NASTEL
Driving Business Transaction Performance ®

AP/SR 121.000

AutoPilot® MQSpeedtest for IBM MQ

Quick Start Guide

Member of AutoPilot® On-Demand for IBM MQ Family

Version 1.2.1

PUBLISHED BY:

RESEARCH & DEVELOPMENT

NASTEL TECHNOLOGIES, INC.

88 SUNNYSIDE BLVD, SUITE 101

PLAINVIEW, NY 11803

DOCUMENT TITLE: **AUTOPILOT MQSPEEDTEST FOR IBM MQ QUICK START GUIDE**

VERSION: **1.2.1**

DOCUMENT RELEASE DATE: **APRIL 2019**

NASTEL DOCUMENT NUMBER: **AP/SR 121.000**

ACKNOWLEDGEMENTS:

Contents

Figures

Tables

Chapter 1: Introduction

Welcome to the *Nastel AutoPilot® MQSpeedtest for IBM MQ Quick Start Guide*. This guide describes the installation, configuration, and usability of the MQSpeedtest utility. Please review this guide carefully before using the product.

1.1 How This Guide is Organized

Chapter 1: Identifies the users and history of the document and supplies support and reference information.

Chapter 2: Contains a brief description of MQSpeedtest.

Chapter 3: Provides instructions for downloading and deploying MQSpeedtest.

Chapter 4: Explains how to run MQSpeedtest.

Chapter 5: Explains how to optionally integrate MQSpeedtest with AutoPilot.

Chapter 6: Explains how to integrate MQSpeedtest with Nastel XRay.

Chapter 7: Provides troubleshooting information.

Appendix A: Installing MQSpeedtest for MQ on z/OS

Index: Contains an alphanumeric cross-reference of all topics and subjects of importance.

1.2 History of This Document

Table 1-1. Document History

Release Date	Document Number	Version	Summary
August 2012	AP/SR 100.001	1.0	Initial release
January 2013	AP/SR 100.002	1.0	Added support for Solar and z/OS (Mantis 7644)
June 2013	AP/SR 110.001	1.1.0	Errata
August 2017	AP/SR 110.002	1.1.0	Update Nastel's street address and technical support link.
April 2018	AP/SR 120.001	1.2.0	Update for 1.2.0, added additional troubleshooting information. The product was renamed MQSpeedtest from MQSonar, but the commands and directories still use "MQSonar" and "mqsonar".
May 2018	AP/SR 120.002	1.2.0	Errata
November 2018	AP/SR 120.003	1.2.0	Errata
April 2019	AP/SR 121.000	1.2.1	Updates for new product names.

1.3 Release Notes

See **`README.htm`** files on your installation.

1.4 Intended Audience

This document is intended for personnel deploying, running, administering, and maintaining IBM MQ. The user should be familiar with:

- Target operating system environment
- Procedures for installing software on the target platform
- Configuring and using IBM MQ Queue Managers.

1.5 Technical Support

For technical support, contact Nastel at https://www.nastel.com/company/contact-us/.

Chapter 2: About MQSpeedtest

2.1 Introduction

The purpose of the MQSpeedtest utility is to measure various times related to queue manager message flow. Much like traditional sonar technology, this is completed by sending a ping which is picked up by the target application and echoed back. By measuring the time spent it can determine various characteristics of the message path. In IBM MQ terms, it sends messages to a listening application which replies with a response to those messages. MQSpeedtest (previously named MQSonar) captures various times along the message path as well as the round-trip times.

The echo application can be the IBM MQ command server or the MQSpeedtest echo program.

- When using the command server, MQSpeedtest can be run without any changes to the queue manager. However, this requires that the utility be run under an ID with administrative privileges.
- When using the MQSpeedtest echo program to generate the replies, minimal changes to the queue manager are required. In many cases, these changes may already be in place and no actual changes are required. The advantage of using the echo program is that it can be run by any user that has put/get privileges.

MQSpeedtest creates a batch of messages and submits them to IBM MQ. The queue used can be configured as local, alias, cluster or remote queue across one or more queue managers as long as it points eventually to the target queue being processed by the echo application.

Users can execute as many instances of MQSpeedtest as needed, to collect statistics about different queues' performance. The following are some basic configurations that can be implemented using MQSpeedtest:

- Single queue manager (Figure 2-1):
  - determine if the queue manager is up and responding
  - determine standard throughput for a (set of) queues
  - compare the behavior of different queue configurations such as using persistent versus non-persistent messages.

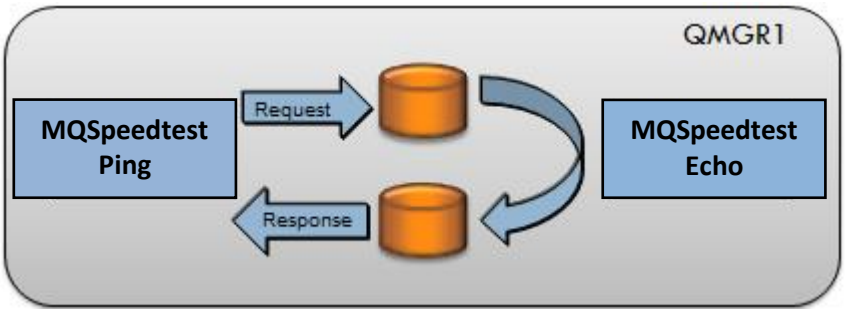

Figure 2-1. Single Queue Manager Flow

Multiple queue managers (Figures 2-2 and 2-3):

- Identify slowdowns in inter-queue manager communication
- Identify queue managers that contribute to delays
- Identify differences in behaviors of different queue managers
- Verify that a path from one sending application to the receiving application is properly configured.

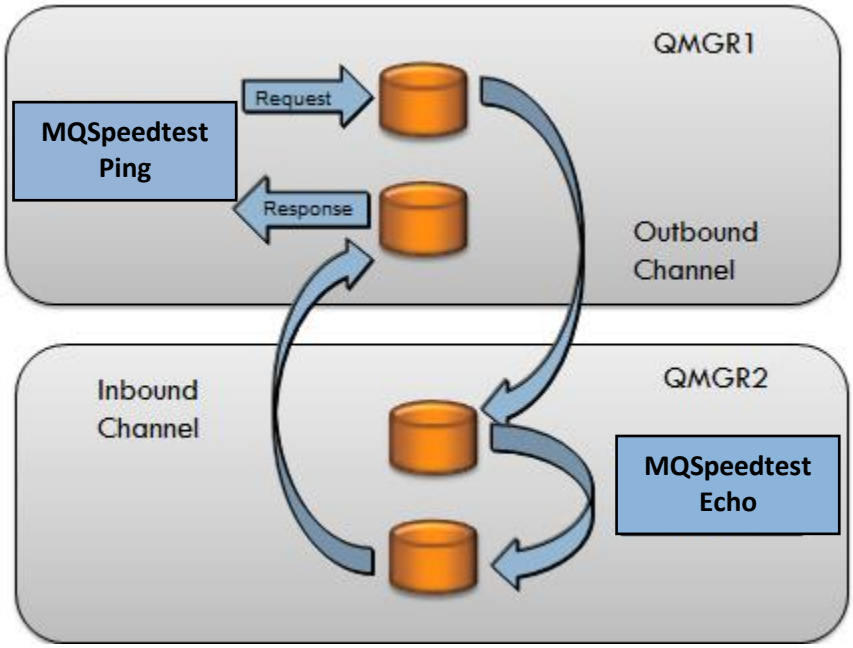

Figure 2-2. Two Queue Manager Flow Example

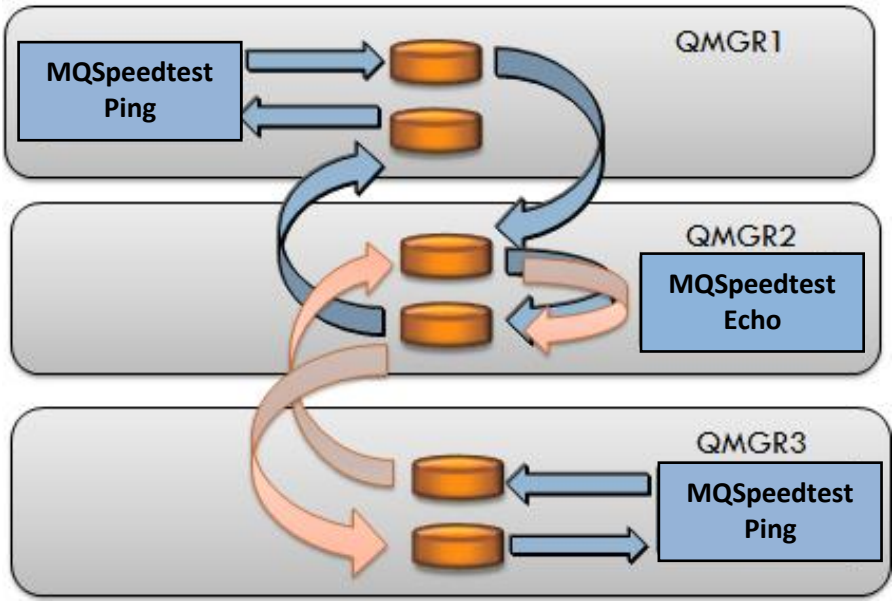

Figure 2-3. Multiple Queue Manager Flow Example

As a standalone utility, MQSpeedtest displays its output to the console which can be captured into a file and reviewed, or processed by scripts to provide a more permanent record of activity.

MQSpeedtest can optionally be integrated with Nastel AutoPilot to provide a more comprehensive solution, including alerting when abnormal conditions occur, providing graphical views of the environment, and capturing historical trending. This is covered in Chapter 5, Integrating MQSpeedtest with AutoPilot M6.

Appendix B contains a sample configuration for MQSpeedtest using sender-receiver channels between two queue managers.

This page intentionally left blank.

Chapter 3: Installing MQSpeedtest

3.1 Pre-requisites

MQSpeedtest requires:

- On the MQSpeedtest ping component machine:
  - MQSpeedtest can be installed on an MQ server or an MQ client machine. MQSpeedtest is built as both a server application and a client application, which supports client connections.
  - The MQSpeedtest installation machine must be running on one of the supported platforms (AIX, Linux, Solaris, Windows, or z/OS).
  - The MQ client library must be installed.
  - If the streams option will be used to integrate with Nastel XRay (see Chapter 6), Java must be installed, version 1.8.0 or later.
- IBM MQ command server running with
  - Administrative permission to submit messages to command input queue
  - Authority to issue the PCF MQCMD_PING_Q_MGR command

  or
- One or more instances of the MQSpeedtest echo application executing with
  - Authority to put and get messages to the request and response queues.
- Know your target queue managers authentication requirements. Check the CONNAUTH attribute and the indicated AUTHINFO settings, including CHKLOCL and CHKCLNT. You may need to provide a valid user ID and password for MQSpeedtest to access the queue manager. See section Troubleshooting, problem 3, item d.

	**IMPORTANT!**	When measuring message performance over remote queues: local and remote queue managers must have a pair of sender/receiver channels defined and active that allows the local and remote queue managers to exchange messages. Alternatively, a client connection can be used when running the script mqspeedtest.cmd (or .sh).

3.2 Installing MQSpeedtest on Windows

The installation is provided as a zip file, `MQSpeedtest_`*`version`*`_WIN_x86-64.zip`, that can be installed at any location on your system. Either:

a) copy the zip file to the desired *install_dir* (for example C:\), right click on the file and select **Extract Here…**; or

b) copy the zip file to a temp directory, right click on the file and select **Open** (or **Open with WinRAR**), then drag the `nastel` folder to the *install_dir* in a Windows Explorer window.

