MA0T

MsgTest Utility

User Guide

V1.3.0

This page left intentionally content-less.
For two sided printing purposes.

Copyright and Disclaimers.

MsgTest (C) uses the PCRE regular expression library. Its copyright information is reproduced below.

1. Description.

MsgTest is a command line based program that provides the ability to execute scripts that simulate the flow of messages between applications that use IBM Websphere MQ (WMQ). A script consists of a series of commands that can be used to:

- Load data from a file to an MQ queue so that an application that reads messages from that queue can be tested.
- Read messages from an MQ queue and compare the actual message with a file to see if it matches the expected result.
- Get messages from one queue and put them on another.
- Put a message with an RFH2 header on an MQ queue to test a message flow written for the Websphere Business Integration Message Broker (a.k.a. MQSI) product.

MsgTest uses a pseudo-threading model that permits multiple tests to be run concurrently. If several tests are eligible for execution then the current command in each test is executed before the processing loop returns to the first test. MQ calls are made in a non-blocking manner so if for example two tests are both eligible for execution and are both issuing MQGET calls then, depending on the values of <ConcurrencyWait> and <CmdMaxWait> several hundred actual MQGET calls might be made before the <GetMsg> command succeeds or exceeds the <CmdMaxWait> value.

2. Installation

Unzip the "MA0T.zip" package into a directory of your choice making sure to use the "Use folder names" option when doing the Extract.

- For execution the following files are required. Their directories need to be in the "Path" system variable.
  - MsgTest.exe ← Extracted to "..\MsgTest\bin"
  - mqic32.dll ← Usually installed in "..\Program Files\IBM\Websphere MQ\bin". Requires Websphere MQ to be installed, either Client or Server versions.
  - libexpat.dll ← Extracted to ..\MsgTest\bin. Also can be extracted from "..\MsgTest\Expat-1.95.8\expat_win32bin_1_95_8.exe
- If you are using an XML editor that can make use of XML Schema then copy the file "../MsgTest/Schema/MsgTest.xsd" to an appropriate directory as per your XML editor's requirements.
- If you want to compile the utility to run on a platform other than Windows and MQ V6.0 then refer to Appendix "D" for more information.
- If you intend running the test scripts in directory ..\TestScripts\Scripts then installing to directory \MsgTest will save you from having to change the <Dir> values in the scripts.

3. New features

3.1 New in V1.1.0

The following bug fixes and features were added as of V1.1.0.

Bugs:

- Program abends with a bad memory reference if no MQMD variable is specified and <Format> is used in <PutMsg> command.
- Program abends if the Encoding of an incoming message is different to the local platform and an RFH2 header is present.
- XML comments could not be used to block out a section of script.

Features:

- Added the <Overlay> command to permit the changing of data in a buffer.
- Added a <NoConvert/> option for the <GetMsg> command which now uses MQGMO_CONVERT by default for the MQGET it issues. **Changed to <Convert> see details in V1.2.6 bugs.**
- Added a <GmoWait> option for the <GetMsg> command so blocking MQ calls can be made.
    - **Note. W**hen using this option a maximum <Concurrency> of 1 applies as the psuedo-threading model cannot handle blocked MQ calls.
- Added performance statistics gathering. See section 9 'Message Rates and Intervals'.
    - <IntervalStart> - Command defining start of a measurement interval
    - <IntervalEnd> - Command defining end of a measurement interval
    - <IntervalInMsgId> - <PutMsg> and <GetMsg> command option to place into or retrieve from the MsgId a high precision time value
    - <IntervalInCorrelId> - <PutMsg> and <GetMsg> command option to place into or retrieve from the CorrelId a high precision time value
- <RandomNum> a random number generation numeric variable.
- <ForEver> loop for ever. Can be exited using <BreakOnRC>. See below.
- Flow control functions for <GetMsg> and <PutMsg>
    - <BreakOnRC> Break out of For loop or Test if RC matches
    - <ContinueOnRC> Skip to end of a For loop or Test if RC matches
    - <SuppressRC> Change the specified reason code to 0 so the test can continue
- <MessageRate> option for the <PutMsg> command which can be used to control the number of messages sent every second. See section 9 'Message Rates and Intervals'.
- The From, To and Incr attributes of the <For> loop control command can now use variable sustitution. I.e. the number of times a put loop is executed can now be specified at program invocation as a parameter.

3.2 New in V1.2.0

The following bug fixes and features were added as of V1.2.0. I decided to release V1.2.0 before the completion of MessageSet support as enough extra useful pieces have been added that I felt a new release was justified.

My thanks to Clem, Neil, Tony, Marcus, Per and Kamy for their feedback and comments.

Bugs:

- Program does not break out of <For> <ForEver> loops if From and To values for MessageRate are the same and Mode="INCREMENTAL"
- Changed C++ // style comments to /* */ style so that the code doesn't break 'C' only compilers
- Corrected invalid code "if(0 <= lpSPB->eElementType)" which should have read "if(0 <= lpSPB->lParseDepth") while this didn't seem to cause a problem it was still invalid. Thanks to Marcus for pointing out the error of my ways.
- It is now no longer an error to attempt to load an empty file.
- The code that supports <SuppressRC> has been moved after that for <ContinueOnRC> and <BreakOnRC> as it was resetting the reason code before use in the later two commands.

Features:

- Added the named <Connection> option to the <Control> section. This permits MsgTest to connect to multiple queue managers at the same time.
- Added the <List> variable command for containing a set of <ListElements>. <List>'s can be loaded from a CRLF/LF delimited file at one <ListElement> per line.
- Added the <ForEach> command for iterating through the contents of a list.
- Added the <Browse Mode="FIRST | NEXT | AUTO"/> option to the <GetMsg> command.
- Flow control options have been added to the <GetFile> and <PutFile> commands.
    - <BreakOnRC> Break out of For loop or Test if RC matches
    - <ContinueOnRC> Skip to end of a For loop or Test if RC matches
    - <SuppressRC> Change the specified reason code to 0 so the test can continue
- Added the <Extract> command to permit retrieval of buffer data into a named variable.
- Added the <Substring> command to permit retrieval of data from a named variable.
- Added the <Double> variable command for declaring and setting floating point values.
- Added the maths functions <Add>, <Subtract>, <Multiply> and <Divide> for manipulating <Num> and <Double> variables.
- Added the <LogAndContinue> option to the <Compare> command so that if a data mismatch occurs a script can continue processing.
- Added the <Finally> element to permit script cleanup to occur even in the event of an error.
- Added the <Append/> to file option for the <PutFile>.
- Added the <Append/> to file option for <TestLog>, <TestReport>, <TestInterval> and <TestUserLog> control.
- Added the <StdIn> command for reading input from the console.
- Added the <StdOut> command for writing output to the console.
- Added the <UserLog> command and the<TestUserLog> control option.
- Added the DataVar="" attribute to the <PutMsg> command so a variable can be directly output without having to be written to a <Buffer> first.

- Added approximately 20 automatic variables mainly for use in the <UserLog> and <StdOut> commands.
- File related errors now contain the errno from errno.h.
- Extra statistics are now written to the Log and Report files at Test and Script completion.
- Message set support is not yet complete. Whilst the parsing code and some sample scripts have been generated and included with this release they only exist as I felt the effort to remove them would have delayed their future release even further. Also they are there in the hope that some of this supportpacs users might comment on the overall concept

3.3 New in V1.2.1

The following bug fixes and features were added as of V1.2.1.

Bugs:

- Escape character strings in the Format="" attributes of the various commands were not properly resolved.

Features:

- Added support for the use of a whole range of constant strings from the cmqc.h header file for use in the various fields of an MQMD variable and the new options elements of the <GetMsg> and <PutMsg> commands. See in Appendix E for more details.
- Added the ability to set the MQGMO.Options <GmoOptions>, MQGMO.MatchOptions <GmoMatchOptions> and MQPMO.Options <PmoOptions> values for the <GetMsg> and <PutMsg> commands respectively.
- Added the DataVar="" attribute to the <PutFile> command so a variable can be directly output without having to be written to a <Buffer> first.
- Added the <Group Mode="NEXT | LAST | AUTO"/> option to the <PutMsg> command to set the MQMF_MSG_IN_GROUP and MQMF_LAST_MSG_IN_GROUP flags.
- Added the <Clear/> option flag to the <List> variable command so that <List>'s inside a loop can be cleared before they reload their <ListElement>'s and <File>'s

3.4 New in V1.2.2

The following bug fixes and features were added as of V1.2.2.

Features:

- Added message set support. Message sets provide the capability to define message content including the capability to generate various headers such as RFH2, MQIIH and MQCIH.

3.5 New in V1.2.3

The following bug fixes and features were added as of V1.2.3.
My thanks to Frenk Ochse for contributing the SSL connection code.

Features:

- Added simple SSL connection support for the Client connections.
- Added the MqscCommand command so that MQSC script style commands can be submitted to a queue manager in PCF escaped text format. I.e. you can define and delete test queues etc. on the fly.
- Added the optional <RemoveDLH/> and <RemoveXQH/> flags to the <GetMsg> command.

3.6 New in V1.2.4

The following bug fixes and features were added as of V1.2.4.

Features:

- Added a Server version of the program MsgTestS.
- Added conditional execution with the addition of the <If…>, <ElseIf…> and <Else/> constructs.
- Added extra flow control commands <Continue/>, <Break/>, <ExitTest/> and <FailTest/>
- <Control> and <Default> elements and their sub-elements can now be omitted providing that a default queue manager is available when running the server version or the MQSERVER environment variable is defined when using the client version.
- Log and Report files now default to NONE if not specified in the <Control> section.
- Added <Length> function for variables and buffers.
- Added <Position> function for variables and buffers.

3.7 New in V1.2.5

The following bug fixes and features were added as of V1.2.5.

Features:

- Added a <MaxMsgLength> parameter to the Connection aggregate to permit the sending of very large messages.
- Added optional attributes FileExtension="" and Delimiter="" to the <FilePrefix> element of the <TestInterval> aggregate so the name and content of the Interval Information Record files can be tailored.
- Added a timestamp value as the first column in the Interval Information Record file.
- Added a Rem="" attribute to the Divide maths function.
- Added the <Eval> maths function so that the target variable can be different to the variable involved in the function.

3.8 New in V1.2.6

The following bug fixes and features were added as of V1.2.6.

Bugs:

- Although the detail for the <NoConvert> flag says that MQGMO_CONVERT is the default behaviour for MsgTest in fact this was not the case and so rather than potentially break a whole series of existing scripts the flag has been renamed to <Convert> and existing behaviour whereby the messages are untranslated will remain in effect.
- There was an error in the conversion of the length values in the RFH2 header that was causing an error on the MQPUT. Also the Encoding and CCSID of a message retrieved by <GetMsg> was not being preserved.

Features:

- Changed the <NoConvert> flag to <Convert> to reflect actual program behaviour.

3.8 New in V1.2.7

The following bug fixes and features were added as of V1.2.7.

Bugs:

- Fixed a bug that caused incorrect behaviour of <If> statements inside a loop in that once a TRUE condition had occured it wasn't reset for the next iteration.
- Fixed a bug in the IntervalInfoRec initialisation that was causing segment exceptions to occur in Linux.
- Fixed a bug in the internal handling of MsgId and CorrelId that was causing errors when one of the script substution characters "%", "[" and "]" occurred in the MsgId or CorrelId.
- Fixed a bug in the handling of <For> loop incrementing where for example a loop <For Name="I" From="1" To="4" Incr="2"> resulted in an iteration with a value of I=5 being run. Note the loop still ends with I=5 but the contents are no longer executed.

Features:

- Added the ability to access system Environment variables from within a script.
- Changed code so it issues a warning, rather than failing, when an attempt is made to connect to the same queue manager more than once.

3.9 New in V1.2.8

The following bug fixes and features were added as of V1.2.8.

Bugs:

- N/A.

Features:

- Added a new automatic variable “TestCurrentMsgRate” which contains the current rate at which messages are being sent if the <MessageRate…> is specified for in a <PutMsg…> command.
- Changed the <Compare…> command so it is possible to just compare the content of RFH2 headers. **Note**. This means that there is no longer a default compare file name of “Compare.txt’.

3.10 New in V1.2.9

The following bug fixes and features were added as of V1.2.9.

Bugs:

- N/A.

Features:

- Added regular expression support to the <Compare> command.
- Scripts command execution times are now logged.

3.10 New in V1.3.0

The following bug fixes and features were added as of V1.3.0.

Bugs:

- N/A.

Features:

- Whenever possible report the location of the mismatch between a buffers contents and the regular expression it is being matched against.

4. Background.

Over the years I have written quite a number of utility programs to manipulate messages on MQ queues. To handle the various options and parameters that can be set for an MQ message the utilities tended to have lots of parameters and switches. I finally decided that I would have a more flexible tool if I created a scripting language and the program to run it instead of the next variation on good old amqsput and so MsgTest was born.

The scripts containing the commands are written in XML. This was done to take advantage of the capability of several XML editors to use XML Schema Definitions to control editing during data entry. Also it provides syntactical validation of a script through the use of any validating XML parser with Schema Definition support.

The program itself is written in vanilla 'C' as on more than one occasion the lack of a C++ compiler on a given platform or a JVM has stopped me from using one or another of my old utility programs whereas I've nearly always been able to get access to a 'C' compiler.

The code has reached the point at which it should be re-factored and rewritten but I just don't have the time to do so however the program is still functioning correctly and it is still reasonably easy to add new features.

5. Script structure.

The annotated XML fragment shown below is used to illustrate the three main structural elements of a MsgTest script.

```
<MsgTest>
    <!-- The <Control> element contains parameter elements   -->
    <!-- related to queue manager connection, script logging -->
    <!-- and other aspects of overall program control.       -->
    <!-- Only one can be defined. The element is optional.   -->
    <Control>
    </Control>
    <!-- The <Defaults> element is used to set the default   -->
    <!-- values for command parameter elements and declare   -->
    <!-- initial values for variables.                       -->
    <!-- Only one can be defined. The element is optional.   -->
    <Defaults>
    </Defaults>
    <!-- Each <Test> element contains the variable declarations -->
    <!-- and commands that are to be executed. A script can     -->
    <!-- contain multiple <Test>'s which are run independently. -->
    <Test Name="Test01">
    </Test>
</MsgTest>
```

6. Script command summary.

The following list is a summary of the available commands. The commands and their parameter elements are discussed in detail in section "7. Script Commands and Parameters".

6.1 Variable Declaration Commands.

- **<Num>** Declare and set the value of an integer variable.
- **<Double>** Declare and set the value of a floating point variable.
- **<Str>** Declare and set the value of a string variable.
- **<List>** Delcare and set the values of a series of <ListElement>'s.
- **<RandomNum>** Declare and set the value of a numeric variable to a random number
- **<MQMD>** Declare and set the supported values of an MQMD control block
- **<RFH2>** Declare and set the supported values of an RFH2 control block

6.2 Buffer Manipulation Commands.

- **<GetFile>** Read a file into a buffer.
- **<PutFile>** Write the contents of a buffer into a file.
- **<InlineData>** Load inline data into a buffer.
- **<Overlay>** Overlay the buffer with the specified data.
- **<Extract>** Extract buffer data into a named variable
- **<Substring>** Retrieve a substring from a variable
- **<GetMsg>** Get a message from an MQ queue into a buffer.
- **<PutMsg>** Put the contents of a buffer onto an MQ queue as a message.
- **<Compare>** Compare the contents of a buffer with a file.
- **<Select>** Select an message from a message set and load it into a buffer.
- **<Length>** Get the length of a variable or buffer.
- **<Position>** Get the position of a specified string in a variable or buffer.

6.3 Wait Commands.

- **<WaitOnMsgDepth>** Wait until an MQ queue reaches a specified depth.
- **<WaitOnTime>** Wait for a specified number of seconds, milli-seconds or until a specified time has been reached.

6.4 Flow Control Commands.

- **<For>** Perform the commands within the <For> loop a specified number of times.
- **<ForEver>** Loop until a <BreakOnRC> exits
- **<ForEach>** Iterate through the contents of a <List>
- **<If><ElseIf><Else>** Conditional execution of contained script elements
- **<Continue>** Skip the remaining statements in a containing <For…> loop or skip to the end of the test if not inside a <For…> loop.
- **<Break>** Exit from the containing <For…> loop or test.
- **<ExitTest>** Exit from a <Test>.
- **<FailTest>** Fail a <Test> and exit.
- **<Finally>** Script execution resumes here in the event of a command failure.

6.5 System Commands.

- **<System>** Call the operating system to execute a command.
- **<MqscCommand>** Send an MQSC script command to a queue manager.

6.6 Interval Commands.

- **<IntervalStart>** Signifies the starting point of an interval.
- **<IntervalEnd>** Signifies the ending point of an interval.

6.7 User Input and Output Commands.

- **<UserLog>** Write information to the user log file.
- **<StdIn>** Read information from the console.
- **<StdOut>** Write information to the system console.

6.8 Maths Commands.

- **<Add>** Add a value to a <Num> or <Double> variable.
- **<Subtract>** Subtract a value from a <Num> or <Double> variable.
- **<Multiply>** Multiply a <Num> or <Double> variable by a value.
- **<Divide>** Divide a <Num> or <Double> variable by a value.
- **<Eval>** Evaluate a simple expression that operates on <Num> or <Double> variables

6.9 Transactional Commands.

- **<Commit>** Commit a unit of work for a connection.
- **<Backout>** Backout a unit of work for a connection.

