messages.

  **Note:** Exercising this option negatively affects the performance of the stage.

If **Topic** is not empty and a message format is other than MQRFH2, whether the message is read depends on the setting of message format filtering. If there is no format filtering, all non-MQRFH2 messages are allowed to pass to the next stage in the job.

  **Note:** If you want to read only the requested topic, enter MQRFH2 in **Message Format.** Note this is not MQRFH2.

**Processing the MQRFH2 Header**

**About this task**

The stage does limited processing with the MQRFH2 header. If you select **Remove MQRFH2 header** on the Options page, only the message body is passed to the next stage. If you clear Remove MQRFH2 header, both the MQRFH2 header and
the message body are passed to the next stage. You can track the topic in the MQRFH2 header by selecting Publication Topic on the **Message Tracking Options** tab. A column named TopicName is added to the Column page. See "Message Tracking Options Tab".

**Columns Tab**

This tab contains the column definitions for the data being output on the chosen link. For information about the data elements in the IBM WebSphere MQ stage, see “Using Column Data Elements”.

The selection of message tracking options affects the column definitions on the output link as seen in the following table representing the **Columns** tab:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Derivation</th>
<th>Key</th>
<th>SQL type</th>
<th>Length</th>
<th>Scale</th>
<th>Nullable</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageID</td>
<td>No</td>
<td>Char</td>
<td>24</td>
<td>No</td>
<td>24</td>
<td>No</td>
<td>24</td>
</tr>
<tr>
<td>Priority</td>
<td>No</td>
<td>Integer</td>
<td>10</td>
<td>No</td>
<td>10</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>GroupID</td>
<td>No</td>
<td>Char</td>
<td>24</td>
<td>No</td>
<td>24</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>PutDate</td>
<td>No</td>
<td>Char</td>
<td>8</td>
<td>No</td>
<td>8</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>PutTime</td>
<td>No</td>
<td>Char</td>
<td>8</td>
<td>No</td>
<td>8</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MSGFld1</td>
<td>Yes</td>
<td>Char</td>
<td>6</td>
<td>No</td>
<td>6</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MSGFld2</td>
<td>No</td>
<td>VarChar</td>
<td>100</td>
<td>No</td>
<td>100</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>MSGFld3</td>
<td>No</td>
<td>VarChar</td>
<td>10</td>
<td>No</td>
<td>10</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

The first column name of an output link is always MessageID. This column is used in transactional situations to do syncpoint reads from the source queue. For information, see "Processing Units of Work".

The four column names following MessageID represent the message tracking options selected on the **Message Tracking Options** tab for the Output page. You can modify these column names to avoid column-naming conflicts.

We recommend that you do not modify:

- The data element type that the WebSphere MQ stage uses to identify these special columns. However, you can select alternative data element types for the **PutDate** and **PutTime** tracking options. For a description of the data elements that represent the various message tracking options, see "Using Column Data Elements".

- SQL type. If the stage detects at runtime that the column definitions do not correspond to the selected message tracking options, the job aborts.

The following table shows the **Data element** field for the columns representing the **Message Tracking Options**:

<table>
<thead>
<tr>
<th>Data element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQ.MSGID</td>
<td>Message ID</td>
</tr>
<tr>
<td>MQ.PRIORITY</td>
<td>Message priority</td>
</tr>
<tr>
<td>MQ.GRPID</td>
<td>Message group ID</td>
</tr>
<tr>
<td>MQ.DATE</td>
<td>Message queue put date (GMT)</td>
</tr>
</tbody>
</table>
Table 15. Data Elements for Message Tracking Options (continued)

<table>
<thead>
<tr>
<th>Data element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQ.TIME</td>
<td>Message queue put time (GMT)</td>
</tr>
</tbody>
</table>

The remaining columns (MSGField1 and so forth) represent the logical column names of the fields contained in the WebSphere MQ message. A WebSphere MQ message is a string-formatted message of fixed-length records, except the last column, which can be shorter. Therefore, you should specify character data types and lengths for these columns.