After installation, you will have directory *install_dir*/nastel/mqspeedtest, wherein you can execute the provided scripts and utilities (mqspeedtest.cmd, mqspeedtest_echo.cmd, nsqping.exe, nsrpl_32.exe and nscrpl_32.exe).

Required Environment Variables
You must set the PATH environment variable to include the mqspeedtest\bin directory for the system to be able to locate the executables and dependent DLLs.

Example:

```
set PATH=C:\nastel\mqspeedtest\bin;%PATH%
```

If you will be using the 'stream' option (see section 4.3), set the PATH and JAVA_HOME environment variables to define where the Java executable and JRE directory are located, respectively:

Example:

```
set PATH=C:\Program Files\java\jre1.8.0_201\bin;%PATH%
set JAVA_HOME=C:\Program Files\java\jre1.8.0_201\bin
```

Customize ReplyTo Queues (optional)

By default MQSpeedtest temporary dynamic queues are created on the target queue manager using the system default model queue, `SYSTEM.DEFAULT.MODEL.QUEUE`. This queue is used to create three temporary dynamic queues with names `NASTEL.REPLY.PING.`*`XXXhhhh`*..., where *XXX* is CMD, INQ, and HASH and *hhhh* is a hexadecimal pattern generated by the queue manager. These queues are automatically deleted after MQSpeedtest ends.

If you want to customize the prefix of these queue names (instead of NASTEL), called the high level and middle level qualifiers, edit and enable the HLQ and MLQ lines in *`install_dir`*`\config\groups\mqqual.ini`

Example:

```
NASTEL.MODEL.QUEUE=SYSTEM.DEFAULT.MODEL.QUEUE
NASTEL.HLQ=MQSPDTST
NASTEL.MLQ=ECHO
```

This would result in queue names of the form `MQSPDTST.ECHO.REPLY.PING.`*`XXXhhhh`*...

3.3 Installing MQSpeedtest on UNIX/Linux

The installation is provided as a compressed tar file, `MQSpeedtest_`*`version`*`_WIN_x86-64.tar.gz`, which can be installed and uncompressed at any location on your system. After installation, you will have a directory *install_dir/*nastel/mqspeedtest, wherein you can execute the provided scripts and utilities (mqspeedtest.sh, mqspeedtest_echo.sh, nsqping, nsrpl_32 and nscrpl_32).

For example, when installing on Red Hat Linux, copy the compressed file to the /tmp directory, then run the following commands:

```
cd  install_dir (for example, /opt)
gunzip*  /tmp/MQSpeedtest_version_Linux_RHEL_version.tar.gz
tar -xvf /tmp/MQSpeedtest_version_Linux_RHEL_version.tar
```

and a nastel/mqspeedtest folder will be created in the /opt directory.

***** When installing on AIX and Solaris, use `uncompress`.

Required Environment Variables

You must set the PATH and library environment variables to include the mqspeedtest/bin and mqspeedtest/lib directories, respectively, for the system to be able to locate the executables and dependent libraries. Example:

```
PATH=/opt/nastel/mqspeedtest/bin:$PATH
LD_LIBRARY_PATH=/opt/nastel/mqspeedtest/lib:$LD_LIBRARY_PATH
export PATH LD_LIBRARY_PATH
```

If you will be using the 'stream' option (see section 4.3), set the PATH and JAVA_HOME environment variables to define where the Java executable and JRE directory are located, respectively:
Example:

```
PATH=/usr/java/jre1.8.0_201/bin:$PATH
JAVA_HOME=/usr/java/jre1.8.0_201
export PATH JAVA_HOME
```

Customize ReplyTo Queues (optional)
See above description for Windows**.**

3.4 Installing MQSpeedtest on z/OS

The installation has three steps:

1. Transfer dataset files to z/OS
2. Create PDS files
3. Configure the JCL.

Refer to Appendix A for detailed installation instructions.

This page intentionally left blank.

Chapter 4: Running MQSpeedtest

4.1 Introduction

If you are using your own queue:

- Start the echo component
- Then run the ping component.

If you are using the IBM MQ Command server:

- Run the ping component.

If you want to or must use sender-receiver channels between the ping and echo components, refer to Appendix B.

4.2 Executing the Echo Component

If you are going to use the IBM MQ Command server to provide the ping, then other than the authority and channel as discussed in section 3.1, no other setup is required and you can continue with section 4.3, Executing the Ping Component.

To use the MQSpeedtest echo application, change to the folder where MQSpeedtest is installed above and run the mqspeedtest_echo (.cmd or .sh) script. When running the provided echo utility, the setup is described below (full details are in section 4.7, nsqrpl Command Line Options):

```
mqspeedtest_echo QueueName Qmgr_name
```

- **Example 1:** Listen on ECHO.QUEUE on queue manager QM2

  ```
  mqspeedtest_echo ECHO.QUEUE QM2
  ```

- **Example 2:** Listen on TEST.QUEUE on queue manager QM1

  ```
  ./mqspeedtest_echo.sh TEST.QUEUE QM2
  ```

	**NOTE:**	The echo component cannot typically be a user application since this application expects an application formatted message and will not be able to process the MQSpeedtest echo message.

4.3 Executing the Ping Component

To measure message performance, the measurement starts by running **mqspeedtest** at the command prompt. Change to the folder where MQSpeedtest is installed above and ensure that the current directory is added to the beginning of your PATH variable, for example export PATH=.:$PATH. Then run the mqspeedtest script (.cmd or .sh). The basic syntax is shown below (full details are in section 4.6, nsqping Command Line Options):

```
mqspeedtest [stream] QueueName Qmgr_name Options
```

where:

QueueName Name of queue to send the ping message to. Either an application/test queue or if testing queue manager response time, use SYSTEM.ADMIN.COMMAND.QUEUE.

stream Option to stream the MQSpeedtest statistics to an Nastel XRay service. A prompt for the required access token will be given. (For details, refer to section 6.1, Publishing Message Flow Performance.)

Options nsqping options other than queue and qmgr. (Refer to section 4.6, nsqping Command Line Options.)

- **Example 1:** Measure performance on queue **ECHO.QUEUE**, queue manager **QM2**:

  ```
  mqspeedtest ECHO.QUEUE QM2
  ```

- **Example 2:** Measure message performance every 10 seconds on a local queue manager **QM1** by sending a batch of 100 messages of 36 bytes each (the default) to the **SYSTEM.ADMIN.COMMAND.QUEUE.** The connection name (-C) specifies how the client application connects to the queue manager:

  - <IP address or host name of the node that hosts the queue manager>[(optional runmqlsr listener port for that queue manager; default 1414)][:<optional svrconn channel name to use; default SYSTEM.DEF.SVRCONN>]

```
mqspeedtest SYSTEM.ADMIN.COMMAND.QUEUE QM1 -C12.13.14.15(1415) -b100 -d10
```

	**NOTE:**	On Windows, when testing with a local queue manager, the connection name is optional.

- **Example 3:** Measure message performance using the nsqping defaults and a test queue. Note that the MQSpeedtest Echo component must be started first for the same queue. (Refer to examples in section 4.2.) Connect to a remote queue manager **QM2**, host machine at IP address 12.13.14.16, where a runmqlsr is listening on port 1416, and use special srvrconn channel MYSVRCONN:

  ```
  mqspeedtest TEST.QUEUE QM2 -C12.13.14.16(1416) :MYSRVCONN
  ```

- **Example 4:** Measure message performance every 20 seconds on a Unix queue manager **QM2** by sending a batch of 1,000 messages of 500 bytes each (the default) to the **REMOTE.CMD.QUEUE**, and perform confirm on arrival (**-ccoa**) and confirm on delivery (**-ccod**) measurements. Connection information is similar to that in example 2. Also given is a user ID and password for authentication by queue manager **QM2**, because channel authentication is set in the QM2 attributes, CHLAUTH(ENABLED):

  ```
  ./mqspeedtest.sh REMOTE.CMD.QUEUE QM2 -C"12.13.14.15(1416):MYSVRCONN" -
  ccoa -ccod -b1000 -s500 -d20 -ujames -pjamespasswd
  ```

4.4 MQSpeedtest Sample Output

The sample below is output from the **mqspeedtest[.cmd or .sh]** command when executed in interactive mode:

4.5 MQSpeedtest Statistics

Below is a table of **MQSpeedtest** statistics and their descriptions.

Table 4-1. MQSpeedtest Statistics

Statistic	Description
MINIMUM_ROUND_TRIP	The minimum time for the ping to be sent and the echo reply to be received, in seconds.
MAXIMUM_ROUND_TRIP	The maximum time for the ping to be sent and the echo reply to be received, in seconds.
AVERAGE_ROUND_TRIP	The average time for the ping to be sent and the echo reply to be received for all messages sent in a batch, in seconds.
AVERAGE_PROPAGATION_TIME	Average time in seconds for the first bit of a ping message to travel over the network link to the echo component. For this statistic to be accurate, the time stamps between the servers sending the ping and sending the echo must be synchronized.
AVERAGE_REFLECTION_TIME	Average time in seconds for the ping message to reflect back from the echo component to the ping component and the echo reply to be read. For this statistic to be accurate, the time stamps between the servers sending the ping and sending the echo must be synchronized.
MESSAGES_SENT	The total number of ping messages sent, included in this report.
CONFIRMED_EXPIRIES	Number of ping messages which expired before being delivered to the echo application (if coe specified).
CONFIRMED_DELIVERIES	Number of ping messages which were delivered to the echo application (if cod specified).
CONFIRMED_ARRIVALS	Number of messages which arrived to the echo application (if coa specified).
CONFIRMED_EXCEPTIONS	Number of messages which resulted in exceptions (if coe was specified).
REPORTS_RECEIVED	Total number of confirmation report messages received.
RESPONSES_RECEIVED	Total number of echo response messages received.
MESSAGES_RECEIVED	Total number of echo response and COA/COD report messages received.
BYTES_SENT	Total number of bytes sent.
BYTES_RECEIVED	Total number of bytes received.
RESPONSE_REQUEST_RATIO	The ratio of echo responses received to ping requests sent. A value of 100 means that all messages sent received responses.
General Performance Indicators	
TOTAL_PUT_TIME	The total time in seconds to put all the messages
TOTAL_GET_TIME	The total time in seconds spent waiting for the responses to arrive
AVERAGE_PUT_RATE	The potential messages put rate calculated based on ping messages generated.
AVERAGE_GET_RATE	The potential messages get rate calculated based on echo messages received.
AVERAGE_PUT_BYTES_PER_SEC	The average of all ping message bytes sent during the batch time.
AVERAGE_GET_BYTES_PER_SEC	The average of all echo message bytes received during the batch time.
PUT_GET_RATE_RATIO	The ratio of put rate to get rate, as a percentage. A value greater than one hundred means that MQSpeedtest could put messages faster than it could get responses.

Table 4-1. MQSpeedtest Statistics

Statistic	Description
Message Performance Indicators	
GROSS_ROUND_TRIP_RATE	Throughput rate (ping requests sent + echo responses received) during the batch interval, msg/sec
GROSS_ROUND_BYTES_PER_SEC	Throughput rate (ping requests sent + echo responses received) during the batch interval, bytes per second
EFFECTIVE_ROUND_TRIP_RATE	Effective throughput rate (ping request messages + echo response messages + COA/COD report messages) during the batch interval, msg/sec
CONFIRMATION_OVERHEAD	Percent of messages resulting from COA and COD report messages
AVERAGE_ARRIVAL_RATE	Average rate of ping messages arrived at the destination queue (based on coa, Confirm On Arrival messages)
AVERAGE_DELIVERY_RATE	Average rate of ping messages delivered to the destination application -- the command server or the MQSpeedtest echo component (based on cod, Confirm On Delivery messages)
AVERAGE_MSG_LATENCY_WITH_QDEPTH	The number of ping messages on queue for the AVERAGE_MSG_LATENCY metric calculation
AVERAGE_MSG_LATENCY	Average time between a ping message arrival at the target queue and delivery to the target application
MAXIMUM_MSG_LATENCY	Maximum time between a ping message arrival at the target queue and delivery to the target application
MAXIMUM_MSG_LATENCY_WITH_QDEPTH	The number of ping messages on queue for the MAXIMUM_MSG_LATENCY metric calculation
TOTAL_BATCH_TIME	The elapsed time in seconds to process the entire batch.
TEST_COMPLETION_CODE	Completion code for the test. 0 indicates that all processing was normal.

