MS03 - WebSphere MQ Save Queue Manager Object Definitions using PCFs

Version 6 Release 1 Modification 3
Version Number 6.1.3

Last Modified:	October 2008
Maintained by:	Jeff Lowrey (jlowrey@us.ibm.com)
Maintainer Emeritus:	Wayne Schutz
Original Author:	Chris Petty

HISTORY/CHANGES

Old history moved to "history" file.

08 June 2005

V6:

a. Updates for WebSphere MQ V6.0.0

b. Removed support for MQ/WIN (aka Mayseed)

c. -o option updates.

d. Error messages contain source file name

e. CURPDEPT reported for local queues if -F not specified

f. Allow SSL client connections via -C and -k flags

g. Added MCATYPE to CLUSRVCR channel types.

h. -F now also suppress ALTDATE & ALTTIME. This should allow library systems like CVS to work ok

i. blanks between -m -r -v -a -x -k and -C flags and specified values are option. So both of these work now: saveqmgr -mTEST
saveqmgr -m TEST

j. -s now suppresses "alter qmgr" command generation

03 Oct 2005

V601:

a. Fix for OpenVMS return code processing

b. Allow putting entire DEFINE on one (or two) lines via the -1 and -2 flags. This makes the output more "machine readable".

c. Added HP executable

d. Fixed problem with CONVERT for zLinux qmgrs

e. Ability to specify client conn exits and data

02 Mar 2006

v602:

a. Added ability to run remotely to a zOS queue manager

b. Added ability to run remotely to a iSeries queue manager

c. fixed bug where lines were running > 70 chars

28 August 2006

V603:

a. -z (-Z) option to generate setmqaut commands

b. fixed small bug where rc from MQGET was being hidden by check for MQADMIN message

c. added options to -R (create reset commands) to allow specifications of the TYPES of channels to reset.

d. fixed problem with 'oneLine (-1)' and chained exits.
Also trim extra space off back of chained exits.

e. iSeries didn't like qmgr names starting with numeric.

f. added additional options on the QSAVEQMGR.CMD panel.

g. If a bad CLWLUSEQ value is returned by the cmd server, we write an error message and set the value to "QMGR"

29 July 2008
V610:

a. Updates for WebSphere MQ V7.0.0
b. Jeff Lowrey now maintainer,
Wayne Schutz is maintainer emeritus
c. Pre-supplied builds for unix are 64bit in most cases

July 2008

v610a: Bugfix release - private release to only a couple of customers

a. Fix for TARGQ not showing up for v6 qmgrs (bad #ifdef) from F.J. Brandelik

December 2008
v611:

a. Gracefully ignore not finding objects of a given type Mainly necessary because of zOS Qmgrs missing AUTHINFO objects.
b. Include long parameter support (--localQmgr)
c. Changed default file extension to MQSC and output name to SAVEQMGR.MQSC from SAVEQMGR.TST
d. Added options to -s/--noSystemObjects to allow user to decide which types of System Objects to skip
e. Fix for TARGQ not showing up for v6 qmgrs (from 610a)
f. Fix for memory crashes with PUBACCT.
g. Added --commandServerQueue to allow users to specify a different name for the command server queue
h. Fix for old(6.0.3a or earlier) bugs with string lengths in services.c
i. Fix for MQRC_UNIT_OF_WORK_NOT_STARTED on NSK - hackery added to mqutils.c to retry with change to MQGMO.