**Rules for Column Length**

Note the following rules about the length of the data for columns on the Input and Output pages:

- If data for a column is shorter than the column width as specified by the Display value, the data is padded with trailing spaces if defined as Char. For VarChar data, Do not pad spaces on VarChar takes precedence.
- If data is longer than the column width as specified by the Display value, it is truncated to the specified column width (Input only).
- If the column width as specified by the Display value is empty, the width is determined based on the SQL type and the data precision (specified by the Length value) as well as on whether Buffer Mismatch (on the Output page) or Do not pad spaces on VarChar (on the Input page) is selected.

**SQL Data Types and Specifications**

The following list documents the SQL data types and the specifications for their column widths:

- **BigInt, Integer, SmallInt, TinyInt.** The column width is the Length plus 1 for the optional sign.
- **Numeric, Decimal.** The column width is the Length plus 2 for the optional sign and decimal point.
- **Float, Real, Double.** The column width is the Length plus 7 for the optional sign, decimal, and exponential expressions, for example, -1.20000e-009.
- **Date.** The column width is 10, by using the YYYY-MM-DD format.
- **Time.** The column width is 8, by using the 24-hour HH:MM:SS format.
- **Timestamp.** The column width is 19, by using the YYYY-MM-DD HH:MM:SS 24-hour format.
- **Others.** (such as Unknown, Char, VarChar, LongVarChar, NChar, NVarChar, LongNVarChar, Binary, VarBinary, and LongVarBinary, Bit) These equal the Length.

Since dates and times are character strings, when messages containing dates or times are read from WebSphere MQ into InfoSphere DataStage or vice versa, the InfoSphere DataStage dates and times are in internal format.

**Using Column Data Elements**

In addition to the message ID, you can choose other message descriptor fields for delivery on the links. The column definitions representing these message properties appear before any columns representing actual message data.

The queue manager generates these message descriptor properties, except **Priority**, in a format that might be meaningless to other downstream stages.
For example, the message **PutTime** is expressed as `HHMMSSTH` where `T` represents tenths of a second and `H` represents hundredths of a second. You might want to express this value as `HH:MM:SS.TH` or in internal IBM InfoSphere DataStage time format. **PutDate** is expressed as `YYYYMMDD`.

The following data elements specify transformations on message put dates and times. IBM WebSphere MQ stage performs these transformations, so a separate Transformer stage is unnecessary.

- **MQ.DATE.TO.TAG**. Converts `YYYYMMDD` to `YYYY-MM-DD`.
- **MQ.DATE.TO.DSDATE**. Converts `YYYYMMDD` to internal InfoSphere DataStage date (days since 12/31/67).
- **MQ.TIME.TO.TAG**. Converts `HHMMSSTH` to `HH:MM:SS.TH`.
- **MQ.TIME.TO.DSTIME**. Converts `HHMMSSTH` to internal InfoSphere DataStage time.

The following data elements associate the message descriptor properties with their corresponding output columns. They are used only for identification purposes, not to imply any transformations on the message data. The stage uses these elements to identify output columns associated with message descriptors. Therefore, you can modify the column names in your job design, if necessary.

- **MQ.MSGID**. The message ID, which is a mandatory output column.
- **MQ.PRIORITY**. The message priority.
- **MQ.CORRID**. The message correlation ID.
- **MQ.GRPID**. The message group ID.
- **MQ.OFFSET**. The message offset for segmented messages.
- **MQ.SEQUENCE**. The message sequence number.
- **MQ.DATE**. The unchanged message put date.
- **MQ.TIME**. The unchanged put time.
- **MQ.MSGTYPE**. The message type
- **MQ.MSGTYPE.TO.STR**. The message type converted to a string value
- **MQ.MSGFORMAT**. The format name of the message data
- **MQ.MSGBUFSIZE**. The length of the message returned from the read
- **MQ.PUTUSERID**. The user identifier
- **MQ.PUTAPPLNAME**. The name of the application that put the message
- **MQ.PUTAPPLTYPE**. The type of application that put the message
- **MQ.PUTAPPLTYPE.TO.STR**. The type of application that put the message converted to a string value

**Note:** Columns for the Message ID, Correlation ID and Group ID data elements contain binary data that is not NLS-mappable. If this data is written to a non-MQ Series stage, turn the NLS off on a per-column basis for these columns in the non-MQ Series stages (set NLS Map to NONE on the Columns tab of the Input page) after selecting the Allow per-column mapping box on the NLS tab.
Chapter 6. Environment variables: WebSphere MQ connector

The WebSphere MQ Connector stage uses these environment variables.

