TIBCO ActiveMatrix BusinessWorks™ Plug-in for WebSphere MQ

User's Guide

Software Release 7.2
May 2013

Two-Second Advantage™

Important Information

Contents

Figures

Tables

Preface

TIBCO ActiveMatrix BusinessWorks™ Plug-in for WebSphere MQ, referred to as the *Plug-in* in this manual, allows the user to design and implement ActiveMatrix BusinessWorks applications that exchange messages with other WebSphere MQ-capable applications across a wide range of hardware and operating systems.

This Plug-in, together with TIBCO ActiveMatrix BusinessWorks Plug-in for Data Conversion, provides ActiveMatrix BusinessWorks with the ability to encode XML-structured data into byte sequences to be sent to a WebSphere MQ-capable application. These byte sequences can populate a memory range within the application and then be interpreted by the application as variable values, as defined by the application programming language (COBOL, RPG or PL/I).

Conversely, the value of variables occupying a memory range within the WebSphere MQ-capable application may be sent to an ActiveMatrix BusinessWorks as a byte sequence, and from there be decoded into XML-structured data for further use.

Topics

Changes from the Previous Release of this Guide

This section itemizes the major changes from the previous release of this guide.

Feature Enhancements

The following enhancements are documented in the sections shown:

- **Supporting Multi-instance Queue Managers** The connection resource in the Plug-in's Connection Table has been enhanced to support using WebSphere MQ Client Channel Definition Tables (CCDTs) for making connections to a queue manager or multi-instance queue manager. All that is required is a working CCDT that can be accessed using the HTTP, FTP, or FILE protocols. CCDTs are the only supported way to connect to a multi-instance queue manager.

  For more information, see WebSphere MQ Connection Shared Resource on page 26. Additionally, see the IBM documentation for a description of CCDTs and how they are to be used.

  Refer to IBM WebSphere MQ documentation regarding High Availability and multi-instance queue managers for configuration and related information.

- **New Application Context Fields** Many new fields related to the application context are exposed on the input and output schemas for all activities. The addition of these fields is facilitated by destination and message option `MQOO_SET_ALL_CONTEXT`. This is now set for all put and publish activities and is not configurable.

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ Documentation

The following documents form the TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ documentation set:

- *TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ Installation* Read this manual for instructions on site preparation and installation.
- *TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ User's Guide* Read this manual for instructions on using the product on all platforms.
- *TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Product Documentation

You may find it useful to read the documentation for the following TIBCO products:

- **TIBCO ActiveMatrix BusinessWorks™ Plug-in for Data Conversion** software: This product is a prerequisite for working with TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ.
- **TIBCO ActiveMatrix BusinessWorks™** software:
  - — *Concepts* Read this manual first. It describes the terminology and concepts of ActiveMatrix BusinessWorks. The other manuals in the documentation set assume familiarity with the information in this manual.
  - — *Getting Started* This manual steps you through a simple example of designing, deploying, and monitoring an ActiveMatrix BusinessWorks process.
  - — *Process Design Guide* This manual describes how to create, edit, and test business processes with ActiveMatrix BusinessWorks.
  - — *Administration* This manual describes how to deploy, manage, and monitor ActiveMatrix BusinessWorks processes with TIBCO Administrator™.
  - — *Palette Reference* This manual describes the palettes in ActiveMatrix BusinessWorks.

  - *Installation* This manual shows you how to install one or more components of ActiveMatrix BusinessWorks and how to set up an ActiveMatrix BusinessWorks domain.
  - *Error Codes* This manual describes the errors returned by ActiveMatrix BusinessWorks.
  - *Release Notes* This document describes the new and changed features as well as the closed and known issues in this release.
- **TIBCO Designer™** software: TIBCO Designer is an intuitive graphical user interface for design-time configuration of TIBCO applications. Online help is available for TIBCO Designer palettes.
- **TIBCO Administrator** software: TIBCO Administrator is the monitoring and managing interface for such new-generation TIBCO products as ActiveMatrix BusinessWorks.

Third-Party Documentation

TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ relates to other products, whose documentation you might find helpful:

- WebSphere MQ description and documentation, which may be found at:

  http://www-01.ibm.com/software/integration/wmq/

- WebSphere MQ for z/OS description and documentation, which may be found at:

  http://www-01.ibm.com/software/integration/wmq/zos/

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
ENV_NAME *TIBCO_HOME*	TIBCO products are installed into an installation environment. A product installed into an installation environment does not access components in other installation environments. Incompatible products and multiple instances of the same product must be installed into different installation environments. An installation environment consists of the following properties: • **Name** Identifies the installation environment. This name is referenced in documentation as *ENV_NAME*. On Microsoft Windows, the name is appended to the name of Windows services created by the installer and is a component of the path to the product shortcut in the Windows Start > All Programs menu. • **Path** The folder into which the product is installed. This folder is referenced in documentation as *TIBCO_HOME*.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [`**`enable`**` | disable]`

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
📝	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
💡	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
⚠	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical OR that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand para1 | param2 | param3`

Table 2 Syntax Typographical Conventions

Convention	Use

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	
	`MyCommand {param1 param2} | {param3 param4}`
	
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	
	`MyCommand {param1 | param2} {param3 | param4}`
	
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	
	`MyCommand param1 [param2] {param3 | param4}`

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support as follows:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 Introduction

This chapter provides an overview of the product TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ, referred to as the Plug-in in this manual. It discusses the product's main features and describes how the various components are designed.

Topics

Product Overview

The Plug-in allows the user to access WebSphere MQ queues and messages with WebSphere MQ applications. While it is possible to exchange free form text data using WebSphere MQ, the Plug-in is specifically intended to be used with the TIBCO ActiveMatrix BusinessWorks Plug-in for Data Conversion. The Plug-in for Data Conversion allows TIBCO ActiveMatrix BusinessWorks applications to use IBM standard copybooks to interface with legacy applications using several transport mechanisms such as this Plug-in.

The Plug-in supports the following features:

- WebSphere MQ 7.x and upward releases.
- A Connection resource that contains all parameters relevant to establishing a connection, including automatic re-connectivity.
- Connections can be pooled and secured with TLS.
- A WebSphere MQ Get activity that gets a message from a queue and returns the message as a byte array.
- A WebSphere MQ Put activity that puts message to a queue. Payloads are binary blobs.
- A WebSphere MQ Listener activity that gets a message from a queue as a process starter. Each process started may access the message read as a byte array. Optionally, the message can be explicitly acknowledged from within the new process. If the message is not acknowledged, it is placed back on the queue.
- The WebSphere MQ Publisher/Subscriber activities perform topic publish and subscribe functions using messages, as described above.
- Handles the following message types:
  - Datagram
  - Request
  - Reply
  - Report (COA, COD, Expiration and Exception types are supported)
- Filters on correlation and message identifiers.
- XA transactions for get and put activities.
- Group messages and segmented message groups.
- Dynamic queues.

- WebSphere MQ message properties.

WebSphere MQ Messages

WebSphere MQ messages consist of three basic sections:

- A header containing fields.
- Message properties that are collection of property names and property values.
- The message body.

The Plug-in uses different mechanisms to provide access to these sections.

Message Headers

Get, put, and listener activities provide access to WebSphere MQ message headers using the input and output schemas for an activity. The root name `mqproperties` contains the fields of message headers. Field names used in `mqproperties` generally map to message header field names used in the WebSphere MQ interface for Java. Only applicable fields are exposed in the input and output schema. For example, the field `accountingToken` is not an input schema field but is an output schema field for the get activity.

Message Properties

Message properties are text strings that can be sent with a put activity to a destination, or can be received from a destination for get and listen activities. Values for a property are strings, and honor the encoding of the message.

Message Body

The message body is exchanged as byte arrays for get, put, and listener activities. If the payload must be a string, you must use the ActiveMatrix BusinessWorks XPATH functions to convert the data:

- `tib:base64-to-string` converts a byte array payload to a string.
- `tib:string-to-base64` converts a string to a byte array payload.

Both functions allow a second parameter that can be used to specify the encoding to be used when converting to and from strings. ActiveMatrix BusinessWorks provides Parse XML and Render XML activities that can consume binary content and produce binary message bodies.

WebSphere MQ message payloads are often complex combinations of string, integer, and other fields types. It is recommended that the Plug-in for Data Conversion be used to parse and render such complex message bodies.

Chapter 2 Working with the Plug-in

This chapter describes topics particular to the TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ.

Topics

Transactions with WebSphere MQ

If a WebSphere MQ put or get activity appears within the context of an XA transaction group, it participates in the XA transaction. Resources committed or rolled back at the end of the transaction include those of the WebSphere MQ activities in the group.

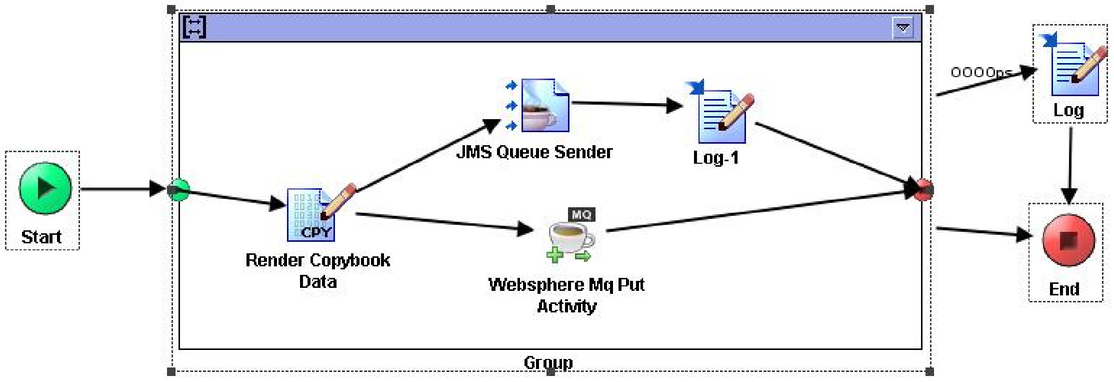
Figure 1 depicts a scenario where two XA Transaction aware activities exist within an XA Transaction group. In the case of the WebSphere MQ Plug-in, the resource manager enlisted on the XA Transaction is the WebSphere MQ Queue Manager which allows it to participate as a full member in the distributed transaction, thus ensuring the integrity of both the WebSphere MQ queue and the state of any other XA resource coordinated by this transaction. Thus, the access to a WebSphere MQ queue, a database and a JMS queue can all be synchronized through their participation in this distributed XA Transaction.

This scenario assumes that the client API used by the plug-in is an Extended Transactional Client (ETC). ETC is the default client delivered in all 7.5 client installations and is a free of charge option for older versions (bearing in mind that the Plug-in requires a 7.x client installation).

Please refer to the IBM support statement entitled "How do I download the Extended Transactional Client (XA)" at:

http://www-01.ibm.com/support/docview.wss?uid=swg21605388

Figure 1 Transactions



Due to limitations in WebSphere MQ, a transaction can only reference one WebSphere MQ queue manager. The queue manager and the queues opened using get and put activities will remain open for the life of the transaction. They are automatically closed and released when the transaction ends.

Using the Data Conversion Plug-in with WebSphere MQ

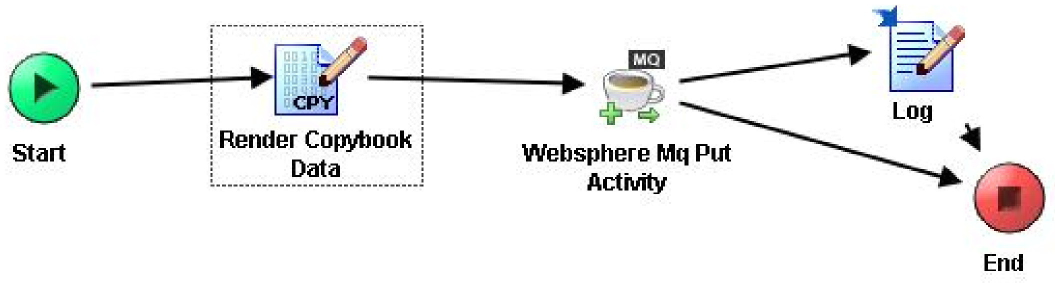
The TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ does not attempt to provide tools for the manipulation of the message body itself. This requirement is fulfilled by the Plug-in for Data Conversion. Messages can be free form text (like XML documents) or they can have internal structure. In the case of most legacy applications, that structure is usually expressed in the form of a copybook document like this one:

```
*   *===========================================================*
*   * TIBCO Mainframe sample COBOL Copybook                     *
*   *===========================================================*

    01  WS-IVP-FIELDS.
        05  IVP-ID                 PIC  X(8)      USAGE DISPLAY.
        05  IVP-DESCR              PIC  X(56)     USAGE DISPLAY.
        05  IVP-RESULT             PIC  X(80)     USAGE DISPLAY.
        05  IVP-BES-NAME           PIC  X(8)      USAGE DISPLAY.
        05  IVP-BES-TRAN-ID        PIC  X(8)      USAGE DISPLAY.
        05  IVP-BES-PGM-NAME       PIC  X(8)      USAGE DISPLAY.
        05  IVP-USERID             PIC  X(8)      USAGE DISPLAY.
        05  IVP-START-TIME         PIC  X(8)      USAGE DISPLAY.
        05  IVP-REPLY-TIME         PIC  X(8)      USAGE DISPLAY.
        05  IVP-SUBJ-SFX           PIC  X(30)     USAGE DISPLAY.
        05  IVP-TYPE               PIC  X(1)      USAGE DISPLAY.
        05  IVP-BES                PIC  X(1)      USAGE DISPLAY.
        05  IVP-WRITES             PIC S9(8)      USAGE COMP.
        05  IVP-BES-TASKNO         PIC S9(8)      USAGE COMP.
        05  IVP-BES-DELAY          PIC S9(8)      USAGE COMP.
        05  IVP-BTCH-SEQ-NO        PIC S9(8)      USAGE COMP.
        05  IVP-START-DATE-CHAR    PIC  X(16)     USAGE DISPLAY.
        05  IVP-START-TIME-CHAR    PIC  X(16)     USAGE DISPLAY.
        05  IVP-REPLY-DATE-CHAR    PIC  X(16)     USAGE DISPLAY.
        05  IVP-REPLY-TIME-CHAR    PIC  X(16)     USAGE DISPLAY.
```

The Data Conversion Plug-in converts copybooks to XML schemas for use in other activities. The Data Conversion Plug-in render and parse activities convert canonical data to memory based formats and vice-versa. If your WebSphere MQ project converses with an application that uses these formats, simply create Data Conversion Plug-in activities that represent the message structure for those applications and pipe the values in and out of message activities to communicate with them.

Figure 2 Using the Data Conversion Plug-in to Convert Canonical Data



Creating Secure Connections to the Queue Manager

You can create a secure connection between the Plug-in and the WebSphere MQ queue manager using Transport Layer Security (TLS).

To enable secure connections, you must:

1. Configure the WebSphere MQ queue manager to accept TLS connections, and create a keystore for the certificates and trusts used by the queue manager. This step is described in WebSphere MQ Queue Manager Setup below.

   The WebSphere MQ queue manager configuration is covered extensively in the IBM documentation. If any of the terms used in this section are unfamiliar, review the IBM documentation for clarification.

2. Configure TIBCO ActiveMatrix BusinessWorks to connect using TLS. This step is described in TIBCO ActiveMatrix BusinessWorks Setup on page 16.

Note that there are many variables that control secure connections, and all must be correctly enabled for TLS to function. If problems occur, it may be difficult to identify the source. The section Troubleshooting on page 18 offers a few suggestions to aid in identifying the source of any problems.

WebSphere MQ Queue Manager Setup

To enable the WebSphere MQ queue manager to accept TLS connections, you must create a keystore for the certificates and trusts used by the queue manager.

These examples use self-signed certificates. Although self-signed certificates are acceptable in a design environment, they should never be used in production.

Task A Create IBM keystores for the queue manager and client.

Each end of the TLS connection must have a keystore.

A sample script is included in the Plug-in's `samples` directory. This script can be used to create keystores for the queue manager and client. The script is annotated, and is intended to be used as template to aid you in the creation of your own scripts.

Use this script to assist in the creation of TLS connections in an evaluation or development environment only. Because all the certificates created by the script are self-signed, they are inappropriate for production use.

Subsequent steps in this sample setup process rely on certificates generated using this script.

Versions of the script are available for UNIX and Windows. The scripts are:

UNIX *TIBCO_HOME*`/bw/plugins/mq/samples/createQueueManagerKeystore.sh`

Windows *TIBCO_HOME*`\bw\plugins\mq\samples\sslcert.bat`

The objective of each script is to produce two IBM CMS type keystores, each containing an identity and the other's signing CA certificate. That way, a TLS connection can be instantiated using one keystore at each end. For example, on a unix system running the command:

`createQueueManagerKeystore.sh` *keystordir qm-name keystore-password*

where:

- *keystoredir* is the directory in which to create the keystores. This can be a working directory or the SSL directory in the queue managers data directory, so long as the queue manager's keystore is eventually placed in the location configured in its SSL parameters.
- *qm-name* is the name of the queue manager for which the keystore is being created. This name must use lower case characters only.
- *keystore-password* is the password used to secure the keystore.

Running this script generates a number of files, most importantly:

- *qm-name*`.sth` — is the stash file for the queue managers key store.
- *qm-name*`.kdb` — is the IBM CMS format keystore for the queue manager.
- `client.kdb` — is the IBM CMS format keystore to be used to make a Java keystore for the client.

You may either customize the scripts to produce keystores that suit your environment's standards, or enter the commands discretely to accomplish the same thing.

Example The following steps describe the configuration of the queue manager to use certificates generated by the script using the command:

```
createQueueManagerKeystore.sh /var/mqm/qmgrs/qmwn/ssl qmwn
password
```

Task B Configure the queue manager to use generated keystore.

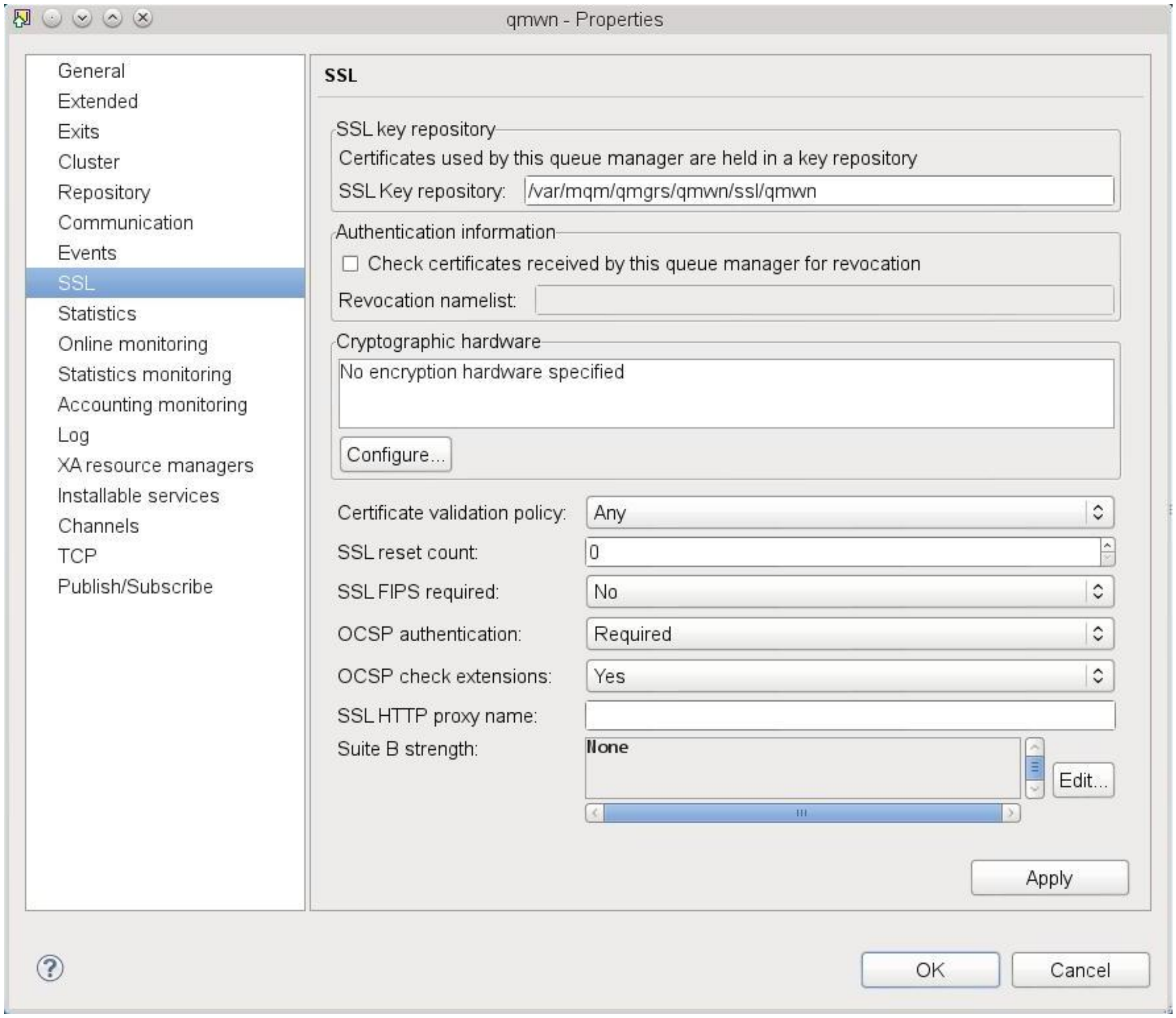
After the script has created the keystores, you must configure the queue manager to use the generated keystore.

In this example, the keystore is named `qmwn.kdb` and the queue manger name was `qmwn`. To configure the queue manager, set its SSL Key Repository parameter to this file without the suffix.

The keystore created with the Example above is `/var/mqm/qmgrs/qmwn/ssl/qmwn.kdb`. The value for the corresponding key database for the queue manager would be `/var/mqm/qmgrs/qmwn/ssl/qmwn`.

Figure 3 shows an example of the IBM WebSphere MQ Explorer queue manager properties for SSL.

Figure 3 WebSphere MQ



SSL

SSL key repository

Certificates used by this queue manager are held in a key repository

SSL Key repository: /var/mqm/qmgrs/qmwn/ssl/qmwn

Authentication information

☐ Check certificates received by this queue manager for revocation

Revocation namelist:

Cryptographic hardware

No encryption hardware specified

Certificate validation policy: Any

SSL reset count: 0

SSL FIPS required: No

OCSP authentication: Required

OCSP check extensions: Yes

SSL HTTP proxy name:

Suite B strength: None

Task C Review the Personal Certificate name using the IBM Key Management tool.

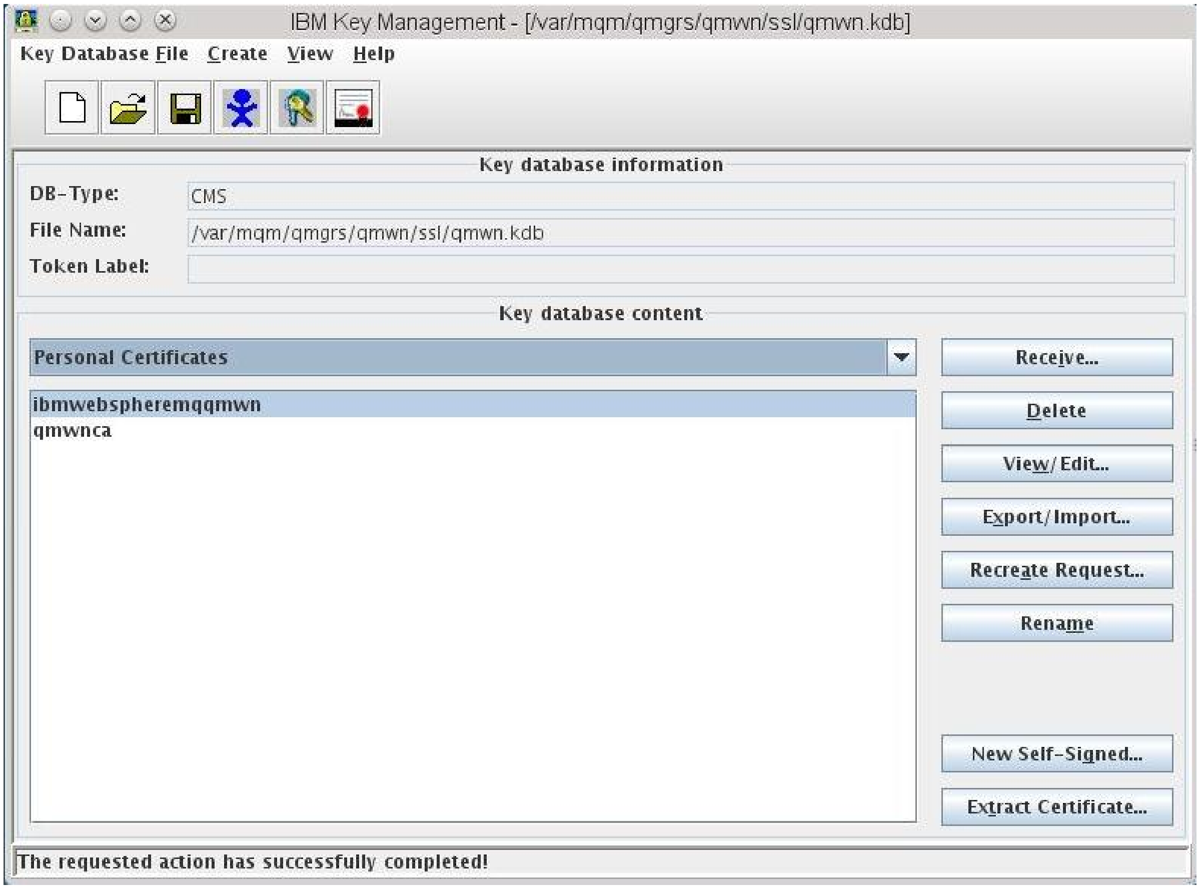
Open the keystore using the IBM Key Management tool. This tool is normally started by the script in `mqm/bin/strmqikm`, and the executable is called `iKeyMan`.

Figure 4 shows the keystore created in Task A. Note the name of the personal certificate is `ibmwebspheremqqmwn`. This special alias is composed of the tag `ibmwebspheremq` with the name of the queue manager concatenated onto it. In this example, the queue manger name is `qmwn`.

The Personal Certificate name *must* use all lower case characters. If your queue manager name contains upper case characters, rename this alias so that all characters are in lower case. If this alias is not correct, the queue manager will not be able to open the keystore.

Note that the IBM Key Management tool (`iKeyMan`) does not let you simply change an uppercase character to a lowercase character. To retain the same name, you will have to use an intermediate name for the rename function to work.

Figure 4 IBM Key Management Tool: iKeyMan



DB-Type: CMS
File Name: /var/mqm/qmgrs/qmwn/ssl/qmwn.kdb
Token Label:

ibmwebspheremqqmwn
qmwnca

The requested action has successfully completed!

You can also review the CA certificate that was copied using the script to instruct the queue manager to "trust" the client's certificate. To see this information, select **Signer Certificates** from the drop-down box below the **Key database content** heading.

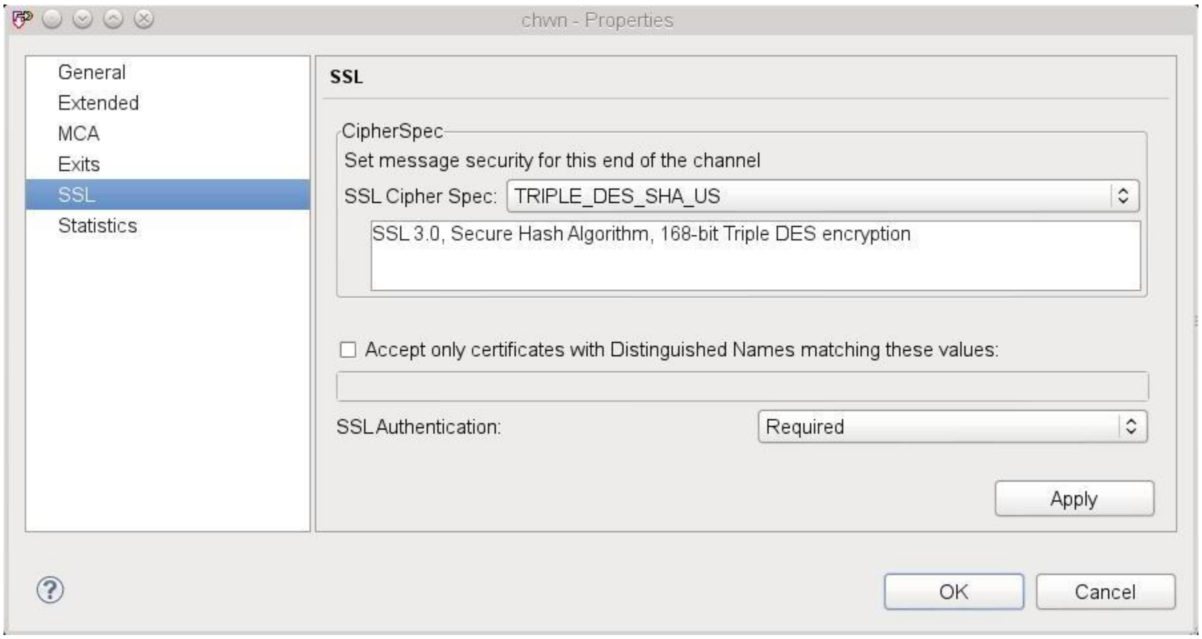
Task D Create a TLS connection for the queue manager.