May 2009
v612:

a. Gracefully ignore MQRCCF_PUBSUB_INHIBITED Needed when QMGR has PSMODE(DISABLED)
b. add -w/--waitTime to allow user to set MQGMO.WaitInterval in case of slow command server
c. patch to OAM.c to warn if authorities are found for queues that weren't found during queue discovery
d. fixes to --noSysObjs to actually function as documented also actually implement --qm.
e. Revert a silent change of behavior on CLWLUSEQ when a bad value was populated during v5.3 migration.
Behavior now identical to saveqmgr v6.0.3a
f. Fix a bug with SSL support for saveqmgrc, that prevented any use of SSL at all.
g. Fix a bug with TPIPES under --oneLine and --twoLines
h. known issue: --noSystemObjects doesn't work, but the short version -s is fine.
i. known issue: --quiet may not function, but -q does
j. fix for bug with TOPICSTR, now null when it needs to be

NOTES

DEPENDENCIES = Websphere MQ for multiplatforms server
RESTRICTIONS = None
MODULE TYPE = INFOZIP compressed and TAR/COMPRESS compressed

CONTENTS

This SupportPac interrogates the attributes of all the objects defined to a queue manager (either local or remote) and saves them to a file or files. Queue manager objects (such as queue, process, namelist, topic, etc definitions) are saved to a file in a format suitable for use with runmqsc.

This tool DOES NOT provide a full backup of an entire queue manager.

If the queue manager is Version 6 or greater, you can also save the OAM definitions to a file. The output of this file is the same as "amqoamd -s". (This does not apply to a zOS queue manager).

It is therefore possible to use this SupportPac to save the definitions of objects known to a queue manager and subsequently recreate those objects on a new queue manager.

This SupportPac can run remotely against either a zOS or a iSeries queue manager and will generate correct MQSC for that platform. See the section below regarding saving zOS queue manager definitions.

The sample code has been developed and tested (using more than 700 test cases) using WebSphere MQ for Linux and WebSphere MQ for Windows/XP. However, any of the WebSphere MQ family which support PCFs can be used.

You must build the executable using "make", or "nmake" on Windows.
See the directions below.

— Special Instructions for the AS/400 ——

Please see the included file AS400READ.ME for specifics regarding the iSeries platform.
These files are located in ms03_iseries.zip.

OPERATION

MS03 is a command line program and can be invoked with a -? flag to display which parameters it will accept:
./saveqmgr.<platform> -?
where <platform> is a platform specific extension, ie .exe for windows, .solaris for solaris, etc.

Usage :

Flag are presented as "--longform/-shortform <*parameters*>: description"
where

--longform is the long name for the flag
-shortform is the short name for the flag
<*parameters*> is an indication of what parameters the flag takes, if any.
description describes the meaning of the flag and the parameters it takes.

- --help/-h -? : gives help (this)
- --localQMgr/-m *lqmgr* : is the name of the local qmgr to connect (MQCONN)
- --remoteQmgr/-r *rqmgr* : is the name of the remote qmgr (XMITQ name)
- --outputFile/-f *file* : allows the output file to be named, if -f is not specified, the output file defaults to SAVEQMGR.MQSC if -f is specified without a filename, it is named \"qmgrname\".MQSC.
- --outputFileNoHeader/-F *fil*] : behaves the same as -f except no header line is created in the output *file* (and CURDEPTH, ALTDATE and ALTTIME are not reported)
- --replaceOutput/-o : specifies to replace (overwrite) the output file
  if the -o flag is not specified, then the default is to append if the file exists
  (on iSeries, *REPLACE,*APPEND must be specified after the -o flag)
- --stdout/-c : directs output to STDOUT and overrides use of -f flag
- --noSystemObjects/-s [*list*]: suppress creation of objects that begin with \"SYSTEM*\"
  list is a comma separated selection from
  QMGR, QUEUE, NAMELIST, CHANNEL, PROCESS, AUTHINFO, LISTENER, SERVICE, TOPIC, SUB, NONE
  Specifying NONE is equivalent to specifying -s without any options and will override all other choices
  This is an inclusion list, System objects of the types listed **will** be reported in the output file.
- --qm : A shortcut option that is exactly equivalent to --noSystemObjects QMGR.
  The ALTER QMGR command will be produced, all other system objects will be skipped.
- --resetChannels/-R [*flags*] : create RESET CHANNEL commands
  flags can be any combination of the follow to create RESET for that channel type:
  s (SENDER) v (SERVER) r (RECEIVER) q (REQUESTER) c (CLUSSDR) d (CLUSRCVR)
  the default is to create RESET commands for all the above channel types
- --permDyn/-p : Create 'local queue' definitions from PERMDYN queues.
- --ignoreErrors/-i : Ignore 'pcf' error codes 'damaged object' 'unknown object'.
  Sets RetCode to WARNING if this error encountered.
- --quiet/-q : Quiet mode, do not issue warning about unknown PCF attributes
- --oneLine/-1 : generate entire DEFINE command on one line (which does NOT include ALTDATE etc)
- --twoLines/-2 : on a 2nd line, generate the DEFINE as a comment (which DOES include ALTDATE etc)