**CC_MSG_LEVEL**

Set this environment variable to specify the minimum severity of the messages that the connector reports in the log file.

At the default value of 3, informational messages and messages of a higher severity are reported to the log file.

The following list contains the valid values:

- 1 - Trace
- 2 - Debug
- 3 - Informational
- 4 - Warning
- 5 - Error
- 6 - Fatal

**CC_SE_TIMESTAMP_FF**

Set this environment variable to specify whether decimal point and fractional digits are included in the timestamp values, when the connector runs in server jobs.

When the environment variable is set to a value other than NONE, MICROSECONDS or SCALE, the behavior is the same as if the environment variable was not set. The environment variable values are case sensitive. When the environment variable is not set, the timestamp values that are produced by the job include a trailing decimal point and six fractional digits.

You can set the environment variable to the following values:

**NONE**

The trailing decimal point and the fractional digits are both omitted.

**MICROSECONDS**

The trailing decimal point and six fractional digits are included.

**SCALE**

The trailing decimal point and $S$ fractional digits are included, where $S$ represents the value of the Scale attribute in the timestamp column definition. When the Scale attribute value is not defined for the column, the Scale attribute value of zero is assumed.

**CC_USE_DEPRECATED_TRANS_PROP**

Set this environment variable to control the behavior of the **Usage > Transaction > End of wave > End of data** property.
When the Usage > Transaction > End of wave > End of data property is not set or undefined, an end of wave marker is not inserted before the end of the data. If the property is defined or set, the connector inserts an end of wave marker before the end of the data.

**CC_WSMQ_FAIL_FOR_ANY_NAMELIST_ERROR**

Set this environment variable to stop a job that is currently running.

This environment variable stops the current job when the following conditions are met:

- When WebSphere MQ returns MQCC_WARNING completion code.
- When WebSphere MQ returns MQRC_MULTIPLE_REASONS reason code.
- One or more of the queues in the namelist failed.
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 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.

The documentation that is in the information center is also provided in PDF files, which are not fully accessible.

For information about the accessibility features of the information center, see Accessibility and keyboard shortcuts in the information center.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.
Appendix B. Reading command-line syntax

This documentation uses special characters to define the command-line syntax.

The following special characters define the command-line syntax:

- **[]** Identifies an optional argument. Arguments that are not enclosed in brackets are required.
- **...** Indicates that you can specify multiple values for the previous argument.
- **|** Indicates mutually exclusive information. You can use the argument to the left of the separator or the argument to the right of the separator. You cannot use both arguments in a single use of the command.
- **{}** Delimits a set of mutually exclusive arguments when one of the arguments is required. If the arguments are optional, they are enclosed in brackets ([ ]).

**Note:**
- The maximum number of characters in an argument is 256.
- Enclose argument values that have embedded spaces with either single or double quotation marks.

For example:

```bash
wsetsrc [-S server] [-l label] [-n name] source
```

The `source` argument is the only required argument for the `wsetsrc` command. The brackets around the other arguments indicate that these arguments are optional.

```bash
wlsac [-l | -f format] [key...] profile
```

In this example, the `-l` and `-f format` arguments are mutually exclusive and optional. The `profile` argument is required. The `key` argument is optional. The ellipsis (...) that follows the `key` argument indicates that you can specify multiple key names.

```bash
wrb -import {rule_pack | rule_set}...
```

In this example, the `rule_pack` and `rule_set` arguments are mutually exclusive, but one of the arguments must be specified. Also, the ellipsis marks (...) indicate that you can specify multiple rule packs or rule sets.
Appendix C. How to read syntax diagrams

The following rules apply to the syntax diagrams that are used in this information:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).


- Optional items appear below the main path.


- If you can choose from two or more items, they appear vertically, in a stack. If you must choose one of the items, one item of the stack appears on the main path.


- If choosing one of the items is optional, the entire stack appears below the main path.


- An arrow returning to the left, above the main line, indicates an item that can be repeated.
If the repeat arrow contains a comma, you must separate repeated items with a comma.

A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.