Using the IBM WebSphere MQ Explorer, create a server connection channel for the new queue manager and set its SSL properties so that client authorization is enabled (if desired) and a cipher suite is selected. For example, TRIPLE_DES_SHA_US is a specification which should have a matching cipher in BusinessWorks.

Note that the terms *SSL* and *TLS* are used interchangeably in the IBM documentation, and most of the parameter settings for TLS are labeled **SSL**. Insofar as this document is concerned, TLS is functionally equivalent to SSL V3.

Figure 5 shows the SSL setting configured using the IBM WebSphere MQ Explorer tool.

Figure 5 SSL Cipher Suite Selection



Set message security for this end of the channel

SSL Cipher Spec: TRIPLE_DES_SHA_US

SSL 3.0, Secure Hash Algorithm, 168-bit Triple DES encryption

☐ Accept only certificates with Distinguished Names matching these values:

SSL Authentication: Required

Task E Verify that all SSL files can be read by the mqm group.

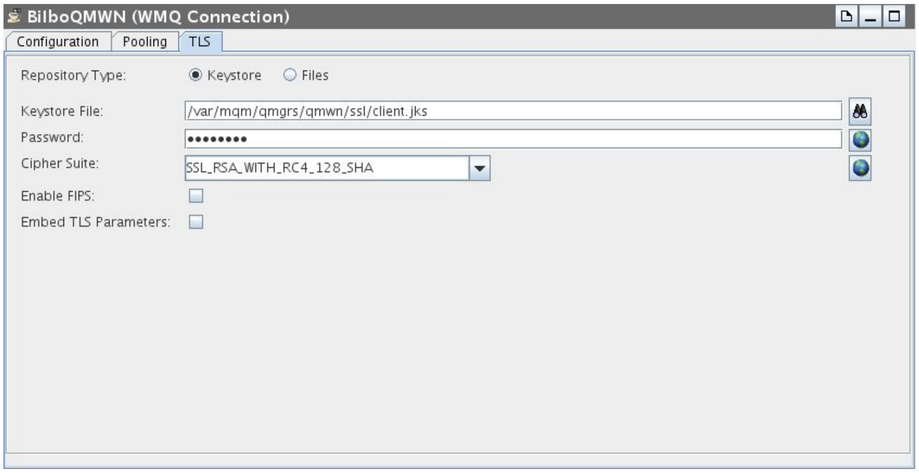
Verify the security of all the files in the directory which you configured as the `SSL Key Repository` property of the queue manager are readable by the `mqm` group. (This is the security group `mqm` that was created during the installation of WebSphere MQ on the platform.) If they are not, adjust their access properties or the queue manager will not be able to read them.

The queue manager is now capable of making a secure connection with a client. Using the Plug-in, you can now create a TLS-secured connection to the queue manager.

TIBCO ActiveMatrix BusinessWorks Setup

Once the queue manager is set up, you can configure the BusinessWorks to use the secure channel:

1. In TIBCO Designer, open an existing project or create a new one.
2. Create a new WebSphere Plug-in for MQ Connection resource.
3. On the Configuration Tab, set up the normal TCP/IP parameters for a remote connection, and check the **TLS Enabled** option.
4. On the Pooling Tab, enable pooling and set reasonable parameters for the pool. Because TLS connections are more resource intensive to establish, there is even more reason to pool these connections.
5. Open the TLS panel:

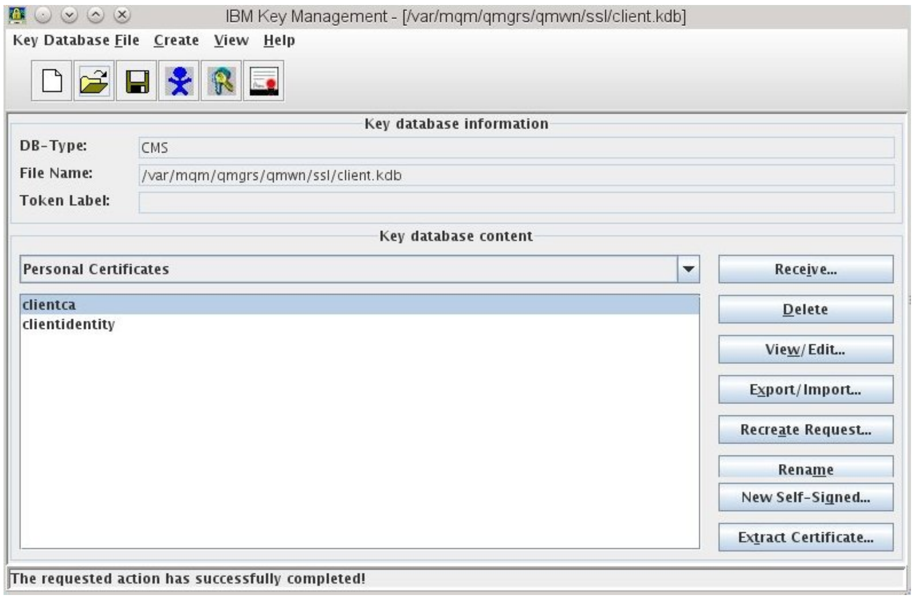


Repository Type: [✓ Keystore] Files
Keystore File: /var/mqm/qmgrs/qmwn/ssl/client.jks
Password: ••••••••
Cipher Suite: SSL_RSA_WITH_RC4_128_SHA
Enable FIPS: ☐
Embed TLS Parameters: ☐

As you can see above, the connection is configured to use a Java keystore file for its credentials. This is the easiest option to configure for use with the IBM Key Manager, and is used in this example. However, you can alternately use an identity, in `pkcs12` format, in combination with a directory of trusted certificates.

6. Start the IBM Key Manger tool using the `strmqikm` program. This program is located in the `mqm/bin` directory in the WebSphere MQ server installation.

7. Using the IBM Key Manager, open the client keystore created by the script during the WebSphere MQ Queue Manager Setup.



Key database information

DB-Type: CMS

File Name: /var/mqm/qmgrs/qmwn/ssl/client.kdb

Token Label:

Key database content

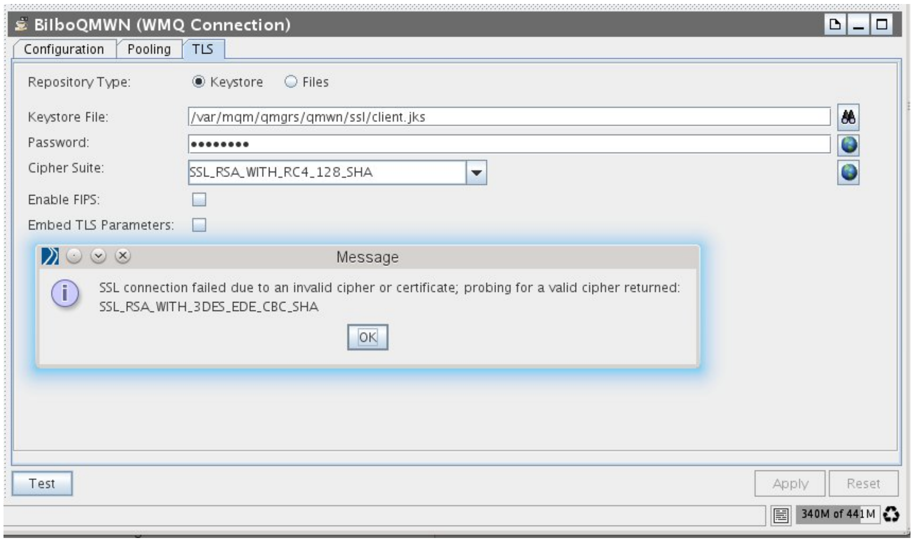
Personal Certificates

clientca
clientidentity

The requested action has successfully completed!

8. Create a Java keystore from the contents of the IBM keystore:
   a. Select the client identity alias and select **Save As** from the File menu.
   b. In the "save as" dialog, change the keystore type to **JKS** and enter a new name for the client keystore. This will be the file you reference from the connection resource, as pictured above.
   c. Set the Key file type to **JKS**.
   d. Click **Save**.
   e. At the prompt, enter a password to be used with the keystore.
9. In TIBCO Designer, navigate to the TLS Tab.
10. In the **Keystore File** field, select the Java keystore just created, and enter the password used to secure it in the **Password** field.
11. In the **Cipher Suite** field, select the cipher `SSL_RSA_WITH_3DES_EDE_CBC_SHA`. This should be the first cipher in the list, and corresponds with the `TRIPPLE_DES_SHA_US` cipher chosen for the queue manager's channel.
12. Test the connection by clicking the **Test** button at the of the Designer panel.

    This test attempts to connect with the queue manager using the parameters specified. If the cipher does not match, the test probes for a correct match from all of the ciphers that the current JRE supports. If a cipher is found, the test prints the cipher in the informational dialog that displays output from the test button. That cipher name will also be printed in the Designer log, from which it can be easily copied.



Repository Type: [✓ Keystore] Files

Keystore File: /var/mqm/qmgrs/qmwn/ssl/client.jks

Password: ••••••••

Cipher Suite: SSL_RSA_WITH_RC4_128_SHA

Enable FIPS: ☐

Embed TLS Parameters: ☐

SSL connection failed due to an invalid cipher or certificate; probing for a valid cipher returned: SSL_RSA_WITH_3DES_EDE_CBC_SHA

Troubleshooting

- Review the TIBCO Designer log file. If there was an error internal to the Plug-in or Designer, look in the Designer log for more information.
- Disable client authentication. This means that the queue manager will not check that it trusts the signer of the plug-in's certificate. If it works with client authentication disabled but not with it enabled, then the two parties do not have all the signing CA certificates in place.
- Enable JSSE debug messages in TIBCO Designer to display messages related to creating the TLS connection:

  ```
  java.property.javax.net.debug   ssl
  ```

  This option enables tracing of all access to certificates as well as the handshake process itself.
- Check the queue manager's log files. These files are in the queue manager's work directory under "errors". For the queue manager above, this file is:

  ```
  /var/mqm/qmgrs/qmwn/errors/AMQERR01.LOG
  ```

A typical error from that file looks like:

```
----- amqrmrsa.c : 889 --------------------------------------------------------
12/07/2012 02:46:09 PM - Process(15433.5) User(jsmith) Program(amqrmppa)
                    Host(bilbo.jrr.org) Installation(Installation1)
                    VRMF(7.5.0.0) QMgr(qmwn)

AMQ9660: SSL key repository: password stash file absent or unusable.

EXPLANATION:
The SSL key repository cannot be used because MQ cannot obtain a password to
access it. Reasons giving rise to this error include:
(a) the key database file and password stash file are not present in the
  location configured for the key repository,
(b) the key database file exists in the correct place but that no password
  stash file has been created for it,
(c) the files are present in the correct place but the userid under which MQ is
  running does not have permission to read them,
(d) one or both of the files are corrupt.

The channel is '????'; in some cases its name cannot be determined and so is
shown as '????'. The channel did not start.
ACTION:
Ensure that the key repository variable is set to where the key database file
is. Ensure that a password stash file has been associated with the key database
file in the same directory, and that the userid under which MQ is running has
read access to both files. If both are already present and readable in the
correct place, delete and recreate them. Restart the channel.
----- amqccisa.c : 5540 -------------------------------------------------------
12/07/2012 02:46:09 PM - Process(15433.5) User(jsmith) Program(amqrmppa)
                    Host(bilbo.jrr.org) Installation(Installation1)
                    VRMF(7.5.0.0) QMgr(qmwn)
```

This error occurred because the queue manager does not have read permission to the files in the SSL directory. This happened because the default file creation mask does not allow group read.

Using the Plug-in with Large Messages

WebSphere MQ cannot accept an arbitrarily large message. It has restrictions based on the size of the buffers in the queue manager and its various channels. There are two techniques to deal with this: Message Groups and Segmented Messages. Both mechanisms use the concept of a "virtual message" composed of many physical member messages all uniquely identified by the same group ID.

Segmented messages are not supported on z/OS.

Sending Long Messages

The `Long Messages` option in the Advanced tab of a put activity allows an application to send payloads that exceed the size of a MQ channel or queue manager buffer. Such message payloads can be automatic broken down into smaller messaged by specifying `Use Message Groups`, or can be programmatically segmented by the application using the specifying `Use Segmented Messages`.

Message Groups

When `Use Message Groups` is specified for the `Long Messages` option, a put activity breaks down the message into smaller messages and puts the each message in sequential order onto the destination queue. The size of the message chunks is defined by the `Maximum Segment Size` option. If the option is set to `0` (zero), then the put activity obtains the maximum message size from the queue manager. The put activity asks the queue manager to automatically generate a group identifier for the group of messages and to maintain message order within the group.

When using message groups on z/OS, your queue definition extended property `Index Type` must be set to `group-id`. If the queue is not indexed by group ID, correct ordering of the results is not guaranteed.

Segmented Messages

When `Use Segmented Messages` is specified for the `Long Messages` option, the ActiveMatrix BusinessWorks application is responsible for sending each message segment of the segment group. Typically, such put activities are within an ActiveMatrix BusinessWorks iterative group and the put activity is invoked within the loop to put each message segment into the destination queue.

The put activity asks the queue manager to generate a group identifier for group of message segments and to maintain message order within the segment group. The ActiveMatrix BusinessWorks application is responsible for identifying the last segment of the segmented message group by setting the `lastSegment` input schema field of the put activity to the string **`true`**.

Note that the segmented message feature is not supported by WebSphere MQ on z/OS.

Receiving Long Messages

A get activity automatically aggregates a grouped or segmented message into a single blob when one is encountered. The ActiveMatrix BusinessWorks application does not need to perform any special action or set any special options. If the message is part of a group, then the message order is honored and the group ID is used to combine the sub-messages into a single payload blob.

However, the ActiveMatrix BusinessWorks application can chose to process each message of a segmented group individually by setting the `Segmented Messages` check box option in a get activity Configuration tab. When set, the `lastSegment` field is present in the output schema for the get activity. After a call to such a get activity, `lastSegment` is set to the string `true` if the message read is the last message of a segment group. Otherwise, the field is set to the string `false`. Typically such get activities are placed within an iteration group and the field `lastSegment` is used to terminate the loop.

Queue Managers and Segmented Messages on Open Systems

In most cases, a queue manager instance is released after a get or put activity completes. However, for get and put activities with an XA transaction, the queue manager is left open until the XA transaction completes. Additionally, the queue manager remains open longer outside a transaction group when:

- In a put activity, `Use Segmented Messages` is specified for the `Long Messages` option.
- In a get activity, the `Segmented Messages` option is checked.

In these circumstances, queue managers are kept open until the last segment of a segment message group is read or written. If segments in a loop are read from one queue and written to another, the same queue manager is shared between the get and put activities. The queue manager is closed and released only after last segment is detected for both the get and the put activities.

Using Message Filters

When reading messages from WebSphere MQ, you can limit which messages are read using message filters. The get and listener Plug-in activities allow you to read only messages that match a specified correlation ID or message ID.

Assigning IDs

These values are assigned to the WebSphere MQ message when it is created. You can specify the desired ID using the `mqproperties` field on the put activity Input tab. However, most applications let the queue manager automatically generate an identifier.

Specifying Filters

Filters can be set for get and listener activities.

- For get activities, the filter can be set using the `correlationId` and `messageId` fields of the Input tab. These fields are grouped under the `mqproperties` field heading.
- For listener activities, the filter is set using the `Correlation Id` and `Message Id` fields on the Advanced tab.

Filter Scenario: Request-Reply

One common use for filters is in a request-reply interaction. In this scenario, a message publisher puts a Request message on the queue that is handled by a service application. The Request message specifies a `Reply To Queue` name, indicating the queue where the requestor will look for the reply message.

Because the same Reply to Queue is often used by many message requestors, the message requester can use the correlation ID generated by the put activity to identify those reply messages that are sent in response to its own request.

To enable this functionality, map the `correlationId` from the Output tab of the put activity to the `correlationId` field on the Input tab of the get activity that will read the reply message.

It is the responsibility of the application replying to the request to set the correct correlation ID in the reply message. The correlation ID is present in the request message header.

Filter Scenario: Reports

Filters can also be used to identify the confirmation of delivery report associated with a put message.

When a put message activity requests a delivery report using the `Reports` field on the Input tab, the queue manager puts a report message onto the specified `Report Reply Queue` queue.

Correlation IDs are used to relate the original put message to the report messages. Using the `Pass Correlation ID` check box, you chose whether the original message ID or correlation ID is used for the report message correlation ID.

As in the previous scenario, the process can pass the `correlationId` shown on the Output tab of the put activity to the `correlationId` field on the Input tab of get activity that will read the report messages.

Chapter 3 WebSphere MQ Palette

This chapter describes the capabilities of the WebSphere MQ palette.

Topics

WebSphere MQ Connection Shared Resource

The connection resource contains all the parameters necessary to connect to a queue manager. It supports these modes:

- **Local** A local connection uses the C language libraries and JNI to locate and connect to queue managers on the local machine.
  - Local connections are not eligible for XA transactions.
  - Local connections require a WebSphere MQ server installation on the local machine.
  - Local connections use the credentials for the logged on user.
- **Remote** A remote connection uses TCP/IP to connect to the listening port on any queue manager on the network or the local machine.
  - Remote connections support XA transactions.
  - Remote connections support the use of specific user credentials.
  - Remote connections support secure transports (TLS/SSL).
- **Client Connection Table** This type of connection uses a Client Channel Definition Table (CCDT) exported by the queue manager upon saving a client channel definition. This is the only supported method to connect to a multi-instance queue manager.
  - Client connection table connections support XA transactions. However, note that transactions are not guaranteed to survive reconnections to the backup server in a failover situation.
  - Client connection table connections support the use of specific user credentials.
  - Client connection table connections support secure transports (TLS/SSL).

  See the IBM documentation for a description of CCDTs and how they are to be used.

Configuration Tab

Table 3 lists and describes the fields under the Configuration tab.

Table 3 Fields on the Configuration Tab: WMQ Connection Resource

Field	Global Variable	Description
`Name`	N	The name of the Region Resource. Default: `WebSphere MQ Connection`
`Description`	N	Description of the resource.
`Binding`	N	`Local`, `Remote`, or `Client Connection Table`. See WebSphere MQ Connection Shared Resource above for more information.
`Client Connection Table URL`	Y	When using a Client Connection Table, enter the URL of the CCDT file that was exported by the server to which you are trying to connect. This file will should be located in the queue manager home directory in the `@ipcc/AMQCLCHL.TAB` file. You can either reference this file from this location, or copy it to a location where it is accessible to the running BusinessWorks instance. This URL supports the HTTP, FTP and FILE protocols.
`Hostname`	Y	For remote connections, provide the name of the machine hosting the queue manager. Default: `localhost` This field is only available for remote bindings.
`Port`	Y	For remote connections, provide the TCP port number on which the queue manager is listening. Default: `1414` Range: `1-65535` This field is only available for remote bindings.
`User Name`	Y	For secure remote connections, the user name which will be used to create the connection. If a user name is not provided, the credentials of the current security context is used.

Table 3 Fields on the Configuration Tab: WMQ Connection Resource

Field	Global Variable	Description
`Password`	N	For remote connections, the password associated with the specified user name. Note that encrypted password fields are not eligible for global variable substitution.
`Queue Manager Name`	Y	For Local and CCT connections, the name of the queue manager to connect to. If not specified, the default queue manager is selected.
`Server Channel Name`	Y	For remote connections, the server channel to connect to. Queue managers can use multiple server channels if needed, for example to support TLS and plain connections.
`TLS Enabled`	N	When checked, the TLS tab is enabled. This option enables the use of Transport Layer Security for the connection resource. Relevant parameters a configured on the TLS Tab. This option is only enabled for `Remote` connections.

Pooling Tab

Table 4 lists and describes the fields under the Pooling tab.

Table 4 Fields on the Pooling Tab: WMQ Connection Resource

Field	Global Variable	Description
`Pooling Enabled`	N	When checked, pooling is active for this connection. The primary consideration for choosing pooling parameters is the number of available connections to the queue manager. Choose values which will not create unnecessary resource consumption in the queue manager and leave available connections for other applications (including other pooled connections, should this application be deployed on multiple servers). If pooling is not active, a new non pooled connection is acquired and released for each activity.

Table 4 Fields on the Pooling Tab: WMQ Connection Resource

Field	Global Variable	Description
`Max Connections`	Y	Determines the maximum number of connections allowed in the pool. When this limit is reached, subsequent activities wait for a connection to become available before starting.
`Maximum Unused`	Y	Determines the maximum number of idle connections allowed in the pool. When the number of unused connections reaches this number, the idle connections are disconnected and closed, freeing resources on the server. Amounts over the `Max Connections` value are ignored.
`Timeout`	Y	The length of time an inactive connection is kept in the pool. A connection that has remained unused for this number of milliseconds are closed and removed from the pool, freeing resources on the server.

TLS Tab

The TLS tab contains the parameters required to make a secure connection to the queue manager using the Transport Layer Security protocol. Before TLS can be used, the queue manager and server channel used by clients must be configured. For more information, see Creating Secure Connections to the Queue Manager on page 10.

Note that TLS panel is enabled only if the `Binding` type is `Remote` and the `TLS Enabled` option is selected on the main Configuration tab.

Table 5 lists and describes the fields under the TLS tab.

Table 5 Fields on the TLS Tab: WMQ Connection Resource

Field	Global Variable	Description
`Repository Type`	N	Specifies the mechanism used to store the identity and trust information for the connection. Options are: • `Keystore` — the TLS certificate and trusted signer certificates are expected to be in a Java keystore file. • `Files` — the TLS certificate is expected to be in a `pkcs12` file and the trusted certificates are located in a directory by themselves. Note that if client authorization is required on the channel to which this Plug-in connects, the CA which signed the Plug-in's certificate must be imported into the server's keystore. See the Creating Secure Connections to the Queue Manager on page 10 section for more information.
`Keystore File`	Y	If the repository type is `Keystore`, this field contains the name of a Java keystore file containing the client certificate and its signer certificate(s). If client authentication is enabled on the server-connection channel, then the CA which signed the queue manager's certificate must be imported here to create a trust relationship.
`Client Identity File`	Y	If the repository type is `Files`, this field that contains the name of a `pkcs12`-formatted certificate with the private key included. This file is secured with a password. `pkcs12`-formatted certificates can be suffixed `pkcs12` or `p12`.
`Trusted Cert Dir`	Y	If the repository type is `Files`, this field contains the directory where all the certificates necessary to trust the server's certificate are located. These certificates can be PEM- or DER-encoded.
`Password`	Y	If the repository type is `Keystore`, this field contains the password used to secure the Java keystore. If the repository type is `Files`, this field contains the password used to secure the identity file.

Table 5   Fields on the TLS Tab: WMQ Connection Resource

Field	Global Variable	Description
`Cipher Suite`	Y	Contains the cipher specification which matches the one selected in the queue manager's server-channel SSL parameters. **Note:** If *all* of the other required fields in this connection resource are correct but the cipher specification is unknown, the **Test** button probes for an acceptable cipher specification and reports it in the pop-up as well as printing it in the log. **Note:** If a client connection table is in use for this connection, the cipher suite is encoded in the table and this value is ignored.
`Enable FIPS`	N	When checked, Federal Information Processing Standards (FIPS) are enabled on platforms where the Java runtime environment supports it. For a FIPS connection to be possible, both the queue manager and the Java runtime environment in which the Plug-in runs must be configured to be FIPS capable. Consult the FIPS documentation to determine which key lengths, algorithms and ciphers are appropriate for your installation's needs.
`Embed TLS Parameters`	N	When checked, the keystore represented by the parameters on this panel is stored in the application's configuration data. This allows the configuration to be deployed to platforms where these files do not exist. The certificates chosen must be valid for all systems to which this application is deployed. If that is not the case, do not select this option and ensure that the files specified are present on all target environments. Global variables may be used to facilitate the necessary flexibility here. Note that, if the parameters here are of the `Files` type and the `Embed TLS Parameters` option is enabled, an internally represented Java keystore will be created from the files and stored within the project's configuration data.

WebSphere MQ Properties Shared Resource

The application properties shared resource provides a mechanism for creating and using application properties on MQ messages. The appearance and function of the resource is identical to the one provided for JMS Properties. The following subset of JMS property types is supported:

- boolean
- byte
- bytes
- short
- int
- long
- float
- double
- string

The node represented by this Application Properties resource will be present on the input schema of put activities and on the output schema of get and listener activities. Fields which contain values are mapped onto sent messages. When processing a received message, all fields in the schema which have corresponding named counterparts in the message are mapped onto the output schema. If a type mismatch occurs when mapping fields onto the output schema, the field is skipped.

If application properties are used on a message, the properties are included in an MQRFH2 header. This header is visible to and must be handled by non-Java applications.

WebSphere MQ Put Activity

The put activity places a message on the configured queue. The activity is capable of sending any of the four message types to permanent or dynamic queues. The capabilities of the activity are determined by the configuration, with some runtime behaviors controlled by the input schema.

The type of message being sent may determine what fields are applicable for the put operation. The four message types are:

- **Datagram** — Basic message.
- **Request** — A message for which the sending application expects a response and for which a reply to queue is required.
- **Reply** — A message sent in response to received Request type message. Usually the destination for this message has been mapped from the `Reply To Queue` field of the request message.
- **Report** — A message containing monitoring data. In most cases, report messages are generated by queue managers. However, it is possible for applications to do so using this message type.

Messages can be sent to predefined queues or dynamic queues. If the destination or the reply to queue is dynamic, you must specify a properly configured model queue. The queue manager constructs the dynamic queue based on that template. The name of a unique dynamic queue can be generated if the queue name provided ends with an `*` character. It makes sense to use such names only if it is possible to configure a receiving application with the name of the newly created queue. This name can be taken from he output schema's `destination` field.

The put activity supports options which request that the queue manager send confirmation reports when the message arrives or is delivered.

Segmented Messages

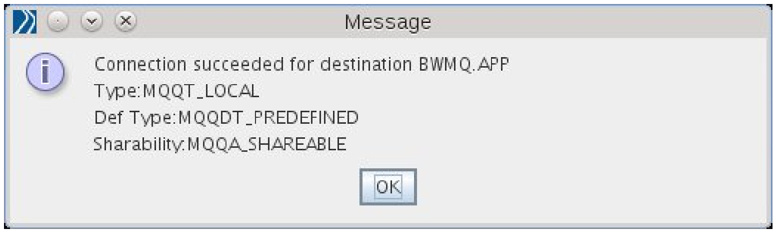
The put activity has the ability to place a group of segmented messages on the queue. (See the `Long Messages` on the Advanced tab.) This feature is most often used in a loop so that the last message has its `lastSegment` field set to `true`. However, if an error prevents the loop from ending and the last message is not correctly written, the queue remains open for output and the message group is incomplete on the queue.

To prevent this, ensure that error recovery in the activity is coded so that the `lastSegment` message is written. It is the responsibility of the application to ensure that segmented message groups are correctly terminated.

Test Button

Each activity includes a test button. The test button for the put activity attempts to validate the selections for queue manager and queue by opening the queue manager and instantiating the queue object. For dynamic queues, only the presence of the model is verified. For predefined queues which were successfully opened, a status box is presented.

For example:



Connection succeeded for destination BWMQ.APP
Type:MQQT_LOCAL
Def Type:MQQDT_PREDEFINED
Sharability:MQQA_SHAREABLE

Configuration Tab

Table 6 lists and describes the fields under the Configuration tab.