When running with report options, the following messages are produced:

RESPONSE: MSGID(1) RND_TRP_MS(0.00) RND_TRP_ON_Q_MS(15.00) MSG_AGE_MS(1.00) PUT_APPL_NAME(nsqecho.exe) PUT_DATE(20120713) PUT_TIME(20305078)

Response: Shows the time that the ping message completed the circuit and arrived back as an echo message at the nsqping application. Verbose option 1.

MQFB_COA: MSGID(1) RND_TRP_MS(0.00) RND_TRP_ON_Q_MS(15.00) MSG_AGE_MS(1.00) PUT_APPL_NAME(Prod_QMGR1) PUT_DATE(20120713) PUT_TIME(20305078)

Confirm on Arrival: Shows the time that the ping message arrived on the target queue. Verbose option 2.

MQFB_COD: MSGID(1) RND_TRP_MS(0.00) RND_TRP_ON_Q_MS(15.00) MSG_AGE_MS(1.00) PUT_APPL_NAME(Prod_QMGR1) PUT_DATE(20120713) PUT_TIME(20305078)

Confirm on Delivery: Shows the time that the target message was read from the target queue by the application. The time difference between arrival and delivery is time spent waiting for the application to process the message. Verbose option 2.

4.6 nsqping Command Line Options

MQSpeedtest uses the program nsqping. The table below lists the nsqping options. These options can be specified when running the mqspeedtest script file.

Table 4-2. nsqping Command Line Options

Property	Description
-a	Append to existing file and does not include a header line. Used with the –fFilename option.
-b<*Batch*>	Number of messages in a batch.
-C<*connection_name*>	The information that defines the communication link to the queue manager (optional). Format is: <ip_host>[(port)][:svrconn_chl_name] **ip_host:** The IP address or host name where the queue manager resides; e.g., 12.45.56.78 or myhost **port:** The listener port for the given queue manager, within parenthesis; optional. Default: 1414 **svrconn_chl_name:** The name of the server-conn channel for an MQ client connection to the queue manager; optional. Default: SYSTEM.DEF.SVRCONN When running on Unix, be sure to surround the entire option string with single or double quotes if using a port number in parentheses. **Example:** "-Cmy.host.com(1414)" or -C"my.host.com(1414)"
-ccoa	Generate confirm-on-arrival report.
-ccod	Generate confirm-on-delivery report.
-cdlq	Generate message that will go onto DLQ if undelivered.
-cexc	Generate confirm-on-exception report.
-cexp	Generate confirm-on-expiry report.
-d<*Interval*>	Interval, in seconds, that ping messages are generated. An interval of 0 (zero) indicates to ping only once.
-e<*Expiry*>	Message expires after Expiry time in 0.1 seconds.
-fFilename	Write statistics data to this file in csv format.
-h<*Host*>	Host name of the target AutoPilot M6 Process Wrapper. (Refer to Chapter 5.)
-m<*QmgrName*>	Specifies the name of the local queue manager.
-p<*AuthenticatePassword*>	Password for authenticating a user that can connect to the queue manager (optional).
-P<AP_*Port_#*>	Port number of the target AutoPilot M6 Process Wrapper. (Refer to Chapter 5.)
-q<*QueueName*>	Queue name of any type (application processing messages put to this queue must not require an application formatted message).
-s<*Bytes*>	Size, in bytes, of generated messages (maximum 32K).
-u<*AuthenticateUserID*>	User ID for authenticating a user that can connect to the queue manager (optional).
-U<*AltUser*>	Issue ping on behalf of another user.
-v<0|1|2>	Set verbose mode to 0 – summary; 1 – reports; 2 – all.
-w<*Wait*>	Wait interval, in milliseconds, before timeout.

4.7 nsrpl Command Line Options

The mqspeedtest_echo script uses the nsrpl program. The table below lists the **nsrpl** options, which can be used to modify the script executed when running with MQSpeedtest.

Table 4-3. nsrpl Command Line Options

Parameters	Description
QueueName1	Input queue name to read ping messages from.
QueueName2	Output queue to put the ReplyTo echoed message or asterisk (*). When set to "*", nsrpl will use the MD.ReplyToQ as the queue to put the echoed message. Must be the second parameter for the program.
QmgrName	Specifies the name of the local queue manager.
Waittime	Time (milliseconds) to spend waiting for a message before the application will terminate. In this way, the application can be launched at the start of a test and will terminate automatically when messages are no longer being sent. Optional. Default: -1 meaning wait forever; no timeout.
NO|YES	If NO is specified, the response message is exactly the same as the request message. If NO is not specified **OR** YES is specified, the message will have "-REPLY GENERATED" added at its end. This may be useful when analyzing message flow in Nastel XRay.
-count	Optional. Suppress display of "Messages replied: nnnnnnn" Default: As messages are received and replied to, an updated count of the messages processed is displayed. **Windows and Unix only**.

4.7.1 nscrpl Command Line Options

To collect timing statistics from some application or test queue on an MQ Appliance where you cannot install software, the Nastel client form of the reply application, nscrpl, will be useful. It will read messages from a queue on the remote queue manager and write to another queue on the same queue manager. Maximum message size is 200,000 bytes. The options for the command are mostly the same as those of nsrpl and are described in Table 4-4.

Table 4-4. nscrpl Command Line Options

Parameters	Description
QueueName1	Input queue name to read ping messages from.
QueueName2	Output queue to put the ReplyTo echoed message. When set to "*", nsrpl will use the MD.ReplyToQ as the queue to put the echoed message. Must be the second parameter for the program.
ConnectionName	Required. This is the information needed by the client application to make a connection to the queue manager: IPadrs_or_host (listener_port_nbr):[srvconn_chl] Default: localhost(1414):SYSTEM.DEF.SRVCONN
QmgrName	Required. Specifies the name of a local or remote queue manager.
Waittime	Time (milliseconds) to spend waiting for a message before the application will terminate. In this way, the application can be launched at the start of a test and will terminate automatically when messages are no longer being sent. Optional. Default: -1 meaning wait forever: no timeout.
NO|YES	If NO is specified, the response message is exactly the same as the request message. If NO is not specified, the message will have "- REPLY GENERATED" added at its end. This may be useful when analyzing message flow in Nastel XRay.

`-count`	Optional. Suppress display of "Messages replied: nnnnnnn" Default: As messages are received and replied to, an updated count of the messages processed is displayed. **Windows and Unix only.**
`-u`*`user_id`*	The ID needed for the connection authentication of the user. Default: none.
`-p`*`password`*	The password needed for the connection authentication of the user.

4.8 Capture Historical Data

MQSpeedtest statistics can be captured in a CSV (comma separated variables) file and used to produce a spreadsheet of historical data. For example, you can capture data on a daily basis, and append the results to an existing file each day. (Refer to Table 4-2 for a description of the various command line options.)

The `-fFilename` and `-a` options are used together with `-f` specifying the name of a file to write to and `-a` controlling if the first row is a header row and whether it should be appended to the existing file. These can also be used with the `-d` option which repeats the test periodically.

Examples:

1. `-fmqspeedtest.csv`

   Writes a header and a single result line to `msqspeedtest.csv`.

2. `-fmqspeedtest.csv -a`

   Writes only a result line to `mqspeedtest.csv`.

3. `-fmqspeedtest.csv -a -d60`

   Writes a header and a result line every 60 seconds to `msqspeedtest.csv`.

4. Import this file into Excel or other spreadsheet to produce charts such as the figure below.

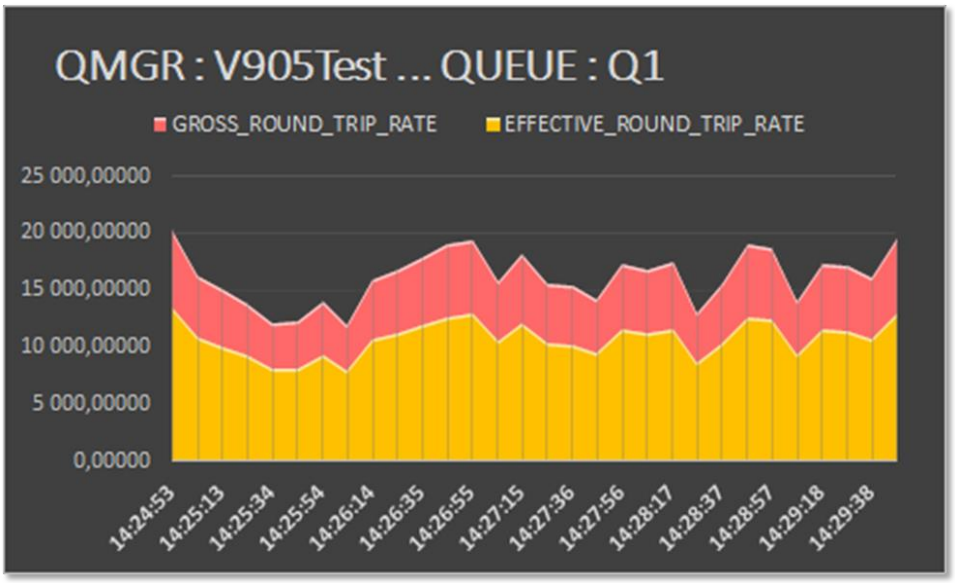

Figure 4-1. MQSpeedtest Spreadsheet Chart

The resulting file contains a time stamp (day of week, month, day, time, and year), queue manager, queue name, and the fields described in Table 4-1.

This page intentionally left blank.

Chapter 5: Integrating MQSpeedtest with AutoPilot M6

MQSpeedtest can be integrated into Nastel's AutoPilot M6 to provide detailed message performance metrics. These metrics can be used to analyze the behavior of the queue manager message performance.

	NOTE:	AutoPilot M6 is shipped separately and not included in the MQSpeedtest distribution.

The following diagram depicts the integration path.

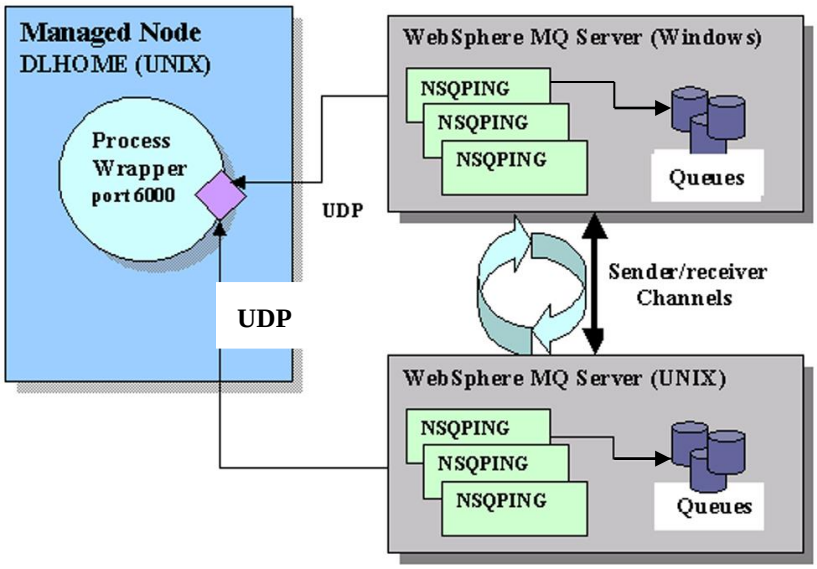

Figure 5-1. MQSpeedtest and AutoPilot M6 Integration Path

Integrate as follows:

1. **AutoPilot M6 Configuration:** Deploy an instance of AutoPilot M6 process wrapper on a CEP Server close to or on the same machine as the source of the metrics. Refer to *AutoPilot M6 Configuration*, section 5.1.
2. **Configuring Process Wrapper Options:** Configure process wrapper to accept IBM MQ performance metrics from MQSpeedtest. Refer to *Configuring Process Wrapper*, section 5.1.2.