- --saveSecurity -z [*file*] : create ‘security definitions (like amqoamd -s)’ in file ‘*file*’(only >=V6)
  if *file* is not specified, it defaults to \”qmgrname\”.AUT in the current directory.
- --saveSecurityNoHeader/-Z [*file*]: same as -z only without the header line
- --commandServerQ/-cs [*queue*]: the name of the Command Server Input queue,
  defaults to SYSTEM.ADMIN.COMMAND.QUEUE
- --version/-v *version* : determines which version of MQSC to generate and can be
  ‘2’,’5’,’51’,’52’,’53’,’6’, or ‘7’

The default is to generate mqsc at the version of the connected queue manager
See “Notes on the ‘-v’ flag” in this document before using this flag.

The Client version of this program supports these additional flags:
If either the -a or -x flag is used, then MQCONNX will be used for the client connection

- --hostname/-a *host* : is the address of the host for a client connection (default is “localhost”)
- --channel/-x *channel* : is the SVRCONN name for a client connection (default is “SYSTEM.DEF.SVRCONN”)
- --sslCipher/-C *cipher spec* : indicates to use the given SSL cipher spec for this client connection
- --keyRepos/-k *key repository* : name of key respository directory if using SSL for client connection
- --channelExitX/-eX *channel exit*: Specifies the channel exit to use, with a qualifier to indicate the type of exit. X = ‘y’ for security, ‘s’ for send, ‘r’ for receive
- --channelDataX/-dX *channel exit data*: Specifies the channel exit data to pass, with a qualifier for the type of exit. X = ‘y’ for security, ‘s’ for send, ‘r’ for receive.

Notes on the '-v' flag:

Normally, the "-v (version)" flag does not need to be specified.
DO NOT USE THE -v FLAG UNLESS YOU MUST.
This is not a change for v6.1.0, this is a more emphatic statement of existing behavior.
saveqmgr will automatically generate the correct level of MQSC based on the CMDLEVEL setting of the queue manager that it is connected to. So, if it is connected to a v5.3 queue manager, it will generate v5.3 compatible mqsc, regardless of the actual level of saveqmgr itself. The "-v" flag should only be needed to generate back-level MQSC.
For example, if saveqmgr is connected to a V6 queue manager and you want to generate v5.3 MQSC.
The delivered executables will work with ALL versions of queue managers. However, if you compile the package yourself on a back-level of the queue manager, that version will not work with a more current version.
So, for example, if you recompile the package with a V6.0 qmgr, you cannot run that executable against a V7.0 qmgr.
This program does NOT save other qmgr information such as is contained in qm.ini or the Windows registry.

Note on MQ v7 Subscription objects:

Certain parameters of MQ v7 Subscription objects are not changeable through ALTER SUB or DEFINE SUB...REPLACE - including SELECTOR.
Saveqmgr v6.1.0 currently produces DEFINE SUB commands with REPLACE specified. If you attempt to run the resulting MQSC against a queue manager that has existing Subscriptions that you are trying to alter (including SYSTEM.* subscriptions), you will get MQSC errors.

The current maintainer needs your feedback on this.