Table 6 Fields on the Configuration Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`Name`	N	The name of the Put activity. Default: `Websphere Mq Put Activity`
`Description`	N	Optional free-form description.
`Connection Resource`	Y	Choose the connection resource which associates this activity with the queue manager that has access to the desired queue.

Table 6 Fields on the Configuration Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`Queue Name`	Y	The name of the queue onto which the message will be placed. Queue names can be chosen using the **Choose Queue** button to the right of this field. If the queue is to be created dynamically, enter the name of the dynamic queue. The name must follow the MQSeries rules for dynamic queue names. If the queue name has been dynamically mapped using the input schema's destination field, this field is not required. However, note that the **Test** button cannot verify dynamically mapped queue names.
`Dynamic Queue`	N	A flag that enables the creation of a dynamic queue. When checked, the `Model Queue Name` field is enabled. Dynamic queues are ones which are created as needed and disappear when the lifespan dictated by the model is reached.
`Model Queue Name`	Y	Check the `Dynamic Queue` option to enable this field. Enter the model queue name, or select a name by clicking the **Choose Queue** button. The queue manager uses the specified model queue when creating this dynamic queue.
`Message Type`	N	Select the type of message that is to be sent: • `Datagram` • `Request` • `Reply` • `Report` If a `Request` message is chosen, the `Reply To Queue` field is enabled. All other message types enable the `Reports` field.
`Reply To Queue`	Y	This field is enabled for `Request` message types only. The queue name which will be placed into the message header's reply to queue field. This queue need not exist and is not checked, but can be chosen using the **Choose Queue** button on the right side of the field.

Table 6 Fields on the Configuration Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`Dynamic Reply Queue`	N	A flag that enables the creation of a dynamic queue. When checked, the `Model Queue Name` field is enabled, and the reply queue will be dynamically created based on the model in the `Model Queue Name` field. If a dynamic reply queue is specified, it will be created during the put operation so that it is available to the responding application. This field is available only for `Request` message types.
`Model Queue Name`	Y	Specify the model after which the reply queue is to be patterned. It can be selected form models using the **Choose Queue** button to the right. This field is only available if the above `Dynamic Reply Queue` option is enabled.
`Application Properties`	Y	Select the Application Properties resource to be included as part of this message. Application properties are similar to regular message header fields except that the name and type of the fields can be determined by a predefined schema. To use custom application properties, create an instance of the WebSphere MQ Application Properties sharable resource and map it to this field.

Advanced Tab

Table 7 covers the fields under the Advanced tab.

Table 7 Fields on the Advanced Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`Message Priority`	Y	Set the priority of the message as it is to be retrieved by the queue manager. The values range from `0` to `9`. If not specified, the default is the destination queue's default priority.

Table 7 Fields on the Advanced Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`Message Expiration`	Y	The time, in tenths of a second, that the message will persist on the queue before being deleted by the queue manager. The default, `0`, is to persist forever. Values range from `0` to `2147483647`. The default is `0`, which means the message never expires.
`Generate Correlation Id`	N	Enable this field to cause the queue manager to generate a unique correlation identifier for the message when it is sent. This means that the message identifier, which is always generated, is copied to the correlation identifier field. Note that if this option is enabled, the queue manager overlays any correlation identifier that is mapped on the input schema.

Table 7 Fields on the Advanced Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`Long Messages`	N	Some messages are too big for either the queue manager or the server channel's buffers and cannot be sent in a single message. This field determines how these large messages are handled: • `Use Message Groups` separates the long message into smaller messages and sends them in a group. The group is identified by a unique group ID, and each message in the group is numbered. The final message has a last message indication. For more information about the size of the intermediate messages, refer to the `Maximum Segment Size` field description. • `Use Segmented Messages` sends a group of logically related messages as a message group having the same group ID. These messages are not concatenated in any way, but are intended to be processed as individual parts of a larger unit of work. For these messages, the input schema features a `lastSegment` field which is set to `true` on the last message. This enables the content of the logical message to be processed serially in segments. This is useful on systems which don't have the resources to proeces the entire message at once. Because the `lastMessage` field must be set for the final segment, these messages are almost always processed within the context of a ActiveMatrix BusinessWorks loop activity, where the loop termination sets the `lastMesage` indicator. See the sample application for an example of such a process.
`Maximum Segment Size`	Y	Used when `Long Messages` is set to `Use Message Groups`. Specifies the size a message, in bytes, when a long messages is broken down into smaller messages. If the option is set to zero, then the put activity uses the maximum message length defined for the queue manager. Default: `0`

Table 7 Fields on the Advanced Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`Persistence`	N	Determines whether messages on the queue manager persist across restarts. Select one of the following settings: • `Persistent` ensures that the message persists if the queue manager fails or restarts. • `Not Persistent` discards the message after a restart. • `Persist as queue def` uses the persistence level that is defined in the queue definition.
`Reports`	N	Several kinds of report message can be triggered by a message's arrival a a particular point in the WebSphere MQ system. Select the type of report message and event trigger using the combination box. Events that can trigger a report are: • `Confirm on arrival` • `Confirm on delivery` • `Exception` • `Expiry` Report types are: • `None` The report is not enabled. • `With No Data` The body of the message does not accompany the report. • `With Data` The first 100 bytes of the message body accompanies the report. • `With Full Data` The entire body of the message accompanies the report. Reports are delivered to the reply to queue specified in the `Report Reply Queue` field. Because this field maps to the message's `replyToQueueName` field, reports are mutually exclusive with `Request` type messages. That is, if the message type is `Request`, a report confirmation is not sent.

Table 7 Fields on the Advanced Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`Report Reply Queue`	Y	Enter the name of the queue to which Report messages should be sent. This queue need not exist and is not checked, but can be chosen using the Choose Queue button on the right side of the field. The value entered here maps onto the message's `replyToQueueName` field.
`Pass Correlation ID`	N	Determines the setting for the report message's correlation ID. • When checked, the correlation ID of the sent message is copied to the correlation ID field of the report message. • When not checked, the message ID of the sent message is copied into correlation ID field of the report message.

Input Tab

Table 8 lists and describes the fields under the Input tab.

Table 8 Fields on the Input Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`InteractionInput`	N	This input schema field is the root node for all the input provided to the activity.
`mqproperties`	N	The node containing all the message header fields (plus the destination override) that are relevant on input. • **destination** Specify a destination here in order to override the queue name from the configuration panels. This allows the queue to be dynamically specified. • **destqmgr** Destination queue manager name. The name of the queue manager on which this queue resides. Note that a transmission queue and a sender channel must be configured for this queue manager for it to work.

Table 8 Fields on the Input Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
		• **correlationId** Provide a correlation identifier for this message by mapping a byte array to this field. The XPath expression `tib:string-to-base64("`*correlationid*`")` is sufficient to populate this field. • **messageId** Specify a message ID. This value overrides the default value generated by the queue manager. The rules are the same as for `correlationId`. • **characterSet** Sets the IBM Coded Characterset ID of this message. Valid values are: — `850` — the commonly used ASCII codeset — `819` — the ISO standard ASCII codeset — `37` — the American EBCDIC codeset — `1200` — Unicode — `1208` — UTF-8 The queue manager may apply this codeset when doing conversions for the purpose of creating report messages. • **accountingToken** Provide a binary (base 64 encoded) accounting token to be used on host systems to assist in accounting for resources used by the application. • **applicationIdData** Provide a string that can be used on all end points to identify this application. • **applicationOriginData** Provide information related to the originating application node for this message.

Table 8 Fields on the Input Tab: WebSphere MQ Put Activity

Field	Global Variable	Description

		• **putApplicationType** Provide either one of the IBM documented application type enumerators for this application or one of your choosing. See the IBM WebSphere MQ documentation for the meaning of the documented application types. These are the documented enumerations:
		`MQAT_AIX`
		`MQAT_CICS`
		`MQAT_DOS`
		`MQAT_IMS`
		`MQAT_MVS`
		`MQAT_OS2`
		`MQAT_OS400`
		`MQAT_QMGR`
		`MQAT_UNIX`
		`MQAT_WINDOWS`
		`MQAT_JAVA`
		`MQAT_UNKNOWN`
		`MQAT_NO_CONTEXT`
		`MQAT_CICS_VSE`
		`MQAT_VMS`
		`MQAT_GUARDIAN`
		`MQAT_VOS`
		`MQAT_DEFAULT`
		`MQAT_NSK`
		`MQAT_CICS_BRIDGE`
		`MQAT_NOTES_AGENT`
		`MQAT_WINDOWS_NT`
		`MQAT_IMS_BRIDGE`
		`MQAT_XCF`
		`MQAT_NO_CONTEXT`
		
		• **putApplicationName** Provide information about the originating application in this field.
		
		• **userId** Provide the userid to be used when authorizing this message as it traverses the messaging system. For operations with report options, this field is mandatory. If a value is not provided here, it is taken from the connection resource if one is available there, and from the environment if it is not.
		
		• **replyToQueueName** Specify the name of an existing queue that the target application can use as a destination for its response. This enables the common "request response" pattern.

Table 8 Fields on the Input Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
		• **replyToQmgrName** Destination queue manager name. The name of the queue manager on which this queue resides. Note that a transmission queue and a sender channel must be configured for this to work. • **format** Describes the nature of the data in the message. It defaults to `"  "`, or `MQFMT_NONE`. A user defined format can be used to assist the receiving application in decoding the message, for instance in the selection of a Data Conversion parse activity for this message type. • **expiry** Indicates the length of time (in tenths of a second) that the message is kept on the queue before being deleted. Defaults to `-1`, which means there is no expiration. This field overrides the configuration value for expiry if it is mapped. Permissible Values: A positive integer or `-1` • **priority** The priority (1-9) of the message. This field overrides a configuration value for priority if it is mapped.
`appProperties`	N	Contains fields that correspond to the schema as it is defined in the application properties resource that is mapped to this activity. Application properties are included in an MQRFH2 header which is dynamically added to the message. Note that `appProperties` is only present if a WMQ Properties Resource has been configured for the activity.
`bytes`	N	Map the main content of the message body here. In general, this is the output of a Plug-in for Data Conversion render activity, but could be any array of bytes. If string data is to be mapped to this field, use the XPath function `tib:string-to-base64` to convert it to bytes.
`lastSegment`	N	If segmentation was specified for this activity, this field is present. When the application sends the last message of a group it must map `true` to this field to terminate the message group.

Output Tab

Table 9 lists and describes the fields under the Output tab.

Table 9 Fields on the Output Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`InteractionOutput`	N	The root node for all the output provided by the activity.
`mqproperties`	N	The node containing all the message header fields (plus the destination override) which are relevant on input. • **destination** Documents the destination for this message. If the queue was a dynamic queue, the generated name is reported here. • **destqmgr** Documents the remote queue manager specified for the put operation. • **correlationId** Documents the correlation ID, if any. • **messageId** Documents the message ID. • **characterSet** Documents the character set used by the queue manager to process this message. • **replyToQmgrName** Destination queue manager name. The name of the queue manager on which this queue resides. Note that a transmission queue and a sender channel must be configured for this to work. • **format** The character string placed into the format field by the sending application. Plug-in applications do not support modification of this field and for those applications it will be empty. However, if a custom application sent the message its contents would be visible here. • **expiry** Indicates the number of tenths of a second that the message is kept on the queue before being deleted. Defaults to `-1`, which means there is no expiration. • **priority** The priority (1-9) with which the message was sent. • **groupId** Documents the group ID, if any.

Table 9 Fields on the Output Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
		• **messagetype** Enumerated value of the chosen message type, where: — 8 is Datagram — 2 is Reply — 4 is Report — 1 is Request • **backoutCount** The number of times this message was backed out before being committed. • **feedback** For a Report message received by a WMQ Get or Listener activity, this field defines the nature of the report. The field is unused with a value of 0 in Put activities. • **userId** the effective user ID used to connect to the queue manager. • **putDateTime** The time that the message was placed onto the queue • **version** Documents the version of the message. This Plug-in supports type 2 messages only. • **replyToQueueName** Same as input. • **accountingToken** As provided on the input schema. • **applicationIdData** As provided on the input schema. • **applicationOriginData** As provided on the input schema. • **putApplicationType** As provided on the input schema. • **putApplicationName** As provided on the input schema.

Table 9 Fields on the Output Tab: WebSphere MQ Put Activity

Field	Global Variable	Description
`Responsetimemillisec`	N	Indicates, in milliseconds, how long it took to send the message.

Error Output Tab

Errors produced by this activity throw an exception of the type `BwmqException`.

WebSphere MQ Get Activity

The get activity retrieves a message from the configured queue. Messages can be retrieved based on filter criteria specified on the input document.

The get activity can be used to:

- Check the queue with a wait interval. The activity waits on the queue for the specified interval, then returns whether or not a message was received.
- Check the queue without a wait interval. The activity checks the queue and returns immediately, regardless of whether a message was read.
- Listen indefinitely on the queue. The activity returns a message if one is available, or waits forever for a message to arrive.

When designing your applications, TIBCO strongly recommends that you use remotely bound queue manager connections, even if they are on the same machine. This prevents problems that occur when using the Stop button on the activity tester, which interrupts the listen on the queue. If the connection resource is of the locally bound type it is impossible to interrupt this operation and one of two things happens:

- The designer fails to stop the test must be shut down, or
- The test terminates and leaves a thread hung on the queue listen operation.

You can see this behavior by looking at the `open input` count for the queue in the WebSphere MQ Explorer.

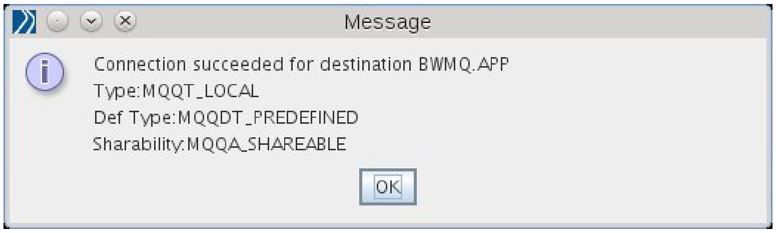
Threadpool Size

It is important to consider the impact of the wait interval on ActiveMatrix BusinessWorks thread resources. If the wait is likely to consume an ActiveMatrix BusinessWorks worker thread for a long enough that the engine performance could be affected, then use a thread pool for best results. This can be accomplished by setting the `Threadpool Size` to a number greater than 0. If you elect to use a thread pool, ensure that it is large enough to accommodate all instances of this activity. If the a thread is not available, an activity will wait indefinitely for an available thread.

Test Button

The test button for the get activity attempts to validate the selections for queue manager and queue by opening the queue manager and instantiating the queue object. For dynamic queues, only the presence of the model is verified. For predefined queues which were successfully opened, a status box is presented.

For example:



Connection succeeded for destination BWMQ.APP
Type:MQQT_LOCAL
Def Type:MQQDT_PREDEFINED
Sharability:MQQA_SHAREABLE

Configuration Tab

Table 10 lists and describes the fields under the Configuration tab.

Table 10 Fields on the Configuration Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
`Name`	N	The name of the Get activity. Default: `Websphere Mq Get Activity`
`Description`	N	Optional free-form description.
`Connection Resource`	Y	Choose the connection resource which connects this activity with the queue manager that has access to the desired queue.
`Threadpool Size`	Y	Because gets necessarily block for long periods of time, it is important not to occupy worker threads for this activity. Enter the number of threads that this deployed instance should dedicate to the named activity. Note that the activity will wait for an available thread if one is not open. Because waits for this type of activity can be long to indefinite, it is important to size this pool so that each activity can obtain a thread when it starts. If the field is `0` or empty, no thread pool is allocated and the activity will run in a TIBCO ActiveMatrix BusinessWorks worker thread. Default: `2` Valid Range: `0` to `256`

Table 10 Fields on the Configuration Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
`Queue Name`	Y	The name of the queue from which the message will be retrieved. Queue names can be chosen using the **Choose Queue** button to the right of this field. If the queue is to be created dynamically, enter the name of the dynamic queue. The name must follow the MQSeries rules for dynamic queue names. If the queue name has been dynamically mapped using the input schema's destination field, then it is not required here. However, note that the **Test** button cannot test dynamically mapped queue names.
`Close Option`	N	Select the close option for this get operation. These options are for use on permanent dynamic queues only. • `None` — No action is taken. • `Delete` — deletes the queue, if it is empty. If the queue is not empty,no action is taken. • `Purge Delete` — deletes the queue and any messages on it. If the get is being performed on a permanent dynamic queue and the intention is for the queue to be deleted, you should choose either `Delete` or `Purge Delete`. Delete will delete the queue if it is empty, and Purge Delete will delete the queue and any messages that are on it. Do *not* select `Delete` or `Purge Delete` for permanently defined queues. Depending on your security access, this may delete them.

Table 10 Fields on the Configuration Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
`Dynamic Queue`	N	A flag that enables the creation of a dynamic queue. When checked, the `Model Queue Name` field is enabled. Depending on the nature of the model queue selected, the dynamic queue could be either "temporary dynamic" or "permanent dynamic". Temporary queues are deleted by the queue manager as soon as their message and connection counts are zero. A permanent dynamic queue behaves exactly like a predefined queue except that its life cycle is intended to be managed by the application. Dynamic queues are most frequently used as reply-to destinations in messaging applications.
`Model Queue Name`	Y	Provide the name of the dynamic queue. This field is only available if the above `Dynamic Queue` option is enabled. Note: It is recommended that static queue names (queue names that do not contain a pattern) be used for get activities. Should a pattern be used for a get activity queue name, then the name generated by the queue manager is not available until the get activity completes.
`Access Mode`	N	Select the access mode for the queue: • **Shared** allow other listeners to access the queue. • **Exclusive** do not allow other listeners to access the queue. If another WebSphere MQ application has the queue open then this activity fails • **Queue Default** Accept the access mode established when the queue was created.

Table 10 Fields on the Configuration Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
`Segmented Messages`	N	Permits the reading of individual messages in a segmented group. When this box is checked, the Plug-in is expected to execute within an iteration group bounded by a "repeat until true" condition which is mapped from the output schema `lastSegment` field. So the following iteration condition statement would terminate the loop correctly: `$WebSphere-MQ-Get-Activity/InteractionOutput/las tSegment = "true"` When the last segment message is detected, the output schema field `lastSegment` is set to `true` rather than `false`. Note that if the `lastSegment` output schema field is present only if the `Segmented Messages` option is enabled. It is recommended, when receiving segmented messages, that only one such activity is active on a given queue at a time. Even if the process is expecting a segmented message, it will process a simple or grouped message should one arrive: • If a non-segmented message is read from the queue, then the message body is returned in the bytes output field and the field `lastSegment` is set to `true`. This allows a loop reading segmented messages to end if a none-segmented message is read. • If a non-segmented, grouped message is read, then all the messages for the group are read and aggregated into a single bytes output. In this circumstance the output schema field `lastSegment` is also set to the string `true`.
`Wait Forever`	N	If checked, the get activity will wait forever for a message on the queue. When disabled, the `Wait Timeout` field is enabled. Default: Checked

Table 10 Fields on the Configuration Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
`Wait Timeout`	Y	The number of milliseconds this activity waits for a message before failing. If significant wait intervals are allowed, a separate thread pool should be used. See Threadpool Size on page 47. This field is not available if `Wait Forever` is checked.
`Application Properties`	Y	Select the Application Properties resource to be used to map the output of this get activity. Application properties are similar to regular message header fields except that the name and type of the fields can be determined by a predefined schema. If the received message contains properties, they are mapped according to this schema.

Input Tab

Table 11 lists and describes the fields under the Input tab.

Table 11 Fields on the Input Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
`InteractionInput`	N	The root node for all the input provided to the activity.

Table 11 Fields on the Input Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
`mqproperties`	N	The node containing all the message header fields (plus the destination override) which are relevant on input. • **destination** Override the destination queue name with the string specified in this field. • **destqmgr** The name of the remote queue manager from which to fetch the message. Not that the queue managers involved must be configured with the necessary sender and receiver channels in place and started. • **correlationId** Provide a correlation identifier for this message by mapping a byte array to this field. The XPath statement `tib:string-to-base64("`*correlationid*`")` is sufficient to populate this field. The provided correlation ID is used to filter messages from the queue. • **messageId** Specify a message ID. This value overrides the default value generated by the queue manager. The rules are the same as for `correlationId`. If a message ID is provided, it is used to filter the get results. • **characterSet** Sets the IBM Coded Characterset ID of this message. Valid values are: — `850` — the commonly used ASCII codeset — `819` — the ISO standard ASCII codeset — `37` — the American EBCDIC codeset — `1200` — Unicode — `1208` — UTF-8 The queue manager may apply this codeset when doing conversions for the purpose of creating report messages.

Output Tab

Table 12 lists and describes the fields under the Output tab.

Table 12 Fields on the Output Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
`InteractionOutput`	N	The root node for all the output provided by the activity.
`mqproperties`	N	The node containing all the message header fields (plus the destination override) which are relevant on output. • **destination** Documents the destination for this message. If the queue was a dynamic queue, the generated name is reported here. • **destqmgr** The name of the remote queue manager supplied on the input schema. • **correlationId** Documents the message correlation ID, if any. • **messageId** Documents the message ID. • **characterSet** Documents the character set the queue manager used to process this message. • **replyToQueueName** The name of the queue to which a reply should be sent. If the type of the message is `request`, then this is the name of the queue that the application should sent the "reply" message to. • **replyToQmgrName** The name of the remote queue manager for this message's reply-to destination, if one was provided. • **format** Describes the nature of the data in the message. The sender of the application can use standard names (these names are prefixed by the text `MQ`) or application-generated names. If the message originated from a TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ application, this is an empty string.

Table 12 Fields on the Output Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
		• **expiry** The expiration time (in tenths of a second), after which the message is eligible to be discarded by the queue manager. The default value of `-1` means there is no expiration. • **priority** The priority (1-9) with which the message was sent. • **groupId** Documents the group ID used for this message, if any. • **messagetype** Enumerated value indicating the message type, where: — `8` is Datagram — `2` is Reply — `4` is Report — `1` is Request • **backoutCount** The number of times this message was backed out before being committed. • **feedback** If the message was a Report message, this field defines the nature of the report. Defaults to `0` for non report messages. • **userId** the effective user ID used to connect to the queue manager. • **putDateTime** The time that the message was placed onto the queue. • **version** Documents the version of the message. This Plug-in supports type 2 messages only. • **accountingToken** The accounting token provided by the sending application.

Table 12 Fields on the Output Tab: WebSphere MQ Get Activity

Field	Global Variable	Description

		• **applicationIdData** Application ID data. This is application data was provided by sender of the message.
		
		• **applicationOriginData** Origin data as provided by either the sender of the message or the queue manager.
		
		• **putApplicationType** The type of application that put the message. Documented enumerations are:
		
		`MQAT_AIX`
		`MQAT_CICS`
		`MQAT_DOS`
		`MQAT_IMS`
		`MQAT_MVS`
		`MQAT_OS2`
		`MQAT_OS400`
		`MQAT_QMGR`
		`MQAT_UNIX`
		`MQAT_WINDOWS`
		`MQAT_JAVA`
		`MQAT_UNKNOWN`
		`MQAT_NO_CONTEXT`
		`MQAT_CICS_VSE`
		`MQAT_VMS`
		`MQAT_GUARDIAN`
		`MQAT_VOS`
		`MQAT_DEFAULT`
		`MQAT_NSK`
		`MQAT_CICS_BRIDGE`
		`MQAT_NOTES_AGENT`
		`MQAT_WINDOWS_NT`
		`MQAT_IMS_BRIDGE`
		`MQAT_XCF`
		`MQAT_NO_CONTEXT`
		
		• **putApplicationName** Application name data as provided by either the sender of the message or the queue manager.

`appProperties`	N	This field maps the application properties present in the received message to the schema attached to the activity. Properties which are missing or not present in the schema are not shown here.

Table 12 Fields on the Output Tab: WebSphere MQ Get Activity

Field	Global Variable	Description
`Bytes`	N	The output body of the message. Typically, this byte array is to be mapped as input to a TIBCO ActiveMatrix BusinessWorks Plug-in for Data Conversion parse activity.
`Responsetimemillisec`	N	Indicates how long it took to receive the message. This value includes any time spent waiting on an empty queue.

Error Output Tab

Errors produced by this activity throw an exception of the type `BwmqException`.

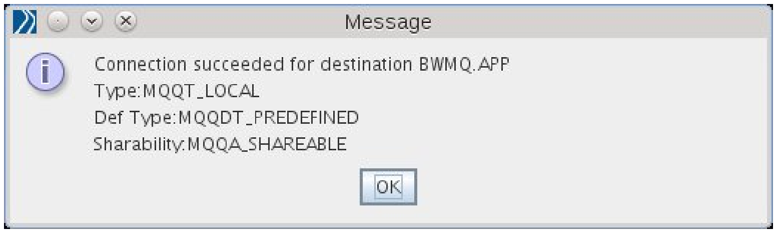
WebSphere MQ Listener Activity

The listener activity listens on a queue and when a message arrives that satisfies the configured filters it retrieves it and starts a process including the contents of the message. The listener can poll a queue based on interval and a timeout values, or it can wait indefinitely on a queue. Regardless of which mode it is in, when messages arrive on the queue they are each processed immediately with no intervening interval time.

Test Button

The test button for the listener activity attempts to validate the selections for queue manager and queue by opening the queue manager and instantiating the queue object. For dynamic queues, only the presence of the model is checked. For predefined queues which were successfully opened, a status box is presented.

For example:



Connection succeeded for destination BWMQ.APP
Type:MQQT_LOCAL
Def Type:MQQDT_PREDEFINED
Sharability:MQQA_SHAREABLE

Configuration Tab

Table 13 lists and describes the fields under the Configuration tab.

Table 13 Fields on the Configuration Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description
`Name`	N	The name of the process listener activity Default: `Websphere Mq Listener Start`
`Description`	N	Optional free-form description.
`Connection Resource`	Y	Choose the connection resource which connects this activity with the queue manager that has access to the desired queue.

Table 13   Fields on the Configuration Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description
`Queue Name`	Y	The name of the queue from which the message will be retrieved. Queue names can be chosen using the **Choose Queue** button to the right of this field.
`Dynamic Queue`	N	A flag that enables the creation of a dynamic queue. When checked, the `Model Queue Name` field is enabled.
`Model Queue Name`	Y	Provide the name of the model queue after which this dynamic queue will be patterned. This field is only available if the above `Dynamic Queue` option is enabled. Note: It is recommended that static queue names (queue names that do not contain a pattern) be used for listener activities. Should a pattern be used for a listener activity queue name, then the name generated by the queue manager is not available until the listener creates a process which makes it impossible for a sender to know the name.
`Fail If Quiescing`	N	Enable this option to prevent this listener from holding up a queue manager quiesce operation. If the queue manager does terminate, the activity attempts to reconnect indefinitely.
`Access Mode`	N	Select the access mode for the queue: • **Shared** allow other listeners to access the queue. • **Exclusive** do not allow other listeners to access the queue. During execution, if the queue is open by another application then the activity fails. • **Queue Default** Accept the access mode established when the queue was created.

Table 13 Fields on the Configuration Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description
`Require Client Confirmation`	N	Enable this option to request explicit confirmation of a received message. The confirm activity can be used to confirm the message's arrival and remove it from the queue. The Plug-in uses the queue manager's Syncpoint capability so that, when the option is enabled, each received message uses a new connection to the queue manager. As a result, this feature involves more overhead than non-explicit confirmation. If the activity fails to confirm the message, it will loop and repeatedly process the message.

Advanced Tab

Table 14 lists and describes the fields under the Advanced tab.