	NOTE:	Due to unreliable nature of UDP, it is highly recommended that process wrapper be deployed on the same machine as NSQPING. This also means that AutoPilot Server should be running on or in close network proximity to the machine running MQSpeedtest.

Once AutoPilot M6 process wrapper is deployed and configured, do the following:

1. **Measuring Message Performance:** Run MQSpeedtest as a daemon to collect message performance metrics. Refer to *Publishing Messaging Flow Performance*, section 5.2.
2. **Monitor Message Metrics:** Create business view to monitor collected metrics. Refer to *Monitor Message Metrics*, section 5.3.

5.1 AutoPilot M6 Configuration

The following steps are performed to accept IBM MQ message performance metrics from MQSpeedtest.

1. **Deploy process wrapper:** Process wrapper is a generic AutoPilot M6 monitor capable of accepting facts from other applications via UDP or TCP. It can also start processes, restart processes and record published information into a JDBC database.
2. **Configure process wrapper:** Set the options required to accept information from MQSpeedtest. This step is required to monitor IBM MQ message performance metrics.

5.1.1 Deploying Process Wrapper

To deploy and configure an instance of a process wrapper, do the following:

1. Right click on the desired CEP Server (managed node).
2. Select **Deploy Expert > Wrappers > Process Wrapper**.
3. Configure Process Wrapper for context, name, and to accept fact on the UDP port (6000) that matches the port configured in MQSpeedtest. (Refer to *Configuring Process Wrapper*, section 5.1.2.)
4. Click **Deploy** to deploy the expert on the managed node.

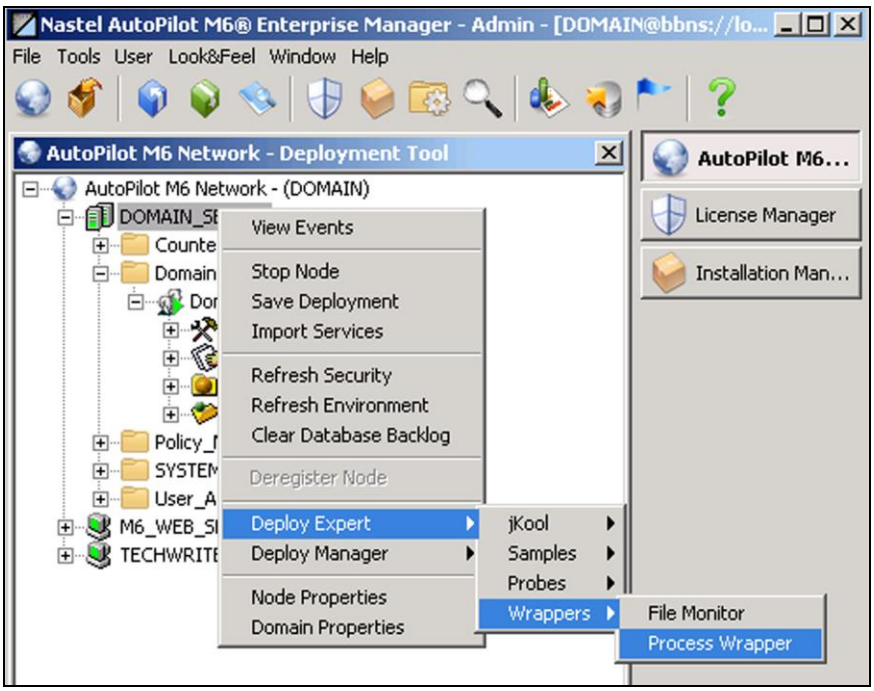

Figure 5-2. Deploying Process Wrappers

Once selected, the process wrapper configuration properties will be displayed, as shown below.

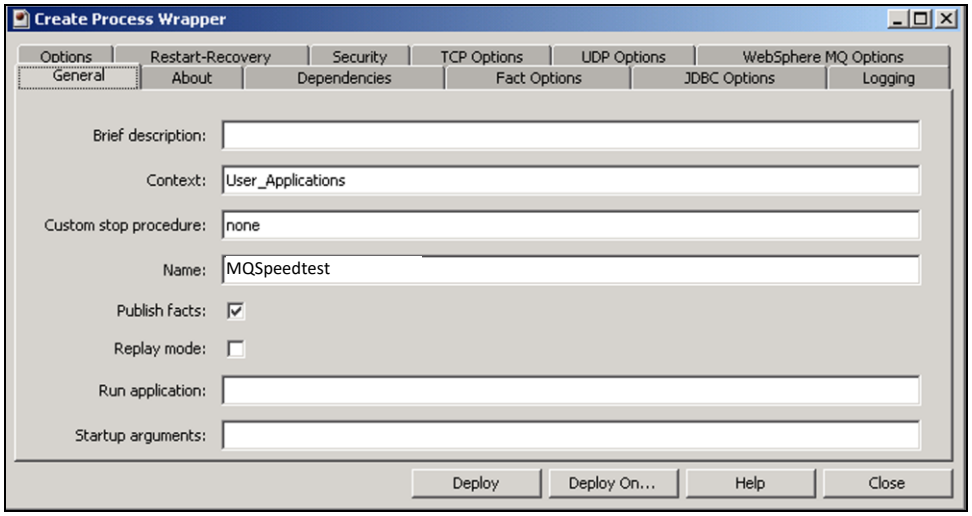

Figure 5-3. Process Wrapper Configuration Properties

5.1.2 Configuring Process Wrapper

The following process wrapper properties must be configured:

1. **General:** At a minimum *Context* and *Name* must be configured.

Table 5-1. Process Wrapper Expert: General

Property	Description
Brief description	A short, user-defined description of the service.
Context	A user define category that will be registered with the domain server. The default is: User_Applications
Custom stop procedure	Application or script that gracefully shuts down a started application. The default is: none
Name	Name that uniquely identifies the service in the domain. The default name is system assigned with the word “service” and 12 random digits. Example: MQSpeedtest
Publish facts	Check to enable/disable. Publishes received facts locally when enabled.
Replay mode	Check to enable/disable. Emulates fact source and replays all received facts.
Run application	Fully qualified name of application/script/process to run. Example: `usr/bin/tar`
Startup arguments	Start–up parameters that are passed to the application. Example: `cvf mytar/opt/nastel`

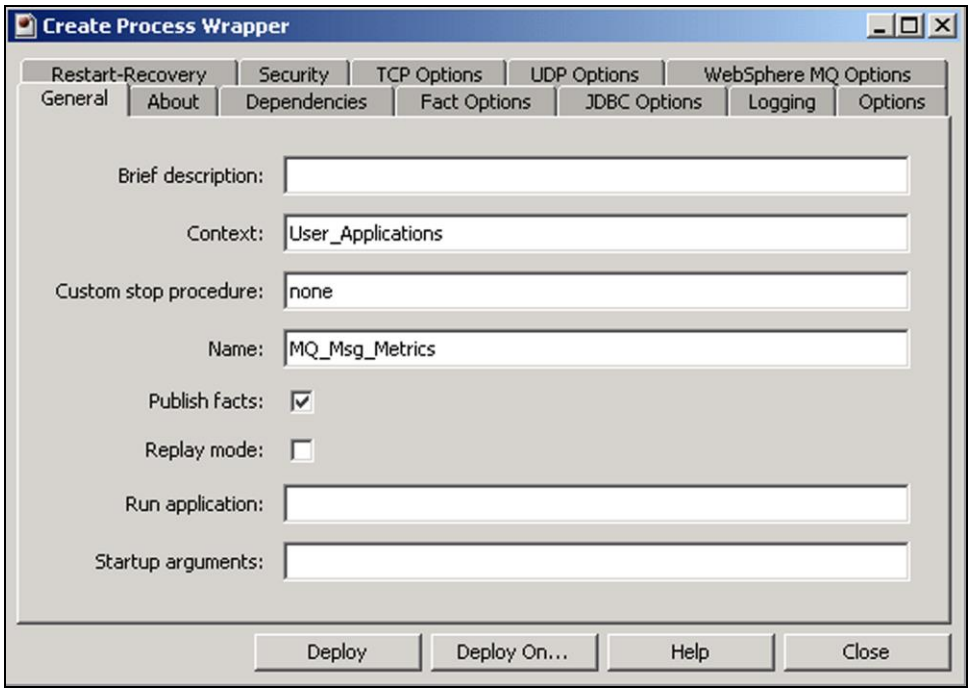

Figure 5-4. Process Wrapper General Configuration Properties

2. **UDP Options:** Both properties on this screen must be configured.

Table 5-2. Process Wrapper Expert: UDP Options

Property	Description
Accept UDP facts	Check to enable/disable.
UDP port	Unique port on which process wrapper will accept the incoming performance metrics. Default is 6000.

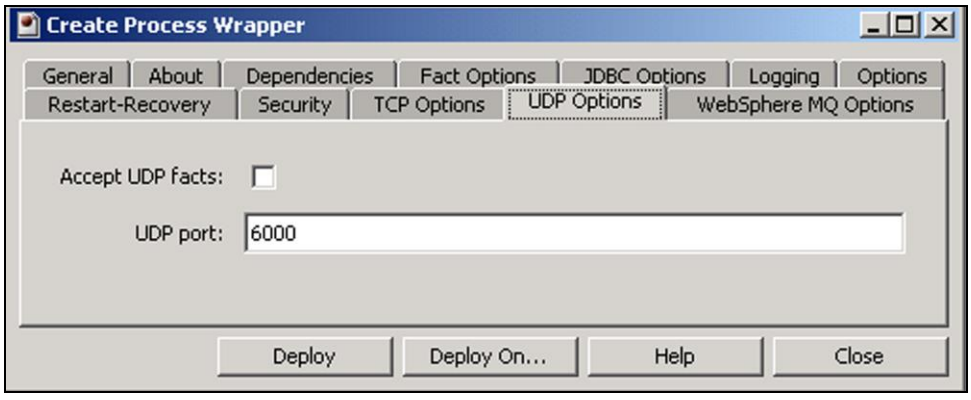

Figure 5-5. Process Wrapper UDP Option Configuration Properties

5.2 Publishing Message Flow Performance

To publish performance metrics to the AutoPilot M6 MQSpeedtest Expert, first create a process wrapper using the APM6 Enterprise Manager as shown in section 5.1.1. Then execute the mqspeedtest command adding the `-h` host name and `-p` UDP options to match configuration of the process wrapper. Multiple instances of MQSpeedtest can publish performance metrics to the same process wrapper.

```
mqspeedtest[Unix:.sh] QueueName Qmgr_name [-CConnectionName [-uuserID
-ppassword]] -hHostname -PUdpPort
```

- **Example 1:** Measure message performance every 10 seconds on a Windows queue manager QM1, over SYSTEM.ADMIN.COMMAND.QUEUE publishing the data to the AP MQSpeedtest Expert on server localhost at port 6060.

  ```
  mqspeedtst SYSTEM.ADMIN.COMMAND.QUEUE QM1 -b100 -d10 -hlocalhost -P6060
  ```

- **Example 2**: Measure message performance every 20 seconds on a local Unix queue manager QM2 over REMOTE.CMD.QUEUE, perform confirm on arrival (-ccoa) and confirm on delivery (-ccod) measurements, publishing the data to the AP MQSpeedtest Expert on server serverpc at port 6000.

  ```
  mqspeedtst.sh REMOTE.CMD.QUEUE QM2 -ccoa -ccod -b100 -d20
  -C"localhost(1414)" -hserverpc -P6000
  ```


5.3 Monitor Message Metrics

All message performance metrics are available under the process wrapper where they are published by MQSpeedtest, grouped by queue manager and queue, as shown in the figure below.