**Fragment-name:**

- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown.
- Variables appear in all lowercase italic letters (for example, column-name). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).
Appendix D. 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 16. 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/</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/software/sw-training/">http://www.ibm.com/software/sw-training/</a></td>
</tr>
</tbody>
</table>
Appendix E. 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 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://pic.dhe.ibm.com/infocenter/iisinfsv/v9r1/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**

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 comment on the information center, click the Feedback link on the top right side of any topic in the information center.
- 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.
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
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.

**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, LACS<sup>link</sup>, ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS and United States Postal Service. IBM Corporation is a non-exclusive DPV and LACS<sup>link</sup> licensee of the United States Postal Service.

Other company, product or service names may be trademarks or service marks of others.
error queues (continued)
    using with input links or reject links 55
    using with output links 55
examples
WebSphere MQ stages 139
executing a queue commit 140
expiry of messages
WebSphere MQ stages 144
Expiry property 84

F
Feedback property 85
    Custom value property 79
    System value 122
Filter messages property 85
filtering
WebSphere MQ stages 151
format headers 60
Format property 86
    Custom value property 80
    System value 123

G
General options property for the
Deregistration property 86
General options property for the
Registration property 87
    General tab
Input page
WebSphere MQ stages 144
Output page
WebSphere MQ stages 148
Stage page
WebSphere MQ stages 143
Group ID property 88
    Hex 88
    Value 129

H
Header version property 88
Hex property for the Accounting token
property 88
Hex property for the Alternate security
ID property 88
Hex property for the Correlation ID
property 88
Hex property for the Group ID
property 88
Hex property for the Message ID
property 89

I
Identity options property 89
IDS 14
    infinite loops
WebSphere MQ stages 140
Informix
    configuring access 14
Informix CLI 14
Informix Databases
    configuring access 14
Informix Dynamic server 14
Informix Extended Parallel Server 14
Informix IDS
    configuring access 15
Informix Load 14
Informix XPS 14
input links
    specifying dynamic queues 59
    using with reject links 55
Input page
    Columns tab
        WebSphere MQ stages 145
        description
        WebSphere MQ stages 142, 144
        General tab
        WebSphere MQ stages 144
        Transaction Handling tab
        WebSphere MQ stages 146
    installation prerequisites
    publish/subscribe 68, 108
    WebSphere MQ connector 51

J
    jobs 1
        designing the WebSphere MQ
        connector 53
    migrating to use connectors 2, 4

K
    Key column property 90

L
    legal notices 169
    length of data, rules
        WebSphere MQ stages 155
    Library path environment
        configuring 9
        link execution, order of
        WebSphere MQ stages 144
    links 55
    Linux 19
    logical messages 62

M
    Match all property 90
    MCAUSER attribute
        using 27
    Message content descriptor property 91
        Message type 97
    Message conversion property 91
        Coded character set ID property 76
        Encoding 83
        message descriptor fields
        WebSphere MQ stages 155
    Message flags property 91
        Value 130
        message groups 62
        message header fields
        setting values 40
message headers 60
Message ID property 92
    Hex 89
    Value 130
Message options property 92
Message order and assembly
    property 92
Message padding property 93
message payloads 60
Message quantity property 94
Message read mode property 94
Message sequence number property 94
Message sequence number property for
    the Publication settings property 95
Message service domain property 95
Message set property 96
Message Tracking Options tab
    WebSphere MQ stages 152
Message truncation property 96
Message type property 96
    Custom value property 80
    System value 123
Message type property for the Message
    content descriptor property 97
Message write mode property 97
messages 60
    components 63
datagram 62
details of
    WebSphere MQ stages 137
displaying 43
    expiry of
    WebSphere MQ stages 144
    filtering 38, 60
    groups of 62
    limit of
    WebSphere MQ stages 148
    limits of
    WebSphere MQ stages 140
    parts of 62
    passing by reference 37
    persistence of
    WebSphere MQ stages 144
    priority
    WebSphere MQ stages 144
    queues for
    WebSphere MQ stages 136
    reading 36, 38
    receiving from the queue 63
    reply 61
    reply and report
    WebSphere MQ stages 141
    report 61
    rows in
    WebSphere MQ stages 137
    sending to the queue 63
    types of 61
    WebSphere MQ stages 144
    writing 38, 39
    WebSphere MQ stages 140
metadata 67
migrating to use connectors 1
Mode property 100
MQCHLLIB environment variable
    using 35
MQCHLTAB environment variable
    using 35
MQRFH
installation requirements 68, 108
MQRFH2
installation requirements 68, 108
MQSERVER environment variable
using 35