Table 14 Fields on the Advanced Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description
`Enable Polling`	N	When enabled, the activity instantiates a connection to the queue manager and opens the queue. It then periodically issues timed gets on the queue where the timeout value is equal to the polling timeout and the period between gets is equal to the polling interval. However, if there are multiple messages on the queue they are processed without an intervening polling interval, thus diminishing application latency. When disabled, the activity waits forever on the queue, processing messages as they arrive. If the queue manager connection fails the activity attempts to reconnect based on the reconnection interval. Note that a poor choice of polling interval and polling timeout values could impose a high overhead on the engine as well as the queue manager to which the activity is connected. Use values here which take into account the latency in the network between this activity and the queue manager, as well as the processing demands you are willing to place on that queue manager.
`Polling Interval`	Y	If polling is enabled, this is the interval at which a new polling cycle is initiated. Specify time in milliseconds. Valid range: `100` to `1000000` Default: `1000`

Table 14 Fields on the Advanced Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description
`Polling Timeout`	Y	If polling is enabled, this is the timeout value for each individual listen. The value specified here should be significantly smaller than the `Polling Interval`. This minimizes the number of concurrent connections to the queue and reduces the processing resources needed. Specify time in milliseconds. Valid range: `10` to `1000000` Default: `1000`
`Reconnection Interval`	Y	The length of time, in milliseconds, that the listener waits between attempts to reconnect to the queue. This setting is used only when polling is disabled. If polling is not enabled, the listener listens indefinitely on the queue. However, if an error occurs and the queue manager is subsequently unavailable (shut down for instance), this value is the number of milliseconds between reconnection attempts. Valid range: `2000` to `1000000` Default: `10000`
`Correlation Id`	Y	The listener instructs the queue manager to filter the request based on this correlation ID. Only messages with exactly the same ID are accepted. Note that WebSphere MQ uses byte arrays for correlation IDs. However, it is not possible to provide a byte array as a configuration parameter. When filtering a queue this way, be sure to use a clear text correlation ID on the application that did the put.
`Message Id`	Y	The listener instructs the queue manager to filter the request based on the specified message ID. Note that, in contrast to correlation IDs message IDs are always provided. If the listener does not explicitly state a message ID, it accepts any message. If an ID is provided, the listen is filtered for that message ID.

Table 14 Fields on the Advanced Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description
`Application Properties`	Y	Select the Application Properties resource to be used to map the output of this get. Application properties are similar to regular message header fields except that the name and type of the fields can be determined by a predefined schema. If the received message contains properties, they are mapped according to this schema.

Misc Tab

This tab contains fields that can be used by the activity to effect workload balancing across threads and even engines.

Sequencing Key

This field can contain an XPath expression that specifies which processes should run, and in which in order. Process instances with sequencing keys that evaluate to the same value are executed sequentially in the order the process instance was created.

See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about controlling the execution order of process instances and about XPath expressions.

Custom Id

This field can contain an XPath expression that specifies a custom ID for the process instance. This ID is displayed in the View Service dialog of TIBCO Administrator, and it is also available in the `$_processContext process` variable.

Output Tab

Table 15 lists and describes the fields under the Output tab.

Table 15 Fields on the Output Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description
`InteractionOutput`	N	The root node for all the output provided by the activity.

Table 15 Fields on the Output Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description
`mqproperties`	N	The node containing all the message header fields (plus the destination override) which are relevant on output. • **destination** Documents the destination for this message. If the queue was a dynamic queue, the generated name is reported here. • **destqmgr** If the message was routed through a remote queue manager, documents the name of the destination queue manager. • **correlationId** Documents the message correlation ID, if any. • **messageId** Documents the message ID. • **characterSet** Documents the character set the queue manager used to process this message. • **replyToQueueName** The name of the queue to which a reply should be sent. If the type of the message is `Request`, then this is the name of the queue that the application should sent the reply message to. • **replyToQueueMgrName** The name of the remote queue manager to be used when processing any reply or report message. • **format** A name that describes the nature of the data in the message. The sender of the application can used standard names (these names are prefixed by the text `MQ`) or application generated names. If this message originated from a TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ, then this field contains an empty string. • **expiry** The expiration time (in tenths of a second), after which the message is eligible to be discarded by the queue manager. This contains whatever value was used when the message was sent.

Table 15 Fields on the Output Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description
		• **priority** The priority (1-9) with which the message was sent. • **groupId** Documents the group ID used for this message, if any. • **messagetype** Enumerated value indicating the message type, where: — `8` is Datagram — `2` is Reply — `4` is Report — `1` is Request • **backoutCount** The number of times this message was backed out before being committed. • **feedback** If the message was a Report message, this field defines the nature of the report. Defaults to `0` for non report messages. • **userId** the effective user ID used to connect to the queue manager. • **putDateTime** The time that the message was placed onto the queue. • **version** Documents the version of the message. This Plug-in supports type 2 messages only. • **accountingToken** The accounting token provided by the sending application. • **applicationIdData** Application ID data as provided by the sending application. • **applicationOriginData** Additional information about the origin of the message provided by the sender of the message.

Table 15 Fields on the Output Tab: WebSphere MQ Listener Activity

Field	Global Variable	Description

		• **putApplicationType** The type of application that put the message. Documented enumerations are:
		`MQAT_AIX`
		`MQAT_CICS`
		`MQAT_DOS`
		`MQAT_IMS`
		`MQAT_MVS`
		`MQAT_OS2`
		`MQAT_OS400`
		`MQAT_QMGR`
		`MQAT_UNIX`
		`MQAT_WINDOWS`
		`MQAT_JAVA`
		`MQAT_UNKNOWN`
		`MQAT_NO_CONTEXT`
		`MQAT_CICS_VSE`
		`MQAT_VMS`
		`MQAT_GUARDIAN`
		`MQAT_VOS`
		`MQAT_DEFAULT`
		`MQAT_NSK`
		`MQAT_CICS_BRIDGE`
		`MQAT_NOTES_AGENT`
		`MQAT_WINDOWS_NT`
		`MQAT_IMS_BRIDGE`
		`MQAT_XCF`
		`MQAT_NO_CONTEXT`
		• **putApplicationName** The name of the application that put the message.

`appProperties`	N	This field maps the application properties present in the received message to the schema attached to the activity. Properties which are missing or not present in the schema are not shown here.
`Bytes`	N	The output body of the message. Typically, this byte array is to be mapped as input to a Data Conversion Plug-in parse activity.

Error Output Tab

Errors produced by this activity are written to the log and the activity continues to listen. If the polling interval or the reconnection interval is small and the activity is attempting to reconnect to an unavailable queue manager, it could produce a lot of output in the activities log.

WebSphere MQ Publish Activity

The publish activity publishes a message to a dynamic or a predefined topic. The activity is capable of sending any of the four message types described in the WebSphere MQ Put Activity section. The capabilities of the activity are determined by the configuration, with some runtime behaviors controlled by the input schema. The type of message being sent may determine what fields are applicable for the publish operation, for instance request messages are incompatible with report options. Even though this is a pub/sub type of activity, the report and reply-to queues are real queues not topics.

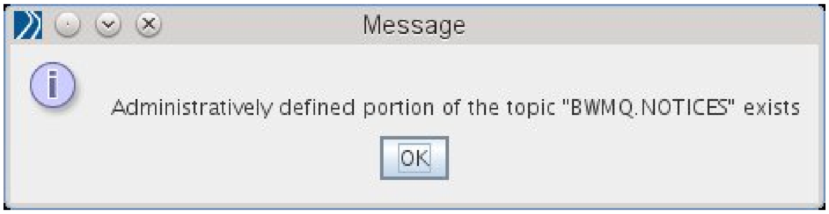
In WebSphere MQ, a topic is defined as a string of characters starting with a root '/' character and can contain many topic nodes to the right of that. WebSphere MQ supports a mechanism for separating administratively-defined topic strings from application-defined ones. This is exposed on the publish activity by the two fields `Topic` and `Topic Dynamic`. The intent is that administratively defined portions of the topic string are represented by selecting a permanently defined topic for the first part of the topic string and concatenating the dynamic portion to the right of it. This way an application can be deployed in several environments without naming conflicts.

There is no difference between the message object published by this activity from the one sent by the Plug-in's Put activity.

Test Button

Each activity includes a test button. The test button for the publish verifies that any administratively defined topic objects exist. Dynamic portions are not checked, and if the entire topic is dynamically defined, only an informational message is printed.

For example:


Message

Administratively defined portion of the topic "BWMQ.NOTICES" exists

Configuration Tab

Table 16 lists and describes the fields under the Configuration tab.

Table 16 Fields on the Configuration Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description
`Name`	N	The name of the Publish activity. Default: `WebSphere MQ Publish Activity`
`Description`	N	Optional free-form description.
`Connection Resource`	Y	Choose the connection resource which associates this activity with the queue manager that has access to the desired topic.
`Topic`	Y	The name of the administratively defined topic to which the message will be published. Topics can be chosen using the **Choose Destination** button to the right of this field. This field is optional in the event that the topic string is to be entirely specified in the `Topic Dynamic` field.
`Topic Dynamic`	Y	The contents of this field are concatenated to the topic name referenced by the administratively defined topic above. If no administratively defined topic is provided this string forms the entire topic string.
`Message Type`	N	Select the type of message that is to be sent: • Datagram • Request • Reply • Report If a Request message is chosen, the Reply To Queue field is enabled and the report options are disabled. All other message types allow report options.
`Reply To Queue`	Y	This field is enabled for Request message types only and contains the queue name which will be placed into the message header's reply to queue field. This queue need not exist and is not checked, but can be chosen using the Choose Queue button on the right side of the field.

Table 16 Fields on the Configuration Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description
`Application Properties`	Y	Select a schema which represents the application properties to be placed in this message.
`Retain Message`	N	Enable this option to indicate that the message is to be retained in the queue manager even if no subscribers exist. This allows future subscribers to receive the message when they are created. This is useful for conveying application state information.

Advanced Tab

Table 17 lists and describes the fields under the Advanced tab.

Table 17 Fields on the Advanced Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description
`Message Priority`	Y	Set the priority of the message as it is to be retrieved by the queue manager. The values range from 0 to 9. If not specified, the default is the destination queue's default priority.

Table 17 Fields on the Advanced Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description
`Reports`	N	Several kinds of report message can be triggered by a message's arrival at a particular point in the WebSphere MQ system. Select the type of report message and event trigger using the combination box. Events that can trigger a report are: • `Confirm on arrival` • `Confirm on delivery` • `Exception` • `Expiry` Report types are: • `None` The report is not enabled. • `With No Data` The body of the message does not accompany the report. • `With Data` The first 100 bytes of the message body accompanies the report. • `With Full Data` The entire body of the message accompanies the report. Reports are delivered to the reply to queue specified in the `Report Reply Queue` field. Because this field maps to the message's `replyToQueueName` field, reports are mutually exclusive with `Request` type messages. That is, if the message type is `Request`, a report confirmation is not sent.
`Report Reply Queue`	Y	Enter the name of the queue to which Report messages should be sent. This queue need not exist and is not checked, but can be chosen using the Choose Queue button on the right side of the field. The value entered here maps onto the message's `replyToQueueName` field.

Input Tab

Table 18 lists and describes the fields under the Input tab.

Table 18 Fields on the Input Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description
`InteractionInput`	N	This input schema field is the root node for all the input provided to the activity.
`mqproperties`	N	The node containing all the message header fields (plus the destination override) that are relevant on input. • **destination** Specify a destination here in order to override the topic name from the configuration panels. • **destqmgr** Destination queue manager name. The name of the queue manager on which this topic resides. Note that a transmission queue and a sender channel must be configured for this queue manager for it to work. • **topicdynamic** The dynamic portion of the topic string. This can be the entire topic string and will override any dynamic topic specified on the configuration panel. • **correlationId** Provide a correlation identifier for this message by mapping a byte array to this field. The XPath expression `tib:string-to-base64("`*correlationid*`")` is sufficient to populate this field. This field is only valid if the topic referenced is an administratively defined unmanaged topic. In all other cases it will be ignored by the queue manager. Consult the IBM WebSphere MQ documentation for descriptions of managed and unmanaged subscriptions. • **messageId** Provide a message identifier for this message by mapping a byte array to this field. The XPath expression `tib:string-to-base64("`*messageId*`")` is sufficient to populate this field. This field is only valid if the topic referenced is an administratively defined unmanaged topic. In all other cases it will be ignored by the queue manager.

Table 18 Fields on the Input Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description
		• **characterSet** Sets the IBM Coded Characterset ID of this message. Valid values are: — `850` — the commonly used ASCII codeset — `819` — the ISO standard ASCII codeset — `37` — the American EBCDIC codeset — `1200` — Unicode — `1208` — UTF-8 The queue manager may apply this codeset when doing conversions for the purpose of creating report messages. • **accountingToken** Provide a binary (base 64 encoded) accounting token to be used on host systems to assist in accounting for resources used by the application. • **applicationIdData** Provide a string that can be used on all end points to identify this application. • **applicationOriginData** Provide information related to the originating application node for this message. • **putApplicationName** The name of the application that put the message. Provide information about the originating application in this field.

Table 18 Fields on the Input Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description

		• **putApplicationType** Provide either one of the IBM documented application type enumerators for this application or one of your choosing. See the IBM WebSphere MQ documentation for the meaning of the documented application types. The documented enumerations are:
		`MQAT_AIX`
		`MQAT_CICS`
		`MQAT_DOS`
		`MQAT_IMS`
		`MQAT_MVS`
		`MQAT_OS2`
		`MQAT_OS400`
		`MQAT_QMGR`
		`MQAT_UNIX`
		`MQAT_WINDOWS`
		`MQAT_JAVA`
		`MQAT_UNKNOWN`
		`MQAT_NO_CONTEXT`
		`MQAT_CICS_VSE`
		`MQAT_VMS`
		`MQAT_GUARDIAN`
		`MQAT_VOS`
		`MQAT_DEFAULT`
		`MQAT_NSK`
		`MQAT_CICS_BRIDGE`
		`MQAT_NOTES_AGENT`
		`MQAT_WINDOWS_NT`
		`MQAT_IMS_BRIDGE`
		`MQAT_XCF`
		`MQAT_NO_CONTEXT`
		• **userId** Provide the userid to be used when authorizing this message as it traverses the messaging system. For operations with report options, this field is mandatory. If a value is not provided here, it is taken from the connection resource if one is available there, and from the environment if it is not.
		• **replyToQueueName** Specify the name of an existing queue that the target application can use as a destination for its response. This enables the common "request response" pattern.

Table 18 Fields on the Input Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description
		• **replyToQmgrName** Destination queue manager name. The name of the queue manager on which this queue resides. Note that a transmission queue and a sender channel must be configured for this to work. • **format** Describes the nature of the data in the message. It defaults to `"  "`, or `MQFMT_NONE`. A user defined format can be used to assist the receiving application in decoding the message, for instance in the selection of a Data Conversion parse activity for this message type. • **expiry** Indicates the length of time (in tenths of a second) that the message is kept on the queue before being deleted. Defaults to `-1`, which means there is no expiration. This field overrides the configuration value for expiry if it is mapped. Permissible Values: A positive integer or `-1` • **priority** The priority (1-9) of the message. This field overrides a configuration value for priority if it is mapped.
`appProperties`	N	Contains fields that correspond to the schema as it is defined in the application properties resource that is mapped to this activity. Note that `appProperties` is only present if a WMQ Properties Resource has been configured for the activity.
`bytes`	N	Map the main content of the message body here. In general, this is the output of a Plug-in for Data Conversion render activity, but could be any array of bytes. If string data is to be mapped to this field, use the XPath function `tib:string-to-base64` to convert it to bytes.

Output Tab

Table 19 lists and describes the fields under the Output tab.

Table 19 Fields on the Output Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description
`InteractionOutput`	N	The root node for all the output provided by the activity.
`mqproperties`	N	The node containing all the message header fields (plus the destination override) which are relevant on input. • **destination** Documents the destination for this message. If the topic was a dynamic topic, the generated name is reported here. • **destqmgr** Documents the remote queue manager specified on the input schema. • **topicdynamic** Documents the effective topic string for this publication. This includes the root portion from any administratively defined topic provided, as well as the any dynamic portion. • **correlationId** Documents the message correlation ID, if any. • **messageId** Documents the message ID. • **characterSet** Documents the character set the queue manager used to process this message. • **replyToQueueName** Same as input. • **replyToQmgrName** Same as input. • **format** The character string placed into the format field by the sending application. • **expiry** Indicates the number in tenths of a second that the message is to be kept on the queue before being deleted. Defaults to `-1`, which means there is no expiration. • **priority** The priority (1-9) with which the message was sent.

Table 19 Fields on the Output Tab: WebSphere MQ Publish Activity

Field	Global Variable	Description
		• **groupId** Documents the group ID assigned to the message by the queue manager. • **messagetype** Enumerated value indicating the message type, where: — 8 is Datagram — 2 is Reply — 4 is Report — 1 is Request • **backoutCount** The number of times this message was backed out before being committed. • **feedback** If the message was a Report message, this field defines the nature of the report. Defaults to 0 for non report messages. • **userId** the effective user ID used to connect to the queue manager. • **putDateTime** The time that the message was placed onto the queue. • **version** Documents the version of the message. This Plug-in supports type 2 messages only. • **accountingToken** As provided on the input schema. • **applicationIdData** As provided on the input schema. • **applicationOriginData** As provided on the input schema. • **putApplicationName** As provided on the input schema. • **putApplicationType** As provided on the input schema. • **responsetimemillisec** Indicates, in milliseconds, how long it took to publish the message.

Error Output Tab

Errors produced by this activity throw an exception of the type `BwmqException`.

WebSphere MQ Subscriber Activity

The subscriber activity listens on a topic or administratively defined subscription and starts a process that provides the contents of the message. The subscriber can poll a topic based on interval and a timeout values, or it can wait indefinitely. Regardless of which mode it is in, when messages arrive they are each processed immediately with no intervening interval time.

When polling is enabled, messages published to the subscribed topic are not lost during the interval when the subscriber is not actively listening because the subscriber exists for the duration and is not created and destroyed across intervals. However, this does not necessarily apply during error recovery or lost connection scenarios.

The topic string is composed of two optional parts:

- An administratively defined topic. This option implies a root topic string defined on the server.
- A dynamic topic string which is concatenated to the topic string for the defined topic, if one is specified.

If neither of these fields is provided but an existing durable subscription is specified, the activity subscribes to the durable subscription.

The queue manager inserts slash ('/') characters as necessary to compose a well formed topic string.

While the listener is active, a temporary dynamic subscription is created (in the queue manager) so that all messages appropriate for the topic are routed there. This subscription will cease to exist when the listener is stopped, or in error recovery. In order to avoid missing messages destined for the topic, you can enable the `Durable Subscription` option to have the subscription persisted indefinitely in the queue manager using the name provided. Durable subscriptions accumulate messages even when there are no listeners, much like a queue would.

Test Button

The test button for the subscriber will confirm the existence of the administratively defined topic and durable subscription, if either are specified.

Configuration Tab

Table 20 lists and describes the fields under the Configuration tab.

Table 20 Fields on the Configuration Tab: WebSphere MQ Subscriber Activity

Field	Global Variable	Description
`Name`	N	The name of the subscriber activity. Default: `WebSphere MQ Subscriber Start`.
`Description`	N	Optional free-form description.
`Connection Resource`	Y	Choose the connection resource which associates this activity with the queue manager that has access to the desired topic.
`Topic`	Y	The name of the administratively defined portion of the topic string. If a `Topic Name` is specified, the topic string associated with that name in the queue manager forms the root or left-most portion of the topic string. Existing topics can be selected using the **Choose Destination** button to the right of this field.
`Wildcard Scheme`	N	WebSphere MQ supports two wild card schemes: • `Topic` — Associated with current queue managers. • `Char` — Associated with version 6 and earlier of WebSphere MQ. Either scheme can be used. Consult the WebSphere MQ documentation for information about these schemes.
`Topic Dynamic`	Y	Enter the dynamic portion of the topic string here. This field can contain the entire topic string or only the right-most portion of it, depending on whether it is to be used in conjunction with an administratively defined topic.
`Durable Subscription`	N	When checked, the subscription created by this subscriber is durable. Durable subscriptions survive subscriber and queue manager restarts, and accumulate messages until they are explicitly deleted.

Table 20 Fields on the Configuration Tab: WebSphere MQ Subscriber Activity

Field	Global Variable	Description
`Subscription Name`	Y	Enabled if `Durable Subscription` is selected. This is the name of the permanent subscription. A subscriber can be started using only the durable subscription name. If the subscription exists, no topic string or dynamic topic are necessary.
`New Publications Only`	N	Enable this option for non-durable subscriptions where you do not want to receive the last message published with the `Retain Message` option. In essence, this is a way of opting out of the application state message.
`Fail If Quiescing`	N	Enable this option to prevent this subscriber from holding up a queue manager quiesce operation. If the queue manager does terminate, the activity attempts to reconnect indefinitely.

Advanced Tab

Table 21 lists and describes the fields under the Advanced tab.

Table 21 Fields on the Advanced Tab: WebSphere MQ Subscriber Activity

Field	Global Variable	Description
`Enable Polling`	N	When enabled, the activity instantiates a connection to the queue manager and opens the topic. It then periodically issues timed gets on the topic where the timeout value is equal to the polling timeout and the period between gets is equal to the polling interval. However, if there are multiple messages available, they are processed without an intervening polling interval, thus diminishing application latency. When disabled, the activity waits indefinitely on the topic, processing messages as they arrive. If the queue manager connection fails, the activity attempts to reconnect based on the reconnection interval. Note that a poor choice of polling interval and polling timeout values could impose a high overhead on the engine as well as the queue manager to which the activity is connected. Use values here which take into account the latency in the network between this activity and the queue manager, as well as the processing demands you are willing to place on that queue manager.
`Polling Interval`	Y	If polling is enabled, this is the interval at which a new polling cycle is initiated. Specify time in milliseconds. Valid range: `100` to `1000000` Default: `1000`
`Polling Timeout`	Y	If polling is enabled, this is the timeout value for each individual listen. The value specified here should be significantly smaller than the `Polling Interval`. This minimizes the number of concurrent connections to the queue and reduces the processing resources needed. Specify time in milliseconds. Valid range: `10` to `1000000` Default: `1000`

Table 21 Fields on the Advanced Tab: WebSphere MQ Subscriber Activity

Field	Global Variable	Description
`Reconnection Interval`	Y	The length of time, in milliseconds, that the listener waits between attempts to reconnect to the topic. This setting is used only when polling is disabled. If polling is not enabled, the listener listens indefinitely on the queue. However, if an error occurs and the queue manager is subsequently unavailable (shut down for instance), this value is the number of milliseconds between reconnection attempts. Valid range: `2000` to `1000000` Default: `10000`
`Application Properties`	Y	If the messages expected on this topic contain application specific topics, a shared schema resource should be created to represent those properties and it should be mapped here. Messages which contain any or all of these properties will have them provided as input to the process created by this starter.

Misc Tab

This tab contains fields that can be used by the activity to effect workload balancing across threads and even engines.

Table 22 lists and describes the fields under the Misc tab.

Table 22 Fields on the Misc Tab: WebSphere MQ Subscriber Activity

Field	Global Variable	Description
`Sequencing Key`	N	This field can contain an XPath expression that specifies which processes should run, and in which in order. Process instances with sequencing keys that evaluate to the same value are executed sequentially in the order the process instance was created. See the *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about controlling the execution order of process instances and about XPath expressions.

Table 22 Fields on the Misc Tab: WebSphere MQ Subscriber Activity

Field	Global Variable	Description
`Custom Id`	N	This field can contain an XPath expression that specifies a custom ID for the process instance. This ID is displayed in the View Service dialog of TIBCO Administrator, and it is also available in the `$_processContext` process variable.

Output Tab

Table 23 lists and describes the fields under the Output tab.

Table 23 Fields on the Output Tab: WebSphere MQ Subscriber Activity

Field	Global Variable	Description
`InteractionOutput`	N	The root node for all the output provided by the activity.
`mqproperties`	N	The node containing all the message header fields (plus the destination override) which are relevant on input. • **destination** Documents the destination for this message. This field contains the complete topic string for the message. • **destqmgr** Documents the remote queue manager specified on the input schema. • **correlationId** Documents the message correlation ID. In most cases this is an internal queue manager value and is not available for filtering purposes. • **messageId** Documents the message ID. • **characterSet** Documents the character set the queue manager used to process this message. • **replyToQueueName** The name of the queue to which a reply should be sent if the message is a request. • **replyToQmgrName** If this message arrived by way of a remote queue manager specification, this field documents the queue manager to which this subscriber is connected.

Table 23 Fields on the Output Tab: WebSphere MQ Subscriber Activity

Field	Global Variable	Description
		• **format** A name that describes the nature of the data in the message. • **expiry** The expiration time (in tenths of a second), after which the message is eligible to be discarded by the queue manager. This contains whatever value was used when the message was sent. • **priority** The priority (1-9) with which the message was sent. • **groupId** The group ID of the message, if it was part of a group. • **messagetype** Enumerated value indicating the message type, where: — 8 is Datagram — 2 is Reply — 4 is Report — 1 is Request • **backoutCount** The number of times this message was backed out before being committed. • **feedback** If the message was a Report message, this field defines the nature of the report. Defaults to 0 for non report messages. • **userid** the effective user ID used to connect to the queue manager. • **putDateTime** The time that the message was placed onto the topic. • **version** Documents the version of the message. • **accountingToken** The accounting token that allows work done as a result of the message to be appropriately billed.

Table 23 Fields on the Output Tab: WebSphere MQ Subscriber Activity

Field	Global Variable	Description

		• **applicationIdData** Application ID data. This is application data was provided by sender of the message.
		
		• **applicationOriginData** Additional information about the origin of the message provided by the sender of the message.
		
		• **putApplicationType** The type of application that put the message. Documented enumerations are:
		`MQAT_AIX`
		`MQAT_CICS`
		`MQAT_DOS`
		`MQAT_IMS`
		`MQAT_MVS`
		`MQAT_OS2`
		`MQAT_OS400`
		`MQAT_QMGR`
		`MQAT_UNIX`
		`MQAT_WINDOWS`
		`MQAT_JAVA`
		`MQAT_UNKNOWN`
		`MQAT_NO_CONTEXT`
		`MQAT_CICS_VSE`
		`MQAT_VMS`
		`MQAT_GUARDIAN`
		`MQAT_VOS`
		`MQAT_DEFAULT`
		`MQAT_NSK`
		`MQAT_CICS_BRIDGE`
		`MQAT_NOTES_AGENT`
		`MQAT_WINDOWS_NT`
		`MQAT_IMS_BRIDGE`
		`MQAT_XCF`
		`MQAT_NO_CONTEXT`
		
		• **putApplicationName** The name of the application that put the message, if one was provided.

`appProperties`	N	This field maps the application properties present in the received message to the schema attached to the activity. Properties which are missing or not present in the schema are not shown here.
`Bytes`	N	The output body of the message. Typically, this byte array is to be mapped as input to a Data Conversion Plug-in parse activity.

Error Output Tab

Errors produced by this activity are written to the log and the activity continues to listen. If the polling interval or the reconnection interval is small and the activity is attempting to reconnect to an unavailable queue manager, it could produce a lot of output in the activities log.

Chapter 4 Sample ActiveMatrix BusinessWorks Projects

This chapter describes the sample ActiveMatrix BusinessWorks project included with the Plug-in.

Topics

Opening the Sample Project

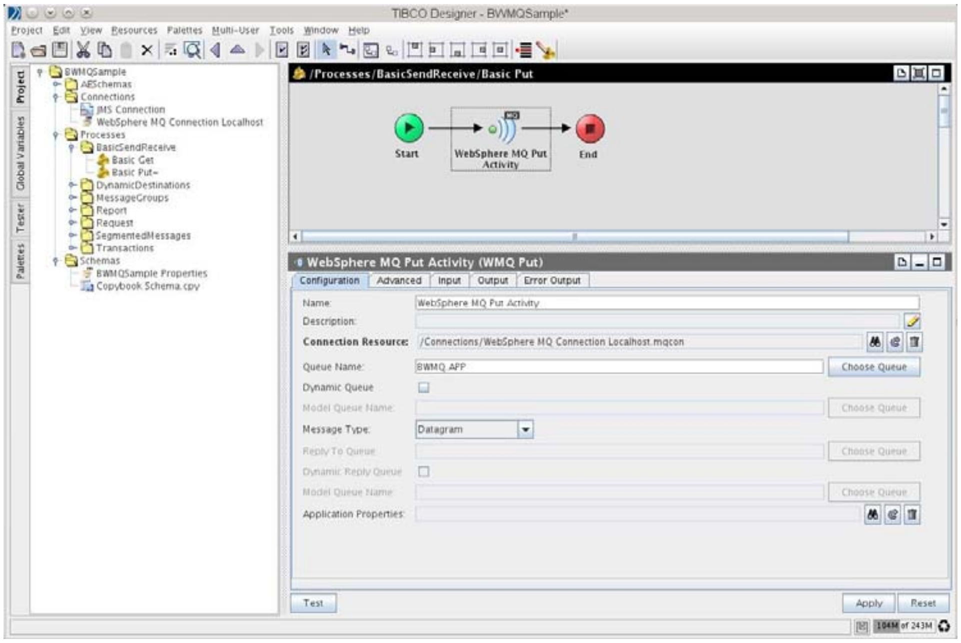
The Plug-in contains a sample ActiveMatrix BusinessWorks project, which is located in the directory
TIBCO_HOME`\bw\plugins\mq\samples\bwmqSample.zip`.