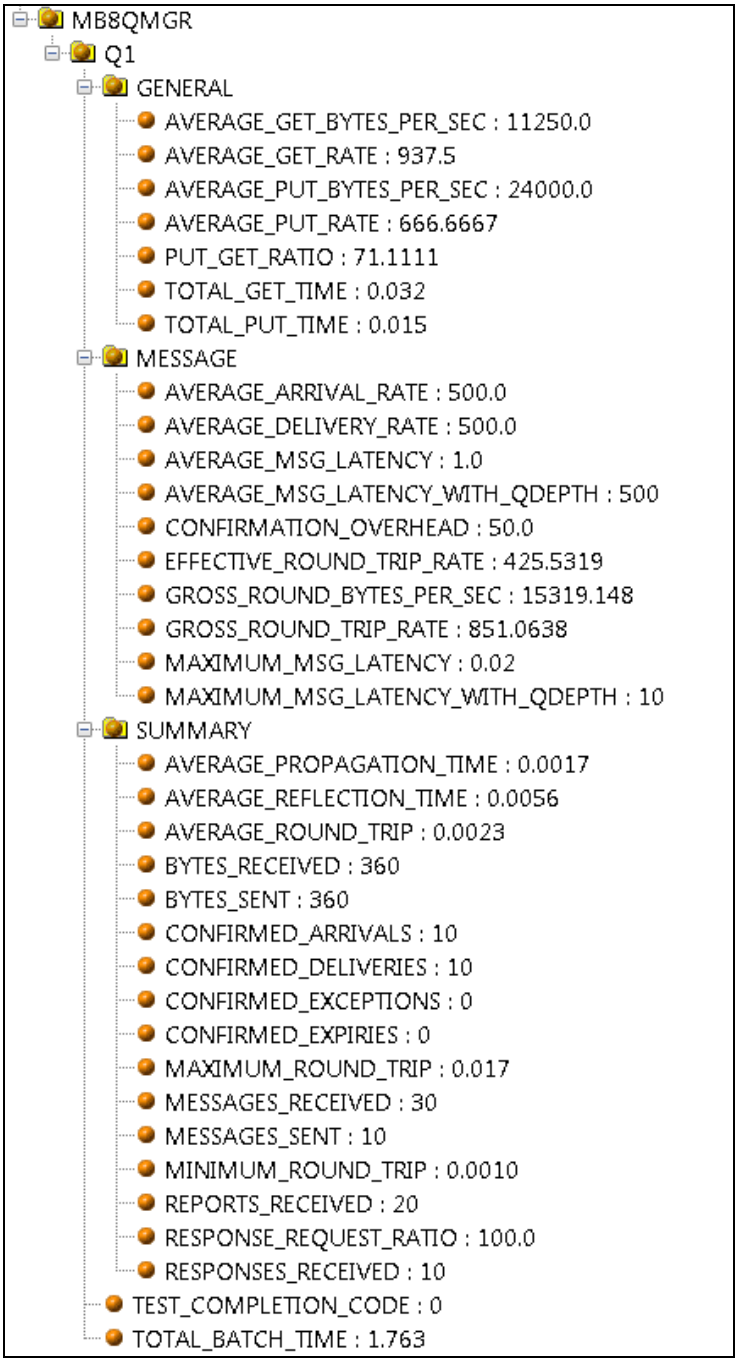

Figure 5-6. Published Message Performance Metrics

All metrics published by MQSpeedtest can be included in user defined business views with rules, automation, alerts and notifications.

	NOTE:	Refer to *AutoPilot M6 User's Guide* for more information about business views.

Chapter 6: Integrating MQSpeedtest with Nastel XRay

MQSpeedtest can be integrated into Nastel XRay (formerly AutoPilot Insight), either using an on-premises edition of Nastel XRay or in the cloud, using **jKoolcoud.com** to provide detailed message performance analytics. These metrics can be used to analyze the behavior of the queue manager and queue message performance over time to better understand their behavior.

	**NOTE:**	Nastel XRay is shipped separately and not included in the MQSpeedtest distribution. However, the streams parser package, tnt4j-streams, is included in the distribution. If you want to use it, go to [install_dir]\streams and unzip **Windows:** tnt4j-streams-1.5.0.zip or **Unix:** tnt4j-streams-1.5.0.tar.gz

This integration is called **streaming** and the data is captured by MQSpeedtest and forwarded to Nastel XRay directly. The destination is your repository to be used for MQSpeedtest data. If you do not have an on-premises edition of Nastel XRay, you can sign up for a free personal account using the cloud edition at https://www.jkoolcloud.com/. **The key element you need to identify is your repository access token.**

If using jKoolcloud, the access token will be contained in the Welcome email, specifically the following three pieces of information are needed in the following steps:

- **Your User ID is:** YOUR-USER-ID
- **Your jKool Dashboard is** @ https://jkool.jkoolcloud.com
- **Your jKool API Access Token is:** YOUR-ACCESS-TOKEN (required for streaming)

6.1 Streaming Message Flow Performance

To stream metrics to the AutoPilot M6 MQSpeedtest Dashboard, execute the **mqspeedtest** command adding `stream` as the first parameter.

All of our instances of MQSpeedtest should publish performance metrics to the same repository so you can compare results.

```
mqspeedtest[Unix:.sh] stream QueueName Qmgr_name [-CConnectionName
[-uuserID -ppassword]] [options]
```

- **Example 1:** Measure message performance using 100 messages seconds on a Windows queue manager QM1, over SYSTEM.ADMIN.COMMAND.QUEUE

```
mqspeedtest stream SYSTEM.ADMIN.COMMAND.QUEUE QM1 -b100
```

- **Example 2**: Measure message performance for 100 messages of 1,000 bytes on a remote Unix host, remotehost, queue manager QM2, queue TEST.DATA.QUEUE over a client connection using channel `REMOTE_TEST_CHANNEL`

```
mqspeedtest.sh stream TEST.DATA.QUEUE QM2
-C"remotehost(1414):REMOTE_TEST_CHANNEL" -b100 -s1000
```

6.1.1 Providing Your Access Token

The first time you execute script mqspeedtest.[cmd or sh], it will prompt you for your access token.

At the prompt, **Enter your Access Token:** enter the access token from your Welcome email or your local repository.

MQSpeedtest will execute and stream the data to Nastel XRay. If you see the message, **Stream session status (SUCCESS) message sent!**, you have streamed successfully and you can move on to the next step to view the results. If you see an error, review the troubleshooting section (Chapter 7).

	NOTE:	If you need to change your access token, delete the file `mqspeedtest_tnt4j.properties` and you will be prompted for the new one.

6.2 Nastel XRay Configuration

6.2.1 Logging on to Nastel XRay

1. Log on to your account.
2. From the landing page, select **Go to Dashboard**. If you see a sample repository, in the upper right, click on the repository and change to your personal repository (this is based on your user ID as shown in Figure 6-2.)

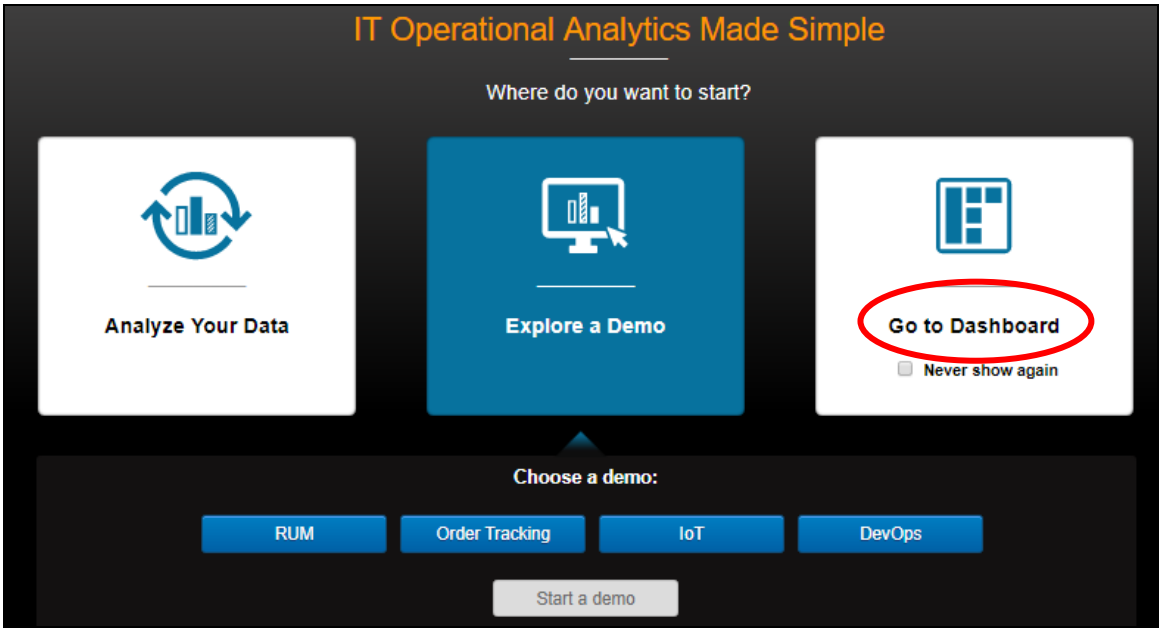

Figure 6-1. Landing Page

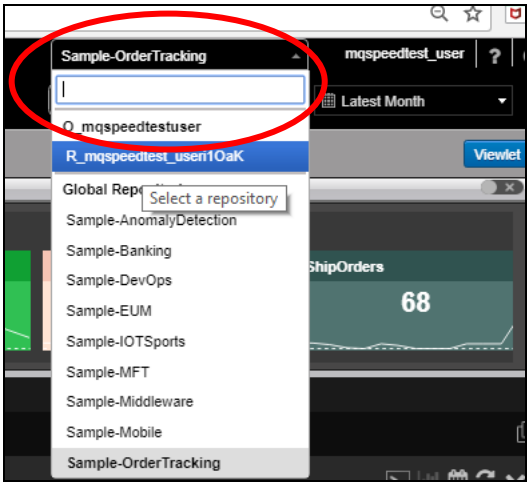

Figure 6-2. Sample Repositories

3. If prompted to create a dashboard, click **Cancel**.

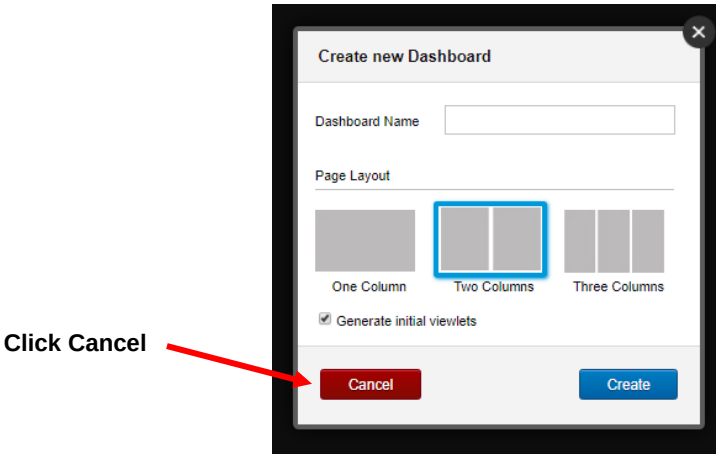

Figure 6-3. Create New Dashboard

6.2.2 Importing the Sample Dashboard

4. Click the **Menu** icon and select **Import/Export**. Then **Dashboards** to display the **Import/Export** dialog box (Figure 6-5).