Note. Sample scripts are provided in Appendix B.

7. Script Commands and Parameters.

The XML elements that make up a script are described below.

After each element is a suffix of the form (x-y-z) which describes how the element can be used where:

- "x" is the element type o(Optional), m(Mandatory), f(Flag) or c(Contains sub-elements)
- "y" is v(Variables permitted), f(Fixed values only) or n(N/A)
- "z" the element has a d(Default value) or n(No default value)

7.1 <Control> Elements.

The sub-elements of the <Control> element are:

- **<Connection Name="">** – (oc-n-n) – This element is used to define the parameters necessary for connecting to a queue manager as a client. Multiple connections are permitted. It is used to contain the <QMgr>, <Channel>, <Host> and <Port> elements that define a connection although these maybe still be defined outside of a <Connection> element for backwards compatibility.
  Note. Even though multiple client connections are permitted they do not behave as part of a co-ordinated Unit-Of-Work.
  Its sole attribute is:
  - **Name**="" – (o-f-d) – This attribute is used to specify the name of the connection and is referenced through the Connection="" attributes of the <GetMsg>, <PutMsg> and <WaitOnMsgDepth> commands. It has a default value of "DefConn".

  Its parameters are:
  - **<QMgr>** – (m-v-n) – Name of the queue manager to connect to as a client. If specified as NONE then a connection will not be attempted.
  - **<Channel>** – (m-v-n) – Name of the SVRCONN channel for the client connection.
  - **<Host>** – (m-v-n) – Name of the host on which the <QMgr> queue manager is running. Used in making the client connection
  - **<Port>** – (m-v-n) – Port number on which the connection is to be made
  - **<KeyRepository>** – (o-f-n) – Fully qualified path and filename without extension of the location of the key database file. Refer to the Websphere MQ Application Programming Reference manual for more details.
  - **<SSLCipherSpec>** – (o-f-n) – For SSL enabled client connection, this specifies the SSL CipherSpec used on the channel. Refer to the Websphere MQ Security manual for more details.
  - **<SSLPeerName>** – (o-f-n) – For SSL enabled client connections, this specifies an SSL peer name. Refer to the WebSphere MQ Script (MQSC) Command Reference manual for more details.
  - **<MaxMsgLength>** – (o-f-n) – Maximum message length that can be sent over the client connection. The actual SVRCONN channel definition and queue manager maximum length are also involved in the negotiation process.
  - **Note**. For the server version only the <QMgr> parameter is required. All the other <Connection> parameters are ignored if present.

- ***Deprecated* - <QMgr>** – (m-v-n) – Name of the queue manager to connect to as a client. If specified as NONE then a connection will not be attempted.
- ***Deprecated* - <Channel>** – (m-v-n) – Name of the SVRCONN channel for the client connection
- ***Deprecated* - <Host>** – (m-v-n) – Name of the host on which the <QMgr> queue manager is running. Used in making the client connection
- ***Deprecated* - <Port>** – (m-v-n) – Port number on which the connection is to be made.
- **<TestLog>** – (oc-n-n) – The test log file contains the detailed log of the execution of the script. It contains two elements used to designate where it is written:
    - **<File>** – (m-v-n) – File name of the test log file. If specified as NONE then logging is suppressed.
    - **<Dir>** – (m-v-n) – Directory to contain the test log file.
    - **<Append>** – (o-f-n) – Append log output to any pre-existing file.
    - **Note**. If <TestLog> is not specified it defaults to NONE.
- **<TestReport>** – (oc-n-n) – The test report file contains a high level report on the execution of the script. Namely when a test starts and finishes and the overall results of running the script.
    - **<File>** – (m-v-n) – File name of the test report file. If specified as NONE then reporting is suppressed.
    - **<Dir>** – (m-v-n) – Directory to contain the test report file.
    - **<Append>** – (o-f-n) – Append report output to any pre-existing file.
    - **Note**. If <TestReport> is not specified it defaults to NONE.
- **<TestInterval>** – (oc-n-n) – If this optional element is specified then Interval Information Records will be written to files contained in the specified directory.
    - **<FilePrefix FileExtenstion="" Delimiter="">** – (m-v-n) – File name prefix for the test interval files. Actual file names are built up from the FilePrefix, the interval name and a file type of "iir". See section 9 'Message Rates and Intervals' for more detail.
      It also has two optional attributes
        - **FileExtension=""** – (o-f-d) – The file extension to be used for the Interval Information Record files. The default value is ".iir"
        - **Delimiter=""** – (o-f-d) – The value to be used as the delimiter between the elements of each row of the Interval Information Record file. Use a prefix of 0x to specify a hex value for example in an ASCII runtime environment a value of 0x09 creates a tab delimited file.
    - **<Dir>** – (m-v-n) – Directory to contain the test report file.
    - **<Append>** – (o-f-n) – Append interval output to any pre-existing file.
- **<TestUserLog>** – (oc-n-n) – The test user log file contains output generated by the use of the <UserLog> command within a script.
    - **<File>** – (m-v-n) – File name of the test use log file.
    - **<Dir>** – (m-v-n) – Directory to contain the test user log file.
    - **<Append>** – (o-f-n) – Append user log output to any pre-existing file.

- **<TestMsgSet Name="">** – (oc-n-n) – Test message set files contain message templates and messages. You can specify multiple TestMsgSet elements to load all the files that make up a message set.

  Its sole attribute is:

  - **Name**="" – (o-f-d) – The name of the MsgSet to which the contents of this file are to be appended.

  Its parameters are:

  - **<File>** – (m-v-n) – File name of the test message set file.
  - **<Dir>** – (m-v-n) – Directory to contain the test message set file.
  - **<ParserName>** – (o-f-n) – The name of the parser to be used whenreading the file. Current parsers are MSGSET and NEWLINE.
- **TestMaxWait>** – (o-f-d) – Maximum time, in seconds, a test will wait for the successful completion of a command. If it is exceeded the test will fail. The default value is 60 seconds.
- **<NextLogMsgInterval>** – (o-f-d) – The time, in seconds, between the writing of log messages whilst a command is waiting to complete. The default value is 5 seconds.
- **<Concurrency>** – (o-f-d) – The number of tests that can be running at once. If a value of 1 is specified the tests will be run sequentially. The default value is 1.
- **<ConcurrencyWait>** – (o-f-d) The time, in milliseconds, to wait if all the concurrent tests are waiting for an external event like the arrival of a message or the expiry of a time interval. The default value is 20 milli-seconds.
- **<OnError>** – (o-f-d) – Defines the behaviour if an error occurs during the execution of a test. It has two values LOG and FATAL. If set to LOG the failure of the test is written to the log and any other tests continue ruuning. If set to FATAL then all testing stops **at once**. The default value is LOG.

7.2 <Defaults> Elements.

The <Defaults> element is used to define the default values for command parameter elements used in command elements that make up a <Test> definition.

7.2.1 Variable Declaration Commands.

Note. Variables defined in the <Defaults> element are used as the initial value for that variable in any <Test> that uses it. They do not function as global variables.
Note. Refer to section "8. Script Variables" for more information on how variables behave.

<Num>

<Num Name="" Format=""> – (o-v-n) – is used to declare a numeric variables' value.
Its attributes are:

- **Name**="" – (m-f-n) – is used to specify the variables name.
- **Format**="" – (o-f-d) – is used to specify a 'c' style printf format string that is to be used when variable substitution is performed, it default value is "%d".

Examples:

```
<Num Name="Count">1</Num>
<Num Name="Count" Format="%02d">1</Num>
<Num Name="Count" Format="FixedBit%03d">%1%</Num>
```

<Double>

<Double Name="" Format=""> – (o-v-n) – is used to declare a floating point numeric variables' value.
Its attributes are:

- **Name**="" – (m-f-n) – is used to specify the variables name.
- **Format**="" – (o-f-d) – is used to specify a 'c' style printf format string that is to be used when variable substitution is performed, it default value is "%lf".

Examples:

```
<Double Name="Count">1</Double>
<Double Name="Count" Format="%08.2lf">1</Double>
<Double Name="Count" Format="Count=%08.2lf">%1%</Double>
```

<Str>

<Str Name="" Format=""> – (o-v-n) – is used to declare a string variables' value.
Its attributes are:

- **Name**="" – (m-f-n) – is used to specify the variables name.
- **Format**="" – (o-f-d) – is used to specify a 'c' style printf format string that is to be used when variable substitution is performed, it default value is "%s".

Examples:

```
<Str Name="FileName">TestFile.In</Str>
<Str Name="FileName">%1%</Str>
<Str Name="FileName" Format="FixedBit%s">%1%</Str>
```

<RandomNum>

<RandomNum Name="" From="" To="" Format=""> – (o-v-n) – is used to declare a pseudo random number variable. In the first example the pseudo random number generator is always seeded with the number 1 in the second example where no value is specified then the current time in seconds is used as a seed value. I.e. where you require the pseudo-random sequence to be repeated specify your own seed value.
Its attributes are:

- **Name**="" – (m-f-n) – is used to specify the variables name.
- **From=""** (o-v-n) – Minimum value for the pseudo random number.
- **To=""** (o-v-n) – Maximum value for the pseudo random number.
- **Format**="" – (o-f-d) – is used to specify a 'c' style printf format string that is to be used when variable substitution is performed, it default value is "%d".

Examples:

```
<RandomNum Name="Count" From="1" To="10">1</RandomNum>
<RandomNum Name="Count" From="1" To="10" Format="%02d"/>
```

<List>

<List Name=""> – (o-v-n) – is used to declare a List variable.
Its sole attribute is:

- **Name**="" – (m-f-n) – is used to specify the variables name.

Its parameters are:

- **<ListElement>** – (o-f-n) – is used to define the value of a list element. Multiple <ListElement>'s can be defined and the may be used in combination with the <ListFile> option.
- **<ListFile>** – (o-v-d) – is the name of a CRLF/LF delimited file the lines of which are to be loaded as <ListElements>.
- **<Dir>** – (o-v-d) – is the name of the directory where the <ListFile> is to be found. Its default value is the variable substitution string %Dir%.
- **<Clear/>** – (o-f-n) – this flag can be used so that <List>'s inside a loop are cleared before they load their list elements from the specified <ListElement>'s and <ListFile>.

<MQMD>

<MQMD Name=""> – (o-v-n) – is used to declare an MQMD control block. Refer to the IBM's documentation for the meaning and default values of the various parameters.
Note. Refer to Appendix E for the values of cmqc.h header files constants that can be used.
Its sole attribute is:

- **Name**="" – (m-f-n) – is used to specify the variables name.

Its parameters are:

- **<Version>** – (o-f-d) – Version number.
- **<Report>** – (o-f-d) – Report options. Can be specified in hex by using 0x as a prefix
- **<MsgType>** – (o-f-d) – Message type.
- **<Expiry>** – (o-f-d) – Expiry time in tenths of a second
- **<Feedback>** – (o-f-d) – Feedback code
- **<Encoding>** – (o-f-d) – Encoding. Can be specified in hex by using 0x as a prefix
- **<CodedCharSetId>** – (o-f-d) – Coded character set id.
- **<Format>** – (o-f-d) – Format.
- **<Priority>** – (o-f-d) – Priority
- **<Persistence>** – (o-f-d) – Persistence
- **<MsgId>** – (o-v-d) – Message Id. Can be specified in hex by using 0x as a prefix.
- **<CorrelId>** – (o-v-d) – Correlation Id. Can be specified in hex by using 0x as a prefix.
- **<ReplyToQ>** – (o-v-d) – Reply to queue.
- **<ReplyToQMgr>** – (o-v-d) – Reply to queue manager.

For the following parameters to take effect you will need to set <PutMsg> option <SetIdentityContext/> or <SetAllContext>

- **<UserIdentifier>** – (o-v-d) – User identifier.
- **<AccountingToken>** – (o-v-d) – Accounting token. Can be specified in hex by using 0x as a prefix.
- **<ApplIdentityData>** – (o-v-d) – Application identity data.

For the following parameters to take effect you will need to set <PutMsg> option <SetAllContext/>

- **<PutApplType>** – (o-v-d) – Putting application type.
- **<PutApplName>** – (o-v-d) – Putting application name.
- **<PutDate>** – (o-v-d) – UTC date the message was put on the queue.
- **<PutTime>** – (o-v-d) – UTC time the message was put on the queue.
- **<ApplOriginData>** – (o-v-d) – Application origin data.

Version 2 and beyond MQMD parameters

- **<GroupId>** – (o-v-d) – Group Id. Can be specified in hex by using 0x as a prefix.
- **<MsgSeqNumber>** – (o-v-d) – Message sequence number.
- **<Offset>** – (o-v-d) – Segmentation offset value.
- **<MsgFlags>** – (o-v-d) – Message flags. Grouping and segmentation.
- **<OriginalLength>** – (o-v-d) – Original length.

File load parameters

- **<MQMDFile>** – (o-v-d) – is the name of a file containing some or all of the MQMD parameters defined above. Its default value is an empty string meaning no parameters are to be loaded from a file. Refer to section "9. Files" for its layout.
- **<Dir>** – (o-v-d) – is the name of the directory where the <MQMDFile> is to be found. Its default value is the variable substitution string %Dir%.

Internally the MQMD is created as follows. First the default structure “MQMD_DEFAULT” is copied to the control block, then any values specified for the variable of the same name in the <Defaults> <MQMD> variable element are applied, then any values from the <MQMDFile> are used, and finally the parameters defined in the <MQMD> element of the test are applied. The default for <MQMDFile> is an empty string meaning no file is to be loaded.

Examples:

```
<MQMD Name=”MD01”>
    <MsgId/> ← Causes the message id to be set to nulls.
    <CorrelId>MyCorrelId</CorrelId>
</MQMD>

<MQMD Name=”MD02”>
    <MsgType>1</MsgType>
    <Priority>6</Priority>
    <MsgId>0x414d51204d5154553033343532302020 4cdb2b4020000201</MsgId>
    <CorrelId/>
</MQMD>

<MQMD Name=”MD03”>
    <MQMDFile>MyMQMD.xml</MQMDFile>
</MQMD>

<MQMD Name="MD04">
    <!—See Appendix E for details of available constants →
    <Version>MQMD_VERSION_2</Version>
    <Report>MQRO_NONE</Report>
    <MsgType>MQMT_DATAGRAM</MsgType>
    <Expiry>MQEI_UNLIMITED</Expiry>
    <Feedback>0</Feedback>
    <Encoding>MQENC_NATIVE</Encoding>
    <CodedCharSetId>MQCCSI_Q_MGR</CodedCharSetId>
    <Format>MQFMT_NONE</Format>
    <Priority>MQPRI_PRIORITY_AS_Q_DEF</Priority>
    <Persistence>MQPER_NOT_PERSISTENT</Persistence>
    <MsgId>MQMI_NONE</MsgId>
    <CorrelId>MQCI_NONE</CorrelId>
    <ReplyToQ>MyReplyToQueue</ReplyToQ>
    <ReplyToQMgr>MyReplyToQueueManager</ReplyToQMgr>
    <GroupId>MQGI_NONE</GroupId>
    <MsgSeqNumber>1</MsgSeqNumber>
    <Offset>0</Offset>
    <MsgFlags>MQMF_MSG_IN_GROUP+MQMF_LAST_MSG_IN_GROUP</MsgFlags>
    <OriginalLength>MQOL_UNDEFINED</OriginalLength>
</MQMD>
```

<RFH2>

<RFH2 Name=""> – (o-v-n) – is used to declare an RFH2 control block.
Its sole attribute is:

- **Name**="" – (m-f-n) – is used to specify the variables name.

Its parameters are:

- **<Folder>** – (m-f-n) – This element is used to contain the data for an RFH2 folder. As the data is in XML format it must be enclosed in a CDATA section. More than one <Folder> element is permitted. At least one is required. Folders are named after the highest-level XML element, so the folder in the example below would be named "four" and "three". When a folder with the same name is specified more than once the last one declared is used.
- **<Format>** – (o-f-d) – This optional value can be used to set the format of the data following the RFH2 header. It defaults to MQFMT_NONE.

- **<RFH2File>** – (o-v-d) – is the name of a file containing the RFH2 parameters defined above. Its default value is an empty string meaning no parameters are to be loaded from a file. Refer to section "9. Files" for its layout.
- **<Dir>** – (o-v-d) – is the name of the directory where the <RFH2File> is to be found. Its default value is the variable substitution string %Dir%.

The RFH2 is created as follows. First the default structure "MQRFH2_DEFAULT" is copied to the control block, then any values specified for the <RFH2> variable in the <Defaults> element are applied, then any values specified as parameters in the <RFH2File> are used to set their respective fields, and finally the parameters defined in the <RFH2> are applied. Folders are appended in the order they are defined.

Example:

```
<RFH2 Name="RF01">
    <Folder><![CDATA[<four>4</four>]]></Folder>
    <Folder><![CDATA[<three>3</three>]]></Folder>
    <Format>MQSTR</Format>
</RFH2>
```

Note. As of V1.2.2 RFH2 headers can also be generated by using MessageSet functionality.