To open the sample project in TIBCO Designer, do the following:

1. Unzip the `bwmqSample.zip` file.
2. Start TIBCO Designer and click **Open existing project**.
3. In the Project Directory file chooser, navigate to the directory where the unzipped file is located. For example:

   TIBCO_HOME`\bw\plugins\mq\samples\bwmqSample.zip`
4. Click **OK**.

Figure 6 The Sample Project



Name: WebSphere MQ Put Activity
Description:
Connection Resource: /Connections/WebSphere MQ Connection Localhost.mqcon
Queue Name: BWMQ.APP
Dynamic Queue: ☐
Model Queue Name:
Message Type: Datagram
Reply To Queue:
Dynamic Reply Queue: ☐
Model Queue Name:
Application Properties:

Using the Sample Project

This sample project is intended to help familiarize you with the components of the TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ, as well as the modes these components were intended to be used in.

Recommended Software

The sample project references two additional Plug-in products:

- **TIBCO ActiveMatrix BusinessWorks Plug-in For Data Conversion**

  To run the samples without the Plug-in for Data Conversion, replace the references with an XPath expression that provides some input to the affected activity.

- **TIBCO ActiveMatrix BusinessWorks XA Transaction Manager**

  The XA Transaction sample cannot be run without the transaction manager plug-in.

These Plug-ins are frequently used with the Plug-in for WebSphere MQ. However, most samples can still be run if you do not have either plug-in.

Required Queues and Models

You must create these basic queues in order to use the sample:

- BWMQ.APP
- BWMQ.DEST.QUEUE
- BWMQ.FROM.QUEUE
- BWMQ.REPLY
- BWMQ.REPORT

You must also create these basic model queues:

- BWMQ.MODEL — a model for a temporary dynamic queue.
- BWMQ.DURABLE.MODEL — a model for a permanent dynamic queue.

Overview of Samples

The sample contains folders for Connections, Processes, and Schemas.

- The activities contained in the `Connections` folder are described in Connections on page 93.
- The resources contained in the Schemas folder are covered in Schemas on page 95.
- The activities and samples in the Processes folder are described in Processes on page 97.

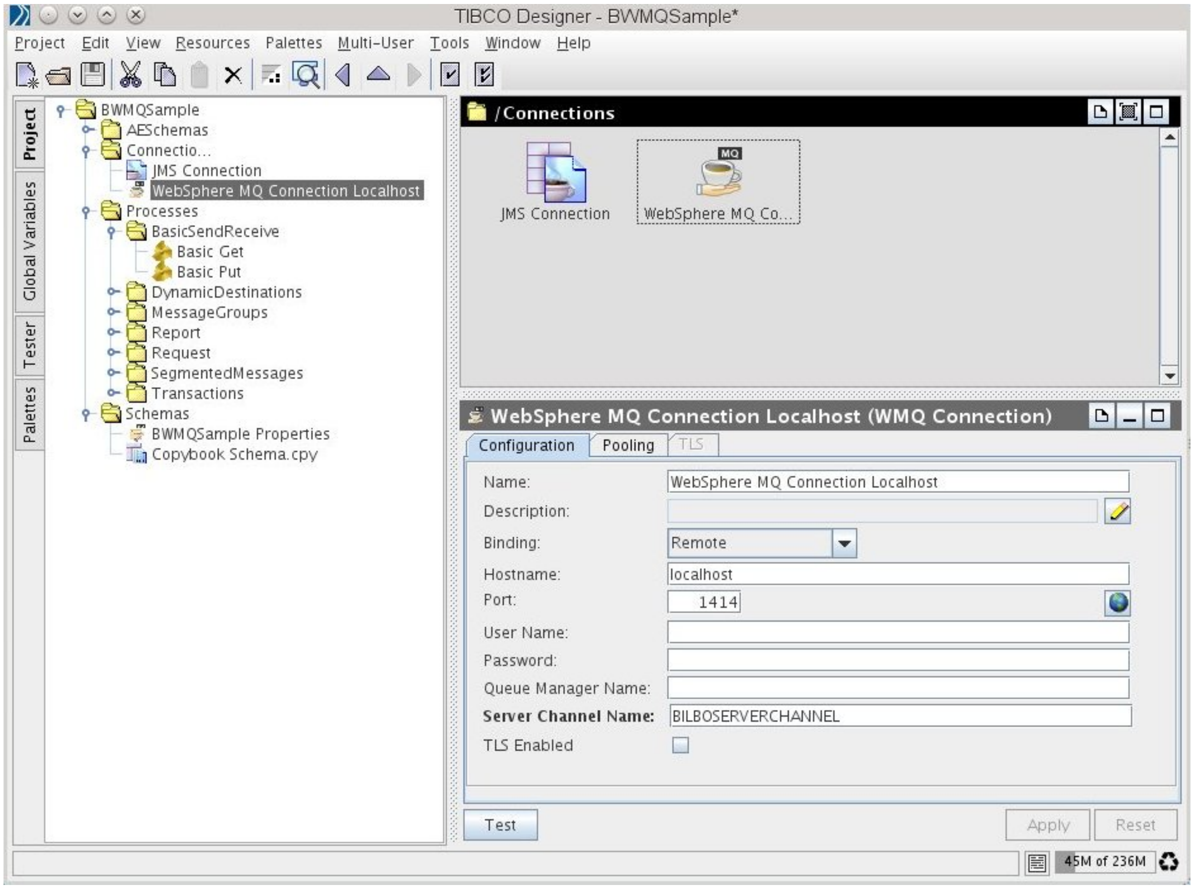
Not all of the capabilities of the Plug-in are demonstrated by these samples. Should you have general questions about the capabilities of the product, please visit the TIBCOmmunity website.

Connections

The Connection resource represents the WebSphere MQ connection and contains all the information needed to instantiate one. Configure the connection resource to connect to the queue manager that you intend to test with.

For best results, use a remote connection for most Designer activities. This prevents difficulties that may occur when attempting to interrupt a connection to the queue manager over a local connection.

Figure 7 The Connection Resource



Name:	WebSphere MQ Connection Localhost
Description:	
Binding:	Remote
Hostname:	localhost
Port:	1414
User Name:	
Password:	
Queue Manager Name:	
Server Channel Name:	BILBOSERVERCHANNEL
TLS Enabled	☐

The connection resource can be used to configure two types of connection:

- **Remote Binding** connects to the queue manager using TCP/IP.
- **Local Binding** connects to the queue manager using the Java Native Interface (JNI). Note that this binding type is not recommended for use during testing in TIBCO Designer because locally bound connections cannot be interrupted using the Stop button.

Depending on which type of binding is selected, a different subset of fields are enabled to create the connection. These fields are described in Table 3, Fields on the Configuration Tab: WMQ Connection Resource, on page 27.

Use the **Test** button at the bottom of the configuration panel to create a connection to the WebSphere MQ Queue Manager using the parameters you have provided.

JMS Connection Folder

The Connections folder also contains a JMS Connection resource, which is used in the XA transaction examples. These samples are described in Transactions on page 120.

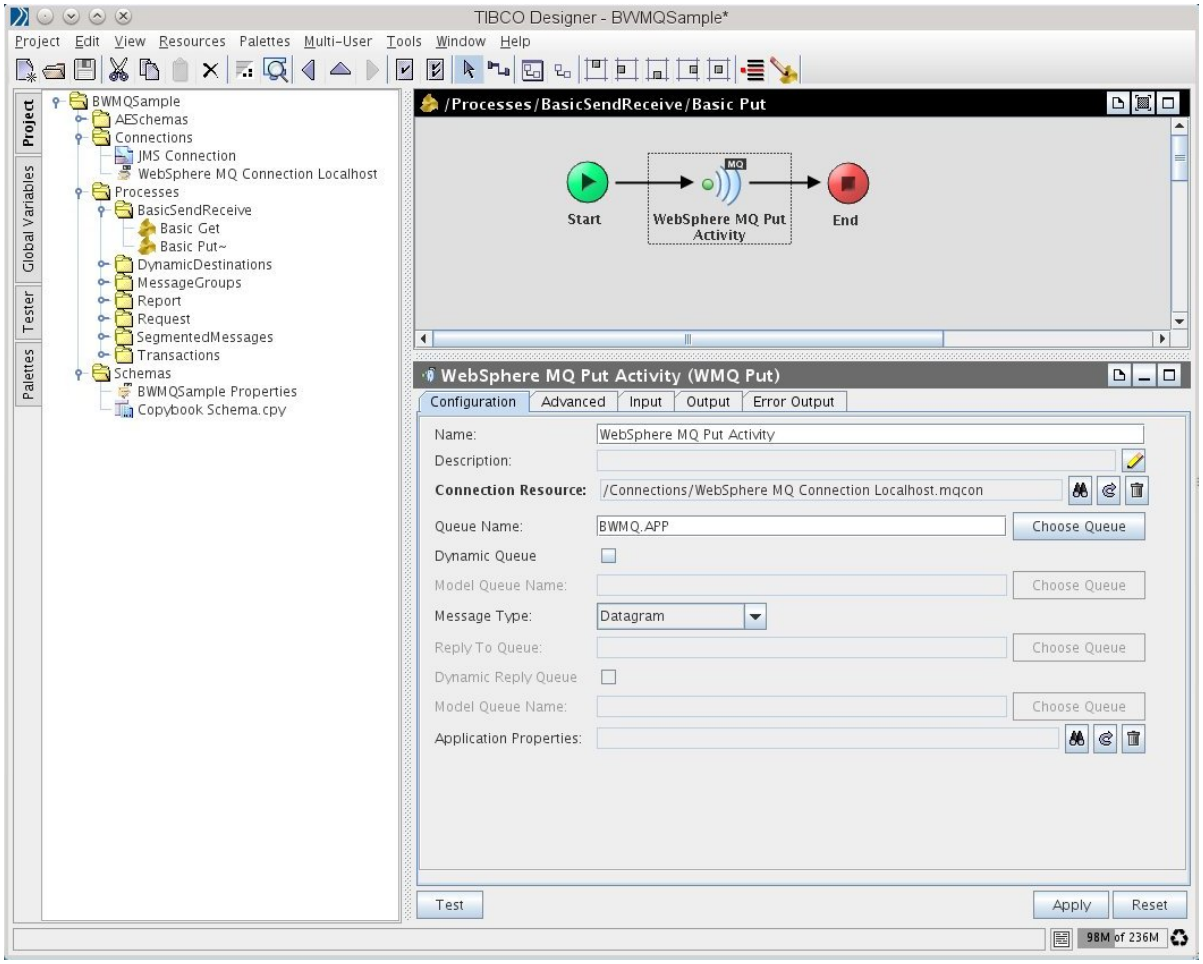
Schemas

The Schemas folder contains two schema resources that are used in this project:

- The WebSphere MQ Application Properties schema, which is created from the JMS Application Properties resource that is used to map various properties into and out of a WebSphere MQ message.
- The Data Conversion Plug-in copy book schema. The Plug-in for Data Conversion can take any memory layout represented by a copy book and convert it to an XML schema that is used in ActiveMatrix BusinessWorks to translate canonical data into formats that adhere to the definitions in the copy book. The conversion activities are called render and parse. The TIBCO ActiveMatrix BusinessWorks Plug-in for WebSphere MQ was designed to work hand in hand with the Plug-in for Data Conversion to facilitate communication with WebSphere MQ applications on any platform.

The application properties resource allows you to define a group of application specific name value pairs to be included with a sent message, or to be extracted from a received one. If one is selected for a WebSphere MQ Plug-in activity, that activity's input or output schema includes fields that relate to this resource. You can then map data into or out of them for use in your application.

Figure 8 Schemas — The WebSphere MQ Application Properties Resource



Name: WebSphere MQ Put Activity
Description:
Connection Resource: /Connections/WebSphere MQ Connection Localhost.mqcon
Queue Name: BWMQ.APP
Dynamic Queue
Model Queue Name:
Message Type: Datagram
Reply To Queue:
Dynamic Reply Queue
Model Queue Name:
Application Properties:

Processes

The Processes folder contains samples of most basic process activities. These activities are described in the following sections:

Basic Send and Receive

The first and simplest activity is a basic send and receive, contained in the `BasicSendReceive` folder. This activity puts a message on the queue, then retrieves the message using a get.

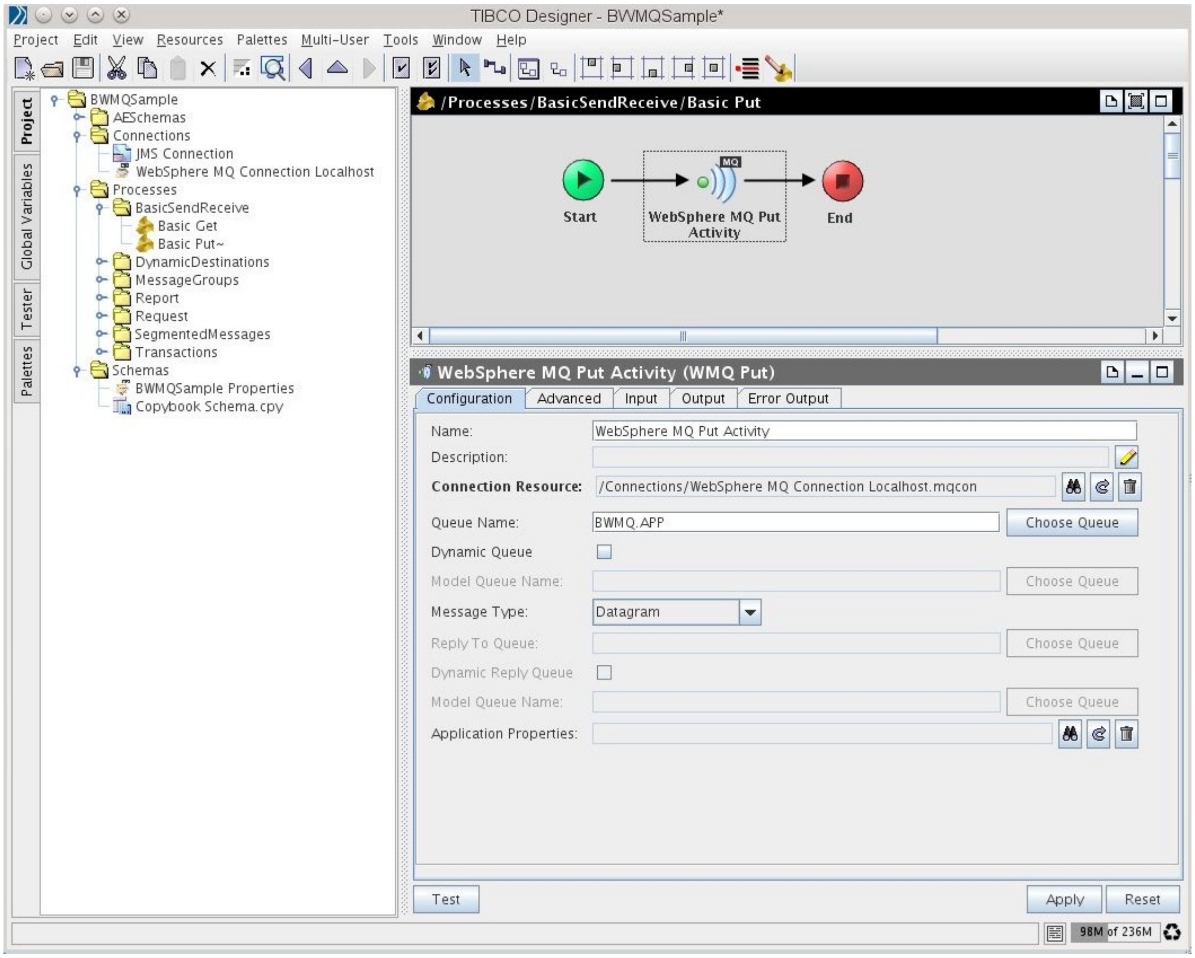
Configuring the Put Activity

Figure 9 shows the configuration panel of the Plug-in put activity. In this sample, the put activity is minimally configured. The sample puts a very simple datagram message on the queue `BWMP.APP`. The minimum configuration for a put activity requires only a connection resource and a queue name.

Use the **Choose Queue** selection buttons to the right of each of the queue and model fields. Use these buttons to select the queues and models you would like to connect to.

The **Test** button at the bottom of the panel creates a connection to the configured queue manager and instantiates an instance of the configured queue object. If there are any errors in the configuration or when creating the connection, an error message appears. If the connection is successfully created, some minimal information about the selected queue is displayed.

Figure 9 The Put Activity — Configuration Tab



Name: WebSphere MQ Put Activity
Description:
Connection Resource: /Connections/WebSphere MQ Connection Localhost.mqcon
Queue Name: BWMQ.APP
Dynamic Queue ☐
Model Queue Name:
Message Type: Datagram
Reply To Queue:
Dynamic Reply Queue ☐
Model Queue Name:
Application Properties:

Configuring the Input Schema

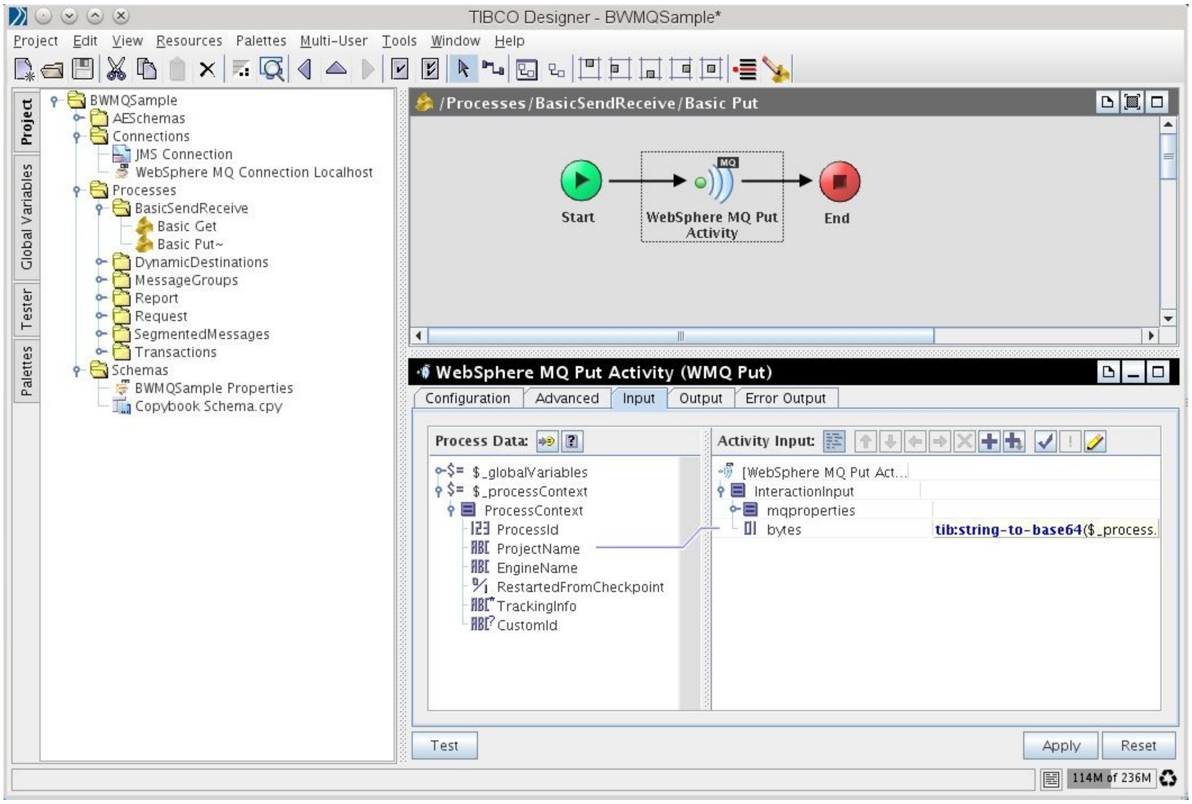
Figure 10 shows the input schema tab for the put activity. This tab exposes all the modifiable input fields in the message header, as well as the destination and payload (bytes) fields.

Because this is a "simple" example, the payload for this message is supplied by an XPath expression which converts a string into a base 64-encoded byte array:

```
tib:string-to-base64($_processContext/ns:ProcessContext/ProjectName)
```

You can use this technique to convert a string to a byte array for any field that requires it.

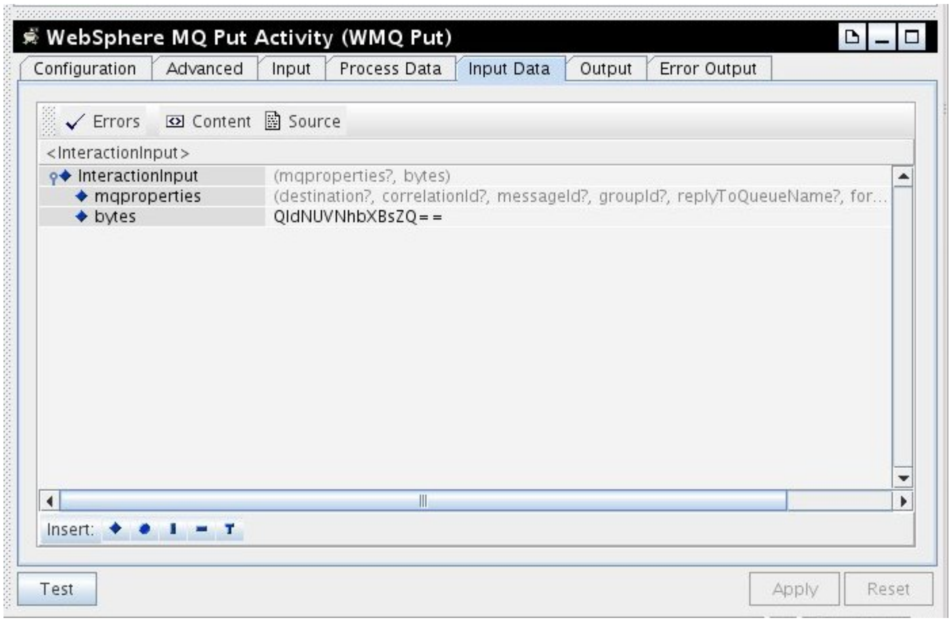
Figure 10 The Put Activity — Input Tab



bytes: tib:string-to-base64($_process.

When the activity is run, the input data can be seen in its base 64 form, as shown in Figure 11.

Figure 11 The Put Activity — Input Data



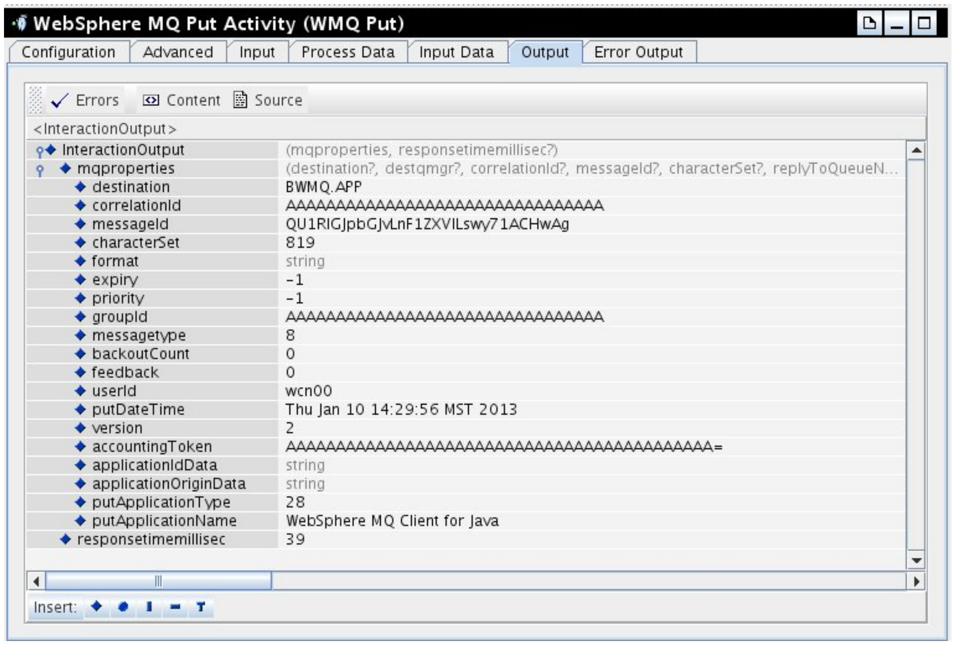
<InteractionInput>

InteractionInput	(mqproperties?, bytes)
mqproperties	(destination?, correlationId?, messageId?, groupId?, replyToQueueName?, for...
bytes	QldNUVNhbXBsZQ==

Reviewing the Output

The Output tab, shown in Figure 12, documents all of the fields as they were modified by the queue manager during the put operation. Other activities in this sample illustrate the use of input and output fields.

Figure 12   The Put Activity — Output Data



<InteractionOutput>

InteractionOutput	(mqproperties, responsetimemillisec?)
mqproperties	(destination?, destqmgr?, correlationId?, messageId?, characterSet?, replyToQueueN...
destination	BWMQ.APP
correlationId	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
messageId	QU1RIGJpbGJvLnF1ZXVILswy71ACHwAg
characterSet	819
format	string
expiry	-1
priority	-1
groupId	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
messagetype	8
backoutCount	0
feedback	0
userId	wcn00
putDateTime	Thu Jan 10 14:29:56 MST 2013
version	2
accountingToken	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=
applicationIdData	string
applicationOriginData	string
putApplicationType	28
putApplicationName	WebSphere MQ Client for Java
responsetimemillisec	39

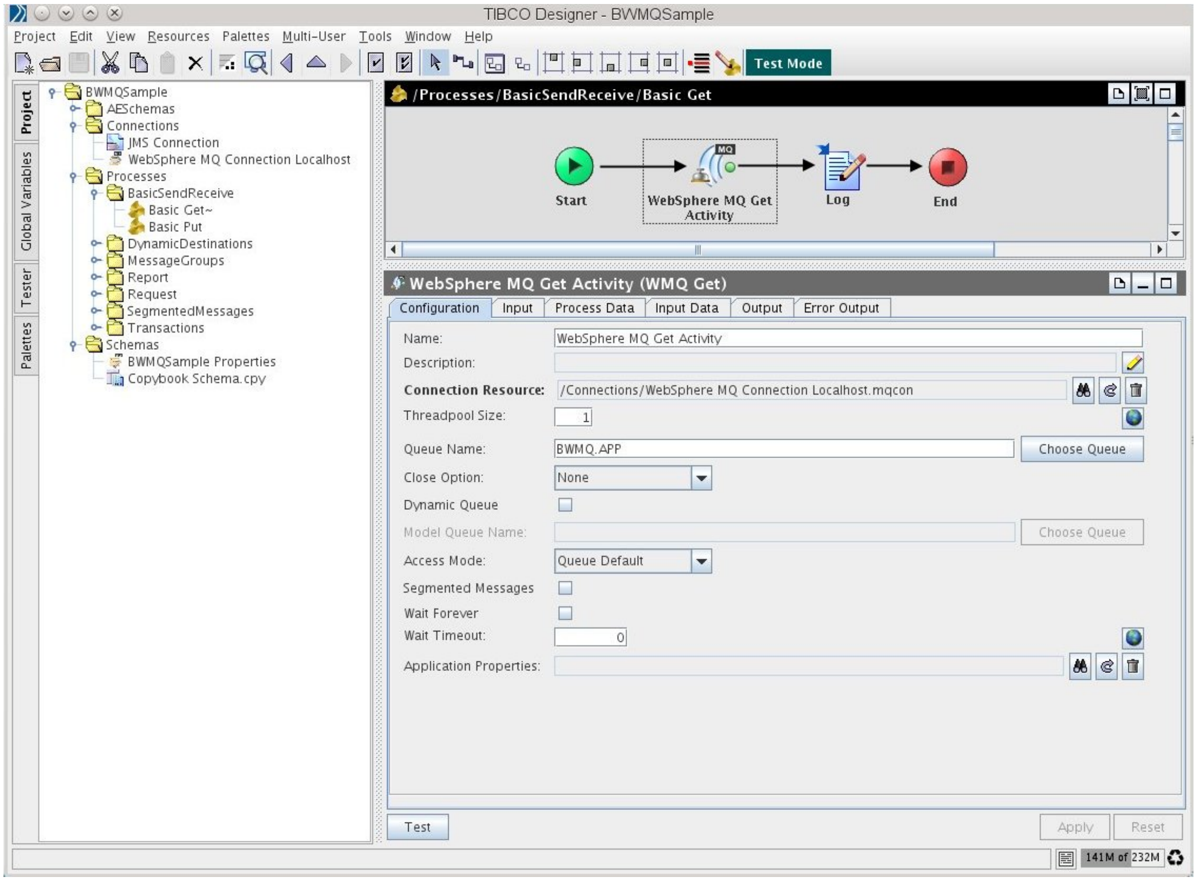
Configuring the Get Activity

Once a message exists on the queue, you can use the basic get sample in this folder to retrieve it.