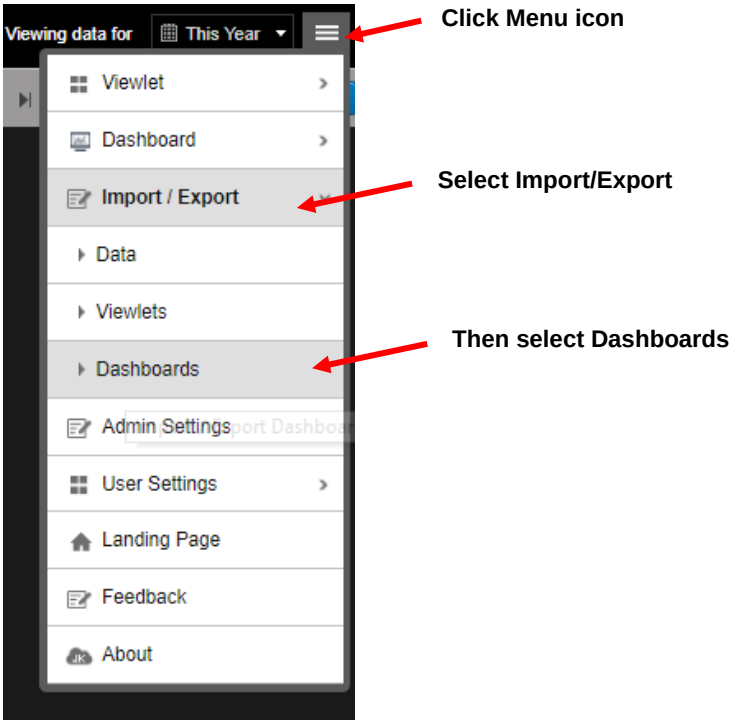

Figure 6-4. Menu

5. Select the **MQSpeedtest_Dashboards.json** file included in your distribution and click **Import**.

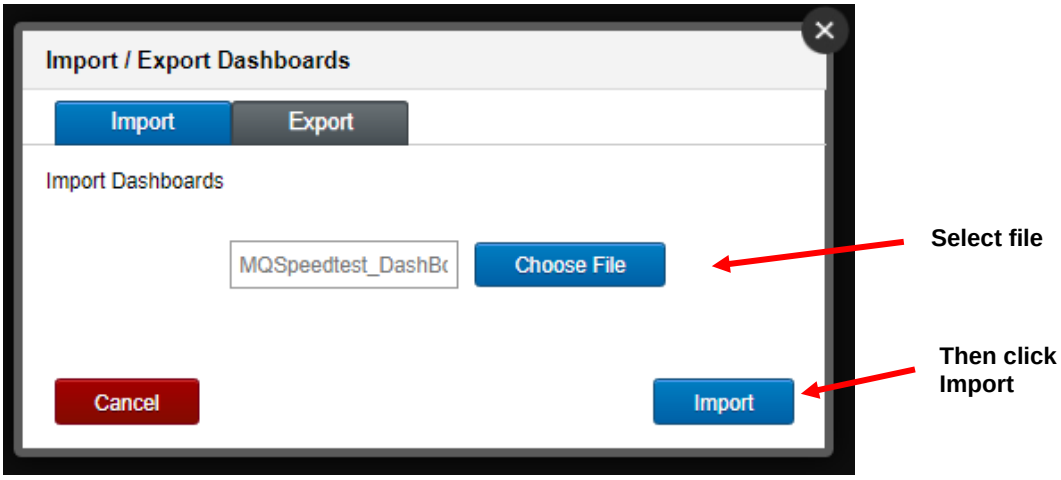

Figure 6-5. Import/Export Dialog Box

6. Select **Menu** > **Dashboards** > **Open** and you will see the dialog in figure 6-6. Select **MQSpeedtest** and click **Open.**

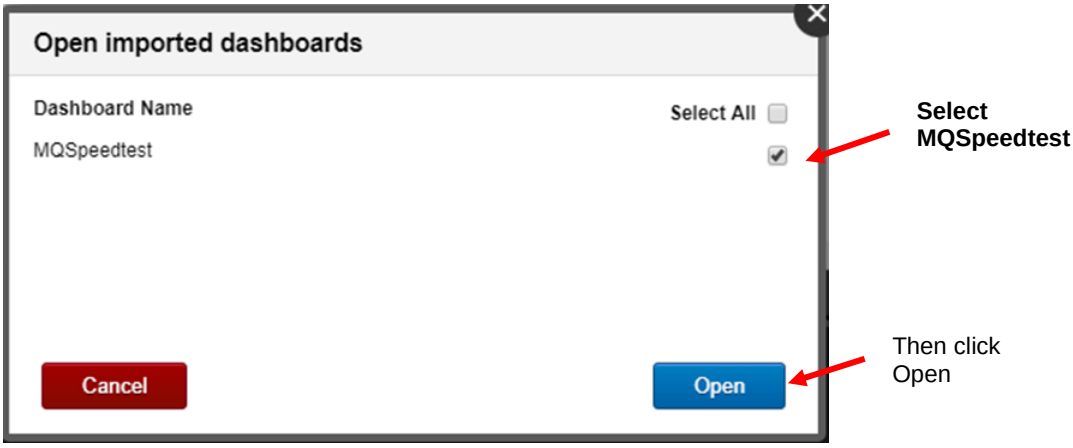

Figure 6-6. Open Imported Dashboards Dialog Box

7. When the dialog box displayed in figure 6-6a opens, click **Open**.

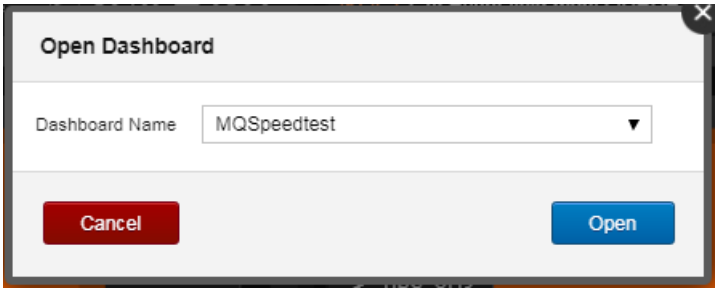

Figure 6-6a Open Dashboard Confirmation Dialog Box

If you have streamed data, you should see it in the imported dashboards as shown in section 6.3. If not, allow at least 2 minutes for the data to show up and then refresh the viewlets or change the new time designation to refresh all viewlets.

6.3 Viewing Message Metrics

There are many metrics published, several examples are shown in the sample dashboard.

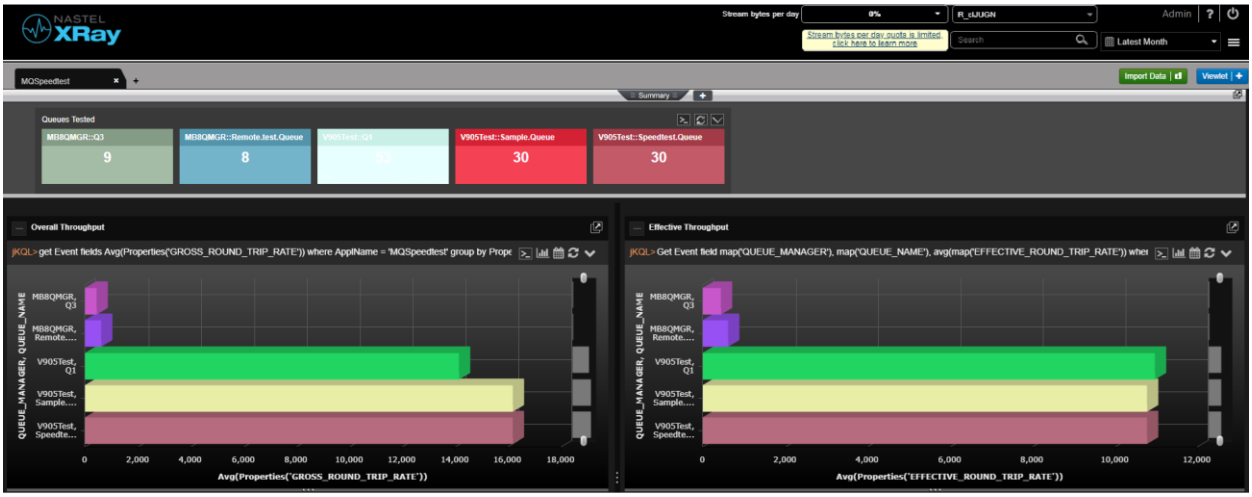

Figure 6-7. Sample Dashboard

The **Summary** shows the total number of executions for various queue managers and queues. The top two views show the effective message rate through the queue manager. (Refer to Table 4-1 for field descriptions.)

These viewlets show the bytes put and received (get) per second.

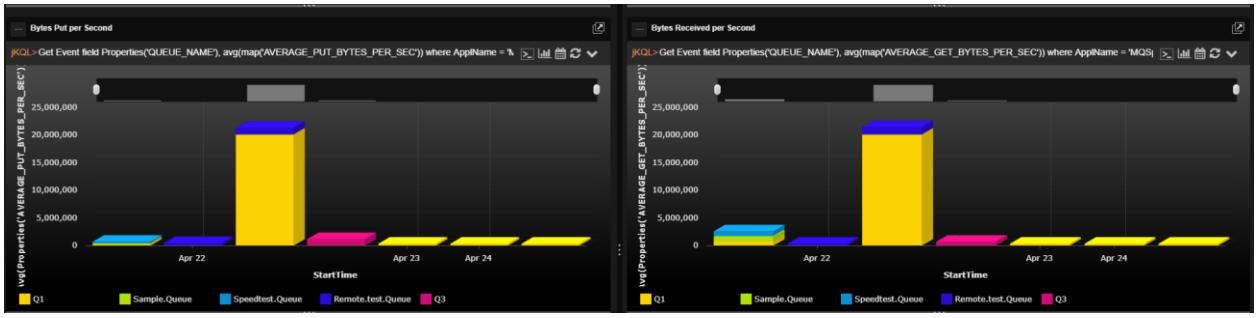

Figure 6-8. Viewlets Showing Bytes Put and Received

These viewlets show the time taken to complete the processing.

Figure 6-9. Viewlets Showing Time to Complete Processing

If you click on the Events Count box in the upper part of the display, a view of the details of the MQSpeedtest metrics is displayed, as in figure 6-10.

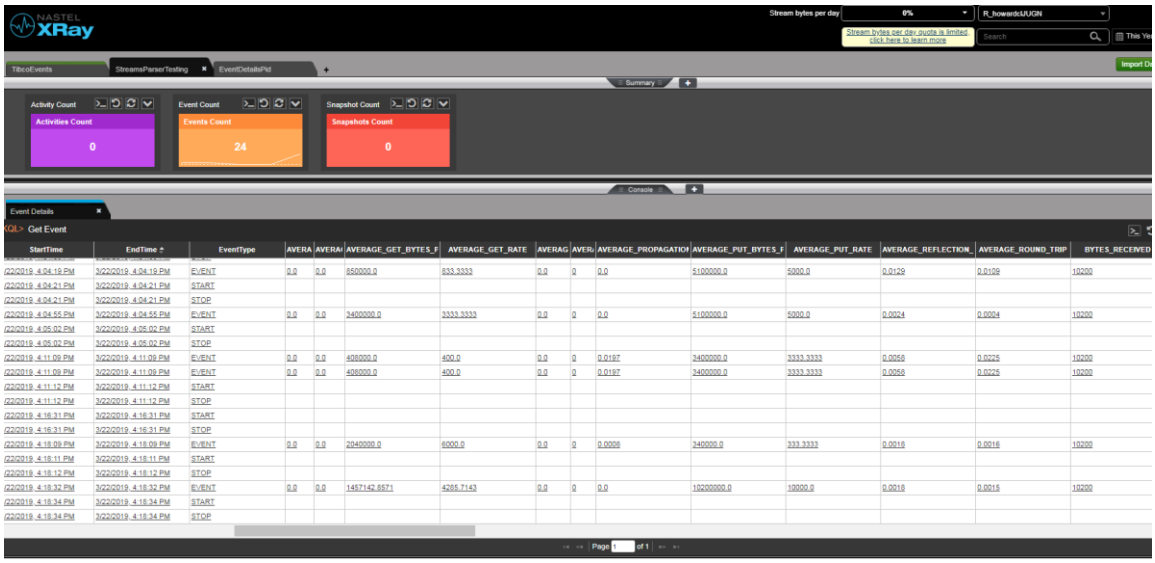

Figure 6-10. Event details for MQSpeedtest

Chapter 7: Troubleshooting

Problem 1: When using the command server as the echo component, nsqping returns the message "Pinging queue manager completed with RC(2033)", no message available

What to do: If running MQSpeedtest in non-client mode and without the echo component, verify that the queue specified is SYSTEM.ADMIN.COMMAND.QUEUE.