7.2.2 Buffer Manipulation Commands.

<GetFile>

<GetFile> – (c-n-n) – This command is used to load a file into a buffer.
Its parameters are:

- **<Buffer>** – (o-v-d) – The name of the buffer into which the file will be loaded. Its default value is "TestBuff". If the buffer does not exist it will be created.
- **<File>** – (o-v-d) – File name of the file to be loaded into the buffer. Its default value is "FileIn.txt".
- **<MQMDFile MQMD="">** – (o-v-d) – File from which MQMD control block information will be loaded. Its default value is an empty string that signifies that there is no MQMD data to be loaded.
    - **MQMD**="" – (o-f-d) – attribute is used to specify the name of the MQMD variable to be created or if it already exists to be set. If not specified the <Test>'s default MQMD variable "TestMQMD" is used.
- **<RFH2File RFH2="">** – (o-v-d) – File name from which the RFH2 folder and format specifier information will be loaded. Its default value is an empty string that signifies that there is no RFH2 data to be loaded.
    - **RFH2**="" – (o-f-d) – attribute is used to specify which RFH2 variable will be to created or if it already exists to be set. If not specified the <Test>'s default RFH2 variable "TestRFH2" is used.
- **<Dir>** – (o-v-d) – Directory containing the file/s. Its default value is the variable substitution string %Dir%.
- **<CmdMaxWait>** – (o-v-d) – The maximum time in seconds to wait before failing the command. If not specified it defaults to the <TestMaxWait> value which in turn defaults to 60 seconds.
- ***<SuppressRC>*** – *<Test> only. See Section 7.3.2.*
- ***<BreakOnRC>*** – *<Test> only. See Section 7.3.2.*
- ***<ContinueOnRC>*** – *<Test> only. See Section 7.3.2.*

<PutFile>

<PutFile DataVar=""> – (c-n-n) – This command is used to write the contents of a buffer to a file.
Its attributes are:

- **DataVar**="" – (o-f-d) – Name of the variable that contains the data to be used instead of the contents of the **<Buffer>**. This option is provided to avoid the need to create a buffer, it is particularly useful in combination with a <ForEach> loop to iterate through the contents of a <List>.
    - **Note.** If this attribute is used the contents of the **<Buffer>** are ignored.
    - **Note.** The contents of the **<Buffer>** are not modified.

Its parameters are:

- **<Buffer>** – (o-v-d) – The name of the buffer from which the file will be written. Its default value is "TestBuff". If the buffer does not exist an error occurs and the test is terminated.
- **<File>** – (o-v-d) – File name of the file to be written from the buffer. Its default value is "FileOut.txt".
- **<MQMDFile MQMD="">** – (o-v-d) – File name to which the MQMD control block information will be written. Its default value is an empty string that signifies that MQMD data is not to be saved.
    - **MQMD**="" – (o-f-d) – attribute is used to specify which MQMD variable is to be saved. If not specified the <Test>'s default MQMD variable "TestMQMD" is saved.
- **<RFH2File RFH2="">** – (o-v-d) – File name of the file that the RFH2 folders and format specifier will be written to. Its default value is an empty string that signifies that the RFH2 data is not to be saved.
    - **RFH2**="" – (o-f-d) – attribute is used to specify which RFH2 variable is to be saved. If not specified the <Test>'s default RFH2 variable "TestRFH2" is saved.
- **<Dir>** – (o-v-d) – Directory to contain the <File>, <MQMDFile>, and <RFH2File> files. Its default value is the variable substitution string %Dir%.
- **<Append>** – (o-f-n) – If the file already exists data is appended otherwise it will be created.
    - **Note.** If the Append option is set as a default there is no way to turn it off in the <PutFile> commands inside a <Test>.
- ***<SuppressRC>*** – *<Test> only. See Section 7.3.2.*
- ***<BreakOnRC>*** – *<Test> only. See Section 7.3.2.*
- ***<ContinueOnRC>*** – *<Test> only. See Section 7.3.2.*

<InlineData>

<InlineData> – (c-n-n) – This command is used to load a buffer with "in-line" data.
Its parameters are:

- **<Buffer>** – (o-v-d) – The name of the buffer into which the data will be loaded. Its default value is "TestBuff". If the buffer does not exist it will be created.
- **<Data>** – (m-f-n) – This element is used to contain the data to be loaded into the buffer. If the data is in XML format or contains XML delimiters i.e. <> then it must be placed in a CDATA section.

<Overlay>

<Overlay Pos="" Len=""> – (c-n-n) – This command is used to overlay the contents of a buffer with the data specified.
Its attributes are:

- **Pos**="" – (m-v-d) – Is the position in the buffer at which the bytes to be overlayed are to be placed. The first byte is defined as being at Pos="1" rather than Pos="0" as most data definitions are written in those terms.
- **Len=**"" – (o-v-d) – Is the length of the data to be written at the specified postion. If Len="0" is used then MsgTest calculates the length of the resolved <Data> string and uses it as the number of bytes being overlayed. Its default value is 0.

If either the Pos="" or Len="" parameters or a combination of both would exceed the end of the existing data then a new buffer that is large enough will be allocated and the existing data copied to it before the overlay operation is executed.
Its parameters are:

- **<Buffer>** – (o-v-d) – The name of the buffer to be overlayed with the specified data. Its default value is "TestBuff".
- **<Data>** – (m-v-n) – This element is used to contain the data to be loaded into the buffer. If the data is in XML format or contains XML delimiters i.e. <> then it must be placed in a CDATA section. It is permissible to use variable substitution. I.e. <Data>%DD%</Data> will be resolved to the two character representation of the day value contained in the %DD% automatic variable.

<Extract>

<Extract Name="" Pos="" Len=""> – (c-n-n) – This command is used to extract data from a buffer into a named variable.
Its attributes are:

- **Name**="" – (m-f-n) – is used to specify the target variables name.
- **Pos**="" – (m-v-d) – Is the position in the buffer from which the bytes to be extracted are located. The first byte is defined as being at Pos="1" rather than Pos="0" as most data definitions are written in those terms.
- **Len=**"" – (m-v-d) – Is the length of the data to be extracted.

If either the Pos="" or Len="" parameters or a combination of both would exceed the end of the existing data then an error occurs.
Its parameters are:

- **<Buffer>** – (o-v-d) – The name of the buffer from which the data is to be extracted. Its default value is "TestBuff".

<Substring>

<Substring Name="" From="" Pos="" Len=""> –
(c-n-n) – This command is used to take a substring from the specified variable.
Its attributes are:

- **Name**="" – (m-f-n) – is used to specify the target variables name.
- **From**="" – (m-f-n) – is used to specify the source variables name.
- **Pos**="" – (m-v-d) – Is the position in the source variable from which the bytes to be extracted are located. The first byte is defined as being at Pos="1" rather than Pos="0" as most data definitions are written in those terms.
- **Len=**"" – (m-v-d) – Is the length of the data to be extracted.

If either the Pos="" or Len="" parameters or a combination of both would exceed the end of the existing data then an error occurs.

<GetMsg>

<GetMsg MQMD="" RFH2="" Connection=""> – (c-n-n) – This command is used to get a message from a Websphere MQ queue into a buffer.
Note. Parameters in "***bold italics***" are <Test> only parameters. Refer to Section 7.3.2 for more information.
Its attributes are:

- **MQMD=**"" – (o-f-d) – Name of the MQMD variable that is to be used.
    - If it is specified then the values contained in it are used to make the MQGET call and then updated after the call has been made.
    - If it is **not** specified then the <Test>'s default MQMD variable "TestMQMD" is set to "MQMD_DEFAULT" before the MQGET call and updated with the results after the call has been made.
    - **Note.** Once an MQMD variable has been updated after a <GetMsg> or <PutMsg> command all fields are set and thus the next time that <MQMD> variable, including the default "TestMQMD", is specifically or implicitly used those are the values that are used.
- **RFH2=**"" – (o-f-d) – Name of the RFH2 variable that is to be updated if the message retrieved by the MQGET call has an RFH2 header. If it is not specified and an RFH2 header is present the <Test>'s default RFH2 variable "TestRFH2" is updated.
- **Connection=**"" – (o-v-d) – Name of a connection defined in the <Control> section. If it is not specified the default connection name "DefConn" is used.

Its parameters are:

- **<Buffer>** – (o-v-d) – The name of the buffer into which the message will be loaded. Its default value is "TestBuff". If the buffer does not exist it will be created.
- **<Q>** – (o-v-d) – The name of the queue the messages are to be read from. Its default value is "MSGTEST.TEST.OUT".
- **<CmdMaxWait>** – (o-v-d) – The maximum time in seconds to wait before failing the command. If not specified it defaults to the <TestMaxWait> value which in turn defaults to 60 seconds.
- **<GmoOptions>** – (o-v-d) – Used to specify MQGMO.Options values for the call. It cannot be used in conjunction with the <Syncpoint> flag. See Appendix E for the values of cmqc.h header constants that can be used.
- **<GmoMatchOptions>** – (o-v-d) – Used to specify MQGMO.MatchOptions values for the call. See Appendix E for the values of cmqc.h header constants that can be used.
- **<GmoWait>** – (o-v-n) – Used to specify an MQGMO.WaitInterval in milliseconds. If this option is used then the <Concurrency> value can only be 1 as the MQGET calls are blocked which stops MsgTest's pseudo-threading model from working properly. However multiple instances of MsgTest can be run to overcome this.
- **<MsgId>** – (o-v-n) – This parameter is provided as a shortcut method for setting the message id to be used in an MQGET call rather having to declare an <MQMD> variable and then reference it in the <GetMsg> command. Refer to the "RequestReply" example in "Appendix B. Example Scripts" for an illustration of both methods of setting a message id. It has no default value, as it is an override for the MQMD variable. Use <MsgId/> to reset the field to nulls i.e. MQMI_NONE.
- **<CorrelId>** – (o-v-n) – This parameter is provided as a shortcut method for setting the correlation id to be used in an MQGET call rather having to declare an <MQMD> variable and then reference it in the <GetMsg> command. Refer to the "RequestReply" example in "Appendix B. Example Scripts" for an illustration of both methods of setting a correlation id. It has no default value, as it is an override for the MQMD variable. Use <CorrelId/> to reset the field to nulls i.e. MQCI_NONE.

- **<GroupId>** – (o-v-n) – This parameter is provided as a shortcut method for setting the group id to be used in an MQGET. Use <GroupId/> to reset the field to nulls i.e. MQGI_NONE.
- **<MsgSeqNumber>** – (o-v-n) – This parameter is provided as a shortcut method for setting the message sequence number to be used in an MQGET. It is only relevant in combination with the <GmoOptions> MQMO_MATCH_MSG_SEQ_NUMBER.
- ***<Browse Mode="FIRST | NEXT | AUTO">*** *– <Test> only. See Section 7.3.2.*
- ***<Convert/>*** *– <Test> only. See Section 7.3.2.*
- ***<IntervalInMsgId/>*** *– <Test> only. See Section 7.3.2.*
- ***<IntervalInCorrelId>*** *– <Test> only. See Section 7.3.2.*
- ***<Syncpoint/>*** *– <Test> only. See Section 7.3.2*
- ***<SuppressRC>*** *– <Test> only. See Section 7.3.2.*
- ***<BreakOnRC>*** *– <Test> only. See Section 7.3.2.*
- ***<ContinueOnRC>*** *– <Test> only. See Section 7.3.2.*

<PutMsg>

<PutMsg MQMD="varname" RFH2="varname" DataVar="varname**" Connection="**connname**">** – (c-n-n) – This command is used to put the contents of a buffer onto a Websphere MQ queue.
Note. Parameters in "***bold italics***" are <Test> only parameters. Refer to Section 7.3.2 for more information.
Its attributes are:

- **MQMD**="varname" – (o-f-d) – Name of the MQMD variable that is to be used.
    - If it is specified then the values contained in it are used to make the MQPUT call and then updated after the call has been made.
    - If it is **not** specified then the <Test>'s default MQMD variable "TestMQMD" is set to "MQMD_DEFAULT" before the MQPUT call and updated with the results after the call has been made.
    - **Note.** Once an MQMD variable has been updated after a <GetMsg> or <PutMsg> command all fields are set and thus the next time that <MQMD> variable, including the default "TestMQMD", is specifically or implicitly used those are the values that are used.
- **RFH2**="varname" – (o-f-d) – Name of the RFH2 variable that contains the folder and format specifier information to be used to generate the RFH2 header for the message being put. If it is not specified then an RFH2 header will not be added even if the <Test>'s default RFH2 variable "TestRFH2" has been populated, i.e. if you want to use the contents of "TestRFH2" then you need to specify RFH2="TestRFH2".
- **Connection**="connname" – (o-v-d) – Name of a connection defined in the <Control> section. If it is not specified the default connection name "DefConn" is used.
- **DataVar**="varname" – (o-f-d) – Name of the variable that contains the data to be used instead of the contents of the **<Buffer>**. This option is provided to avoid the need to create a buffer, it is particularly useful in combination with a <ForEach> loop to iterate through the contents of a <List>.
    - **Note.** If this attribute is used the contents of the **<Buffer>** are ignored.
    - **Note.** The contents of the **<Buffer>** are not modified.

Its parameters are:

- **<Buffer>** (o-v-d) The name of the buffer from which the data is put. Its default value is "TestBuff". If the buffer does not exist an error occurs and the test is terminated.
- **<Q>** (o-v-d) The name of the queue the messages are to be put to. Its default value is "MSGTEST.TEST.IN".

- **<QMgr>** (o-v-d) The name of the target queue manager. This is an optional element it default value is an empty string which means the default queue manager will be used.
- **<PmoOptions>** – (o-v-d) – Used to specify MQPMO.Options values for the call. It cannot be used in conjunction with the <SetIdentityContext>, <SetAllContext> or <Syncpoint> flags. See Appendix E for the values of cmqc.h header constants that can be used.
- **<MsgId>** – (o-v-n) – This parameter is provided as a shortcut method for setting the message id to be used in an MQPUT call rather having to declare an <MQMD> variable and then reference it in the <PutMsg> command. Refer to the "RequestReply" example in "Appendix B. Example Scripts" for an illustration of both methods of setting a message id. It has no default value, as it is an override for the MQMD variable.
- **<CorrelId>** – (o-v-n) – This parameter is provided as a shortcut method for setting the correlation id to be used in an MQPUT call rather having to declare an <MQMD> variable and then reference it in the <PutMsg> command. Refer to the "RequestReply" example in "Appendix B. Example Scripts" for an illustration of both methods of setting a correlation id. It has no default value, as it is an override for the MQMD variable. Use <MsgId/> to reset the field to nulls i.e. MQMI_NONE.
- **<GroupId>** – (o-v-n) – This parameter is provided as a shortcut method for setting the group id to be used in an MQPUT. Use <GroupId/> to reset the field to nulls i.e. MQGI_NONE. To send messages in a group you will need to declare an MQMD variable and set the appropriate <MsgFlags>.
- *<Group Mode=" NEXT | LAST | AUTO"> – <Test> only. See Section 7.3.2.*
- **<MsgSeqNumber>** – (o-v-n) – This parameter is provided as a shortcut method for setting the message sequence number to be used in an MQPUT.
- **<Format>** – (o-v-n) – This parameter is provided as a shortcut method for setting the message format to be used in an MQPUT call rather having to declare an <MQMD> variable and then reference it in the <PutMsg> command. It has no default value, as it is an override for the MQMD variable. Use <CorrelId/> to reset the field to nulls i.e. MQCI_NONE.
- ***<MessageRate>** – <Test> only. See Section 7.3.2.*
- ***<IntervalInMsgId/>** – <Test> only. See Section 7.3.2.*
- ***<IntervalInCorrelId/>** – <Test> only. See Section 7.3.2.*
- ***<SetIdentityContext/>** – <Test> only. See Section 7.3.2*
- ***<SetAllContext/>** – <Test> only. See Section 7.3.2.*
- ***<Syncpoint/>** – <Test> only. See Section 7.3.2*
- ***<SuppressRC>** – <Test> only. See Section 7.3.2.*
- ***<BreakOnRC>** – <Test> only. See Section 7.3.2.*
- ***<ContinueOnRC>** – <Test> only. See Section 7.3.2.*

<Compare>

<Compare> – (c-n-n) – This command is used to compare the contents of the current buffer with a file, optionally controlling the comparison with a masking definition file.
Its parameters are:

- **<Buffer>** – (o-v-d) – The name of the buffer containing the data to be compared. Its default value is "TestBuff". If the buffer does not exist an error occurs and the test is terminated.
- **<File>** – (o-v-n) – File name of the file to be compared on a **byte by byte** basis with the buffer. As of V1.2.8 there is no longer a default value, this permits RFH2 only comparisons.
- **<Mask>** – (o-v-d) – Name of the file that controls the byte-by-byte comparison with the buffer. Its default value is an empty string that means that no masking takes place. The current masking characters are:
    - "=" This means that the corresponding bytes must match exactly.
    - "9" The corresponding byte in the buffer must be numeric.
    - "*" Do not compare the corresponding byte with anything at all any value at this position is acceptable.
- **<Regex>** – (o-v-n) – File name of a file containing a regular expression to be compared with the buffer.
- **<RFH2Comp R1="rfh1" R2="rfh2"/>** – (o-f-d) – This parameter is provided for comparing two <RFH2> variables as part of the comparison operation. **Note.** The match comparison is done on the basis that all folders in R1 must be found in R2 however R2 can have extra folders that are not found in R1.
    - **R1**="rfh1" – (m-f-n) – Name of the first <RFH2> variable.
    - **R2**="rfh2" – (m-f-n) – Name of the second <RFH2> variable.
- **<Dir>** – (o-v-d) – Directory containing the <File> and <Mask> files. Its default value is the variable substitution string %Dir%.
- **<LogAndContinue/>** – (f-f-n) – Use this flag to prevent the script from terminating if the comparison fails.
    - **Note.** The number of successful and failed comparisons are available as Automatic Variables. See Section "8.2 Automatic Variables" for more information.
    - **Note.** Also the number of failed comparisons is added to the value returned by MsgTest itself thus ensuring a non-zero completion code in the event of a compare failing. See section "11 MsgTest Return Code".
    - **Note.** If the LogAndContinue option is set as a default there is no way to turn it off in the <Compare> commands inside a <Test>.