As with the put activity, the minimal configuration for a get activity requires selecting the connection resource and the queue, as shown in Figure 13.

The default behavior of the get activity is to wait forever for a message on the queue. Although this sample uses the default, it is more common to place an upper limit on how long the activity should wait for a message.

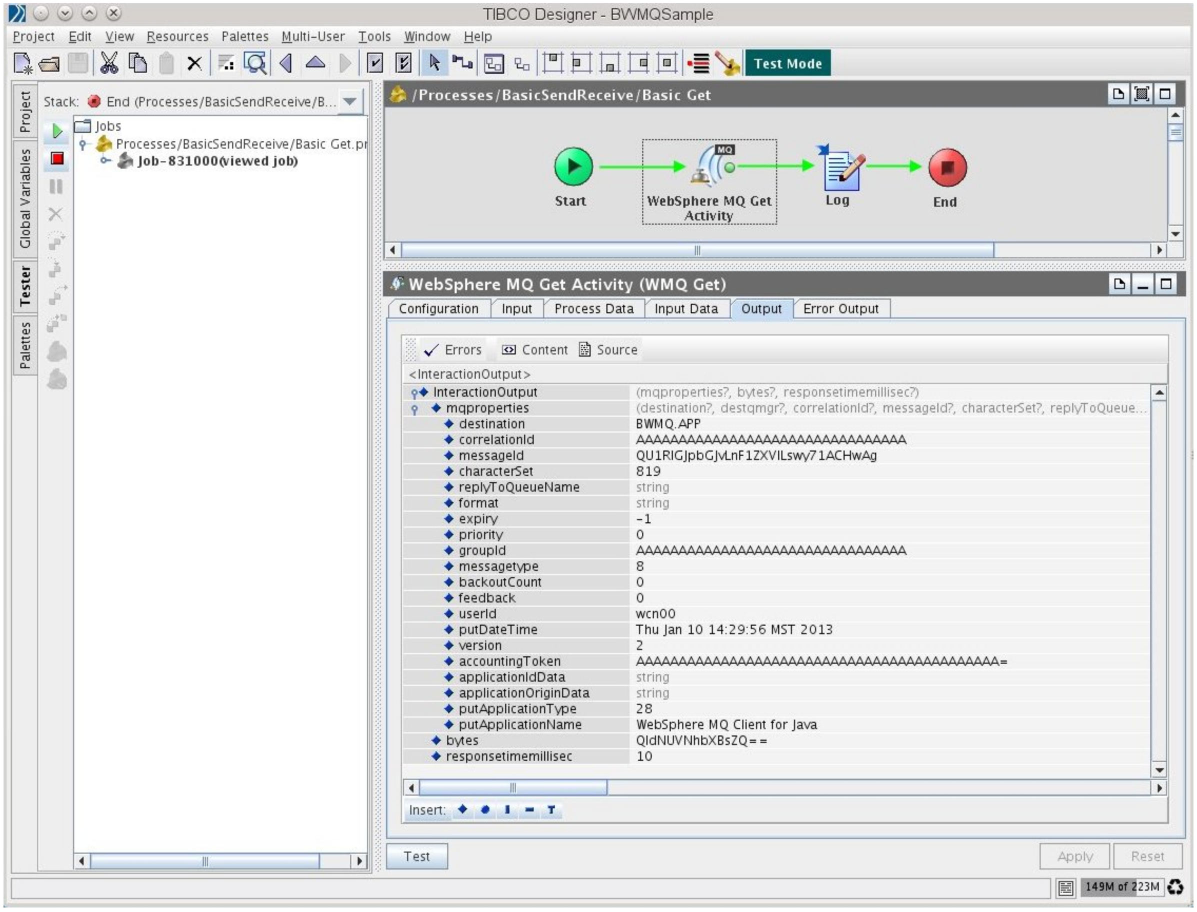
Figure 13 The Get Activity — Configuration Tab



Name:	WebSphere MQ Get Activity
Description:	
Connection Resource:	/Connections/WebSphere MQ Connection Localhost.mqcon
Threadpool Size:	1
Queue Name:	BWMQ.APP
Close Option:	None
Dynamic Queue	☐
Model Queue Name:	
Access Mode:	Queue Default
Segmented Messages	☐
Wait Forever	☐
Wait Timeout:	0
Application Properties:	

The Input and Output tabs are also similar to those of the put activity. Run the activity and examine the output tab, as shown in Figure 14:

Figure 14 The Get Activity — Output Tab


TIBCO Designer - BWMQSample
Project Edit View Resources Palettes Multi-User Tools Window Help
Test Mode
Project
Global Variables
Tester
Palettes
Stack: End (Processes/BasicSendReceive/B...
Jobs
Processes/BasicSendReceive/Basic Get.pr
Job-831000(viewed job)
/Processes/BasicSendReceive/Basic Get
Start
WebSphere MQ Get Activity
Log
End
WebSphere MQ Get Activity (WMQ Get)
Configuration
Input
Process Data
Input Data
Output
Error Output
Errors
Content
Source
Insert:
Test
Apply
Reset
149M of 223M

<InteractionOutput>

InteractionOutput	(mqproperties?, bytes?, responsetimemillisec?)
mqproperties	(destination?, destqmgr?, correlationId?, messageId?, characterSet?, replyToQueue...
destination	BWMQ.APP
correlationId	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
messageId	QU1RIGJpbGJvLnF1ZXVILswy71ACHwAg
characterSet	819
replyToQueueName	string
format	string
expiry	-1
priority	0
groupId	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
messagetype	8
backoutCount	0
feedback	0
userId	wcn00
putDateTime	Thu Jan 10 14:29:56 MST 2013
version	2
accountingToken	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=
applicationIdData	string
applicationOriginData	string
putApplicationType	28
putApplicationName	WebSphere MQ Client for Java
bytes	QldNUVNhbXBsZQ==
responsetimemillisec	10

Notice that the output from the activity is similar to the output of the put activity, with the exception of the `bytes` field which is the payload.

These two simple activities can be combined in many ways using more of their options to create sophisticated queuing applications. The following sections explain the rest of the activities in this sample application paying more attention to the nuances that can be achieved via the other options present on the activity panels.

Dynamic Destinations

The `DynamicDestinations` folder contains the Dynamic Destinations sample. Dynamic destinations are queues which do not exist until they are opened for read or write. These queues are patterned after a model queue (which you must create), and they can be of a temporary or permanent nature.

Temporary dynamic queues disappear as soon as the last subscriber and message are gone from the queue.

Permanent dynamic queues behave much like normal queues in that they survive queue manager restarts and retain their messages as would a normal queue. The main difference is that they can be created and deleted by an application, which frees administrators from some of the maintenance obligations for applications designed to use them.

A dynamic queue can be created either by the put, get or listen activities. The most common use of dynamic queues is for reply messages.

Configuring a Queue Listener to Use a Dynamic Queue

In this example, a queue listener, which is a process starter, is configured to use a dynamic queue whose name and model are taken from global variables. For dynamic queues which have a fixed instead of dynamic name, global variables are a good way to let message senders and receivers know what names to use.

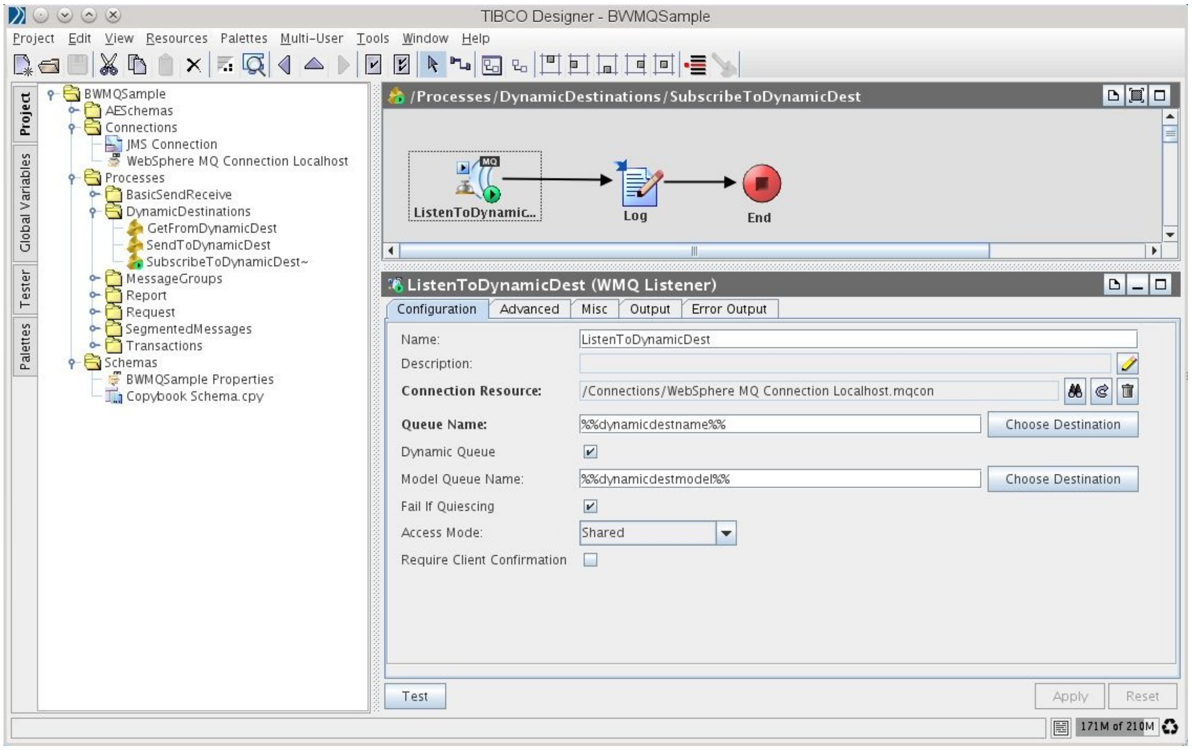
All activities that use dynamic destinations check for the existence of the queue before attempting to create it. If it exists, they open it as though it were a normal queue. In this sample, the listener will typically start before the sender and therefore will create the queue if it does not exist.

This listener monitors the specified dynamic destination and logs the arrival of any messages. Whether this queue is temporary or permanent depends on the model queue chosen. Using this sample, you can experiment with both temporary and permanent queues in order to learn about the lifetime of the resulting dynamic queue.

The `DynamicDestinations` folder has an activity called `GetFromDynamicDest` which has been configured to use a get operation instead of a listener. It must be explicitly started in order to receive the message.

Figure 15 shows the Configuration tab for the queue listener. Note that the **Test** button does not attempt to create a dynamic queue. It merely checks that the model specified exists.

Figure 15 Dynamic Destinations — Listen Activity Configuration Tab

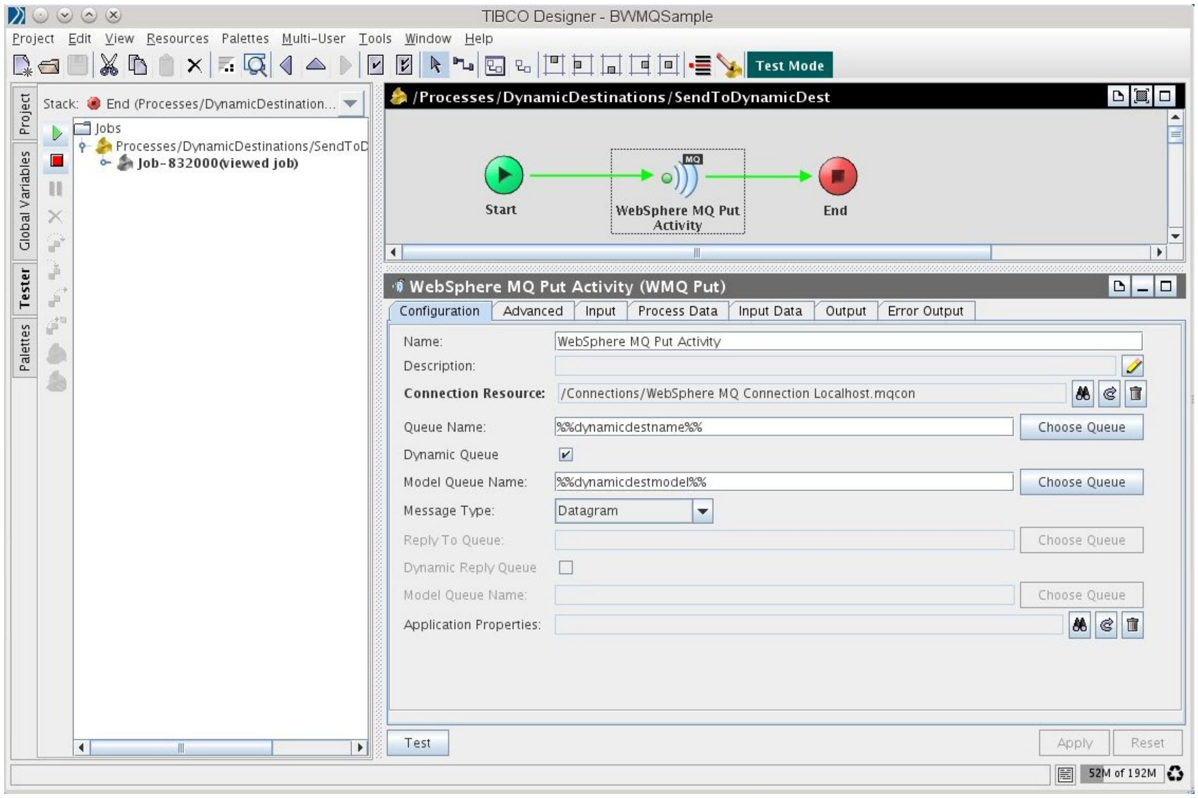


Name:	ListenToDynamicDest
Description:	
Connection Resource:	/Connections/WebSphere MQ Connection Localhost.mqcon
Queue Name:	%%dynamicdestname%%
Dynamic Queue	✓
Model Queue Name:	%%dynamicdestmodel%%
Fail If Quiescing	✓
Access Mode:	Shared
Require Client Confirmation	☐

Sending a Message to a Dynamic Queue

The put activity shown in Figure 16 sends a message to the dynamic queue listener that was created above. If the listener was started before the put activity is run, the queue already exists and the put activity simply opens it for output and places the message there. If the listener was not started first, the put activity itself creates the queue before placing the message on it.

Figure 16 Dynamic Destinations — Put to a Dynamic Queue


TIBCO Designer - BWMQSample
Project Edit View Resources Palettes Multi-User Tools Window Help
Test Mode
Project
Global Variables
Tester
Palettes
Stack: End (Processes/DynamicDestination...
Jobs
Processes/DynamicDestinations/SendToD
Job-832000(viewed job)
/Processes/DynamicDestinations/SendToDynamicDest
Start
WebSphere MQ Put Activity
End
WebSphere MQ Put Activity (WMQ Put)
Configuration
Advanced
Input
Process Data
Input Data
Output
Error Output
Choose Queue
Test
Apply
Reset
52M of 192M

Name:	WebSphere MQ Put Activity
Description:	
Connection Resource:	/Connections/WebSphere MQ Connection Localhost.mqcon
Queue Name:	%%dynamicdestname%%
Dynamic Queue	✓
Model Queue Name:	%%dynamicdestmodel%%
Message Type:	Datagram
Reply To Queue:	
Dynamic Reply Queue	☐
Model Queue Name:	
Application Properties:	

If you select a temporary model queue for this sample, two scenarios are possible regarding the lifetime of the queue:

1. If the listener activity is used to monitor the queue and it is started before the put is run, the temporary queue exists for as long as the listener is running.
2. If the explicit get activity is used to monitor the dynamic queue, it ends as soon as the message arrives. Because this leaves no client attached to an empty queue, the queue is deleted by the queue manager.

Close Options

If you change the model queue used in these activities from a temporary one to a persistent one, the resulting queue remains after the activities have finished. If using a persistent dynamic queue, you can modify this behavior using the **Close Options** dialog box in the get activity:

- If you choose `Delete` and the queue is empty after the get, the permanent dynamic queue is deleted.

- If you choose `Purge Delete`, the queue and any remaining messages are deleted.

Note that you must have the appropriate authorization to perform these actions.

Message Groups

The sample contained in the `MessageGroups` folder is very similar to the previous one except that it demonstrates the handling of large messages using message groups. Message groups are a mechanism designed to deal with the finite size of buffers in the queue manager and the channels between them. If your message is too large to fit in these buffers, it must be split into smaller messages and sent to the destination, where it can be reassembled. The Plug-in facilitates the transmission of large messages using the **Long Messages** configuration option.

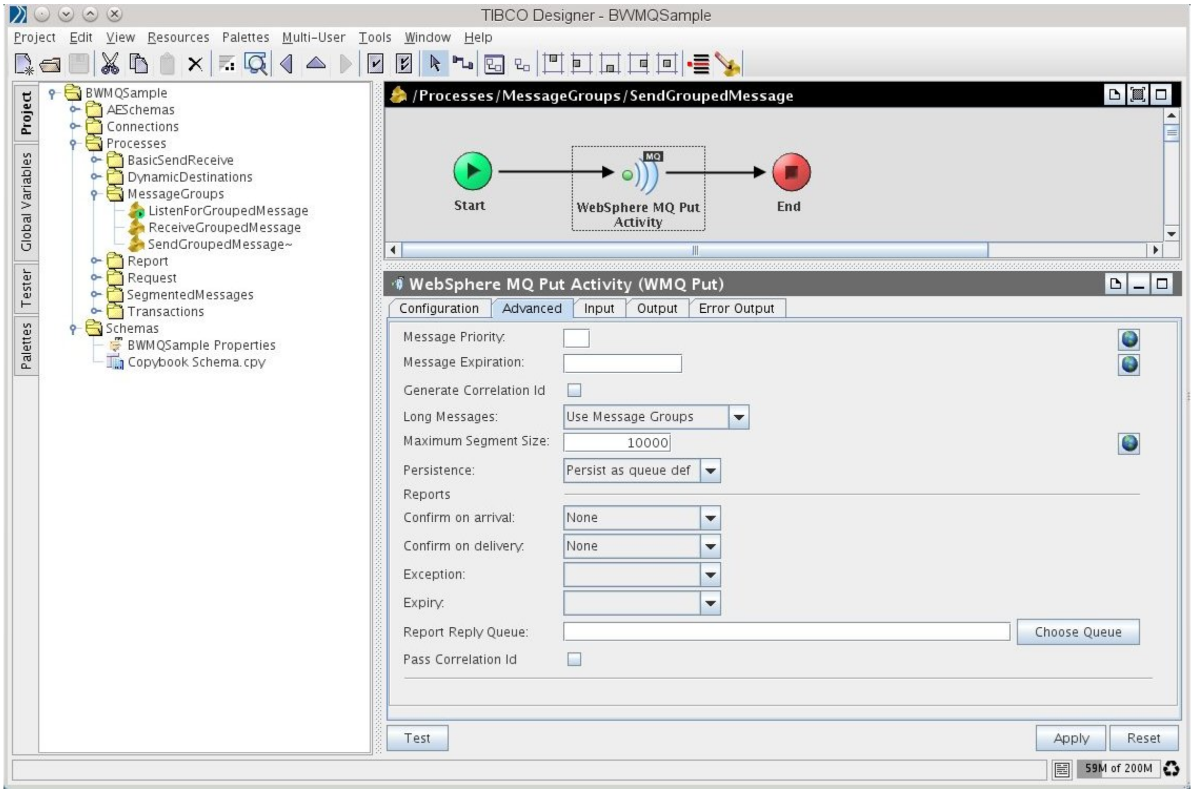
There are two options for handling long messages:

- Message Groups splits the large message into smaller messages and sends them with a unique group ID. When they are received, the application aggregates the smaller messages and reads them as a single unit. This option is described here.
- Segmented Messages do not split an existing message into smaller, ones but are used to send many closely related messages as part of a message group. The contents of this message group will be processed as closely related but individual messages. This behavior is shown in the sample activity described in Segmented Messages on page 117.

To enable Message Groups, open the Advanced tab of the put activity and select `Use Message Groups` in the **Long Messages** field. When that option is selected, the **Maximum Segment Size** field becomes visible and you can enter the size (in bytes) of the largest message your infrastructure can support (or any size smaller than that).

These settings are shown in Figure 17.

Figure 17 Message Groups — Advanced Configuration

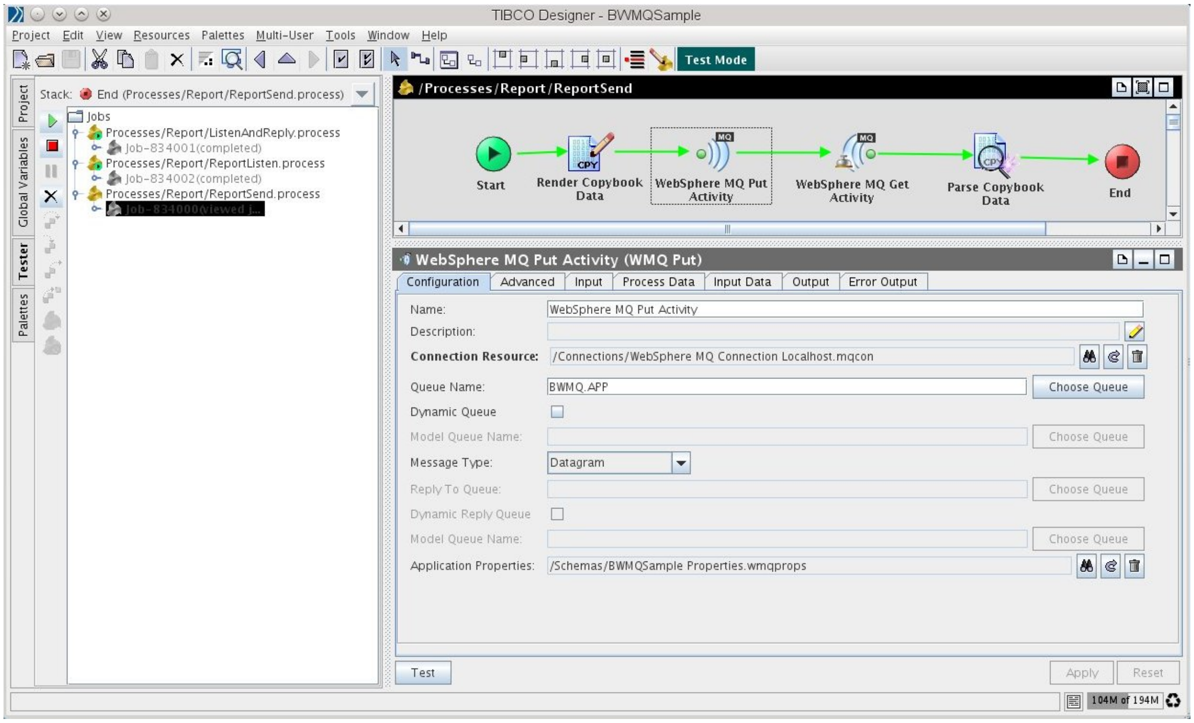


Message Priority:
Message Expiration:
Generate Correlation Id ☐
Long Messages: Use Message Groups
Maximum Segment Size: 10000
Persistence: Persist as queue def
Reports
Confirm on arrival: None
Confirm on delivery: None
Exception:
Expiry:
Report Reply Queue:
Pass Correlation Id ☐

Once the message groups options are configured, any long message you provide is split into segments of this size and sent to the destination using a message group.

If you observe the queue after such a put, you will see a series of messages on it corresponding to your large message. This sample writes a group of 100 messages on the `BWMQ.APP` queue, as shown in Figure 18.

Figure 18 Message Groups — Testing



Stack: End (Processes/Report/ReportSend.process)

Jobs
- Processes/Report/ListenAndReply.process
  - Job-834001(completed)
- Processes/Report/ReportListen.process
  - Job-834002(completed)
- Processes/Report/ReportSend.process
  - Job-834000(viewed ...

/Processes/Report/ReportSend

WebSphere MQ Put Activity (WMQ Put)

Name:	WebSphere MQ Put Activity
Description:	
Connection Resource:	/Connections/WebSphere MQ Connection Localhost.mqcon
Queue Name:	BWMQ.APP
Dynamic Queue	☐
Model Queue Name:	
Message Type:	Datagram
Reply To Queue:	
Dynamic Reply Queue	☐
Model Queue Name:	
Application Properties:	/Schemas/BWMQSample Properties.wmqprops

No special configuration of the either the listener or the get activity is required to receive a grouped message. If either activity detects such a message, it loops over the contents of the queue until all of the segments making up the original message have been received and reassembled. The final output of the get or put process started by a listener is the same as for a normal message.

Report

The `Reports` folder contains a sample that demonstrates the use of report messages.

In WebSphere MQ it is possible to set options on a put message which instruct the queue manager to provide status messages related to the progress of the message through the queuing system. These confirmation messages are automatically delivered to the destination in the message's `replyToQueueName` header field.

To configure these status messages in the Plug-in, use the settings located on the put activity's main Configuration tab. Select the type of confirmation message you want using the **Report Confirmation** field.

It is up to you to monitor the reply-to queue for confirmations.

This folder contains three activities:

- ReportSend   Send a message with a report request.
- ListenAndReply   Listen for the business message and reply to the sender.
- ReportListen   Listen for the reports.

To use this sample, load all three activities. The two process starters, ReportListen and ListenAndReply, start listening immediately. When you run the ReportSend activity, it sends a message to the ListenAndReply activity. The queue manager then sends a report to the ReportListen activity.

The TIBCO ActiveMatrix BusinessWorks Plug-in For Data Conversion

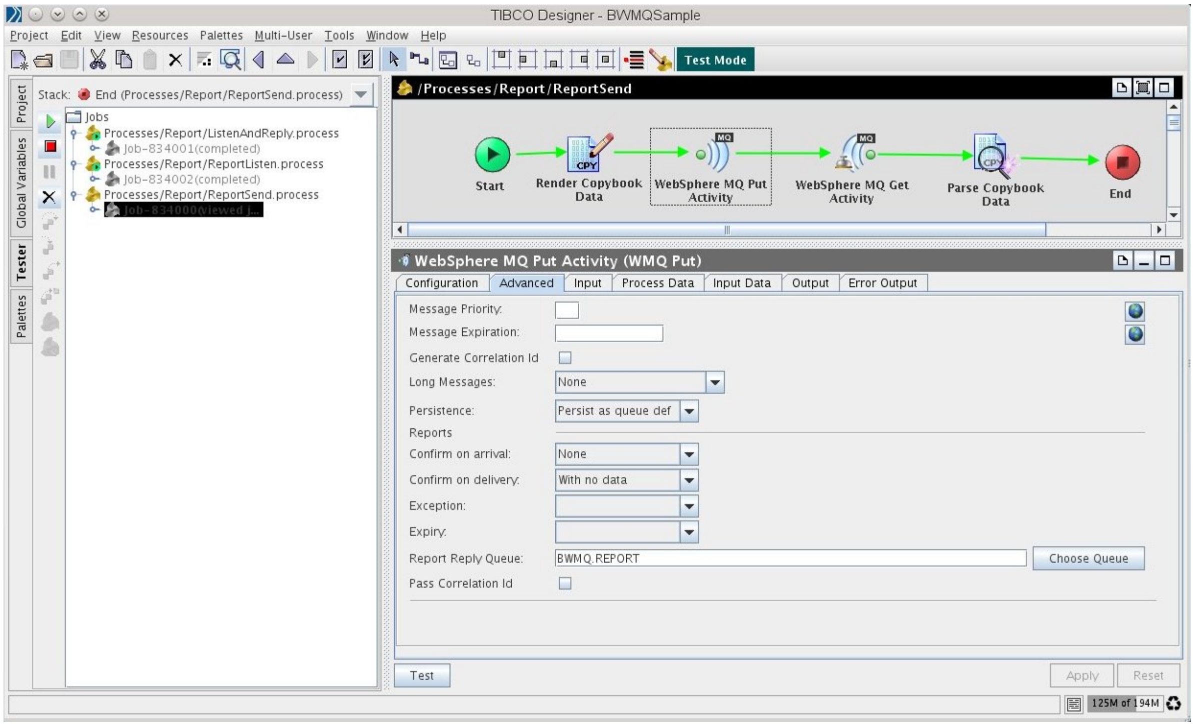
This sample also introduces the use of the The TIBCO ActiveMatrix BusinessWorks Plug-in For Data Conversion to provide input for the WebSphere MQ activities. If you do not have this Plug-in, simply remove those activities and replace the inputs with XPath statements like the ones in previous samples. See Configuring the Input Schema on page 98 for an example.