Various options are available, as follows:

1. Maintain existing behavior, leaving the responsibility with the MQ Admin.
2. Produce the invalid parameters in the MQSC, but commented out like ALTDATE is.
3. Produce a pair of commands, DELETE SUB/DEFINE SUB, to remove existing objects.
4. Do not produce these parameters at all.

Other suggestions will be considered.

Notes for Windows Users:

WebSphere MQ V5.1 and above allows the administrator to configure certain queue manager attributes via the Microsoft Management Console. WebSphere MQ Version 6 and version 7 use the Eclipse based MQ Explorer.
Some of these configuration attributes are stored in the Windows Registry.
An example is "LogPrimaryFiles", which is a property of the queue manager. Because these are not stored in MQSC format, this SupportPac does NOT back up these types of attributes. Provision must be made for manually recording and recovering these attrbiutes.

Notes for OpenVMS users:

OpenVMS support is considered legacy as of this release. If you need support for the OpenVMS platform, please contact the current maintainer at the email address at the top of this document.
To use the command ‘SAVEQMGR’, a symbol has to be defined as follows:

```
$ SAVEQMGR :== “$MYDISK:[MYDIRECTORY]SAVEQMGR”
```

(don’t forget the dollarsign after the first double quote) or use the ‘MCR’ command:

```
$ MCR MYDISK:[MYDIRECTORY]SAVEQMGR
```

The path name ‘MYDISK:[MYDIRECTORY]’ can be omitted if the executable is located in SYS$SYSTEM:.

The VMS CLI (Command Line Interface) is case insensitive. All command line parameters are converted into lowercase. If parameters have to be in uppercase for WebSphere MQ, enclose them by double quotes, eg:

```
$ SAVEQMGR –M “QMGR_NAME”
```

Once the operation is complete a file will have been created in the current directory. (If the file existed before operation began it will have been appended to.) This file is SAVEQMGR.TST (however, see the note about the -f flag above).

Local Operation

Start SAVEQMGR and specify command line parameters as indicated earlier. You must have alreaady started the queue manager and the command server. SAVEQMGR will, in turn, interogate the attributes of queue manager objects. Replies will be sent to a dynamic reply queue and processed as they are received. Each object known to the queue manager will have a separate entry in the resulting save file.
The messages that are sent have an expiry time of five minutes. If the command server is not running the request will eventually time out and the messages will expire. This is the most common cause of errors.

Remote Operation

“Remote” queue manager definitions can be saved in one of two ways:

1. Use the client version (saveqmgrc) of this SupportPac to connect to the queue manager for which you wish to save the definitions. You may use either an MQSERVER variable, client channel tables or pass the connection information as parameters to ‘saveqmgrc’.
2. Connect to a local queue manager that has channel definitions to the queue manager for which you wish to save the definitions. In that case, you would specify a ‘-m’ flag naming the local queue manager and a ‘-r’ flag naming the remote queue manager.

This is much the same as local operation. In this case it is assumed that you already have the WebSphere MQ network in place and that there is a default route to the destination queue manager. The most common way to acheive this is to establish channels between the two queue managers the transmission queue for which is the name of the destination queue manager or use queue manager clustering.
Messages will expire as for local operation. In addition to ensuring that the command server is running at the destination queue manager it is also necessary to ensure that the required channels are active.

Running SAVEQMGR against a zOS queue manager

You can run 'saveqmgr' against a zOS queue manager by using either a client connection or normal MQ routing from a local windows or unix queue manager. In addition to returning MQSC defintions for the objects support by the zOS queue manager, additional information is returned:

1. The CSQ6SYSP, CSQ6LOGP and CSQ6ARVP macros are reconstructed.
2. SET LOG, SET SYSTEM and SET ARCHIVE commands are created (if they have been entered on the queue manager)
3. DEFINE BUFFPOOL, DEFINE LOG, DEFINE PSID, DEFINE CFSTRUCT (if in qsg) commands are created

These definitions are commented out in the MQSC and prefixed by an indicator which describes where these commands may run. An example is:

```
     *(12cr) SET SYSTEM +
*(12cr)        CTHREAD(300) +
*(12cr)        IDBACK(25) +
*(12cr)        IDFORE(105) +
```

In this case, this command can be issued from CSQINP1 (1), CSQINP2 (2), the zOS Console © and the command server and command queue, by means of CSQUTIL, CSQINPX, or applications ®. ZPARM macros are prefixed with '(zparm)'.