<Select>

<Select MsgSetName="name" MsgNum="num" MQMD="varname"> – (c-n-n) – This command is used to select a message by its number from a message set and load it into a buffer.
Note. Parameters in "***bold italics***" are <Test> only parameters. Refer to Section 7.3.2 for more information.
Its attributes are:

- **MsgSetName=**"name" – (o-v-d) – Name of the message set to retrieve the message from. **Note**. The message set name corresponds to that used in the <TestMessageSet> element in the <Control> section.
- **MsgNum=**"num" – (o-v-d) – Zero based message number of the message to be retrieved.
- **MQMD=**"varname" – (o-f-d) – Name of the MQMD variable that is to be used if the message contains an MQMD segment.
    - If it is **not** specified then the <Test>'s default MQMD variable "TestMQMD" is used.

Its parameters are:

- **<Buffer>** – (o-v-d) – The name of the buffer into which the message will be loaded. Its default value is "TestBuff". If the buffer does not exist it will be created.
- ***<SuppressRC>*** – *<Test> only. See Section 7.3.2.*
- ***<BreakOnRC>*** – *<Test> only. See Section 7.3.2.*
- ***<ContinueOnRC>*** – *<Test> only. See Section 7.3.2.*

<Length>

There are no <Defaults> settings for <Length>.

<Position>

There are no <Defaults> settings for <Position>.

7.2.3 Wait Commands.

<WaitOnMsgDepth>

<WaitOnMsgDepth Connection=""> – (c-n-n) – This command is used to wait for a Websphere MQ queue to reach or exceed a specified depth.
Its attributes are:

- **Connection**="" – (o-v-d) – Name of a connection defined in the <Control> section. If it is not specified the default connection name "DefConn" is used.

Its parameters are:

- **<Q>** – (o-v-d) – The name of the queue whose depth is to be checked. Its default value is "MSGTEST.TEST.OUT".
- **<Depth>** – (o-v-d) – The depth that has to be reached or exceeded before the command will complete. Its default value is 1.
- **<CmdMaxWait>** – (o-v-d) – The maximum time in seconds to wait before failing the command. If not specified and no messages arrive on the queue <WaitOnMsgDepth> will wait for 999999 seconds (11.5days).
    - **Note**. It does **not** default to <TestMaxWait> like other commands where a command wait interval is applicable.

<WaitOnTime>

<WaitOnTime> – (c-n-n) – This command is used to delay processing of a <Test> by a specified number of seconds, milliseconds or wait until a specified time has been reached.
Note. If <Concurrency> is greater than 1 then in the case of the <Interval> and <Time> options any other active <Test>'s will continue executing. The <IntervalMillis> option is currently implemented as a blocking call.
Note. The three options are mutually exclusive.
Its parameters are:

- **<Interval>** – (o-v-d) – The interval, in seconds, to wait before continuing with the next command. This is the default option and its default value is 10.
- **<IntervalMillis>** – (o-v-d) – The interval, in milli-seconds, to wait before continuing with the next command.
- **<Time>** – (o-v-d) – The time of day, expressed as a string of the format HHMMSS, at which execution will resume.

7.2.4 Flow Control Commands.

There are no <Defaults> settings for the flow control commands.

7.2.5 System Command.

<System>

<System> – (c-n-n) – This command is used to call the operating system to get it to execute a command or program.

- **<Command>** – (m-v-n) – The command string to be passed to the operating system for execution.
- **<IgnoreRC/>** – (f-n-n) – Flag used to tell MsgTest not to check the Return Code from the command. Only use it if you know the command or program you are calling is badly behaved and does not return a value of zero on success.

The <System> command provides a great deal of flexibility to your scripts. It can be used to:

- Call another utility program to perform an action not supported by MsgTest.
- Execute the application a test message was just loaded for.
- Move files from one directory to another.
- Etc.

<MqscCommand>

There are no <Defaults> settings for MqscCommand.

7.2.6 Interval Commands.

There are no <Defaults> settings for the interval commands.

7.2.7 User Input and Output Commands.

There are no <Defaults> settings for the user input and output commands.

7.2.8 Maths Commands.

There are no <Defaults> settings for the maths commands.

7.2.9 Transactional Control Commands.

WARNING. The <Commit> and <Backout> commands only apply to the specified Connection and result in a single-phase commit i.e. they **do not** provide Unit Of Work coordination across multiple connections. This means that there is a small but real risk that messages can be **lost** or **duplicated** when using this utility especially across multiple connections. Whilst there are strategies to reduce the risk it still exists.

Note. As stated in the usage notes for MQDISC "Because of the differences between environments, applications which are intended to be portable should ensure that the unit of work is committed or backed out before the application ends." MsgTest always issues an MQBACK call prior to the MQDISC.
"MQ's

<Commit>

<Commit Connection=""> – (c-n-n) – This command is used to commit a Unit Of Work on the specified connection.
Its attributes are:

- **Connection**="" – (o-v-d) – Name of a connection defined in the <Control> section. If it is not specified the default connection name "DefConn" is used.

<Backout>

<Backout Connection=""> – (c-n-n) – This command is used to back out a Unit Of Work on the specified connection.
Its attributes are:

- **Connection**="" – (o-v-d) – Name of a connection defined in the <Control> section. If it is not specified the default connection name "DefConn" is used.

7.3. <Test> Elements.

The <Test Name=""> element is used to contain the commands that comprise a given test. The "Name" attribute is used in the <TestLog> and <TestReport> files to identify the test.

7.3.1 Variable Declaration Commands.

When used in a <Test> the variable declaration commands and their parameter elements have the same meaning as when they are defined in the <Defaults> element. Refer to section "7.2.1 Variable Declaration Commands".

7.3.2 Buffer Manipulation Commands.

When used in a <Test> the buffer manipulation commands and their parameter elements have the same meaning as when they are defined in the <Defaults> element, where a default is applicable. Refer to section "7.2.2 Buffer Manipulation Commands".

Over and above this <GetFile>, <PutFile>, <GetMsg> and <PutMsg> have the following <Test> only options.

<GetFile>

- <Buffer> – <Defaults> and <Test>. See Section 7.2.2
- <File> – <Defaults> and <Test>. See Section 7.2.2
- <MQMDFile MQMD="varname"> – <Defaults> and <Test>. See Section 7.2.2
- <RFH2File RFH2="varname"> – <Defaults> and <Test>. See Section 7.2.2
- <Dir> – <Defaults> and <Test>. See Section 7.2.2
- <CmdMaxWait> – <Defaults> and <Test>. See Section 7.2.2
- **<SuppressRC>** – (o-f-n) – Used to suppress the reason code from the fopen() and fread() calls. Up to 16 may be defined. For example <SuppressRC>2</SuppressRC> could be used to suppress a ENOENT(file not found) error and permit the script to continue executing.
- **<BreakOnRC>** – (o-f-n) – If the reason code specified matches the reason code from the fopen() and fread() calls and:
    - The <GetFile> command is inside a loop then the loop is terminated and execution of the script continues at the next statement after the </For>, </ForEver> or </ForEach> end tag.
    - The <GetFile> command exists outside of any loops then execution of the <Test> is terminated as successful and any commands beyond the <GetFile> command containing the <BreakOnRC> option are bypassed.

  Up to 16 may be defined.
- **<ContinueOnRC>** – (o-f-n) – If the reason code specified matches the reason code from the fopen() and fread() calls and:
    - The <GetFile> command is inside a loop then execution skips to the end of the loop bypassing any commands between the <GetFile> command containing the <ContinueOnRC> and the </For>, </ForEver>, </ForEach> end tag. The loop count is then checked and if still valid the loop continues executing.
    - The <GetFile> command exists outside of any loops then execution of the <Test> is terminated as successful and any commands beyond the <GetFile> command containing the <ContinueOnRC> option are bypassed.

  Up to 16 may be defined.

<PutFile>

<PutFile> – (c-n-n) – This command is used to write the contents of a buffer to a file.
Its parameters are:

- **<**Buffer> – <Defaults> and <Test>. See Section 7.2.2
- <File> – <Defaults> and <Test>. See Section 7.2.2
- <MQMDFile MQMD=”varname”> – <Defaults> and <Test>. See Section 7.2.2
- <RFH2File RFH2=”varname”> – <Defaults> and <Test>. See Section 7.2.2
- <Dir> – <Defaults> and <Test>. See Section 7.2.2
- <Append> – <Defaults> and <Test>. See Section 7.2.2
- **<SuppressRC>** – (o-f-n) – Used to suppress the reason code from the fopen() and fwrite() calls. Up to 16 may be defined. For example <SuppressRC>2</SuppressRC> could be used to suppress a ENOENT(file not found) error and permit the script to continue executing.
- **<BreakOnRC>** – (o-f-n) – If the reason code specified matches the reason code from the fopen() and fwrite() calls and:
    - The <PutFile> command is inside a loop then the loop is terminated and execution of the script continues at the next statement after the </For>, </ForEver> or </ForEach> end tag.
    - The <PutFile> command exists outside of any loops then execution of the <Test> is terminated as successful and any commands beyond the <PutFile> command containing the <BreakOnRC> option are bypassed.

  Up to 16 may be defined.
- **<ContinueOnRC>** – (o-f-n) – If the reason code specified matches the reason code from the fopen() and fwrite() calls and:
    - The <PutFile> command is inside a loop then execution skips to the end of the loop bypassing any commands between the <PutFile> command containing the <ContinueOnRC> and the </For>, </ForEver> or <ForEach> end tag. The loop count is then checked and if still valid the loop continues executing.
    - The <PutFile> command exists outside of any loops then execution of the <Test> is terminated as successful and any commands beyond the <PutFile> command containing the <ContinueOnRC> option are bypassed.
- Up to 16 may be defined.

<GetMsg>

- <*Buffer*> – <Defaults> and <Test>. See Section 7.2.2
- <*Q*> – <Defaults> and <Test>. See Section 7.2.2
- <*CmdMaxWait*> – <Defaults> and <Test>. See Section 7.2.2
- <*GmoOptions*> – <Defaults> and <Test>. See Section 7.2.2
- <*GmoMatchOptions*> – <Defaults> and <Test>. See Section 7.2.2
- <*GmoWait*> – <Defaults> and <Test>. See Section 7.2.2
- <*MsgId*> – <Defaults> and <Test>. See Section 7.2.2
- <*CorrelId*> – <Defaults> and <Test>. See Section 7.2.2
- <*GroupId*> – <Defaults> and <Test>. See Section 7.2.2
- <*MsgSeqNumber*> – <Defaults> and <Test>. See Section 7.2.2
- **<Browse Mode="FIRST | NEXT | AUTO"/>** – (m-f-n) – Flag to turn on message queue browsing so message gets are non-destructive.
    - **Mode="FIRST | NEXT | AUTO"** – (m-f-n) – One of three possible modes.
        - **FIRST** – Set get message option MQGMO_BROWSE_FIRST
        - **NEXT** – Set get message option MQGMO_BROWSE_NEXT
        - **AUTO** – If the <GetMsg> command is inside a <For> or <ForEver> loop then on the first iteration of the loop use Mode="FIRST" otherwise use Mode="NEXT". If <GetMsg> is not contained in a loop then use of auto mode is considered to be an error.
- **<Convert/>** – (f-n-n) – Flag used to turn on the MQGMO_CONVERT option on the MQGET call.
- **<IntervalInMsgId Name=""/>** – (o-n-n) – Used indicate that the MQMD.MsgId contains a high precision time value for interval measurement. <IntervalInMsgId> and <IntervalInCorrelId are mutually exclusive.
    - **N**ame="" – (m-f-n) – Name of the interval for which the interval statistics are to be recorded. See section 9 'Message Rates and Intervals' for more information.
- **<IntervalInCorrelId Name=""/>** – (o-n-n) – Used indicate that the MQMD.CorrelId contains a high precision time value for interval measurement. <IntervalInMsgId> and <IntervalInCorrelId are mutually exclusive.
    - **N**ame="" – (m-f-n) – Name of the interval for which the interval statistics are to be recorded. See section 9 'Message Rates and Intervals' for more information.
- **<Syncpoint/>** – (f-n-n) – Execute the MQGET call under syncpoint.
- **<RemoveDLH/>** – (f-n-n) – Remove the Dead Letter Header if it exists.
- **<RemoveXQH/>** – (f-n-n) – Remove the Transmit Queue Header if it exists.
- **<SuppressRC>** – (o-f-n) – Used to suppress the reason code from the MQGET call. Up to 16 may be defined. For example <SuppressRC>2110</SuppressRC> could be used to suppress a message conversion error and permit the script to continue executing.
- **<BreakOnRC>** – (o-f-n) – If the reason code specified matches the reason code from the MQGET call and:
    - The <GetMsg> command is inside a loop then the loop is terminated and execution of the script continues at the next statement after the </For>, </ForEver> or </ForEach> end tag.
    - The <GetMsg> command exists outside of any loops then execution of the <Test> is terminated as successful and any commands beyond the <GetMsg> command containing the <BreakOnRC> option are bypassed.

  Up to 16 may be defined.

- **<ContinueOnRC>** – (o-f-n) – If the reason code specified matches the reason code from the MQGET call and:
    - The <GetMsg> command is inside a loop then execution skips to the end of the loop bypassing any commands between the <GetMsg> command containing the <ContinueOnRC> and the </For>, </ForEver> or </ForEach> end tag. The loop count is then checked and if still valid the loop continues executing.
    - The <GetMsg> command exists outside of any loops then execution of the <Test> is terminated as successful and any commands beyond the <GetMsg> command containing the <ContinueOnRC> option are bypassed.

  Up to 16 may be defined.

<PutMsg>

- <*Buffer*> – <Defaults> and <Test>. See Section 7.2.2
- <*Q*> – <Defaults> and <Test>. See Section 7.2.2
- <*QMgr*> – <Defaults> and <Test>. See Section 7.2.2
- <*PmoOptions*> – <Defaults> and <Test>. See Section 7.2.2
- <*MsgId*> – <Defaults> and <Test>. See Section 7.2.2
- <*CorrelId*> – <Defaults> and <Test>. See Section 7.2.2
- <*GroupId*> – <Defaults> and <Test>. See Section 7.2.2
- **<Group Mode="NEXT | LAST | AUTO"/>** – (m-f-n) – Flag to turn on message group flags in the MQMD.MsgFlags field.
    - **Mode="NEXT | LAST | AUTO"** – (m-f-n) – One of three possible modes.
        - **NEXT** – Set message flag MQMF_MSG_IN_GROUP
        - **LAST** – Set message flag MQMF_LAST_MSG_IN_GROUP
        - **AUTO** – If the <PutMsg> command is inside a <For> or <ForEach> loop then on all iterations except the last it will use Mode="NEXT". On the LAST iteration, i.e. for <For> loops the To="" value matches the loop counter or the last element of the <List> of the <ForEach> loop has been reached, then Mode="LAST" is used. If <PutMsg> is not contained in a loop then use of auto mode is considered to be an error. It is also illegal to use a <Group> switch in a <ForEver> loop.
- <*MsgSeqNumber*> – <Defaults> and <Test>. See Section 7.2.2
- <*Format*> – <Defaults> and <Test>. See Section 7.2.2
- **<MessageRate From="" To="" Incr="" Interval="" Mode="">** – (o-f-n) – Used to control the rate at which messages are sent. See section 9 'Message Rates and Intervals' for more information.
    - **From=""** – (m-f-n) – Minimum message rate. All modes.
    - **To=""** – (m-f-n) – Maximum message rate. All modes.
    - **Incr=""** – (m-f-n) – Message rate increment. Mode="INCREMENTAL" only.
    - **Interval=""** – (m-f-n) – Time in seconds before the message rate is incremented. Mode="INCREMENTAL" only.
    - **Mode="BURST | PACED | RANDOM | INCREMENTAL"** – (m-f-n) – One of four possible modes.
        - **BURST** – In burst mode for each second while a script is running in a <For>/<ForEver> loop a random value between the From="" and To="" is chosen and the chosen number of messages are put on the target queue as fast as possible.