The Data Conversion Plug-in takes information from the canonical model in ActiveMatrix BusinessWorks and transforms it to a byte array which conforms to the copy book from which it was configured. See the documentation for the The TIBCO ActiveMatrix BusinessWorks Plug-in For Data Conversion for a detailed explanation of its use.

Configuring the Put Activity for Reports

Figure 19 shows the configuration of the put activity to request a "Confirm on Delivery" message from the queue manager. Note also that this activity has been configured with a schema resource in the **Application Properties** field. This causes the fields defined in that schema to be available on the Input tab of the put activity.

Figure 19 Reports — Requesting a Confirm on Delivery Report


TIBCO Designer - BWMQSample
Project Edit View Resources Palettes Multi-User Tools Window Help
Test Mode
Project
Global Variables
Tester
Palettes
Stack: End (Processes/Report/ReportSend.process)
Jobs
Processes/Report/ListenAndReply.process
Job-834001(completed)
Processes/Report/ReportListen.process
Job-834002(completed)
Processes/Report/ReportSend.process
Job-834000(viewed J...
/Processes/Report/ReportSend
Start
Render Copybook Data
WebSphere MQ Put Activity
WebSphere MQ Get Activity
Parse Copybook Data
End
WebSphere MQ Put Activity (WMQ Put)
Configuration
Advanced
Input
Process Data
Input Data
Output
Error Output
Choose Queue
Test
Apply
Reset
125M of 194M

Message Priority:
Message Expiration:
Generate Correlation Id
Long Messages: None
Persistence: Persist as queue def
Reports
Confirm on arrival: None
Confirm on delivery: With no data
Exception:
Expiry:
Report Reply Queue: BWMQ.REPORT
Pass Correlation Id

Configuring the Input Schema

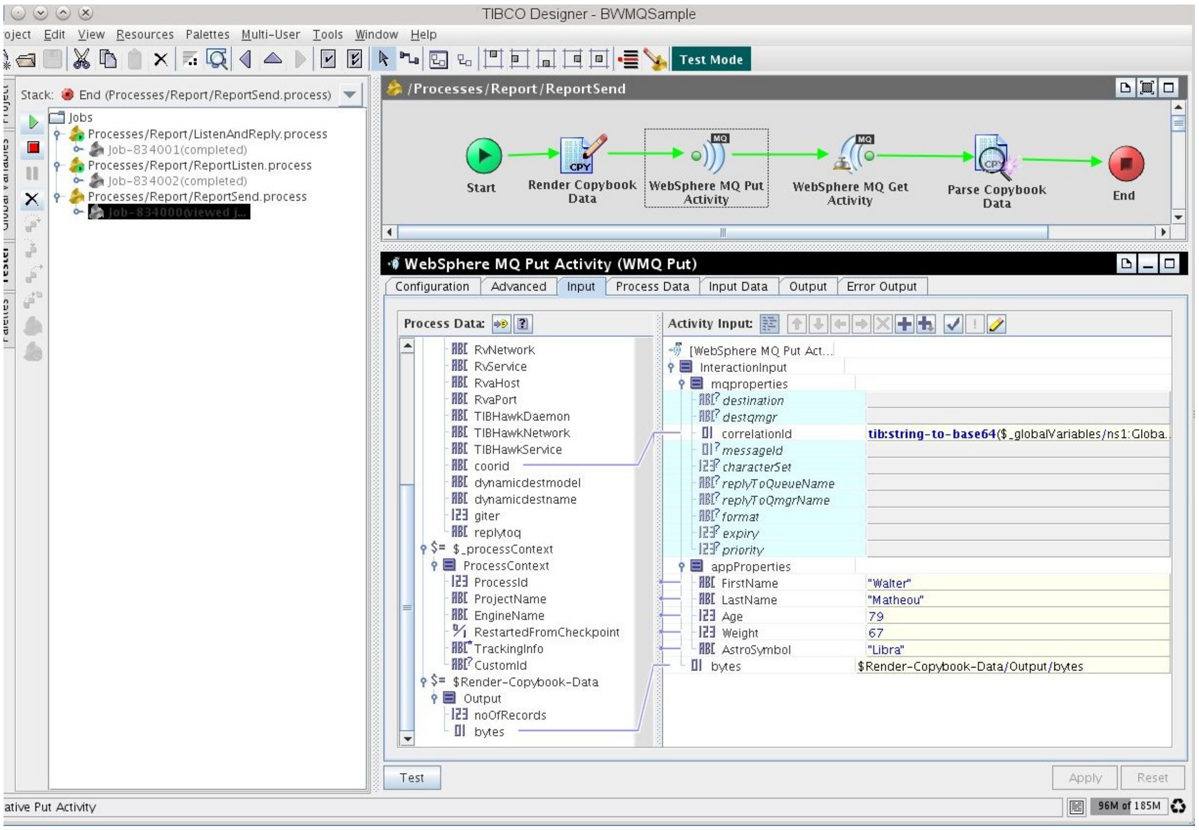
When the put activity has been associated with a schema resource using the Application Properties field, the fields defined in that schema are available on the Input tab.

The contents of the application properties node can be filled out with type checked variables, or they can be mapped from a suitable source. These variables are made available to the receiving application. If the receiving application is another WebSphere MQ activity, it can be made aware of these properties simply by mapping the same schema to that activity.

Figure 20 shows the sample Input tab, which maps these fields using the application properties schema:

- `FirstName`
- `LastName`
- `Age`
- `Weight`
- `AstroSymbol`

Figure 20 Reports — Put Activity Input Tab and Application Properties Node


TIBCO Designer - BWMQSample
Test Mode
/Processes/Report/ReportSend
Start
Render Copybook Data
WebSphere MQ Put Activity
WebSphere MQ Get Activity
Parse Copybook Data
End
WebSphere MQ Put Activity (WMQ Put)
Configuration
Advanced
Input
Process Data
Input Data
Output
Error Output
Test
Apply
Reset

Stack: End (Processes/Report/ReportSend.process)

```
Jobs
  Processes/Report/ListenAndReply.process
    Job-834001(completed)
  Processes/Report/ReportListen.process
    Job-834002(completed)
  Processes/Report/ReportSend.process
    Job-834000(viewed j...
```

Process Data:

```
RvNetwork
RvService
RvaHost
RvaPort
TIBHawkDaemon
TIBHawkNetwork
TIBHawkService
coorid
dynamicdestmodel
dynamicdestname
giter
replytoq
$_processContext
  ProcessContext
    ProcessId
    ProjectName
    EngineName
    RestartedFromCheckpoint
    TrackingInfo
    CustomId
$Render-Copybook-Data
  Output
    noOfRecords
    bytes
```

Activity Input:

[WebSphere MQ Put Act...	
InteractionInput	
mqproperties	
destination	
destqmgr	
correlationId	tib:string-to-base64($_globalVariables/ns1:Globa...
messageId	
characterSet	
replyToQueueName	
replyToQmgrName	
format	
expiry	
priority	
appProperties	
FirstName	"Walter"
LastName	"Matheou"
Age	79
Weight	67
AstroSymbol	"Libra"
bytes	$Render-Copybook-Data/Output/bytes

ative Put Activity 96M of 185M

Note that on this input schema there is a variable mapped to the `correlationId` field. The presence of this variable causes that correlation ID to be placed on the message. Message receivers can select messages by either the correlation ID or message ID. Message IDs tend to be unique, whereas correlation IDs may be common to a whole group of messages that relate to a particular unit of work.

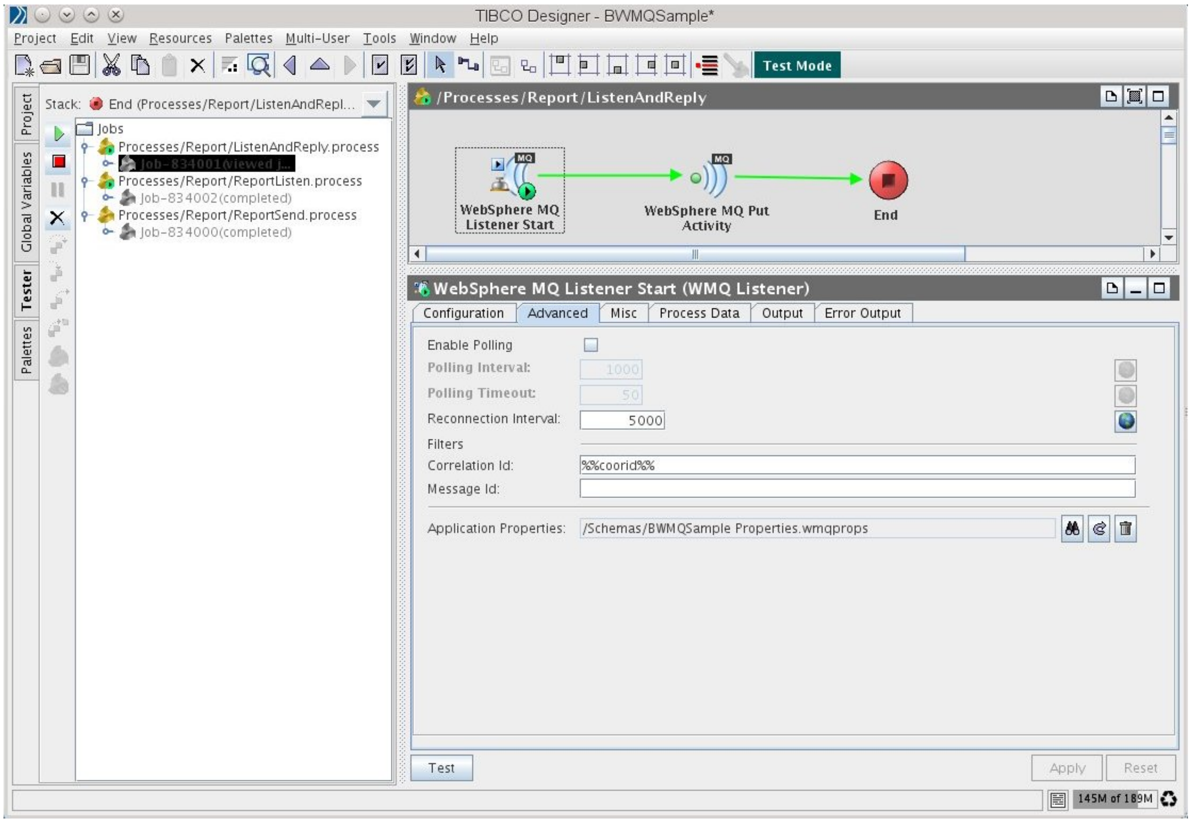
The message listener in this sample is configured to filter on this correlation ID.

Filtering Report Messages

The ListenAndReply process listens for a message and sends the requested reply message. The queue manager takes care of sending the report to the intended destination without any application involvement.

Because a process starter has no Input tab on which to map things like correlation IDs, the Advanced panel of this activity has fields for the correlation and message IDs. Figure 21 shows the settings in the Advanced tab.

Figure 21 Reports — Listener Activities and Filter Fields


TIBCO Designer - BWMQSample*
Project Edit View Resources Palettes Multi-User Tools Window Help
Test Mode
Stack: End (Processes/Report/ListenAndRepl...
Jobs
Processes/Report/ListenAndReply.process
Job-834001(viewed j...
Processes/Report/ReportListen.process
Job-834002(completed)
Processes/Report/ReportSend.process
Job-834000(completed)
/Processes/Report/ListenAndReply
WebSphere MQ Listener Start
WebSphere MQ Put Activity
End
WebSphere MQ Listener Start (WMQ Listener)
Configuration
Advanced
Misc
Process Data
Output
Error Output
Test
Apply
Reset
145M of 189M

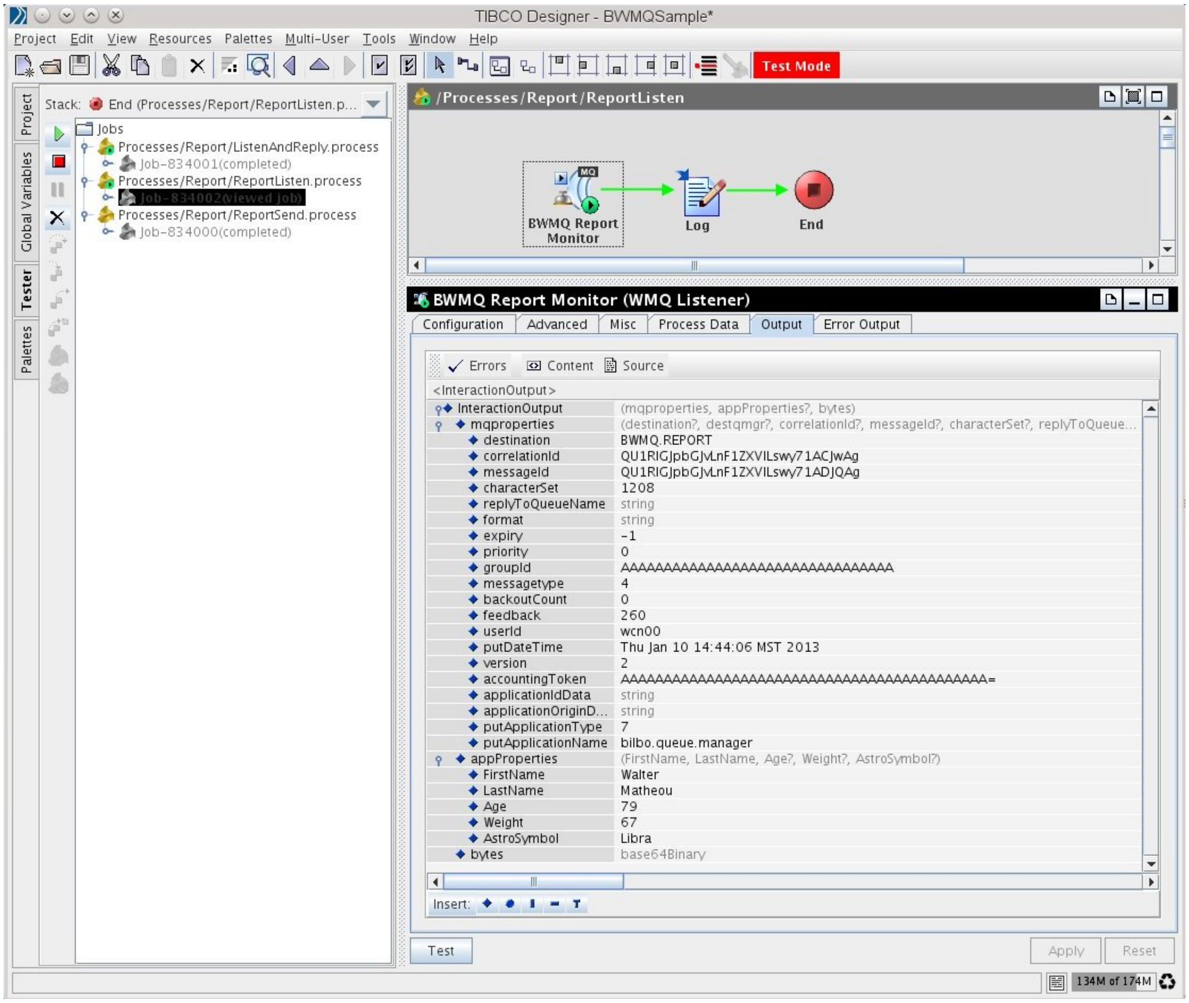
Enable Polling: ☐
Polling Interval: 1000
Polling Timeout: 50
Reconnection Interval: 5000
Filters
Correlation Id: %%coorid%%
Message Id:
Application Properties: /Schemas/BWMQSample Properties.wmqprops

Note that in WebSphere MQ these IDs are binary 24-byte fields and they are never translated at any point in the infrastructure. However, configuration variables in ActiveMatrix BusinessWorks are strings. Therefore if it is necessary to filter using a correlation ID in ActiveMatrix BusinessWorks, be careful to choose an ID that can be reliably represented as a string to this listener.

Viewing the Report Message

The Output tab of the Report Monitor activity shows the nature of the report message sent by the queue manager.

Figure 22 Reports — Report Message Output Tab


TIBCO Designer - BWMQSample*
Project Edit View Resources Palettes Multi-User Tools Window Help
Test Mode
Project
Global Variables
Tester
Palettes
/Processes/Report/ReportListen
BWMQ Report Monitor
Log
End
BWMQ Report Monitor (WMQ Listener)
Configuration Advanced Misc Process Data Output Error Output
Errors Content Source
Insert:
Test
Apply
Reset
134M of 174M

Stack: End (Processes/Report/ReportListen.p...

```
Jobs
  Processes/Report/ListenAndReply.process
    Job-834001(completed)
  Processes/Report/ReportListen.process
    Job-834002(viewed job)
  Processes/Report/ReportSend.process
    Job-834000(completed)
```

<InteractionOutput>

InteractionOutput	(mqproperties, appProperties?, bytes)
mqproperties	(destination?, destqmgr?, correlationId?, messageId?, characterSet?, replyToQueue...
destination	BWMQ.REPORT
correlationId	QU1RIGJpbGJvLnF1ZXVILswy71ACJwAg
messageId	QU1RIGJpbGJvLnF1ZXVILswy71ADJQAg
characterSet	1208
replyToQueueName	string
format	string
expiry	-1
priority	0
groupId	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
messagetype	4
backoutCount	0
feedback	260
userId	wcn00
putDateTime	Thu Jan 10 14:44:06 MST 2013
version	2
accountingToken	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=
applicationIdData	string
applicationOriginD...	string
putApplicationType	7
putApplicationName	bilbo.queue.manager
appProperties	(FirstName, LastName, Age?, Weight?, AstroSymbol?)
FirstName	Walter
LastName	Matheou
Age	79
Weight	67
AstroSymbol	Libra
bytes	base64Binary

Information about the fields presented here and in all other message receive operations is briefly documented in this manual. More detailed descriptions can be found in the IBM WebSphere MQ documentation.

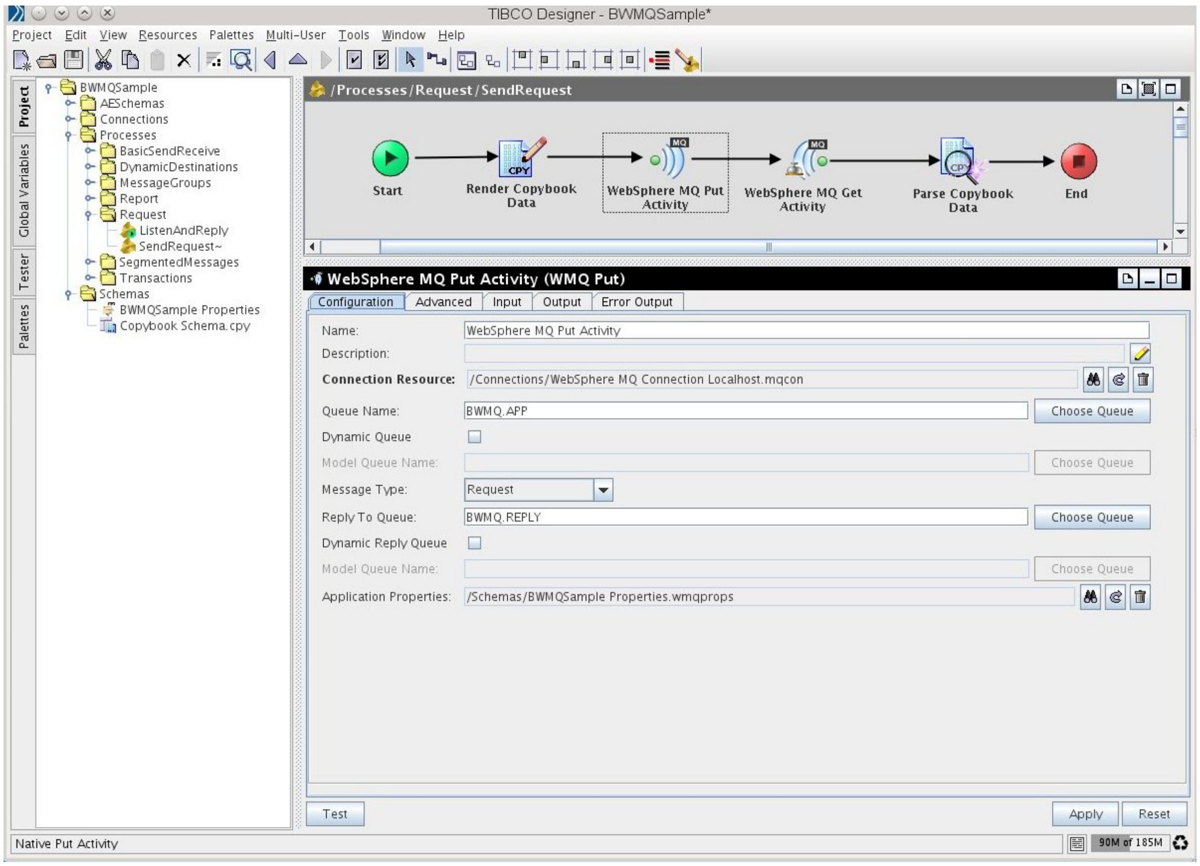
Request

A common use case in all messaging applications is for a client application to send a message to a server application from which it expects a reply. This scenario is demonstrated in the `Request` sample folder.

To make the application architecture flexible, it is best to have the requestor dictate the queue on which the reply is to be sent. This is done using either the **Reply To Queue** field of either the main Configuration tab of the put activity, or the `replyToQueueName` field in the activity's Input schema. While any message can sport a reply-to field, the special Request message type mandates its presence.

Figure 23 shows the Configuration tab for the sample put message activity. Note that the selected message type is `Request`.

Figure 23 Requests — Message Type



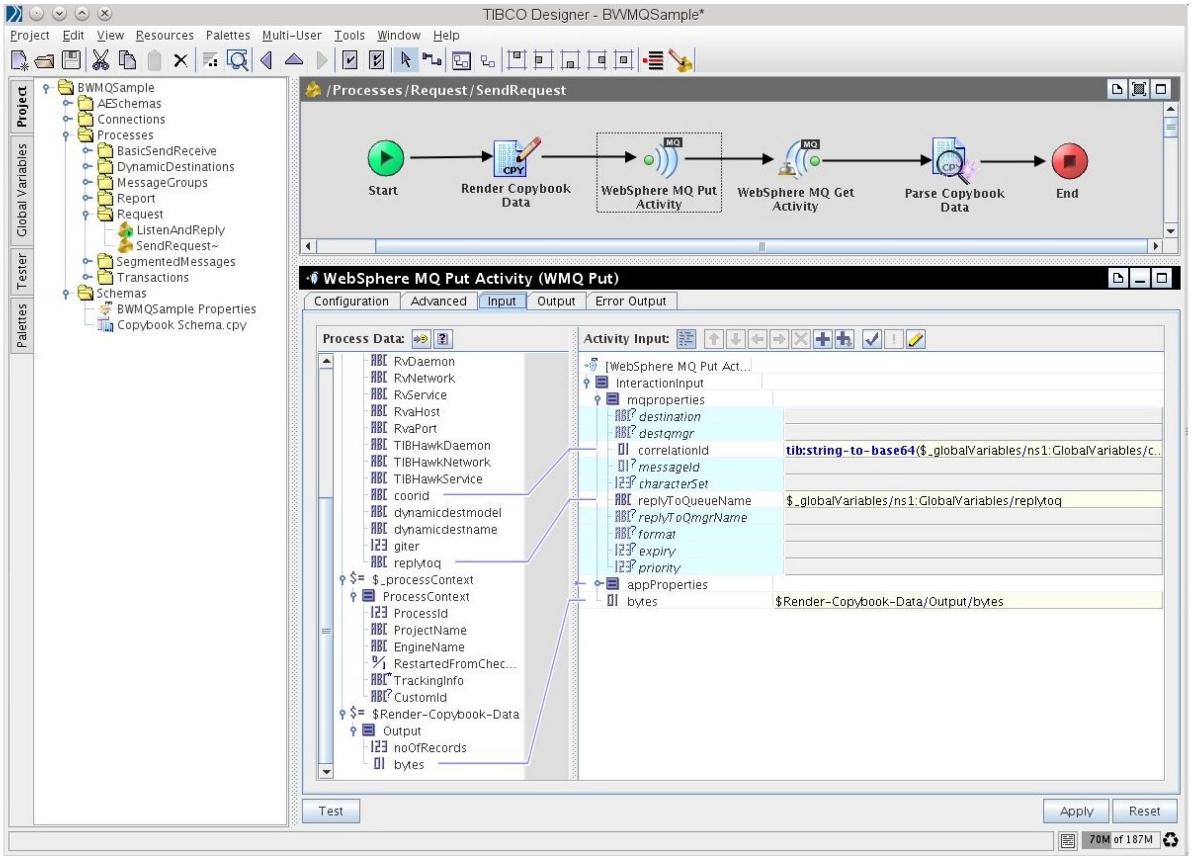
Name: WebSphere MQ Put Activity
Description:
Connection Resource: /Connections/WebSphere MQ Connection Localhost.mqcon
Queue Name: BWMQ.APP
Dynamic Queue ☐
Model Queue Name:
Message Type: Request
Reply To Queue: BWMQ.REPLY
Dynamic Reply Queue ☐
Model Queue Name:
Application Properties: /Schemas/BWMQSample Properties.wmqprops

When `Request` is selected in the **Message Type** drop-down box, the **Reply To Queue** field is enabled. However, the field is not required because it's possible to dynamically map the reply-to queue using the input schema.

Note that when the Request message type is chosen, the option for report confirmation messages is disabled. This is because report messages use the reply-to destination name, making these fields mutually exclusive.

If the specified reply to queue is dynamic, it is best to use a permanent dynamic queue in order to simplify application design. In this sample, the Reply To Queue name is mapped on the input schema using a global variable (see Figure 24). In the case where a configuration variable is present and the input schema is mapped, like this, the input schema takes precedence in all cases.

Figure 24 Requests — Mapping the Reply to Queue Name from a Global Variable



correlationId	tib:string-to-base64($_globalVariables/ns1:GlobalVariables/c...
replyToQueueName	$_globalVariables/ns1:GlobalVariables/replytoq
bytes	$Render-Copybook-Data/Output/bytes

The destination for the server activities response is dynamically mapped from the source messages reply to field. This allows the conversation between the two activities to be completely dynamic.

Segmented Messages

Segmented messages are a special class of grouped message. Simple message groups (see Message Groups on page 107) split a large message up into smaller parts to accommodate small buffers in the message transport. When received by the application, the smaller messages are aggregated and processed as a single message.

Segmented messages are intended to allow processing of individual messages which are inherently related and are intended to be processed as single unit of work. This special class of message must be processed within a group loop construct in ActiveMatrix BusinessWorks. Each message sent with the `Use Segmented Messages` indication (on the Advanced tab of the put activity) represents a member of the message group.

There are two different samples in this folder:

- These activities are intended to be run together, where one simply sends a series of messages to the other.
  - SendStreamedSegments
  - RecStreamedSegments
- TestCopyQueue and its embedded CopyQueue activity copy a segmented message group from one queue to another. This activity can be run as a standalone process.