BUILDING Saveqmgr

Sample makefiles are included:
makefile.linux --for building an Linux version of the executable (add -m31 for 64 bit linux on zSeries, see makefile)makefile.aix -- for building an AIX version of the executable. makefile.solaris --for building a Solaris version of the executable. makefile.hp --for building a HP/UX version of the executable.
makefile.windows - for building a Windows version of the executable.
makefile.win64 - for building a Windows 64-bit version of the executable.
makefile.nsk - for building an Tandem NSK version of the executable.
makefile.compaqtru64unix - for building an Compaq Tru64 Unix version of the executable (Thanks to Ralph Beckers for support)
Descrip.mms - for building an OpenVMS version of the executable using MMS.
Make_savqmgr.com - for building an OpenVMS version of the executable without MMS.
makefile.os400 is no longer included in this package. See the AS400READ.ME file in ms03_iseries.zip for details.
Edit the makefile for your platform (ie makefile.windows) and ensure the variables point to the correct directories, etc.
On UNIX, issue: “make -f makefile.xxx”, where xxx is the extension for the platform you are compiling on.
To recompile on Windows or Win-64, issue: “nmake -f makefile.windows” or
“nmake -f makefile.win64”

HINTS:

You can use the -c flag together with sed to customize your output file. Suppose you want to delete all lines that contain the attribute NPMCLASS, try running:

```
./saveqmgr.linux -c  sed ‘s/^ *NPMCLASS.*/* skipping npmclass/’ > saveqmgr.tst
```

If you want to change all local queues to a maxdepth of 640000, try:

```
./saveqmgr.linux -c  sed ‘s/^ *MAXDEPTH.*/  MAXDEPTH(640000) +/’ > saveqmgr.tst
```

You get the idea...

You can automate the savings of all your qmgrs on a machine with this (for linux):

```
dspmq  grep ‘STATUS(Running)’  sed ‘s/QMNAME(\(.*\)) *STATUS.*/\1/’  xargs -i ./saveqmgr.linux -f
{}.mqsc -m {}
```

With V6 of MQ, you can create a SERVICE object to automatically run saveqmgr when the qmgr itself starts:

```
DEFINE SERVICE (‘Ms03’) DESCR(‘Run saveqmgr when qmgr starts’)
STARTCMD(‘c:\ms03\saveqmgr.exe’) +
STARTARG(‘-m +QMNAME+ -f c:\temp\+QMNAME+.mqsc -o’) STDOUT(‘c:\temp\ms03.stdout’)
+
STDERR(‘c:\temp\ms03.stderr’) CONTROL(STARTONLY) SERVTYPE(COMMAND) REPLACE
```

Here’s a little script to delete all the queues that begin with AMQ (even the PERMDYN ones):

```
export QM=WBRK_QM
./saveqmgr.linux -m$QM -1 -p -c   grep \(\’AMQ  gawk ‘{print “delete ql”,$3}’  runmqsc $QM
```

A script to authorize “phred” to get from all non-SYSTEM queues:

```
./saveqmgr.linux -m$QM -1 -s -c  egrep ‘^DEFINE QLOCALQREMOTEQMODELQALIAS’  awk
‘{print $3}’   \
sed -e “s/(‘\(.*\)’)/\1/”  xargs -i  -t setmqaut -m $QM -t q -p phred -n {} +get
```

If you use GNU diff with the ‘-F’ option on saveqmgr, you can get good detail about changed objects from yesterday:

```
diff -c -F DEFINE /tmp/wschutz.yesterday /tmp/wschutz.today
```