        - **PACED** – In paced mode for each second while a script is running in a <For>/<ForEver> loop a random value between the From="" and To="" is chosen and then chosen number of messages are put on the target queue evenly spaced across that second. I.e. if a target rate of twenty messages a second was chosen then a message will be placed on the target queue as close to every 50 milliseconds as possible.
        - **RANDOM** – In random mode for each second while a script is running in a <For>/<ForEver> loop a random value between the From="" and To="" is chosen and then chosen number of messages are put on the target queue randomly spaced across that second. I.e. if a target rate of ten messages a second was chosen then each message will be placed on the target queue somewhere between 0 and 180 milliseconds.
        - **INCREMENTAL** – Incremental mode is different to the others in that the number of message to be sent each second starts at the From="" value and increases to the To="" value in Incr="" increments with the Incr="" increment being applied every Interval="" seconds. During any given second the messages are evenly spaced across the second as per the paced mode. **Note.** The other main difference for incremental mode is that when the To="" value is reached it breaks out of the containing <For>/<ForEver> loop or <Test> in the same way as a matching <BreakOnRC>, in fact it is specifically designed to be use in a <ForEver> loop.
- **<IntervalInMsgId Name=""/>** – (o-n-n) – Used indicate that the MQMD.MsgId is to be generated by MsgTest and will contain a high precision time value for interval measurement. <IntervalInMsgId> and <IntervalInCorrelId are mutually exclusive.
    - **N**ame="" – (m-f-n) – Name of the interval for which the interval statistics are to be recorded. See section 9 'Message Rates and Intervals' for more information.
- **<IntervalInCorrelId Name=""/>** – (o-n-n) – Used indicate that the MQMD.CorrelId is to be generated by MsgTest and will contain a high precision time value for interval measurement. <IntervalInMsgId> and <IntervalInCorrelId are mutually exclusive.
    - **N**ame="" – (m-f-n) – Name of the interval for which the interval statistics are to be recorded. See section 9 'Message Rates and Intervals' for more information.
- **<SetIdentityContext/>** – (f-n-n) – Turn on the ability to set identity context fields in the MQMD
- **<SetAllContext/>** – (f-n-n) – Turn on the ability to set both identity and origin context fields in the MQMD
- **<Syncpoint>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.
- **<SuppressRC>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.
- **<BreakOnRC>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.
- **<ContinueOnRC>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.

<Select>

- <*Buffer*> – <Defaults> and <Test>. See Section 7.2.2
- **<SuppressRC>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.
- **<BreakOnRC>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.
- **<ContinueOnRC>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.

<Length>

<Length Name="" Of=""> – (c-n-n) – This command is used to determine the length of a variable or buffer.
Its attributes are:

- **Name=**"" – (o-f-n) – Is the name of the variable that is to be updated with the length of the variable or buffer specified in the Of="" attribute.
    - If the Name="" variable exists it will be updated.
    - If the Name="" variable does not exist it will be created.
    - If the Of="" variable does not exist a value of -1 is returned.
- **Of=**"" – (o-f-n) – Is the name of the variable or buffer whose length is to be determined.
    - The list of variables and buffers is searched in the following order. Firstly the <Test>'s local variable list is checked for a match, then the <Default> variables are checked, after that the program invocation variables are checked and finally the <Test>'s buffer list is checked for a matching name.
    - Variable names can be subscripted.

<Position>

<Position Name="" Of="" In="" From=""> – (c-n-n) – This command is used to determine the position of a string in a variable or buffer.
Its attributes are:

- **Name=**"" – (o-f-n) – Is the name of the variable that is to be updated with the position of the Of="" string within the variable or buffer specified by the In="" attribute.
    - If the Name="" variable exists it will be updated.
    - If the Name="" variable does not exist it will be created.
    - If the In="" variable does not exist a value of -4 is returned.
    - If the resolved value of the Of="" string has a length of zero -3 is returned.
    - If the combination of the From="" offset and the length of the resolved value of the Of="" string would overrun the end of the In="" variable or buffer then a value of -2 is returned.
    - If the resolved value of the Of="" string cannot be found -1 is returned.
- **Of=**"" – (o-v-n) – Is the string to be located within the variable or buffer specified by the In="" attribute.
    - The value specified in the Of="" attribute is resolved at run time i.e. it can contain variables so assuming Var=ABC then Of="%Var1%" resolves to Of="ABC".
- **In=**"" – (o-f-n) – Is the name of the variable or buffer that is to be searched for the Of="" string.
    - The list of variables and buffers is searched in the following order. Firstly the <Test>'s local variable list is checked for a match, then the <Default> variables are checked, after that the program invocation variables are checked and finally the <Test>'s buffer list is checked for a matching name.
    - Variable names can be subscripted.
- **From=**"" – (o-v-d) – Is the offset in the variable or buffer specified by In="".
    - The first byte is defined as being at From="1" rather than From="0" as most data definitions are written in those terms.
    - The From="" attribute has a default value of 1.

7.3.3 Wait Commands.

When used in a <Test> the wait commands and their parameter elements have the same meaning as when they are defined in the <Defaults> element. Refer to section “7.2.3 Wait Commands”.

7.3.4 Flow Control Commands.

<For>

<For Name=”” From=”” To=”” Incr=”” Format=””> (c-n-n) This command element is used to repetitively execute the command elements contained within it.
Its attributes are:

- **Name=””** (o-f-d) – Is used to give the <For> loop a name. It is also the name of an implicit <Num> variable that can be used as a substitutable variable in the same manner as an explicitly defined <Num Name=””> style variable.
- **From=””** (o-v-n) – Starting value for the for loop counter.
- **To=””** (o-v-n) – Ending value for the for loop counter.
- **Incr””** (o-v-n) – For loop counter increment. If positive the for loop counter value increases until it reaches the To=”” value, if negative the for loop counter value decreases until it reaches the To=”” value.
- **Format=””** – (o-f-d) – is used to specify a ‘c’ style printf format string that is to be used when variable substitution is performed for the implicit <Num> variable, its default value is “%d”.

<ForEver>

<ForEver Name=”” Format=””> (c-n-n) This command element is used to repetitively execute the command elements contained within it until a <BreakOnRC> condition is matched in a <GetFile>, <PutFile>, <GetMsg> or <PutMsg> command contained therein.
Its attributes are:

- **Name=””** (o-f-d) – Is used to give the <ForEver> loop a name. It is also the name of an implicit <Num> variable that can be used as a substitutable variable in the same manner as an explicitly defined <Num Name=””> style variable.
- Implicit – From – ForEver loops start at a count of 1.
- Implicit – To – ForEver loops never end however the count wraps back to 1 when 2,147,483,647 is reached.
- Implicit – Incr – The ForEver loops implicit Name=”” variable is incremented by 1.
- **Format=””** – (o-f-d) – is used to specify a ‘c’ style printf format string that is to be used when variable substitution is performed for the implicit <Num> variable, its default value is “%d”.

<ForEach>

<ForEach Name="" List="" Format=""> (c-n-n) This command element is used to repetitively execute the command elements contained within it for each element of the specified List.
Its attributes are:

- **Name=""** (o-f-d) – Is used to give the <ForEach> loop a name. It is also the name of an implicit <Str> variable that can be used as a substitutable variable in the same manner as an explicitly defined <Str Name=""> style variable. The value of the named variable is set to the value of the current ListElement in the specified List.
- **List=""** (o-f-d) – Is use to specify the list variable that contains the elements to be iterated over.
- **Format=""** – (o-f-d) – is used to specify a 'c' style printf format string that is to be used when variable substitution is performed for the implicit <Str> variable, its default value is "%s".

<Finally>

<Finally> (c-n-n) This command element can be used to contain commands that are to be executed in the event that the failure of a preceding command would result in the termination of a script. Execution of the script resumes with the command following the opening <Finally> tag of the <Finally> block. It is intended for use in cleaning up resources in the event of a script failure.

- **Note.** The closing </Finally> tag of the <Finally> block must be the last element in a script. This means the <Finally> block and its command elements may not be defined inside a <For/ForEver/ForEach> loop.
- **Note.** If an error occurs once inside the <Finally> block then the script is terminated.

<If>

<If Cond=""> (c-n-n) This command element is used to contain elements that are to be conditionally executed.
Its only attribute is:

- **Cond=""** (o-v-n) – Is used to specify a condition that can be evaluated as TRUE or FALSE.

The comparison is performed by breaking the condition into the two parts on either side of any of the following predicates:

- **.**EQ**.** – EqualTo
- **.**NE**.** – NotEqualTo
- **.**LT**.** – LessThan
- **.**GT**.** – GreaterThan
- **.**LE**.** – LessThanOrEqual
- **.**GE**.** – GreaterThanOrEqual

A numeric comparison is performed unless the value of either of the parts is in a single quoted string or contains a non-numeric character in which case a string comparison is performed. If the either string contains a decimal point but is otherwise numeric a comparison of doubles is perfomed.
Note. If any of the following commands <GetFile>, <PutFile>, <GetMsg>, <PutMsg>, <WaitOnDepth>, <Commit>, <Backout>, <System>, <MqscCommand> which update LastCC and LastRC is immediately followed by an <If> that contains a condition that checks LastCC or LastRC then even if the command fails the <Test> will continue at the <If>. This includes <If> statements that contain an <ElseIf> which checks LastCC or LastRC.
Note. <BreakOnRC>, <ContinueOnRC> and <SuppressRC> still function in the same way i.e. skipping to the end of the current <For> loop or the end of the <Test>

<ElseIf/>

<ElseIf Cond=””/> (c-n-n) This command element serves as a target if the **Cond**ition of the containing <If> statement or a preceding <ElseIf/> statement evaluates to FALSE.
Note. Because the element serves as a target it **MUST** always be specified as an empty tag, see example below.
Its only attribute is:

- **Cond=””** (o-v-n) – Same as in the <If/> statement above.

<Else/>

<Else/> (c-n-n) This command element serves as a target if the **Cond**ition of the containing <If> statement or a preceding <ElseIf> statement evaluates to FALSE.
Note. Because the element serves as a target it **MUST** always be specified as an empty tag, see following example.

```
<Num Name="TheNum"/>
<ForEver Name="I">
    <StdOut>Enter a number between 0 and 999.</StdOut>
    <StdIn Name="TheNum"/>
    <If Cond="0.LE.%TheNum%">
        <If Cond="999.GE.%TheNum%">
            <Break/>
        </If>
    </If>
    <StdOut>Try again. Enter a number between 0 and 999.</StdOut>
</ForEver>
<Num Name="Work">42</Num>
<Subtract Name="Work">%TheNum%</Subtract>
<If Cond="0.LT.%Work%">
    <StdOut>You entered %TheNum% which if I add %Work% becomes 42.</StdOut>
<ElseIf Cond="0.EQ.%Work%"/>
    <StdOut>You entered 42.</StdOut>
<Else/>
    <Multiply Name="Work">-1</Multiply>
    <StdOut>You entered %TheNum%  which if I subtract %Work% becomes 42.</StdOut>
</If>
```

<Break>

<Break> (o-f-n) This command works as follows:

- If the <Break> command is inside a loop then the loop is terminated and execution of the script continues at the next statement after the </For>, </ForEver> or </ForEach> end tag.
- If the <Break> command exists outside of any loops then execution of the <Test> is terminated as successful and any commands beyond the <Break> command are bypassed until either a <Finally> tag or the end of the <Test> is reached.

<Continue>

<Continue> (o-f-n) This command works as follows:

- If the <Continue> command is inside a loop then execution skips to the end of the loop bypassing any commands between the <Continue> command and the </For>, </ForEver> or </ForEach> end tag. The loop count is then checked and if still valid the loop continues executing.
- If the <Continue> command exists outside of any loops then execution of the <Test> is terminated as successful and any commands beyond the <Continue> command are bypassed until either a <Finally> tag or the end of the <Test> is reached.

<ExitTest>

<ExitTest> (o-f-n) This command works as follows:

- Regardless of where the <ExitTest> command exists, i.e. inside or outside of any loops, execution of the <Test> is terminated as successful and any commands beyond the <ExitTest> command are bypassed until either a <Finally> tag or the end of the <Test> is reached.

<FailTest>

<FailTest> (o-f-n) This command works as follows:

- Regardless of where the <FailTest> command exists, i.e. inside or outside of any loops, execution of the <Test> is terminated as failed and any commands beyond the <FailTest> command are bypassed until either a <Finally> tag or the end of the <Test> is reached.

7.3.5 System Commands.

<System>

When used in a <Test> the <System> command and its parameter elements have the same meaning as when they are defined in the <Defaults> element. Refer to section “7.2.5 System Command”.

<MqscCommand>

< MqscCommand Connection=””> – (c-n-n) – This command is used to send an MQSC script command to a connected queue manager as PCF escaped text, there are no default settings for <MqscCommand>.
Its attributes are:

- **Connection=**”” – (o-v-d) – Name of a connection defined in the **<**Control> section. If it is not specified the default connection name “DefConn” is used.

Its parameters are:

- **<Buffer>** – (o-v-d) – The name of the buffer into which the PCF command String output will be placed. Its default value is “MqscBuff”. If the buffer does not exist it will be created.
- **<Q>** – (o-v-d) – The name of the queue the command messages are sent to. Its default value is “SYSTEM.ADMIN.COMMAND.QUEUE”.
- **<MqscReplyDynamicQ>** – (o-v-d) – The name of the reply-to-queue the to which command reply messages will be returned. It is created as a Temp Dynamic queue using “SYSTEM.MQSC.REPLY.QUEUE” as its model queue. Its default value is"MSGTEST.MQSC.REPLY.QUEUE.*”.
- **<Command>** – (m-v-n) – The MQSC Script command string to be passed to the target queue manager for execution.
- **<CmdMaxWait>** – (o-v-d) – The maximum time in seconds to wait before failing the command. If not specified it defaults to 30 seconds. **Note.** This a timeout does not always mean the command itself failed to execute, rather that the reply message/s did not arrive within the designated period.
- **<SuppressRC>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.
- **<BreakOnRC>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.
- **<ContinueOnRC>** – (o-f-n) – Refer to 7.3.2 on <GetMsg> above.

7.3.6 Interval Commands.

Whilst defined in <Test>'s the interval commands operate at a script scope and thus can span multiple tests so an <IntervalStart> can be placed in one <Test> and the corresponding <IntervalEnd> can be placed in another.

<IntervalStart>

<IntervalStart Name=""> (c-n-n) This command element is used to identify when an interval start event should be recorded.
Its attribute is:

- **Name=""** (o-f-d) – Is the name of the interval. Each <IntervalStart> must have a corresponding <IntervalEnd> command and the names must be unique across the entire script.

<IntervalEnd>

<IntervalEnd Name=""> (c-n-n) This command element is used to identify when an interval end event should be recorded.
Its attribute is:

- **Name=""** (o-f-d) – Is the name of the interval. Each <IntervalEnd> must have a corresponding <IntervalStart> command and the names must be unique across the entire script

7.3.7 User Input and Output Commands.

<UserLog>

<UserLog> (o-f-n) This command element is used to specify the content to be written to the user log file specified in the <Control> section by the <TestUserLog> option. The data contained within the tag may include substitutable variables.

<StdIn>

<StdIn Name=""> (o-f-n) This command element is used to get user input from the console. Its attribute is:

- **Name=""** (m-f-n) – Is the name of the variable that is to be set to the value input from the console.
    - **Note**. If the variable has not been previously declared then a new <Str> variable with the specified name is created.

<StdOut>

<StdOut> (o-f-n) This command element is used to specify the content to be written to the console. The data contained within the tag may include substitutable variables.

7.3.8 Maths Commands.

Whether Integer or Floating-Point operations are executed by the maths commands is dependent on the type of the named variable.

<Add>

<Add Name=""> (o-v-n) This command is used to add the specified value to the named variable.

- **Name=""** (m-f-n) – Name of the <Num> or <Double> variable to which the value is to be added.

<Subtract>

<Subtract Name=""> (o-v-n) This command is used to subtract the specified value from the named variable.

- **Name=""** (m-f-n) – Name of the <Num> or <Double> variable from which the value is to be subtracted.

<Multiply>

<Multiply Name=""> (o-v-n) This command is used to multiply the named variable by the specified value.

- **Name=""** (m-f-n) – Name of the <Num> or <Double> variable which is to be multiplied by the value.

<Divide>

<Divide Name="" Rem=""> (o-v-n) This command is used to divide the named variable by the specified value.

- **Name=""** (m-f-n) – Name of the <Num> or <Double> variable which is to be divided by the value.
- **Rem=""** (o-f-n) – Name of the <Num> or <Double> variable which is to receive the value of the Remainder from the Division. **Note**. This is done on a integer arithmetic basis for both <Num>'s and <Double>'s. See the examples in Maths.xml.

<Eval>

<Eval Name=""> (o-v-n) This command is used to evaluate a simple mathematical expression and assign the result to the **Name**d variable.

- **Name=""** (m-f-n) – Name of the <Num> or <Double> variable which is to receive the result from evaluating the mathematical expression.

Expressions take the general form LeftHandSide.Operator.RightHandSide. The valid Operators are:

- .ADD. – Add the RHS to the LHS.
- .SUB. – Subtract the RHS from the LHS.
- .MUL. – Multiply the LHS by the RHS.
- .DIV. – Divide the LHS by the RHS.
- .REM. – Obtain the remainder of the division of the LHS by the RHS.

For example to find the value of variable I divided by 4 and assign it to J you would code <Eval Name="J">%I%.DIV.4</Eval>.

7.3.9 Transactional Control Commands.

When used in a <Test> the commit and backout commands and their parameter elements have the same meaning as when they are defined in the <Defaults> element. Refer to section "7.2.9 System Command".

8. Script Variables.

The scripting language provides for substitutable variables and command parameter element defaults. Substitutable variables have the form %VarName%. Any extra parameters after the script name on the programs' command invocation line are placed into string variables with the names %1%, %2% etc, these are the only variables, along with %ScriptName%, available to the sub-elements of the <Control> element. Named variables can be defined in both the <Defaults> and <Test> elements. A variable declared in the <Defaults> element will be copied to any <Test> using it, i.e. they are **NOT** global variables just an initial value. The following examples illustrate the use of substitutable variables. Also refer to the log output from the example scripts in the install directory "../MsgTest/Logs" to see the results of variable substitution.