If using multiple queue managers and MQSpeedtest in non-client mode, verify that the channels are started in both directions.

Verify that the user running the MQSpeedtest is defined and is authorized to access MQ, the queue specified, and any queues in the path of the MQSpeedtest route.

Include the -ccoa option and execute with -v2 to see if the messages are arriving on the target queue.

If using the WMQ command server as the echo component, the maximum message size is 32K.

Problem 2: When running program nsqping or scripts mqspeedtest or mqspeedtest_echo, the message "error while loading shared libraries: libmqm_r.so" or similar occurs.

What to do: This indicates that the proper Unix IBM MQ libraries have not been found. Check the nsqping (or nsrpl or nscrpl) library dependencies: `ldd nsqping`. If it shows "unknown" for one or more libraries, check the library path environment variable for the platform. If the 64-bit MQ library path, for example, `/opt/mqm/lib64`, is not included, then you can manually append or prepend that path to the variable.

Examples:

Linux and Solaris:

```
echo $LD_LIBRARY_PATH
LD_LIBRARY_PATH=/opt/mqm/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

AIX:

```
echo  $LIBPATH
export LIBPATH=/usr/mqm/lib64:$LIBPATH
```

`[or LIBPATH=/usr/mqm71/usr/mqm/lib64:$LIBPATH` if using an MQ 7.1 queue manager and it has a special installation path]

In a multi-version queue manager installation, the best way to set up the MQ environment is to use the setmqenv command from bin directory associated with the version of the local MQ target queue manager.

Example:

```
. /usr/mqm71/usr/mqm/bin/setmqenv  -m qmgr_name [or -n installation_name] -k
```

-k sets the library path, adding it to the start of the current value, thereby superseding any wrong path that may be included to the right of it in the path list.

To find the installation name for a queue manager to use in the above command, first use this command:

```
dspmq -o all
```

On an AIX system, LIBPATH may then look like this, so as to allow 32-bit and 64-bit applications to link with the correct libraries for an MQ 7.5 queue manager:

```
LIBPATH=/usr/mqm75/usr/mqm/lib:/usr/mqm75/usr/mqm/lib64
```


The PATH environment variable will also be set properly, with the proper bin directory for the selected MQ version added to the start of the PATH current value, for example like this:

```
PATH=/usr/mqm75/usr/mqm/bin:/opt/nastel/mqspeedtest/bin: etc
```

For the nsqping dependencies, append the 'dot' directory to LIBPATH and PATH to indicate the current directory as well, since you most likely are sitting in the mqspeedtest directory when running nsqping or the mqspeedtest scripts:

```
LIBPATH=.:$LIBPATH
PATH=.:$PATH
```

You can alternatively append the fully qualified mqspeedtest directory path to allow the libnsq* libraries to be linked to nsqping and for the nsqping executable to be found without you having to be sitting directly in the mqspeedtest directory.

You can also modify the scripts mqspeedtest.cmd (.sh) and mqspeedtest _echo.cmd (.sh) to specify the correct library path. Be careful the scripts do not change the library path that you set up external to the scripts.

Problem 3: MQCONNX fails with ReasonCode 2035, user not authorized.

What to do:

a. Options –u*username* and –p*password* may be needed or there is an error in the values being used.

b. If you were using –u and –p, try the command again without –u and –p.

c. The MCAUSER attribute in the svrconn channel definition on the target queue manager may be wrong or it may need an authorized user ID value, for example 'mqm', to allow running the svrconn channel.

d. Check the settings of the queue manager attributes.
The `CONNAUTH` attribute can be set to the name of an authentication information (`AUTHINFO`) object. The `AUTHINFO` object in turn has attributes `CHCKLOCL` and `CHCKCLNT`, which can be set to one of these values:
NONE, OPTIONAL, REQUIRED, REQDADM.
If its value is REQUIRED or REQDADM, provide a valid user and password, or if you have the authority, change it to NONE during the MQSpeedtest testing.

See IBM Knowledge Center, **Turning on connection authentication on a queue manager**
https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_8.0.0/com.ibm.mq.sec.doc/q113250_.htm

Problem 4: MQCONNX fails with ReasonCode 2538, host not available.

What to do: The –C*connection_name* option is needed or there is an error in the one being used. Check the IP address or host name, queue manager listener TCP port number, and server-conn channel name being used. Check if the listener is running with that TCP port number on the remote queue manager.

Problem 5: MQCONNX fails with ReasonCode 2540, unknown channel name.

What to do: The –C*connection_name* svrconn channel name part is invalid. It is either a spelling error or the channel does not exist on the destination system.

Problem 6: MQCONNX fails with ReasonCode 2058, queue manager name error.

What to do: The MQ client library could not find a queue manager that matches with the name given in the command line and the destination of the connection name.

a. Check the queue manager and connection names for spelling or other errors.

b. Check that the MQ environment is set properly for the version of a local queue manager being used, as described in Problem 2. For example, use command `env|sort` and observe the variables starting with "MQ" and the PATH variable.

c. Check if "runmqsc *qmgr_name*" can be run. If it cannot, the desired installation environment is set incorrectly.

Problem 7: Pinging *qmgr_name*(*queue_name*) completed with RC 2085, unknown object name.

What to do: Check the queue and queue manager names for spelling errors. Check that the queue name is defined on the target queue manager.

Problem 8: Facts should be published to an AutoPilot MQSpeedtest expert and a message like this appears:

ERROR: failed to write to host(service)=11.0.0.24(6012), sock type 2, RC(58)

Description of last errno: : There is no destination address for the socket operation

What to do: Use the APM6 Enterprise Manager to check if the UDP port number configured is correct for the MQSpeedtest expert.

Problem 9: nsqping: command not found

What to do: The nsqping script expects the current folder to be defined in the path. This could be added as "." (period), such as `export PATH=.:$PATH`

Problem 10: java: command not found

What to do: When streaming, Java is required to process the data. You need to install Java for your respective operating system, and the java.exe path must be defined in the PATH environment variable. Be sure the java version is 1.8.0 or greater.

Problem 11: Failed to authenticate with service='https://data.jkoolcloud.com' token='xxxxx'

What to do: The streaming access token you provided is incorrect and needs to be updated. Do one of the following:

a. Delete the file `mqspeedtest_tnt4j.properties` and re-run the **mqspeedtest** script. It will prompt you for the access token.

OR

b. Manually edit `mqspeedtest_tnt4j.properties` and locate the line that reads `event.sink.factory.Token:` and set the token here.

Problem 12: The streams agent startup on Solaris fails with java exception

java.io.IOException: Failed to connect to uri=https://12.13.14.15, reason=Error during hash calculation

…

Caused by: java.lang.RuntimeException: Could not clone digets

at …

Caused by: java.lang.CloneNotSupportedException: SHA-384

at …

Caused by: sun.security.pkcs11.wrapper.PKCS11Exceptioon: CKR_STATE_UNSAVEABLE

What to do: This problem is documented at
https://www.reddit.com/r/admincraft/comments/59p0a7/client_connection_error_exception_in_thread_user/

Edit the end of the streams agent startup script *mqspeedtest_install_dir*`/streams/tnt4j-streams-1.5.0/bin/tnt4j-streams.sh:`

```
# Add option to disable Public-Key Cryptography Standards (PKCS) security
SECURITY_OPT="-Dsun.security.pkcs11.enable-solaris=false"
"$JAVA_HOME/bin/java" $STREAMSOPTS $SECURITY_OPT -classpath "$LIBPATH" $MAINCLASS
$*
```

Problem 13: The streams agent startup fails with java exception
root cause: java.lang.IllegalStateException: Tracker is not opened to record activity data...
…
Caused by: org.apache.http.conn.ConnectTimeoutException: Connect to data.jkoolcloud.com:443 [data.jkoolcloud.com/52.0.63.104] failed: connect timed out

What to do: Check these items:
a. Check the java version with command `java –version.` It should show 1.8.0 or greater. If not, download such version JRE or JDK from the web, for example,
https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
install it, change the PATH and JAVA_HOME environment variables, and re-run the MQSpeedtest script.

b. The Nastel XRay service URL may not be resolvable to an IP address on your computer.
Try to ping and trace the route to the Nastel XRay data service:

```
ping  data.jkoolcloud.com
Unix: traceoute (Windows: tracert) data.jkoolcloud.com
```

Try to telnet to the Nastel XRay data service, port 443:

`telnet data.jkoolcloud.com` (or its IP address) 443

If these fail, display your computer's routing table and possibly add a new gateway. Confer with your network administrator if necessary.

Appendix A: Installing MQSpeedtest for MQ on z/OS

A.1 Transferring Data Sets to z/OS

The XMI files provided from Nastel Technologies can be uploaded to the z/OS operating system using FTP. If you want to pre-allocate the datasets on z/OS before using FTP, create the files using the attributes shown in Table A-1. It is not necessary to pre-allocate the target datasets on z/OS if you use the QUOTE SITE command with FTP. Ensure your source and target directories are correct.

For example:

ftp <host>

Enter: user <id>

Enter: password <******>

Enter: bin

Enter: quote site lrecl=80 recfm=fb blksize=3120 primary=5 blocks secondary=5 blocks

Enter: put JCL.XMI

Enter: quote site lrecl=80 recfm=fb blksize=3120 primary=210 blocks secondary=25 blocks

Enter: put LOAD.XMI

The required file allocations on the mainframe are listed in Table A-1.

Table A-1. Minimum z/OS File Allocations for Data Sets

	DSORG	RECFM	LRECL	BLKSIZE	BLKS
JCL.XMI	PS	FB	80	3120	5
LOAD.XMI	PS	FB	80	3120	210

JCL.XMI represents the name of the file on the mainframe. It could be USERID.JCL.XMI where USERID is your TSO logon ID.

The same applies for the LOAD file. These files can be allocated using 3.2 in Interactive System Productivity Facility (ISPF) if you use the pre-allocation method. These are temporary files and may be deleted once the installation is successful.

A.2 Creating Files

Once transferred to the mainframe these files must be RECEIVED into PDS files. The RECEIVE command will extract the files and dynamically create the proper space when used as follows.

HLQ represents the high-level qualifier for each dataset. MQSPDTST is the middle-level qualifier. These are the names that are to be substituted in Job Control Language (JCL) streams.

The block sizes for the FB files may be changed to conform to site standards. The load library should remain the same as in Table A-2.

Receive indsn(JCL.XMI)

The response to the prompt is: dsn('*HLQ*.MQSPDTST.JCL')

Receive indsn(LOAD.XMI)

The response to the prompt is: dsn('*HLQ*.MQSPDTST.LOAD')

Table A-2 shows the dataset attributes that will be created when they are received. The space may vary at different installations.

Table A-2. Minimum z/OS File Allocations

	DSORG	RECFM	LRECL	BLKSIZE	BLKS
HLQ.MQSPDTST.JCL	PO	FB	80	27920	4
HLQ.MQSPDTST.LOAD	PO	U	0	27998	32

There are now two installed target libraries:

- HLQ.MQSPDTST.JCL – This library contains JCL to run MQSpeedtest Ping and Echo programs.
- HLQ.MQSPDTST.LOAD – This library contains the load modules for the MQSpeedtest Ping and Echo programs.

A.3 Configuring JCLs

Before submitting the sample JCL, configuration is required. Refer to Chapter 4, Running MQSpeedtest.

A.3.1 MQSpeedtest Ping JCL

The member MQSPING sample JCL is shown below. The basic options are used in this sample.

The sample JCL MQSPING in the package contains an example of MQSpeedtest Ping showing the additional options.

For a full list of all available options, refer to section 4.6, nsqping Command Line Options.