N
name lists 52
authorizations for 38
using 38
WebSphere MQ stages 136
Netezza ODBC driver
configuring 19
NLS tab
WebSphere MQ stages 142, 143
not on palette 6

O
ODBC databases
setting up connectivity 13
ODBC driver
configuring 19, 20
ODBC driver on Window
configuring 20
Offset property 102
Oracle databases
configuring 11
order of link execution
WebSphere MQ stages 144
Original length property 103
Other queue settings property 104
Transmission queue 126
Output page
Columns tab
WebSphere MQ stages 154
description
WebSphere MQ stages 142, 147
General tab
WebSphere MQ stages 148
Message Tracking Options tab
WebSphere MQ stages 152

P
palette
displaying stages 6
parallel engine
connecting to ODBC databases 13
connecting to Teradata databases 18
Password property 104
payload 104
Payload size property 104
Period property 105
permissions
WebSphere MQ stages 142
persistence of messages
WebSphere MQ stages 144
Persistence options property 106
Persistence property 105
Physical format property 106
priority of messages
WebSphere MQ stages 144
Priority property 106

Q
queue manager connection
configuring connector access 57
testing connector access 58
queue manager names
displaying 43
Queue manager property 112
Queue manager property for the Cluster
queue property 111
Queue manager property for the Error
queue property 112
queue managers
client mode location 51
clusters 52
configuration of
WebSphere MQ stages 141
connecting to 41
WebSphere MQ stages 143
description
WebSphere MQ stages 136
server mode location 51
Queue name property 112
Queue name property for the Dynamic
queue property 113
Queue name property for the Dynamic
reply queue property 114
Queue name property for the Error queue
property 115
queue names
displaying 42
queues 52
alias 28
authorizations for 38, 39
details of
WebSphere MQ stages 137
dynamic 39
executing a commit
WebSphere MQ stages 140
model 39
name for
WebSphere MQ stages 136
remote 28
types of 56

R
Refresh property 116
registration
configuring 69
Registration options property 116
Registration property 116
Correlation ID property 79
General options 87
Topic 125
reject 55
reject links
using with input links 55
using with request links 55
Reject tab
relationship to error queue 55
remote queue definition
WebSphere MQ stages 141
Remove MQRFH2 header property 116
reply messages 123
Reply queue close options property 117
Reply queue property 116
Reply to queue manager property 118
Reply to queue property 117
report messages 123
Report property 119
Value 131
request and reply scenario 64
configuring 65
request links
using with reject links 55
request messages 123
request/reply mode
configuring 31
Row buffer count property 119
row length, defining
WebSphere MQ stages 137
rows in a message
WebSphere MQ stages 137
rules, length of data
WebSphere MQ stages 155