8.1 Variable Substitution.

Example1 – Connection parameter variable substitution in the <Control> element. As scripts may be run in multiple environments rather than having to provide a different script for each environment the queue manager name and host name could be supplied as parameters after the script name on the command line.

So if MsgTest is invoked as follows:

```
MsgTest MyTest.xml MyQM01 MyTestHost
```

Where script MyTest.xml contains:

```
<MsgTest>
    <Control>
        <QMgr>%1%</QMgr>
        <Host>%2%</Host>
        …
    </Control>
    …
</MsgTest>
```

Then upon execution the script would be resolved as follows:

```
<MsgTest>
    <Control>
        <QMgr>MyQM01</QMgr>
        <Host>MyTestHost</Host>
        …
    </Control>
    …
</MsgTest>
```

Example2 – Command parameter element defaults and recursive substitutable variable resolution. In the example shown below the root directory name and file name prefix are supplied on the command line during program invocation. The root directory name from program invocation is then used in the string variables %DirIn% and %DirOut% which are declared as <Str> elements in the <Defaults> element to identify the default source and target directories for the <GetFile>, and <PutFile> command elements respectively. The file name prefix is used in the <File> parameter elements of the <GetFile> and <PutFile> commands in the test <Test Name="CopyFiles"> to help generate the name of the file to be copied.

So if MsgTest is invoked as follows:

```
MsgTest MyCopyFile.xml c:\MsgTest TestData
```

Where script MyCopyFile.xml contains:

```
<MsgTest>
    …
    <Defaults>
        <Str Name="DirIn">%1%\DataIn</Str>
        <Str Name="DirOut">%1%\DataOut</Str>
        <GetFile>
            <Dir>%DirIn%</Dir>
        </GetFile>
        <PutFile>
            <Dir>%DirOut%</Dir>
        </PutFile>
    </Defaults>
    <Test Name="CopyFiles">
        <GetFile>
            <File>%2%.in</File>
        </GetFile>
        <PutFile>
            <File>%2%.out</File>
        </PutFile>
    </Test>
</MsgTest>
```

Script resolution of the commands contained in <Test Name="CopyFile"> occurs as follows:

- Variable resolution "%2%.in" to "TestData.in" for the <File> element of <GetFile>
- As a <Dir> element has not been specified for the <GetFile> command the one specified in the <Defaults> element for <GetFile> is used:
    - Variable resolution "%DirIn%" to "%1%\DataIn"
    - Variable resolution "%1%\DataIn" to "c:\MsgTest\DataIn".
- Variable resolution "%2%.out" to "TestData.out" for the <File> element of <PutFile>
- As a <Dir> element has not been specified for the <PutFile> command the one specified in the <Defaults> element for <PutFile> is used:
    - Variable resolution "%DirOut%" to "%1%\DataOut".
    - Variable resolution "%1%\DataOut" to "c:\MsgTest\DataOut".

Thus the <Test> element of the script functionally resolves to:

```
    <Test Name="CopyFiles">
        <GetFile>
            <File>TestData.in</File>
            <Dir>c:\MsgTest\DataIn</Dir>
        </GetFile>
        <PutFile>
            <File>TestData.out</File>
            <Dir>c:\MsgTest\DataOut</Dir>
        </PutFile>
```

```
        </Test>
```

Example3 –Variables are permitted to have simple subscripts. The following example illustrates one use of subscripted variables by saving the %MsgId% automatic variable into subscripted variables %MessageId[01]%, %MessageId[02]%, and %MessageId[03]% which are created in the first <For> loop when the loop variable %PM% is resolved in the string variable command <Str Name="MessageId[%PM%]">%MsgId%</Str>. The variables thus created are used to retrieve the reply messages in the second <For> loop. The script has been annotated with the values that would be substituted in the first iteration of each loop.

```
<Defaults>
    <Str Name="Dir">c:\MsgTest\DataIn</Str>
    <GetMsg><Q>MSGTEST.OUT</Q></GetMsg>
    <PutMsg><Q>MSGTEST.OUT</Q></PutMsg>
</Defaults>
<Test Name="#1 RequestReply">
    <For Name="PM" From="1" To="3" Incr="1" Format="%02d">
        <GetFile>
            <File>FileIn%PM%.txt</File>
            <!-- FileIn01.txt -->
        </GetFile>
        <PutMsg>
            <MsgId/>
            <CorrelId>RequestReply%PM%</CorrelId>
            <!-- RequestReply01 -->
        </PutMsg>
        <Str Name="MessageId[%PM%]">%MsgId%</Str>
        <!-- MessageId[01] -->
    </For>
    <For Name="GM" From="1" To="3" Incr="1" Format="%02d">
        <GetMsg>
            <MsgId/>
            <CorrelId>%MessageId[%GM%]%</CorrelId>
            <!-- MessageId[01] -->
        </GetMsg>
    </For>
</Test>
<Test Name="#2 EchoRequestReply">
    <WaitOnTime>
        <Interval>1</Interval>
    </WaitOnTime>
    <!-- Process in reverse order so msgs stack up for Test #1 -->
    <For Name="L1" From="3" To="1" Incr="-1" Format="%02d">
        <GetMsg>
            <MsgId/>
            <CorrelId>RequestReply%L1%</CorrelId>
            <!-- RequestReply01 -->
        </GetMsg>
        <PutMsg>
            <MsgId/>
            <CorrelId>%MsgId%</CorrelId>
        </PutMsg>
    </For>
</Test>
```

8.2 Automatic Variables.

There is a set of automatically defined variables associated with each <Test>.
Note. The time related automatic string variables are set when a <Test> is assigned to the active list, i.e. the <Test> begins executing.
Note. The MQMD related automatic string variables are set after the execution of an <GetMsg> or <PutMsg> command.
Note Variable names are case sensitive.

Automatic string variables.

- %YYYY% Four digit year. Two digits century. Two digits year.
- %YY% Two digit year value with leading zero.
- %MMM% Month as defined by "Locale" usually three characters e.g. "Jan", "Feb" etc.
- %MM% Two digit month with leading zero.
- %DDD% Day as defined by "Locale" usually three characters e.g. "Mon", "Tue" etc.
- %DD% Two digit day with leading zero.
- %hh% Hour value from time of day with leading zero.
- %mm% Minute value from time of day with leading zero.
- %ss% Seconds value from time of day with leading zero.
- %PID% Process Id of the MsgTest process.
- %ScriptName% Name of the script file without any preceding directory information.
- %MsgId% MQMD.MsgId from last MQPUT or MQGET.
- %CorrelId% MQMD.CorrelId from last MQPUT or MQGET.
- %MsgSeqNumber% MQMD.MsgSeqNumber from last MQPUT or MQGET.
- %ReplyToQ% MQMD.ReplyToQ from last MQPUT or MQGET.
- %ReplyToQMgr% MQMD.ReplyToQMgr from last MQPUT or MQGET.
- %MsgType% MQMD.MsgType from last MQPUT or MQGET.
- %Priority% MQMD.Priority from last MQPUT or MQGET.
- %Persistence% MQMD.Persistence from last MQPUT or MQGET.

Automatic numeric variables.

After command execution.

- %LastCC% Value of the Condition Code set by the last command.
- %LastRC% Value of the Reason Code set by the last command.
- %LastDataLen% Length of data involved in the last command.
- %LastFile% Name of the last file from a <GetFile>/<PutFile> command.
- %LastDir% Name of the last direcotry from a <GetFile>/<PutFile> command.
- %LastQ% Name of the last Q from a <GetMsg>/<PutMsg> command.
- %LastQMgr% Name of the last QMgr from a <GetMsg>/<PutMsg> command.

Test level command statistics.

- %TestTotalGetFile% Total number of <GetFile> commands executed.
- %TestSuccessfulGetFile% Number of successful <GetFile> commands.
- %TestFailedGetFile% Number of failed <GetFile> commands executed.
- %TestTotalPutFile% Total number of <PutFile> commands executed.
- %TestSuccessfulPutFile% Number of successful <PutFile> commands.
- %TestFailedPutFile% Number of failed <PutFile> commands executed.
- %TestTotalGetMsg% Total number of <GetMsg> commands executed.
- %TestSuccessfulGetMsg% Number of successful <GetMsg> commands.
- %TestFailedGetMsg% Number of failed <GetMsg> commands executed.
- %TestTotalPutMsg% Total number of <PutMsg> commands executed.

- %TestSuccessfulPutMsg% Number of successful <PutMsg> commands.
- %TestFailedPutMsg% Number of failed <PutMsg> commands executed.
- %TestTotalCompare% Total number of <Compare> commands executed.
- %TestSuccessfulCompare% Number of successful <Compare> commands.
- %TestFailedCompare% Number of failed <Compare> commands executed.
- %TestCurrentMsgRate% Current value of <MessageRate…> in <PutMsg>.

Script level command statistics.

- %ScriptTotalGetFile% Total number of <GetFile> commands executed.
- %ScriptSuccessfulGetFile% Number of successful <GetFile> commands.
- %ScriptFailedGetFile% Number of failed <GetFile> commands executed.
- %ScriptTotalPutFile% Total number of <PutFile> commands executed.
- %ScriptSuccessfulPutFile% Number of successful <PutFile> commands.
- %ScriptFailedPutFile% Number of failed <PutFile> commands executed.
- %ScriptTotalGetMsg% Total number of <GetMsg> commands executed.
- %ScriptSuccessfulGetMsg% Number of successful <GetMsg> commands.
- %ScriptFailedGetMsg% Number of failed <GetMsg> commands executed.
- %ScriptTotalPutMsg% Total number of <PutMsg> commands executed.
- %ScriptSuccessfulPutMsg% Number of successful <PutMsg> commands.
- %ScriptFailedPutMsg% Number of failed <PutMsg> commands executed.
- %ScriptTotalCompare% Total number of <Compare> commands executed.
- %ScriptSuccessfulCompare% Number of successful <Compare> commands.
- %ScriptFailedCompare% Number of failed <Compare> commands executed.

Automatic control block variables.

- <MQMD Name=”TestMQMD> Default MQMD. Refer to the <GetMsg> and <PutMsg> commands in section “7.2 <Defaults> Elements” for more details.
- <RFH2 Name=”TestRFH2> Default RFH2. Refer to the <GetMsg> and <PutMsg> commands in the “7.2 <Defaults> Elements” for more details.

Updating a time related automatic string variable.

To get any of the time related automatic variables updated to its current value specify the relevant <Str> command as an empty tag. To update the hours, minutes and seconds variables during the execution of a script you would do the following:

<Str Name=”hh”/>
<Str Name=”mm”/>
<Str Name=”ss”/>

Updating a numeric or string variable.

To change the value of a numeric of string variable just re-declare it.

```
<Num Name="Num01" Format="%03d">33</Num>     ← Substitutes as 033
<Num Name="Num01">44</Num>                   ← Now substitutes as 044
<Num Name="Num01" Format="%05d">55</Num>     ← Now substitutes as 00055
```

Updating a compound variable.

To change a given element in the <MQMD> and <RFH2> compound variables just re-declare the compound variable with only the elements you want changed. See the examples below.

```
<MQMD Name=”MD01”>
    <Priority>4</Priority>
    <MsgId>11111</MsgId>
</MQMD>

<MQMD Name=”MD01”>
    <MsgId>22222</MsgId>  ← Only changes the MsgId, Priority is still 4.
</MQMD>
```

Accessing an element of the MQMD compound variable.

In the following script fragment a subscripted <MQMD> variable is created by the <MQMD Name="MDPut[%PM%]"> command in each iteration of the first <For> loop. The MsgId contained in the <MQMD> variables thus created is then accessed in the second <For> loop by specifying %MDPut[%GM%].MsgId% as the <CorrelId> for the <GetMsg> command.

```
<For Name="PM" From="1" To="3" Incr="1" Format="%02d">
    <GetFile>
        <File>FileIn%PM%.txt</File>
    </GetFile>
    <MQMD Name="MDPut[%PM%]">
        <Priority>5</Priority>
        <MsgId/>
        <CorrelId>RequestReply%PM%</CorrelId>
    </MQMD>
    <PutMsg MQMD="MDPut[%PM%]"/>
</For>
<For Name="GM" From="1" To="3" Incr="1" Format="%02d">
    <GetMsg>
        <MsgId/>
        <CorrelId>%MDPut[%GM%].MsgId%</CorrelId>
    </GetMsg>
</For>
```

9. MessageRates and Intervals

MsgTest includes the ability to control the rate at which it delivers messages to a queue, the ability to measure the time taken between two points in a script and how long it has taken to process a given message. These three functions are discussed in the following sub-sections. Also refer to the Interval and MessageRate scripts supplied in the "TestScripts" directory for examples of how they can be used.

9.1 MessageRates

It is possible to control the rate at which messages are sent by the <PutMsg> command from within a <For>/<ForEver> loop by using the <MessageRate> option of the <PutMsg> command. Refer to Section 7.3.2 for details of the various parameters and control modes.

Whilst it is only possible to run with a <Concurrency> of 1 when the <MessageRate> option of <PutMsg> has been specified there is nothing to prevent the running of several instances of MsgTest. For example:

- A baseline rate of 2 messages a second over half a hour period could be established inside the first script by coding a <ForEver> loop with a <MessageRate From="2" To="2" Incr="1" Interval="1800" Mode="INCREMENTAL"> specified
- Random bursts of between 2 and 5 messages a second could be added inside the second script by coding a <For Name="LP" From="1" To="6300" Incr="1"> loop with a <MessageRate From="2" To="5" Mode="BURST"> specified. The To="6300" in the <For> loopvalue being determined by an average message rate of 3.5 messages a second * 1800 seconds.

It is quite possible to achieve fairly sophisticated messaging patterns by using combinations of scripts.

9.2 Script Intervals

To measure the time taken to execute a section in a script you place <IntervalStart> and <IntervalEnd> commands around the portions of the script you want to analyse. Have a look at the samples in the TestScripts directory for some ideas.

9.3 Message Intervals

Another option that is available for the <PutMsg> command is to place a high precision timer value in either the MsgId or CorrelId of the MQMD by specifying either the <IntervalInMsgId> or <IntervalInCorrelId> options respectively.

This value can then be evaluated by a corresponding <GetMsg> also containing either the <IntervalInMsgId> or <IntervalInCorrelId> option.

This relies on the application which is being measured either just passing the value of the MsgId or CorrelId straight through or using the suggested request/reply convention whereby the MsgId of the inbound message is copied to the CorrelId of the outbound message.

Further to this the scripts performing the <PutMsg> and <GetMsg> commands need to both be running on the same machine as otherwise the timer values obtained are unlikely to reflect the real response time. The mechanism used being to get a high precision time value just before the MQPUT and place this in the requested MQMD field and then get another high precision time value just after the MQGET extract the MQPUT's value from the MQMD and do a subtraction to get the result.

To avoid the <PutMsg> and <GetMsg> commands interfering with each other it is strongly recommended that they are run in separate scripts.

Finally if you want to know the time it takes the request and reply scripts to execute their <PutMsg> and <GetMsg> commands then run the "Interval1.xml" script with appropriately sized messages on the same platform as the rest of the testing will be performed on and look at the interval statistics summary in the <TestReport> file.

9.4 Interval statistics reports

While a script that uses intervals is executing the related statistics are being accumulated in memory. When the script completes execution a summary of the statistics is written in the report file. Have a look at the "../TestScripts/Reports" directory for some sample output.

9.4 Interval statistics files

If the <TestInterval> parameter is specified in the <Control> section of a script then an Interval Information Record file will be created for each uniquely named interval in the script. Each record occupies one line in the file and consists of three fields:

- The first field is the time difference between the IntervalStart value of this observation and the IntervalStart value of the preceding observation. This can be useful when looking for anomalies in the running of the <PutMsg> script.
- The second field is the time difference between the IntervalEnd value of this observation and the IntervalEnd value of the preceding observation. This can be useful when looking for anomalies in the running of the <GetMsg> script.
- The third field is the observed response time and is the difference between the IntervalEnd value of this observation and the IntervalStart value of the same observation. The interval statistics summary in the report is generated from all the observations containing this field.

10. Files.

<MQMDFile>

The format of an <MQMDFile> file is shown below. The <MQMD> element contains the values elements for setting the supported fields in an MQMD.

```
<?xml version="1.0"?>
<MQMD>
      <Version></Version>
      <Report></Report>
      <MsgType></MsgType>
      <Expiry></Expiry>
      <Feedback></Feedback>
      <Encoding></Encoding>
      <CodedCharSetId></CodedCharSetId>
      <Format></Format>
      <Priority></Priority>
      <Persistence></Persistence>
      <MsgId></MsgId>
      <CorrelId></CorrelId>
      <BackoutCount></BackoutCount>
      <ReplyToQ></ReplyToQ>
      <ReplyToQMgr></ReplyToQMgr>
      <UserIdentifier></UserIdentifier>
      <AccountingToken></AccountingToken>
      <ApplIdentityData></ApplIdentityData>
      <PutApplType></PutApplType>
      <PutApplName></PutApplName>
      <PutDate></PutDate>
      <PutTime></PutTime>
      <ApplOriginData></ApplOriginData>
      <GroupId></GroupId>
      <MsgSeqNumber></MsgSeqNumber>
      <Offset></Offset>
      <MsgFlags></MsgFlags>
      <OriginalLength></OriginalLength>
</MQMD>
```

<RFH2File>

The format of an <RFH2File> file is shown below. The <MQRFH2> element contains the <Folder> and <Format> elements. As the actual folder data is in XML format it must be contained in a CDATA section.

```
<?xml version="1.0"?>
<MQRFH2>
    <Folder><![CDATA[<four>4</four>]]></Folder>
    <Folder><![CDATA[<three>3</three>]]></Folder>
    <Format>MQSTR</Format>
</MQRFH2>
```

11. Message Sets.

Message Sets are used to define the format and data for messages to be sent to MQ based applications by MsgTest scripts.