```
//MQSPING  JOB (),'NSQPING',MSGCLASS=X,CLASS=A,
//         NOTIFY=&SYSUID
//*-------------------------------------------------------------------
//*
//*  MQSpeedtest Ping Execution
//*
//*  ++hlq++    - Nastel high level qualifier
//*  ++mqser++  - WebsphereMQ high level qualifier
//*  ++qmgr++   - Specifies the name of the local queue manager.
//*  -q         - Specifies the name of queue to process.  If other
//*               than SYSTEM.COMMAND.INPUT, start MQSECHO first
//*
//*-------------------------------------------------------------------
//STEP1    EXEC PGM=NSQPING,
// PARM='-m++qmgr++ -qSYSTEM.COMMAND.INPUT'
//STEPLIB  DD DSN=++hlq++.LOAD,DISP=SHR
//         DD DSN=++mqser++.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//CEEOPTS  DD DUMMY
```

A.3.2 MQSpeedtest Echo JCL

The member MQSECHO sample JCL is shown below. The basic options are used in this sample.

The sample JCL MQSECHO in the package contains an example of MQSpeedtest Echo showing the additional options.

For a full list of all available options, refer to section 4.7, nsrpl Command Line Options.

```
// MQSECHO JOB (),'NSRPL',MSGCLASS=X,CLASS=A,
//         NOTIFY=&SYSUID
//*-------------------------------------------------------------------
//*
//*  MQSECHO Echo execution
//*
//*  ++hlq++   - Nastel high level qualifier
//*  ++mqser++ - WebsphereMQ high level qualifier
//*  ++qmgr++  - Queue Manager
//*  ++queue++ - Queue from which to read the ping message
//*  *         - Param 2, indicates the ReplyTo queue is in the
//*              message descriptor (MQMD) of the ping message
//*  -1        - The MQGET wait time. Minus 1 indicates wait forever
//*  NO|YES     - If NO is specified, the response message is exactly the same
//*               as the request message.  If NO is not specified OR YES is
//*               specified, the message will have "-REPLY GENERATED" added at
//*               its end.
//*
//*-------------------------------------------------------------------
//STEP1    EXEC PGM=NSRPL,
// PARM='++queue++ * ++qmgr++ -1 NO'
//STEPLIB  DD DSN=++hlq++.LOAD,DISP=SHR
//         DD DSN=++mqser++.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//CEEOPTS  DD DUMMY
```


This page intentionally left blank.

Appendix B. Sample Channels Configuration between Two Systems

This is a sample configuration for MQSpeedtest:

- Send and receive channels between two hosts, SYSTEMA and SYSTEMB
- Queue managers QMGRA and QMGRB
- Local queue AAA.LQ on QMGRB
- SYSTEMA is the ping'er, SYSTEMB is the echo'er
- Use runmqsc for all define and ping commands
- runmqlsr listener port 2418 on both systems

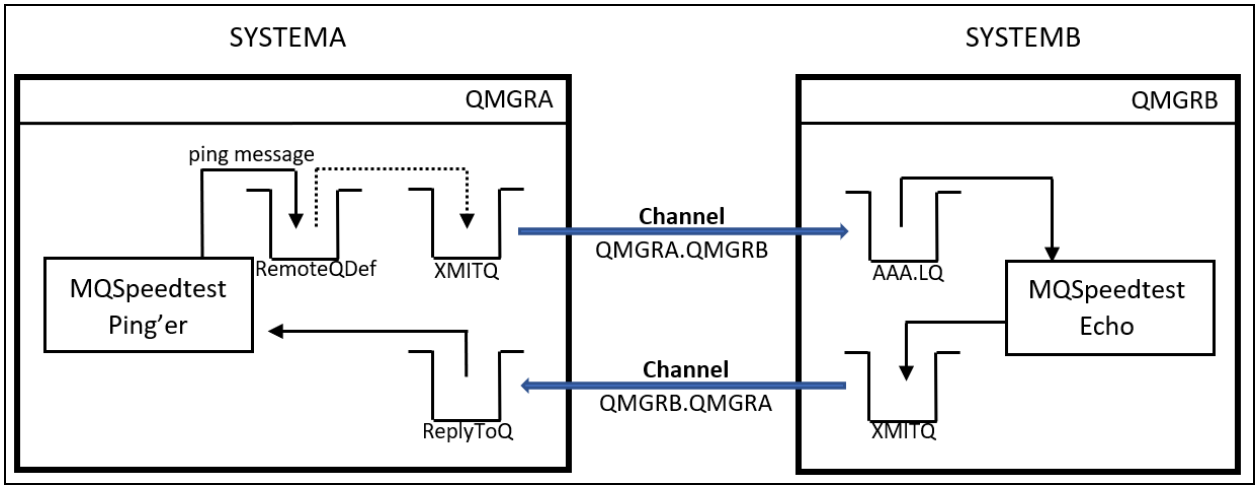

Figure B-1. Two System Channels Configuration

On System A (ping'er):

```
define channel(QMGRA.QMGRB) chltype(SDR) conname('SYSTEMB(2418)') xmitq(QMGRB)
define qlocal(QMGRB)  usage(XMITQ)
start  chl(QMGRA.QMGRB)
define channel(QMGRB.QMGRA) chltype(RCVR)
define qremote(QMGRB_AAA.RQ)  rqmname(QMGRB) rname(AAA.LQ)  xmitq(QMGRB)
define channel(QMGRB.QMGRA) chltype(RCVR)
runmqlsr -m QMGRA -t TCP -p 2418  &
```

On System B (echo'er):

```
define channel(QMGRB.QMGRA) chltype(SDR) conname('SYSTEMA(2418)') xmitq(QMGRA)
define qlocal(QMGRA) usage(XMITQ)
define qlocal(AAA.LQ)
start chl(QMGRB.QMGRA)
define channel(QMGRA.QMGRB) chltype(RCVR)
runmqlsr -m QMGRB -t TCP -p 2418 &
```

On System B, test the channel to System A:

```
ping channel(QMGRB.QMGRA)
```

On System A, test the channel to System B:

```
ping channel(QMGRA.QMGRB)
```


Start echo'er on System B:

```
mqspeedtest_echo.sh  AAA.LQ  QMGRB
```

Start ping'er on System A:

```
mqspeedtest.sh   QMGRB_AAA.RQ    QMGRA -ccoa –ccod
```

Observe on echo'er System B:

```
mqspeedtest_echo.sh AAA.LQ QMGRB
  MQSpeedtest Echo starting
 - Waiting for ping component to send messages
 NSRPL start, V18, Apr 12 2019
 Input queue is AAA.LQ
 Output queue is the ReplyToQ copied from inbound message
 Messages replied: 0000010
```

Observe on ping'er System A:

```
MQSpeedtest 1.2.1, 12-Apr-2019
MQSpeedtest Ping starting
If results are delayed, check echo component status
-C option not used, set env var MQ_CONNECT_TYPE=STANDARD
nsqping, V6.6.0004
 (C)Copyright 1995 - 2019, Nastel Technologies Inc. ALL RIGHTS RESERVED.

Start options:
        -m<QmgrName>                QMGRA
        -C<ConnectionInfo>
        -q<DestQueue>               QMGRB_AAA.RQ
        -d<PingInterval>            0
        -h<AP_Host>                 none [If none, AP facts are not sent]
        -P<AP_Port_#>               6000
        -f<StatsOutputFile>         none
        -a                          False [append to existing file and do not include a header line]
        -b<MsgsInBatch >            10
        -s<MsgSizeBytes>            0
        -w<WaitMsecBeforeTimeout>   60000
        -e<MsgExpiryTime,1/10 secs> -1 [-1 = unlimited]
        -U<AlternateUserID>         none
        -ccoa                       True [generate confirm-on-arrival report]
        -ccod                       True [generate confirm-on-delivery report]
        -cexp                       False [generate confirm-on-expiry report]
        -cexc                       False [generate confirm-on-exception report]
        -cdlq                       False [generate messages that will go into DLQ if undelivered]
        -v<verbose_mode>            0 [0=summary, 1=reports, 2=all]
        -u<AuthenticateUserID>
        -p<AuthenticatePassword>    none given
        -t                          off [trace option]

Tue Apr 16 17:40:38 2019
nsqping: No connection name, using MQCONN in mqmgr.c....
Pinging QMGRA(QMGRB_AAA.RQ) using 36 byte 10(msgs) batch....
Statistics for queue QMGRA(QMGRB_AAA.RQ) :
Summary Performance Indicators :
        MINIMUM_ROUND_TRIP       (8.6730 sec/msg)
        MAXIMUM_ROUND_TRIP       (8.7780 sec/msg)
```

```
        AVERAGE_ROUND_TRIP       (2.9086 sec/msg)
        AVERAGE_PROPAGATION_TIME (72.5476 sec/msg)
        AVERAGE_REFLECTION_TIME  (-69.6384 sec/msg)
        MESSAGES_SENT            (10)
        CONFIRMED_EXPIRIES       (0)
        CONFIRMED_DELIVERIES     (10)
        CONFIRMED_ARRIVALS       (10)
        CONFIRMED_EXCEPTIONS     (0)
        REPORTS_RECEIVED         (20)
        RESPONSES_RECEIVED       (10)
        MESSAGES_RECEIVED        (30)
        BYTES_SENT               (360)
        BYTES_RECEIVED           (360)
        RESPONSE_REQUEST_RATIO   (100.0000%)


General Performance Indicators :
        TOTAL_PUT_TIME           (0.0020 sec)
        TOTAL_GET_TIME           (8.7800 sec)
        AVERAGE_PUT_RATE         (5000.0000 msg/sec [180000.00 bytes/sec])
        AVERAGE_GET_RATE         (3.4169 msg/sec [41.00 bytes/sec])
        PUT_GET_RATE_RATIO       (146333.3333% [1463.33])

Message Performance Indicators :
        GROSS_ROUND_TRIP_RATE    (4.5548 msg/sec [81.99 bytes/sec])
        EFFECTIVE_ROUND_TRIP_RATE(2.2774 msg/sec)
        CONFIRMATION_OVERHEAD    (50.0000% [0.50])
        AVERAGE_ARRIVAL_RATE     (1000.0000 msg/sec])
        AVERAGE_DELIVERY_RATE    (1000.0000 msg/sec])
        AVERAGE_MSG_LATENCY      (1.0000 sec]) WITH QDEPTH(1000)
        MAXIMUM_MSG_LATENCY      (0.0100 sec]) WITH QDEPTH(10)
        TOTAL_BATCH_TIME         (8.7960 sec)
        TEST_COMPLETION_CODE     (0)

Pinging QMGRA(QMGRB_AAA.RQ) completed with RC(0)
```

This page intentionally left blank.

Index

This page intentionally left blank.