S
schemas 63
security
WebSphere MQ stages 141
Segment size property 120
Segmented message property 120
segments 62
SELECT privileges 9
server connection mode
installation requirements 51
server mode
queue manager location 51
Server mode
server-connection channel
definition 27
WebSphere MQ connector 27
Service type property 120
Set header fields property 120
setmqaut command
using 25
setmqaut utility
WebSphere MQ stages 142
setting environment variables for
databases
setting 21, 22
shared cluster queues
using 29
software services
contacting 165
Sort order property 121
source queues 56
special characters
in command-line syntax 161
stage not on palette 6
Stage page
description
WebSphere MQ stages 142
General tab
WebSphere MQ stages 143
NLS tab
WebSphere MQ stages 142, 143
stages
adding to palette 6
staging area
WebSphere MQ stages 140, 141
Start value property 121
Stream name property 121
streams 67
subscribers 67
Subscription identity property 121
Subscription name property 122
Subscription point property 122
subscriptions
configuring 69
support
customer 165
synchronous message processing
configuring 65
synchronous processing 64
syntax
command-line 161
System value property for the Feedback
property 122
System value property for the Format
property 123
System value property for the Message
type property 123
System value property for the Publication
format property 124
System value property for the Put
application type property 124
T
target queues 56
Teradata databases
setting up connectivity 18
Timestamp property 124
Topic property for the Deregistration
property 125
Topic property for the Publication
settings property 125
Topic property for the Registration
property 125
topics 67
trademarks
list of 169
transaction control
WebSphere MQ stages 146
transaction groups
WebSphere MQ stages 146
Transaction Handling tab
Input page
WebSphere MQ stages 146
Transmission queue property for the
Error queue property 125
Transmission queue property for the
Other queue settings property 126
transmission queues 56
Transport type property 126
Treat EOL as row terminator
property 126
troubleshooting
WebSphere MQ connector 49
WebSphere MQ stages 140
Types of messages
WebSphere MQ stages 144
U
unit of work
WebSphere MQ stages 136, 138
Use wildcard property 126
User ID property 126
Username property 127
V
Value property for the Accounting token
property 127
Value property for the Alternate security
ID property 128
Value property for the Correlation ID
property 128
Value property for the Group ID
property 129
Value property for the Message flags
property 130
Value property for the Message ID
property 130
Value property for the Report
property 131
W
wait time
WebSphere MQ stages 148
Wait time property 131
web sites
non-IBM 163
WebSphere MQ connector
alias queues, using 28
alternate user ID, configuring 41
authorization requirements 25
authorizations, required 47
Client mode 27
client-connection channel
definition 27
cursor refreshing 37
error queues, using 30
filtering messages 38
jobs that use Distributed Transaction
stage 44
MCAUSER attribute 27
messages, displaying 43
MQI channel 27
name and password, specifying 36
passing data by reference 37
publish/subscribe mode,
configuring 32
queue manager names, displaying 43
queue managers, connecting to 41
queue names, displaying 42
reading messages 36
remote queues, using 28
request/reply mode, configuring 31
security exit 27
Server mode 27
server-connection channel
definition 27
setmqaut command
user IDs and 26
shared cluster queues, using 29
specifying client-connection channel
definition 35
user IDs 46
user IDs and 26
writing messages 38, 39
WebSphere MQ connectors
configuring as a source 54
configuring as a target 55
configuring client-connection channel
definitions 52
configuring queue manager
connection 57
designing jobs 53
metadata 67
testing queue manager connection 58
WebSphere MQ stages 140
browse mode 140
browsing the queue 136, 138, 147
client only version 135
client/server version 135
clusters 141
column definitions 137
Columns tab
Input page 145
Output page 154
complex message formats 137
collection parameters 124
destructive read 149
designing jobs 140
dynamic processing 140
expiry of messages 144
filtering 151
WebSphere MQ stages (continued)  
functionality 133  
General tab  
  Input page 144  
  Output page 148  
  Stage page 143  
infinite loops 140  
Input page 142, 144  
  General tab 144  
  Transaction Handling tab 146  
  length of data, rules 155  
link execution, order of 144  
message descriptor fields 155  
Message Tracking Options tab 152  
messages  
  details of 137  
  expiry of 144  
  limit of 148  
  limits of 140  
  persistence of 144  
  priority 144  
  queues for 136  
  reply and report 141  
  rows in 137  
  types of 144  
  writing 140  
namelists 136  
NLS tab 142, 143  
  order of link execution 144  
  Output page 142, 147  
  Columns tab 154  
  General tab 148  
  Message Tracking Options tab 152  
permissions 142  
persistence of messages 144  
priority of messages 144  
publish/subscribe 135  
queue managers 136  
  configuration of 141  
  connecting to 143  
queues  
  details of 137  
  executing a commit 140  
  name for 136  
  remote queue definition 141  
  row length, defining 137  
  rows in a message 137  
  rules, length of data 155  
security 141  
setmqaut utility 142  
Stage page 142  
  General tab 143  
  NLS tab 142, 143  
staging area 140, 141  
terminology 134  
transaction control 136, 146  
transaction groups 146  
Transaction Handling tab  
  Input page 146  
  troubleshooting 140  
types of messages 144  
  unit of work 136, 138  
  wait time 148  
  writing messages 140  
XML format 137  
writing messages  
  WebSphere MQ stages 140  
  WSMQ.QUEUENAME data element 38  
XML format 137  

X  

WebSphere MQ stages 140