11.1 MessageSet structure.

There are two main sections within a message set. The <MessageTemplates> section in which the message definitions, segment definitions and element definitions that define the format of a message are placed, and the <Messages> section in which the actual element values for the messages are defined.

The individual messages <Msg>'s in the <Messages> section reference message defintions <MsgDef>'s in the <MessageTemplates> section by type.

The message defintions <MsgDef>'s are built up by using elements definitions <Element>'s and segments 's which refer to segment definitions <SegDef>'s in the <MessageTemplates> section by type.

The segment defintions <SegDef>'s are built up by using elements definitions <Element>'s and segments 's which refer to other segment definitions <SegDef>'s in the <MessageTemplates> section by type.

Each of the elements <Element>'s in the <MessageTemplates> section can be assigned a default value so that only the data that needs to be vary has to be defined in the actual messages <Msg>'s in the <Messages> section.

The XML fragment shown below is used to illustrate the main structural elements of a MsgSet. **Note.** The correspondence between the segment and element definitions in the <MessageTemplates> section and the XML tag names in the <Messages> section.

```
<MsgSet>
    <MessageTemplates>
        <MsgDef Type="MyMsgType" …>
            <Element Name="Hdr …>
            <Segment Type="SegOne" …>
        </MsgDef>
        <SegDef Type="SegOne" …>
            <Segment Type="SegTwo" …>
            <Element Name="EOS"…>
        </SegDef>
        <SegDef Type="SegTwo"…>
            <Element Name="S2A"…>
            <Element Name="S2B"…>
        </SegDef>
    </MessageTemplates>
    <Messages>
        <Msg Type="MyMsgType"…>
            <Hdr>Header</Hdr>
            <SegOne>
                <SegTwo><S2A>AAA</S2A><S2B>BBB</S2B></SegTwo>
                <EOS>EndOfSegment</EOS>
            </SegOne>
        </Msg>
    </Messages>
</MsgSet>
```

To permit reuse of definitions the <MessageTemplates> and <Messages> that make up a message set can be distributed across several files their aggregation into a message set is performed by specifying multiple <TestMsgSet> definitions with the same message set name in the <Control> section of the script that uses the message set. An example of this would be the combination of the standard header definitions, MQRFH2 & MQIIH & MQCIH, supplied in the {InstallDir}/StdMsgTemplates/StdMsgTemplates.xml file with your own message definitions in another file.

11.2 Message Template Elements.

<MsgDef>

<MsgDef Type="" OutputFormat="" Truncate="Yes | No"> – (mc-n-n) – This element is used to contain the individual segment and element definitions that describe the format of a message.
Its attributes are:

- **Type**="" – (m-f-n) – This attribute is used to specify the message type of the message definition. It provides the link between a message definition <MsgDef> in the <MessageTemplates> section and the actual messages <Msg>'s in the <Messages> section.
- **OutputFormat**="" – (o-f-d) – This attribute is used to specify the format of the output that will be generated by the <Select> command for messages of this type. Its default value is FIXED which is also currently the only option available.
- **Truncate="Yes | No"** – (o-f-d) – This attribute is used to control whether or not element data in this message can be truncated to fit into a field. If "Yes" is specified then element data can be truncated and normal processing will continue. If "No", which is the default, is specified then the script will fail as soon as it encounters a value that needs to be truncated to fit into a field.

<SegDef>

<SegDef Type="" OutputFormat=""> – (oc-n-n) – This element is used to contain the individual element and segment definitions that describe the format of a segment of a message. To permit the construction of hierarchical data format definitions segment definitions <SegDef>'s can reference other <Segments>. It is optional in that a message definition need only contain <Element> definitions.
Its attributes are:

- **Type**="" – (m-f-n) – This attribute is used to specify the type of the message segment. It provides the link between a <SegDef> segment definition and the actual that uses that definition.
- **OutputFormat**="" – (o-f-d) – This attribute is used to specify the format of the output that will be generated by the <Select> command for messages of this type. Its default value is FIXED which is also currently the only option available. If specified at segment level it overrides the output format specified in the message definition.

**** – (oc-n-n) – This element is used define the output format and the number of occurrences of a segment within a message definition <MsgDef> or segment definition <SegDef>.
Its attributes are:

- **Type**="" – (m-f-n) – This attribute is used to specify the type of the message segment. It provides the link between the and the <SegDef> that defines its format.
- **OutputFormat**="" – (o-f-d) – This attribute is used to specify the format of the output that will be generated by the <Select> command for messages of this type. Its default value is FIXED which is also currently the only option available. If specified at segment level it overrides the output format specified in the segment and message definitions that contain it.
- **MinOccurs**="" – (o-f-d) – This attribute is used to specify the minimum number of occurrences of the segment within its containing segment or message definition. A value of 0 means that the segment is optional. Its default value is 1.
- **MaxOccurs**="" – (o-f-d) – This attribute is used to specify the maximum number of occurrences of the segment within its containing segment or message definition. If there is no upper limit on the number of segments then specify a value of "unbounded". Its default value is 1.

<Element>

<Element Name="" Type="Char | Hex | Int8 | Int16 | Int32 | Int64 | PD" Len="" Justify="Left | Right" PadChar="" EndAlign="" CmqValues="Yes | No" MinOccurs="" MaxOccurs=""> – (o-n-n) – This element is used define the output format and the number of occurrences of an element within a message definition <MsgDef> or segment definition <SegDef>.
Its attributes are:

- **Name**="" – (m-f-n) – This attribute is used to specify the name of the element. It links the <Element> in the <MessageTemplates> section and the XML element of the same name contained in a <Msg> in the <Messages> section .
- **Type**="**Char | Hex | Int8 | Int16 | Int32 | Int64 | PD**" – (m-f-n) – This attribute is used to specify the type of the message segment. The possible values are:
    - **Char** – The element will contain character data. If an element needs to be set to all blanks then a CDATA section will need to be used.
    - **Hex** – The element will contain hex character data. The data can be prefixed with "0x" for documentation purposes.
    - **Int8** – The element is an eight bit integer.
    - **Int16** – The element is a sixteen bit integer.
    - **Int32** – The element is a thirty two bit integer.
    - **Int64** – The element is a sixty four bit integer.
    - **PD** – The element contains a packed decimal number.
- **Len**="" – (o-f-d) – This attribute is used to specify the length of **Char, Hex** and **PD** elements. If a length value is not specified for an element then it will be derived from the length of the data XML tag in the <Msg>. The length of the various **Int**eger elements is implicit and so any Len attribute will be ignored.
- **Justify="Left | Right"** – For character, **Char**, type elements this attribute can be used to specify whether the field should be left or right justified.
- **PadChar=""** – Use this attribute to specify the pad character for character, **Char**, elements.

- **EndAlign=””** – This attribute is used to specify the end alignment for derived length character ,**Char**, elements. For example if an end alignment value of 4 is specified then extra bytes will be added to the field so that its length is evenly divisible by 4.
- **CmqValues=”Yes | No”** – Use this attribute to specify whether or not MsgTest is to try to resolve the data contained in this element as a CMQ type constant. For example if specified as “Yes” and the XML element in the message contains “MQFMT_STRING” the elements value will be resolved to “MQSTR “.
- **MinOccurs=**”” – (o-f-d) – This attribute is used to specify the minimum number of occurrences of the element within its containing segment or message definition. A value of 0 means that the element is optional. Its default value is 1.
- **MaxOccurs=**”” – (o-f-d) – This attribute is used to specify the maximum number of occurrences of the element within its containing element or message definition. If there is no upper limit on the number of elements then specify a value of “unbounded”. Its default value is 1.

The data value specified for an element in it XML tag in the <Messages> section can be:

- Simple character data that will be interpreted according to the element type for example the numeric characters specified in an integer element will be converted to their binary representation whereas the data in a character element will be justified, padded and aligned as requested by the attributes of its <Element> definition.
- One of the supported CMQ values e.g. MQFMT_STRING. Refer to the Appendix E or MsgTestCmqValues.c for the currently supported values.
- A percent sign enclosed script variable name.
- The special values %SegLen%, %SegLen-1%, %SegLen-2% and %SegLen-4% which will cause the length of the containing segment to be substituted as the value for that element. **Note**. The segment length special values can only be used in integer elements.

12. MsgTest Return Code.

When MsgTest has completed execution it sets the Return Code as follows

0 – If all <Test>’s and all the commands contained therein succeeded either naturally or through the use of <SuppressRC>.

Non-Zero – When something fails the reason code will be set to a non-zero value consisting of the sum of the following values:

- The number of FailedTests
- The number of IncompleteTests
- The number of failed GetFile commands
- The number of failed PutFile commands
- The number of failed GetMsg commands
- The number of failed PutMsg
- The number of failed Compare commands

Appendix A. MA0T Package Description.

The package comes as a Zip file containing the following directories:

MsgTest

This directory contains all the other directories, this document and the bug fixes text file

- MsgTest Fixes.txt
- MsgTest User Guide.doc
- MsgTest User Guide.pdf

MsgTest\bin

This directory contains the MsgText executable and the Expat parser dll.

- mt.bat ← Batch file to simplify script execution(Client)
- mts.bat ← Batch file to simplify script execution(Server)
- libexpat.dll. ← Dll for Expat version 2.0.1
- MsgTest.exe ← Test script execution engine(Client)
- MsgTestS.exe ← Test script execution engine(Server)
- nlinfo.exe ← Exectuion environments national language info
- pcre3.dll ← Dll for Pcre version 6.4.1

MsgTest\DataIn

Deprecated. Example data has now been moved to MsgTest\TestScripts\DataIn

MsgTest\DataOut

Deprecated. Example data has now been moved to MsgTest\TestScripts\DataOut

MsgTest\Examples

Deprecated. Example data has now been moved to MsgTest\TestScripts\Scripts.

- RequestReply.xml ← Appendix B Example
- RFH2Proc.xml ← Appendix B Example

MsgTest\Expat-2.0.1

Contains the windows self extracting exe for V2.0.1 of the expat parser.

- expat_win32bin_2_0_1.exe
- expat-2.0.1.tar.gz

MsgTest\Logs

Deprecated. Example logs have now been moved to MsgTest\TestScripts\Logs

MsgTest\Makefiles

Contains the make files for Unix systems. Only been able to test Linux so far

- Makefile-AIX.mak
- Makefile-Linux.mak
- Makefile-Solaris.mak

MsgTest\MSVC

Contains the Microsoft Visual Studio for C++ project files.

- MsgTest.dsw
- MsgTest.dsp
- MsgTestS.dsp
- nlinfo.dsp

MsgTest\ Pcre-6.4.1

Contains the Pcre regular expression libraries

MsgTest\Reports

Deprecated. Example reports have now been moved to MsgTest\TestScripts\Reports

MsgTest\Schema

Contains the XML schema for validating MsgTest scripts and MessageSets

- MsgTest.xsd
- MsgSet.xsd

MsgTest\Source

Contains the source code for MsgTest

- MsgTest.c
- MsgTest.h
- MsgTestCmqValues.c
- MsgTestConv.c
- MsgTestCtrlBlk.c
- MsgTestFile.c
- MsgTestInit.c
- MsgTestMessages.c
- MsgTestMsgSet.c
- MsgTestRun.c
- MsgTestScript.c
- MsgTestShut.c
- MsgTestStaticAV.h
- MsgTestStaticMD.h
- MsgTestTime.c
- MsgTestUtil.c
- nlinfo.c

MsgTest\StdMsgTemplates

Contains the standard message set definitions. The MQMD control block and MQRFH2, MQIIH and MQCIH headers.

- StdMsgTemplates.xml

MsgTest\TestScripts

Contains the annotated script files used to test the utility and the output from executing them. Not all scripts are valid and this is noted in those scripts that are not so. The scripts are designed to exercise the utility and parser and so some of them will seem pointless in a wider context, however I have included them all as the log files illustrate the functioning of some aspect of the utility. The nine sub-directories contain the following:

- DataIn ← Input data files
- DataOut ← Output data files
- IIR ← Interval information record files
- Logs ← Log files
- MsgSets ← Message set files
- Reports ← Report files
- Scripts ← The test script files
- UserLogs ← User log files
- Work ← Work area for script temporary files

Appendix B. Example scripts.

RequestReply

This script simulates a standard request/reply scenario where the message id of the request message is copied to the correlation id of the reply message. **Note**. The two different ways of setting the MQMD's MsgId and CorrelId as illustrated by the RequestReply <Test>'s below.

```
<?xml version="1.0"?>
<MsgTest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\MsgTest\Schema\MsgTest.xsd">
    <Control>
        <QMgr>MYQMGR</QMgr>
        <Channel>MY.SVRCONN</Channel>
        <Host>127.0.0.1</Host>
        <Port>1444</Port>
        <TestLog>
            <File>RequestReply.log</File>
            <Dir>C:\MsgTest\TestScripts\Logs</Dir>
        </TestLog>
        <TestReport>
            <File>RequestReply.rpt</File>
            <Dir>C:\MsgTest\TestScripts\Reports</Dir>
        </TestReport>
        <TestMaxWait>30</TestMaxWait>
        <NextLogMsgInterval>1</NextLogMsgInterval>
        <Concurrency>3</Concurrency>
        <ConcurrencyWait>20</ConcurrencyWait>
        <OnError>LOG</OnError>
    </Control>
    <Defaults>
        <Str Name="Dir">C:\MsgTest\TestScripts\Data</Str>
        <GetMsg>
            <Q>MSGTEST.OUT</Q>
        </GetMsg>
        <PutMsg>
            <Q>MSGTEST.OUT</Q>
        </PutMsg>
    </Defaults>
    <Test Name="#1 RequestReply">
        <GetFile>
            <File>FileIn1.txt</File>
            <Dir>%Dir%In</Dir>
        </GetFile>
        <MQMD Name="MD01">
            <MsgId/>
            <CorrelId>REQREP1</CorrelId>
        </MQMD>
        <PutMsg MQMD="MD01"/>
        <MQMD Name="MD01">
            <MsgId/>
            <CorrelId>%MsgId%</CorrelId>
        </MQMD>
        <GetMsg MQMD="MD01"/>
        <PutFile>
            <File>FileOut1.txt</File>
            <Dir>%Dir%Out</Dir>
        </PutFile>
    </Test>
```

```
    <Test Name="#2 RequestReply Alternate">
        <GetFile>
            <File>FileIn2.txt</File>
            <Dir>%Dir%In</Dir>
        </GetFile>
        <PutMsg MQMD="MD01">
            <MsgId/>
            <CorrelId>REQREP2</CorrelId>
        </PutMsg>
        <GetMsg MQMD="MD01">
            <MsgId/>
            <CorrelId>%MsgId%</CorrelId>
        </GetMsg>
        <PutFile>
            <File>FileOut2.txt</File>
            <Dir>%Dir%Out</Dir>
        </PutFile>
    </Test>
    <Test Name="#3 EchoRequestReply">
        <GetMsg>
            <MsgId/>
            <CorrelId>REQREP1</CorrelId>
        </GetMsg>
        <PutMsg>
            <MsgId/>
            <CorrelId>%MsgId%</CorrelId>
        </PutMsg>
        <GetMsg>
            <MsgId/>
            <CorrelId>REQREP2</CorrelId>
        </GetMsg>
        <PutMsg>
            <MsgId/>
            <CorrelId>%MsgId%</CorrelId>
        </PutMsg>
    </Test>
</MsgTest>
```

RFH2Processing

This example demonstrates the manipulation RFH2 headers. **Note** the use of program invocation variable %1% for the queue manager name.

```
<?xml version="1.0"?>
<MsgTest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\MsgTest\Schema\MsgTest.xsd">
    <Control>
        <QMgr>%1%</QMgr>
        <Channel>MY.SVRCONN</Channel>
        <Host>127.0.0.1</Host>
        <Port>1444</Port>
        <TestLog>
            <File>RFH2Proc.log</File>
            <Dir>c:\MsgTest\TestScripts\Logs</Dir>
        </TestLog>
        <TestReport>
            <File>RFH2Proc.rpt</File>
            <Dir>c:\MsgTest\TestScripts\Reports</Dir>
        </TestReport>
        <TestMaxWait>30</TestMaxWait>
        <NextLogMsgInterval>1</NextLogMsgInterval>
        <Concurrency>1</Concurrency>
        <ConcurrencyWait>20</ConcurrencyWait>
        <OnError>LOG</OnError>
    </Control>
    <Defaults>
        <Str Name="Dir">c:\MsgTest\TestScripts\DataIn</Str>
    </Defaults>
    <Test Name="#1 RFH2Processing">
        <GetFile>
            <File>FileIn1.txt</File>
            <RFH2File RFH2="RFH2Put">MsgFolders.rf2</RFH2File>
        </GetFile>
        <PutMsg RFH2="RFH2Put">
            <Q>MSGTEST.OUT</Q>
        </PutMsg>
        <GetMsg RFH2="RFH2Get"/>
        <!-- Fudge it by loading same RFH2 file for comparison -->
        <RFH2 Name="RFH2Comp">
            <RFH2File>MsgFolders.rf2</RFH2File>
        </RFH2>
        <Compare>
            <File>FileIn1.txt</File>
            <!-- Same data file also -->
            <RFH2Comp R1="RFH2Get" R2="RFH2Comp"/>
        </Compare>
    </Test>
</MsgTest>
```