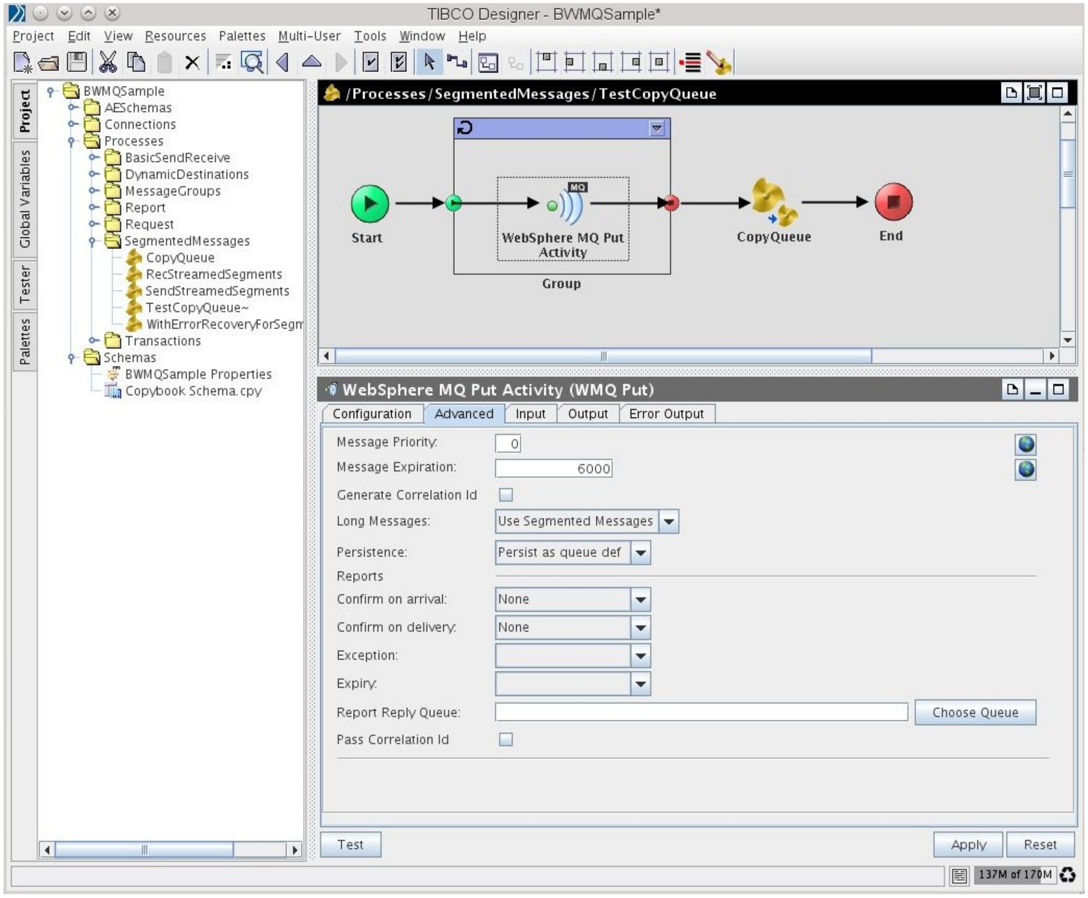
All messages sent as a segment group have the same group ID, which you will see on the output schema of each one.

Sending Segmented Messages

Segmented Messages are configured on the Advanced tab of the put activity, by selecting `Use Segmented Messages` in the **Long Messages** field. When that option is selected, the **Maximum Segment Size** field becomes visible and you can enter the size (in bytes) of the largest message your infrastructure can support (or any size smaller than that).

These settings are shown in Figure 25.

Figure 25 Segmented Messages — Put Activity Advanced Tab


TIBCO Designer - BWMQSample*
Project Edit View Resources Palettes Multi-User Tools Window Help
/Processes/SegmentedMessages/TestCopyQueue
Start
WebSphere MQ Put Activity
Group
CopyQueue
End
WebSphere MQ Put Activity (WMQ Put)
Configuration
Advanced
Input
Output
Error Output
Choose Queue
Test
Apply
Reset
137M of 170M

BWMQSample
- AESchemas
- Connections
- Processes
  - BasicSendReceive
  - DynamicDestinations
  - MessageGroups
  - Report
  - Request
  - SegmentedMessages
    - CopyQueue
    - RecStreamedSegments
    - SendStreamedSegments
    - TestCopyQueue~
    - WithErrorRecoveryForSegm
  - Transactions
- Schemas
  - BWMQSample Properties
  - Copybook Schema.cpy

Message Priority:	0
Message Expiration:	6000
Generate Correlation Id	☐
Long Messages:	Use Segmented Messages
Persistence:	Persist as queue def
Reports	
Confirm on arrival:	None
Confirm on delivery:	None
Exception:	
Expiry:	
Report Reply Queue:	
Pass Correlation Id	☐

Message groups must be terminated by a message that is marked as the "last message". This setting is assigned using the `lastSegment` field, present on the input schema. When `lastSegment` is set to `true`, the get application terminates the segmented message group. To set the `lastSegment` field on the final message in the segmented group, we use the following XPath statement using the loop counter's loop variable:

```
if($iter = 10) then "true" else "false"
```

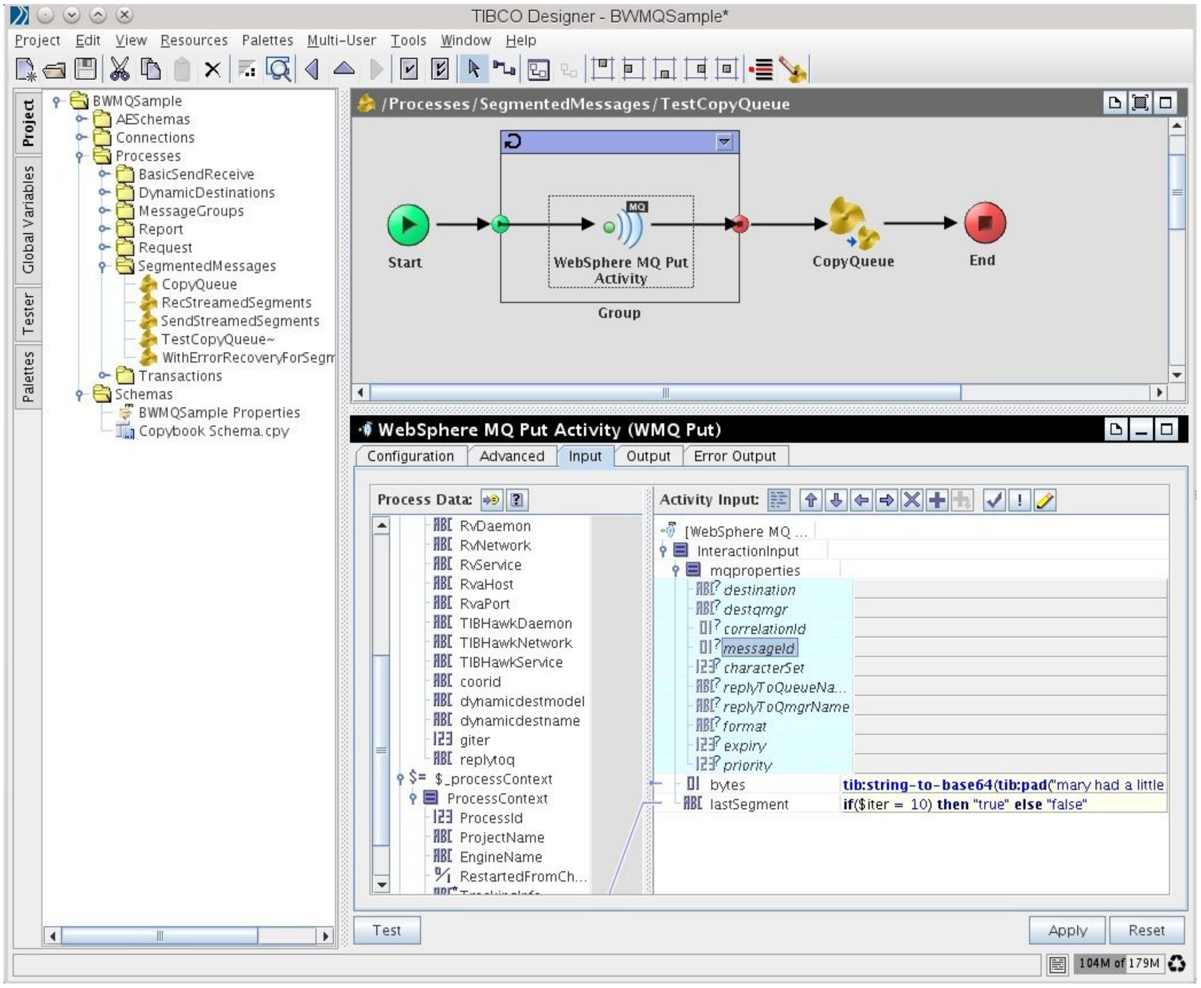
The loop is set to "loop until true" using this expression:

```
$iter = 10
```

Note that it would be better to use some global or process variable field rather than hard coding the number 10 in both places. However, for clarity's sake the sample hard codes the number.

Figure 26 shows the Input tab for the Segmented Messages sample.

Figure 26 Segmented Messages — Assigning the lastSegment Field



Process Data:

```
RvDaemon
RvNetwork
RvService
RvaHost
RvaPort
TIBHawkDaemon
TIBHawkNetwork
TIBHawkService
coorid
dynamicdestmodel
dynamicdestname
giter
replytoq
$_processContext
  ProcessContext
    ProcessId
    ProjectName
    EngineName
    RestartedFromCh...
```

Activity Input:

```
[WebSphere MQ ...
  InteractionInput
    mqproperties
      destination
      destqmgr
      correlationId
      messageId
      characterSet
      replyToQueueNa...
      replyToQmgrName
      format
      expiry
      priority
    bytes          tib:string-to-base64(tib:pad("mary had a little
    lastSegment    if($iter = 10) then "true" else "false"
```

The net result of this activity is to put a series of associated messages on the queue whose input is related but not intended to be merely concatenated.

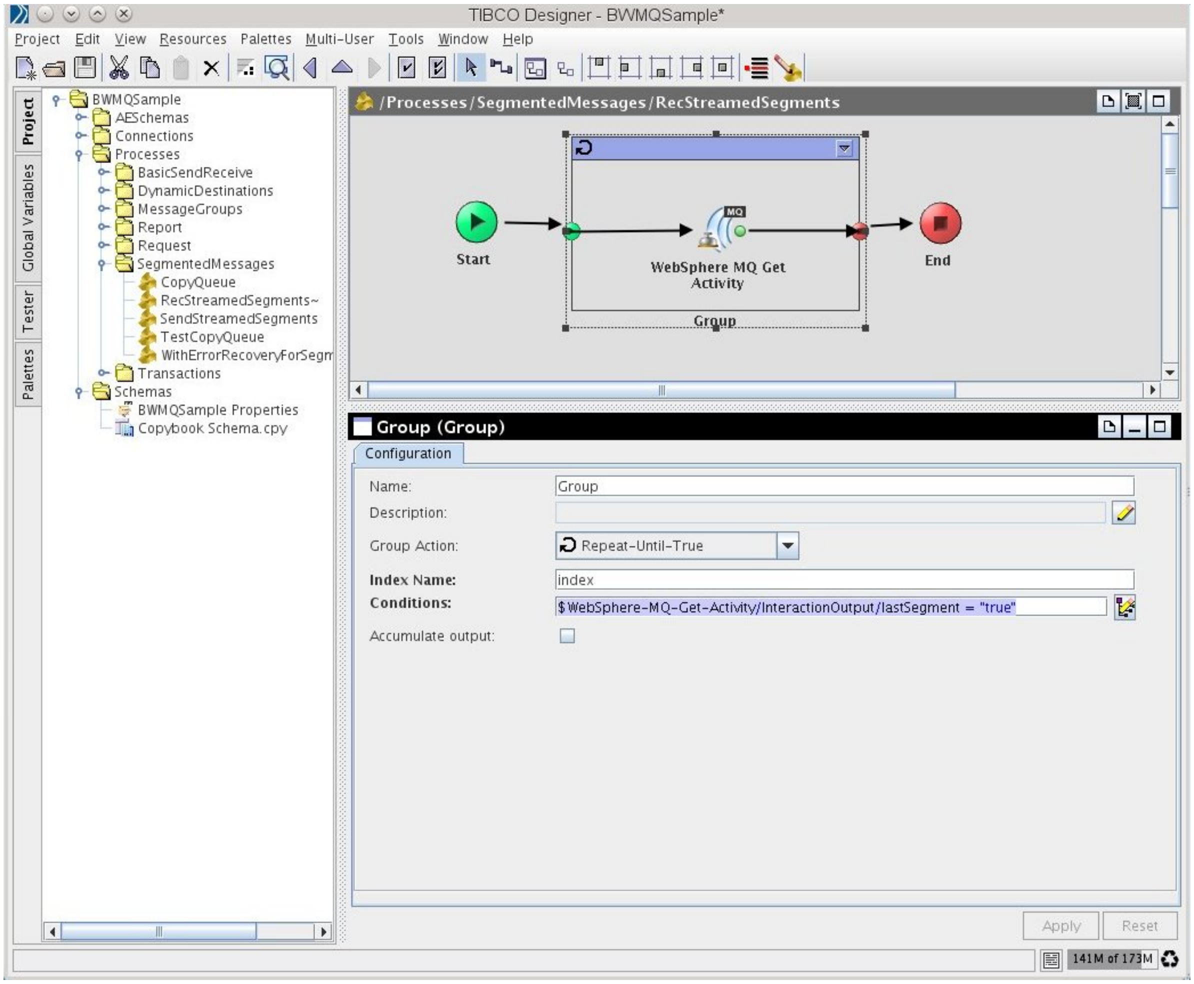
Receiving Segmented Messages

The get activity for segmented messages must also be used within a loop. Additionally, the loop must be configured to terminate when the last message in the group has been processed. This group is configured to "Repeat Until True" and the evaluation statement is:

```
$WebSphere-MQ-Get-Activity/InteractionOutput/lastSegment = "true"
```

This allows the get activity to terminate the loop by setting its output variable `lastSegment` to `true`.

Figure 27 Segmented Messages — Get Activity Designed to Terminate on lastSegment



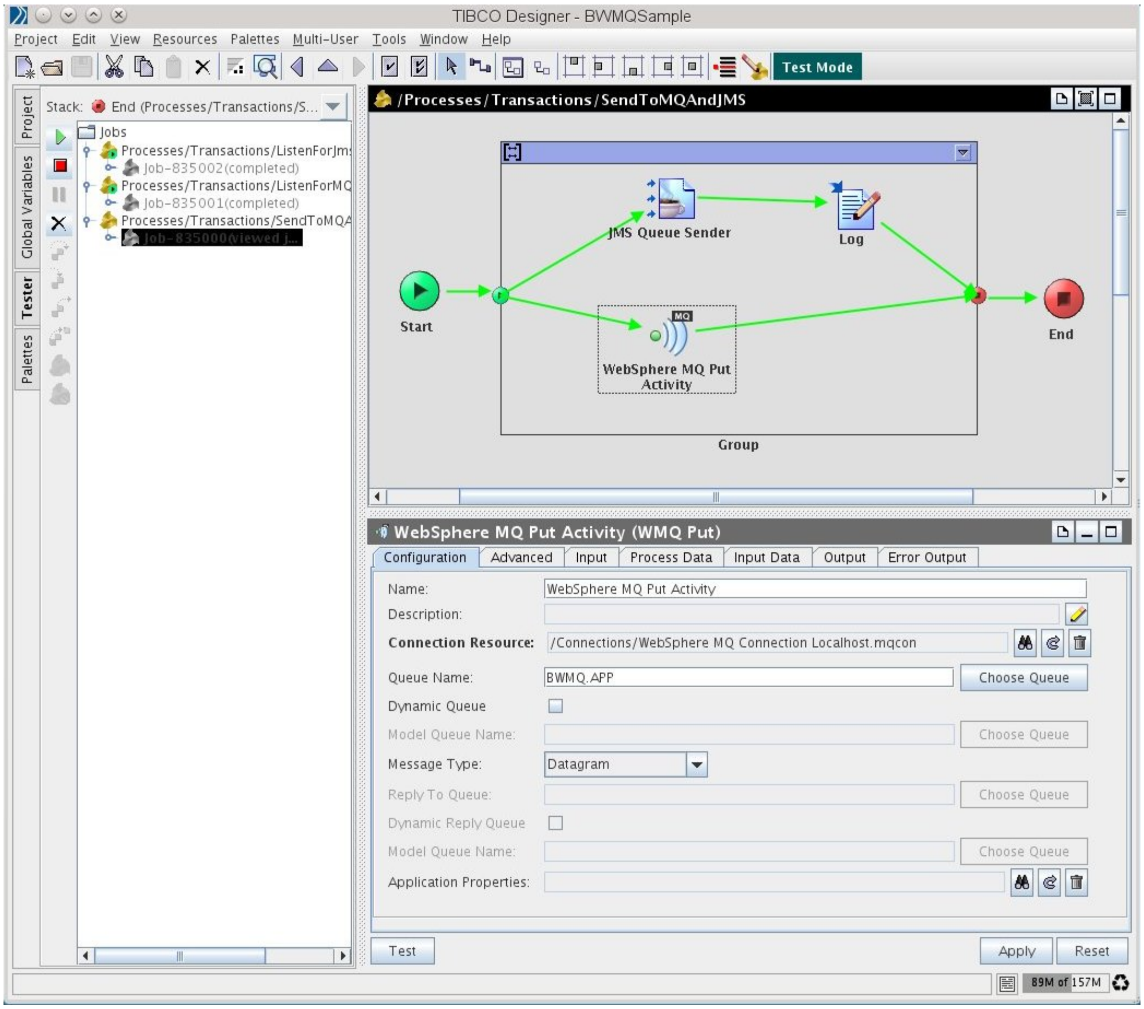
Name: Group
Description:
Group Action: Repeat-Until-True
Index Name: index
Conditions: $WebSphere-MQ-Get-Activity/InteractionOutput/lastSegment = "true"
Accumulate output: ☐

Transactions

WebSphere MQ queue managers support the XA Transaction architecture, and both the get and put activities in this Plug-in can also participate in XA transactions. The Transactions folder contains sample put and get activities that use XA transactions.

To run this sample, you must have installed and configured the TIBCO ActiveMatrix BusinessWorks XA Transaction Manager. Get and put activities that fall inside an XA transaction group automatically participate in the transaction. This sample demonstrates commit coordination between JMS and WebSphere MQ.

Figure 28 XA Transaction — Configuration

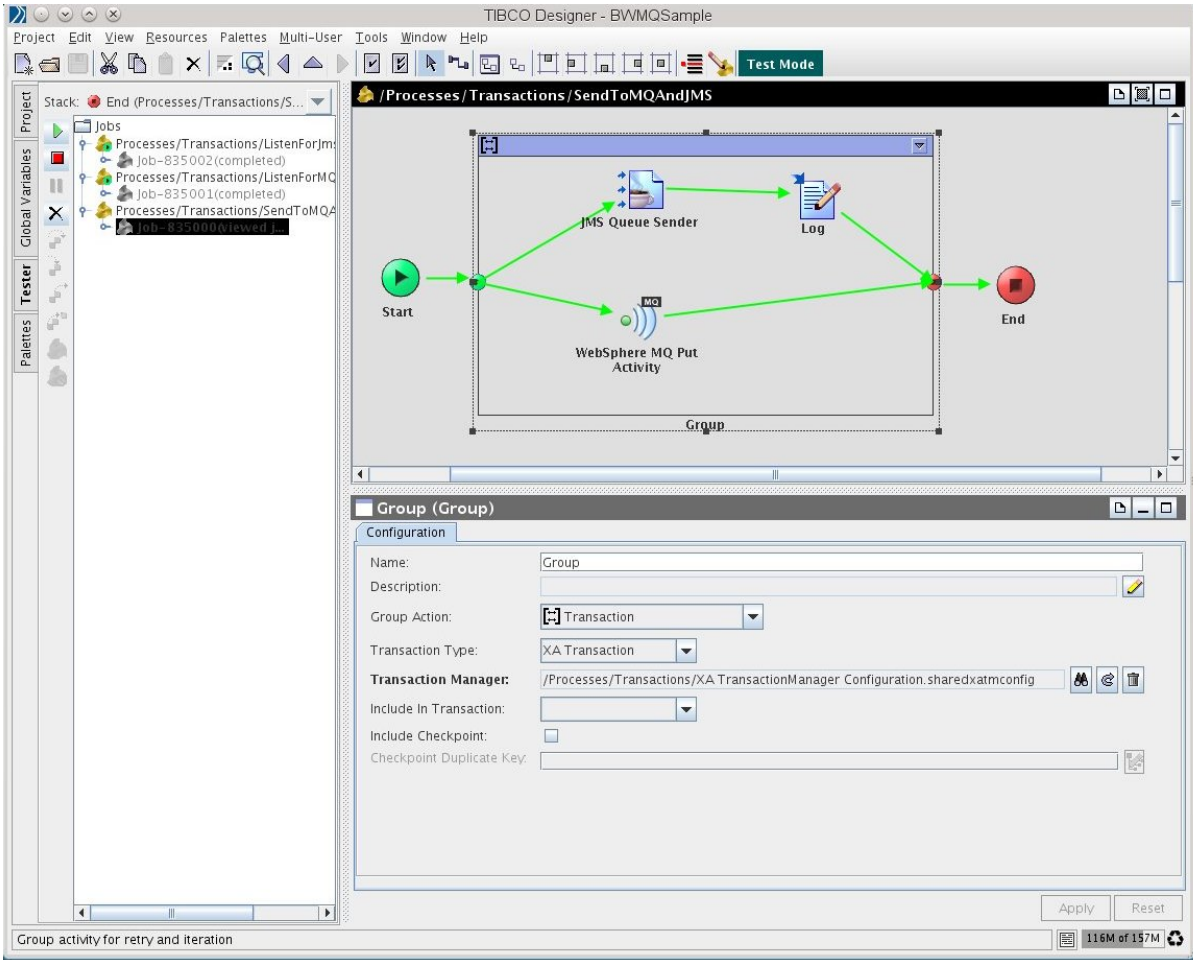

TIBCO Designer - BWMQSample
Project Edit View Resources Palettes Multi-User Tools Window Help
Test Mode
Project
Global Variables
Tester
Palettes
Stack: End (Processes/Transactions/S...
Jobs
Processes/Transactions/ListenForJms
Job-835002(completed)
Processes/Transactions/ListenForMQ
Job-835001(completed)
Processes/Transactions/SendToMQA
Job-835000(viewed j...
/Processes/Transactions/SendToMQAndJMS
Start
JMS Queue Sender
Log
WebSphere MQ Put Activity
End
Group
WebSphere MQ Put Activity (WMQ Put)
Configuration
Advanced
Input
Process Data
Input Data
Output
Error Output
Test
Apply
Reset
89M of 157M

Name: WebSphere MQ Put Activity
Description:
Connection Resource: /Connections/WebSphere MQ Connection Localhost.mqcon
Queue Name: BWMQ.APP [Choose Queue]
Dynamic Queue ☐
Model Queue Name: [Choose Queue]
Message Type: Datagram
Reply To Queue: [Choose Queue]
Dynamic Reply Queue ☐
Model Queue Name: [Choose Queue]
Application Properties:

In this case both a simple JMS and WebSphere MQ message are placed on queues. There are monitoring activities here, should you wish to start them. When run, you will observe that both the JMS and WebSphere MQ messages are sent to their respective queues.

The run shown in Figure 28 indicates that messages to both queues were successfully delivered.

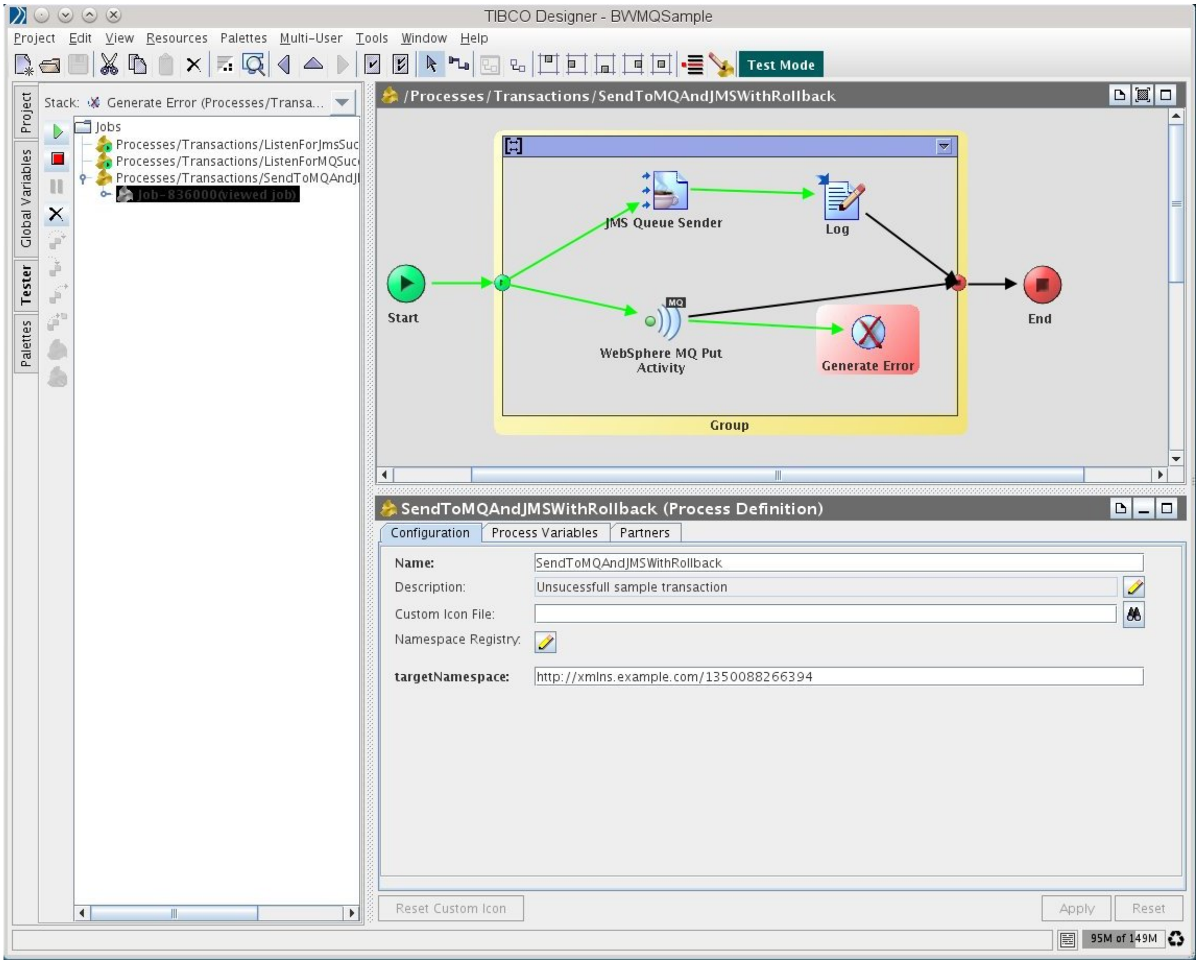
Figure 29 XA Transaction — Success


TIBCO Designer - BWMQSample
Project Edit View Resources Palettes Multi-User Tools Window Help
Test Mode
Stack: End (Processes/Transactions/S...
Jobs
Processes/Transactions/ListenForJms
Job-835002(completed)
Processes/Transactions/ListenForMQ
Job-835001(completed)
Processes/Transactions/SendToMQA
Job-835000(viewed j...
/Processes/Transactions/SendToMQAndJMS
JMS Queue Sender
Log
Start
End
WebSphere MQ Put Activity
Group
Group (Group)
Configuration
Apply
Reset
Group activity for retry and iteration
116M of 157M

Name: Group
Description:
Group Action: Transaction
Transaction Type: XA Transaction
Transaction Manager: /Processes/Transactions/XA TransactionManager Configuration.sharedxatmconfig
Include In Transaction:
Include Checkpoint:
Checkpoint Duplicate Key:

In Figure 30, an intentional error has been introduced to cause an XA rollback on both resource managers. The two receivers have not received any messages and the respective queues are empty.

Figure 30 XA Transaction — Failed


TIBCO Designer - BWMQSample
Project Edit View Resources Palettes Multi-User Tools Window Help
Test Mode
Project
Global Variables
Tester
Palettes
Stack: Generate Error (Processes/Transa...
Jobs
Processes/Transactions/ListenForJmsSuc
Processes/Transactions/ListenForMQSuc
Processes/Transactions/SendToMQAndJI
Job-836000(viewed job)
/Processes/Transactions/SendToMQAndJMSWithRollback
JMS Queue Sender
Log
Start
End
WebSphere MQ Put Activity
Generate Error
Group
SendToMQAndJMSWithRollback (Process Definition)
Configuration
Process Variables
Partners
Reset Custom Icon
Apply
Reset
95M of 149M

Name:	SendToMQAndJMSWithRollback
Description:	Unsucessfull sample transaction
Custom Icon File:	
Namespace Registry:	
targetNamespace:	http://xmlns.example.com/1350088266394