File MsgFolders.rf2

```
<?xml version="1.0"?>
<MQRFH2>
    <Folder><![CDATA[<mcd><msd>XML</msd></mcd>]]></Folder>
    <Folder><![CDATA[<usr><Flag>TRUE</Flag></usr>]]></Folder>
</MQRFH2>
```

MultiMessage

Demonstrates a complex flow of messages. Unlike the previous two examples, which are standalone tests, this example involves two applications that do the following. Application 1 get messages from queue MSGTEST.APPL1.IN transforms them and writes them to queue MSGTEST.APPL2.IN. Application 2 waits for the arrival of a message on its control queue MSGTEST.APPL2.CTRL before reading in all the messages on queue MSGTEST.APPL2.IN and writing them to file MsgCombo.txt.

```
<?xml version="1.0"?>
<MsgTest>
<Control>
    <QMgr>MYQMGR</QMgr>
    <Channel>MY.SVRCONN</Channel>
    <Host>myhost</Host>
    <Port>1414</Port>
    <TestLog>
        <File>MultiMsg.log</File>
        <Dir>c:\MsgTest\Logs</Dir>
    </TestLog>
    <TestReport>
        <File>MultiMsg.rpt</File>
        <Dir>c:\MsgTest\Reports</Dir>
    </TestReport>
    <TestMaxWait>30</TestMaxWait>
    <NextLogMsgInterval>1</NextLogMsgInterval>
    <Concurrency>5</Concurrency>
    <ConcurrencyWait>20</ConcurrencyWait>
    <OnError>LOG</OnError>
</Control>
<Defaults>
    <Str Name="Dir">c:\MsgTest\DataIn</Str>
</Defaults>
<Test Name="#1 End to end Appl1/Appl2">
    <!-- Put twenty messages on queue MSGTEST.APPL1.IN for Appl1 -->
    <!-- Use files MsgIn01.txt, MsgIn02.txt, ... MsgIn20.txt -->
    <For Name=”FileNum” From=”1” To=”20” Incr=”1” Format=”%02d”>
        <GetFile><File>MsgIn%FileNum%.txt</File></GetFile>
        <PutMsg><Q>MSGTEST.APPL1.IN</Q></PutMsg>
    </For>
    <!-- Wait until Appl1 has finished processing the 20 messages -->
    <WaitOnMessageDepth>
        <Q>MSGTEST.APPL2.IN</Q>
        <Depth>20</Depth>
    </WaitOnMessageDepth>
    <!-- Send a control message to Appl2 -->
    <GetFile><File>MsgCtrl.txt</File></GetFile>
    <PutMsg><Q>MSGTEST.APPL2.CTRL</Q></PutMsg>
    <!-- Get the output file from Appl2 and compare it against -->
    <!-- the expected result. Allow 5 minutes for it to arrive -->
    <GetFile>
        <File>MsgCombo.txt</File>
        <MaxCmdWait>300</MaxCmdWait>
    </GetFile>
    <Compare><File>MsgComboCompare.txt</File></Compare>
</Test>
</MsgTest>
```

Appendix C. Sample Log and Report files.

Sample log and report files from running the first two examples in Appendix B are included in the package in directories "..\MsgTest\Logs" and "..\MsgTest\Reports" respectively.

Appendix D. Compling MsgTest.

Windows using Visual Studio for C++ V6.

The project workspace and project files MsgTest.dsw and MsgTest.dsp respectively have been included in directory "..\MsgTest\MSVC". You may need to modify the project so that it points to the Expat, Pcre and MQ library files in the directories where you installed them.

Other Platforms.

To compile and run MsgTest on other platforms you will need to do the following:

1. Compile the Expat parser for that platform. Refer to the instructions and source code contained in the expat-2.0.1.tar.gz file and compile the Expat library.
2. Compile the Pcre regular expression library for that platform. Refer to the instructions and source code contained in the pcre-6.4.tar.gz file and compile the Pcre library.
3. Create a make file to compile the source files
    - MsgTest.c
    - MsgTestCmqValues.c
    - MsgTestConv.c
    - MsgTestCtrlBlk.c
    - MsgTestFile.c
    - MsgTestInit.c
    - MsgTestMessages.c
    - MsgTestMsgSet.c
    - MsgTestRun.c
    - MsgTestScript.c
    - MsgTestShut.c
    - MsgTestTime.c
    - MsgTestUtil.c
4. In the make file link the compiled source files with the Expat library, the Pcre library and the Websphere MQ client library mqic32
5. Sample Makefiles are available in the ..\MsgTest\Makefiles directory for the AIX, Linux and Solaris platforms. They have NOT been updated to include the Pcre library as I do not have access to any Unix platforms at the moment.

Unix.

The following is an outline of the kind of steps needed to get MA0T compiled and running in a Unix environment in particular Linux.

1. Unzip ma0t.zip into a set of folders on Windows
2. On Linux:
    a. Create an install directory: mkdir "directory name"
    b. Set an environment variable to same: export MA0TDIR="directory path"
3. Copy expat-2.0.0.tar.gz from Windows folders to install directory ${MA0TDIR}
4. On Linux, extract expat files:
    a. gunzip expat-2.0.0.tar.gz
    b. tar -xvf expat-2.0.0.tar

    At the end of 4(b) subdirectory ${MA0TDIR}/expat-2.0.0 has been created
5. Go into the expat-2.0.0 subdirectory and:
    a. run the configuration script: ./configure --prefix=${MA0TDIR}/expat
    b. compile expat: make (compiles it into distribution libraries)
    c. install expat: make install

    At the end of 5c the ${MA0TDIR}/expat directory should have been created and the expat shared objects moved into it.
    You need to make a choice as to whether the ${MA0TDIR}/expat directory will be the same on all machines, or will be different.
    If the same, you can compile MsgTest with the LD_RUN_PATH variable set to ${MA0TDIR}/expat see the Linux makefile.
    If different, you need to ensure that the LD_LIBRARY_PATH variable contains the local value of ${MA0TDIR}/expat before running MsgTest.
6. From Windows, copy to ${MA0TDIR} on Linux at least the following subdirectories: DataIn, DataOut, Source, Schema, StdMsgTemplates, Examples. Also create empty subdirectories Logs, Reports, Work. Also consider doing the same for TestScripts and its sub-directories.
7. Inside each of the copied subdirectories, run the command: dos2unix *
8. Inside the Examples directory you can run the command:
    sed -e "s|[Cc]:[\\]MsgTest[\\]|${MA0TDIR}/|g;ta;b" -e :a -e "s|[\\]|/|g" -i *

    to change embedded Windows directories to Unix
9. Copy Makefile_Linux.mak to Source directory and rename to just Makefile
10. Compile: make
    a. All the object modules, executables will be created in the Source directory.
    b. Assuming everything compiled OK, get rid of object modules with the command make cleanobj
    c. Move the MsgTest and MsgTestS executables to a directory of your choice, presumably ${MA0TDIR}/bin

Appendix E. MQMD and Header constants.

MsgTest now supports the use of the following constants in the <MQMD> field elements and the <GmoOptions>, <GmoMatchOptions> and <PmoOptions> elements of the <GetMsg> and <PutMsg> commands. As of V1.2.2 and the implementation of MessageSets the MQIIH, MQCIH and MQRFH2 headers are also now supported. Refer to the IBM Websphere MQ Application Programming Reference for details on the use of the actual values.
Note. Not all values from the cmqc.h header are supported.
Note. The "Interpreted to mean…" values are only done for constants that have a value of zero. Where actual bit settings are available for two opposites then no interpretation is attempted for example specifying MQPMO_SYNCPOINT & MQPMO_NO_SYNCPOINT will result in the <PutMsg> command failing with RC=2046 i.e. MQRC_OPTIONS_ERROR.

<MQMD>

<StrucId>

- MQMD_STRUC_ID

<Version>

- MQMD_VERSION_1
- MQMD_VERSION_2
- MQMD_CURRENT_VERSION

<Report>

- MQRO_EXCEPTION
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_COA
- MQRO_COA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA
- MQRO_COD
- MQRO_COD_WITH_DATA
- MQRO_COD_WITH_FULL_DATA
- MQRO_PAN
- MQRO_NAN
- MQRO_NEW_MSG_ID
  Interpreted to mean turn off the MQRO_PASS_MSG_ID bit
- MQRO_PASS_MSG_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID
  Interpreted to mean turn off the MQRO_PASS_MSG_ID & MQRO_PASS_CORREL_ID bits
- MQRO_PASS_CORREL_ID
- MQRO_DEAD_LETTER_Q
  Interpreted to mean turn off the MQRO_DISCARD_MSG bit
- MQRO_DISCARD_MSG
- MQRO_NONE
  Interpreted to mean reset all report options

<MsgType>

- MQMT_REQUEST
- MQMT_REPLY
- MQMT_REPORT
- MQMT_DATAGRAM

<Expiry>

- MQEI_UNLIMITED

<FeedBack>

- MQFB_NONE

<Encoding>

- MQENC_NATIVE

<CodedCharSetId>

- MQCCSI_UNDEFINED
- MQCCSI_DEFAULT
- MQCCSI_Q_MGR
- MQCCSI_INHERIT
- MQCCSI_EMBEDDED

<Format>

- MQFMT_NONE
- MQFMT_ADMIN
- MQFMT_CHANNEL_COMPLETED
- MQFMT_CICS
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_DIST_HEADER
- MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_MD_EXTENSION
- MQFMT_PCF
- MQFMT_REF_MSG_HEADER
- MQFMT_RF_HEADER
- MQFMT_RF_HEADER_2
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_WORK_INFO_HEADER
- MQFMT_XMIT_Q_HEADER

<Priority>

- MQPRI_PRIORITY_AS_Q_DEF

<Persistence>

- MQPER_NOT_PERSISTENT
- MQPER_PERSISTENT
- MQPER_PERSISTENCE_AS_Q_DEF

<MsgId>

- MQMI_NONE

<CorrelId>

- MQCI_NONE

<GroupId>

- MQGI_NONE

<AccountingToken>

- MQACT_NONE

<PutApplType>

- MQACT_NONE
- MQAT_UNKNOWN
- MQAT_NO_CONTEXT
- MQAT_CICS
- MQAT_MVS
- MQAT_OS390
- MQAT_ZOS

- MQAT_IMS
- MQAT_OS2
- MQAT_DOS
- MQAT_AIX
- MQAT_UNIX
- MQAT_QMGR
- MQAT_OS400
- MQAT_WINDOWS
- MQAT_CICS_VSE
- MQAT_WINDOWS_NT
- MQAT_VMS
- MQAT_GUARDIAN
- MQAT_NSK
- MQAT_VOS
- MQAT_IMS_BRIDGE
- MQAT_XCF
- MQAT_CICS_BRIDGE
- MQAT_NOTES_AGENT
- MQAT_USER
- MQAT_BROKER
- MQAT_JAVA
- MQAT_DQM
- MQAT_CHANNEL_INITIATOR
- MQAT_DEFAULT

<MsgFlags>

- MQMF_SEGMENTATION_INHIBITED
  Interpreted to mean turn off the MQMF_SEGMENTATION_ALLOWED bit
- MQMF_SEGMENTATION_ALLOWED
- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_NONE
  Interpreted to mean reset all message flags

<OriginalLength>

- MQOL_UNDEFINED

<GetMsg>

<GmoOptions>

- MQGMO_WAIT
- MQGMO_NO_WAIT
  Interpreted to mean turn off the MQGMO_WAIT bit
- MQGMO_SET_SIGNAL
- MQGMO_FAIL_IF_QUIESCING
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_NO_SYNCPOINT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_LOCK
- MQGMO_UNLOCK
- MQGMO_ACCEPT_TRUNCATED_MSG
- MQGMO_CONVERT
- MQGMO_LOGICAL_ORDER
- MQGMO_COMPLETE_MSG
- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_NONE
  Interpreted to mean reset all get message options

<GmoMatchOptions>

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID
- MQMO_MATCH_MSG_SEQ_NUMBER
- MQMO_MATCH_OFFSET
- MQMO_MATCH_MSG_TOKEN
- MQMO_NONE

<PutMsg>

<PmoOptions>

- MQPMO_SYNCPOINT
- MQPMO_NO_SYNCPOINT
- MQPMO_NEW_MSG_ID
- MQPMO_NEW_CORREL_ID
- MQPMO_LOGICAL_ORDER
- MQPMO_NO_CONTEXT
- MQPMO_DEFAULT_CONTEXT
- MQPMO_PASS_IDENTITY_CONTEXT
- MQPMO_PASS_ALL_CONTEXT
- MQPMO_SET_IDENTITY_CONTEXT
- MQPMO_SET_ALL_CONTEXT
- MQPMO_ALTERNATE_USER_AUTHORITY
- MQPMO_FAIL_IF_QUIESCING
- MQPMO_NONE
  Interpreted to mean reset all get message options

<MQIIH>

<StrucId>

- MQIIH_STRUC_ID

<Version>

- MQIIH_VERSION_1
- MQIIH_CURRENT_VERSION

<StrucLength>

- MQIIH_LENGTH_1

<Flags>

- MQIIH_NONE
- MQIIH_PASS_EXPIRATION
- MQIIH_UNLIMITED_EXPIRATION
- MQIIH_REPLY_FORMAT_NONE

<Authenticator>

- MQIAUT_NONE

<TranInstanceId>

- MQITII_NONE

<TranState>

- MQITS_IN_CONVERSATION
- MQITS_NOT_IN_CONVERSATION
- MQITS_ARCHITECTED

<CommitMode>

- MQICM_COMMIT_THEN_SEND
- MQICM_SEND_THEN_COMMIT

<SecurityScope>

- MQISS_CHECK
- MQISS_FULL

<MQCIH>

<StrucId>

- MQCIH_STRUC_ID

<Version>

- MQCIH_VERSION_1
- MQCIH_VERSION_2
- MQCIH_CURRENT_VERSION

<StrucLength>

- MQCIH_LENGTH_1
- MQCIH_LENGTH_2

<Flags>

- MQCIH_NONE
- MQCIH_PASS_EXPIRATION
- MQCIH_UNLIMITED_EXPIRATION
- MQCIH_REPLY_WITHOUT_NULLS
- MQCIH_REPLY_WITH_NULLS
- MQCIH_SYNC_ON_RETURN
- MQCIH_NO_SYNC_ON_RETURN

<ReturnCode>

- MQCRC_OK
- MQCRC_CICS_EXEC_ERROR
- MQCRC_MQ_API_ERROR
- MQCRC_BRIDGE_ERROR
- MQCRC_BRIDGE_ABEND
- MQCRC_APPLICATION_ABEND
- MQCRC_SECURITY_ERROR
- MQCRC_PROGRAM_NOT_AVAILABLE
- MQCRC_BRIDGE_TIMEOUT
- MQCRC_TRANSID_NOT_AVAILABLE

<CompCode>

- MQCC_OK

<Reason>

- MQRC_NONE

<UOWControl>

- MQCUOWC_ONLY
- MQCUOWC_CONTINUE
- MQCUOWC_FIRST
- MQCUOWC_MIDDLE
- MQCUOWC_LAST
- MQCUOWC_COMMIT
- MQCUOWC_BACKOUT

<GetWaitInterval>

- MQCGWI_DEFAULT

<LinkType>

- MQCLT_PROGRAM
- MQCLT_TRANSACTION

<OutputDataLength>

- MQCODL_AS_INPUT

<ADSDescriptor>

- MQCADSD_NONE
- MQCADSD_SEND
- MQCADSD_RECV
- MQCADSD_MSGFORMAT

<ConversationalTask>

- MQCCT_YES
- MQCCT_NO

<TaskEndStatus>

- MQCTES_NOSYNC
- MQCTES_COMMIT
- MQCTES_BACKOUT
- MQCTES_ENDTASK

<Facility>

- MQCFAC_NONE

<Function>

- MQCFUNC_MQCONN
- MQCFUNC_MQGET
- MQCFUNC_MQINQ
- MQCFUNC_MQOPEN
- MQCFUNC_MQPUT
- MQCFUNC_MQPUT1
- MQCFUNC_NONE

<StartCode>

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT
- MQCSC_NONE

<MQRFH2>

<StrucId>

- MQRFH_STRUC_ID

<Version>

- MQRFH_VERSION_1
- MQRFH_VERSION_2

<Flags>

- MQRFH_NONE

Appendix F. Registration and Feedback.

Please register your use of this utility by sending me an email at the address below. Doing so will let me know how widely it is being used and if requested permit me to keep you informed of any updates and fixes.

Also if you have any comments, problems or enhancement requests then please send them.

Finally if you are brave enough to modify the source code to add a new feature or fix a bug then please send the changes to me and I'll try to incorporate them into the next release if in my judgement they fit into the overall concept. If you want to modify the source code but don't understand what a particular piece of code is meant to do then I'll try to answer any queries.

Email: ta882324@bigpond.net.au