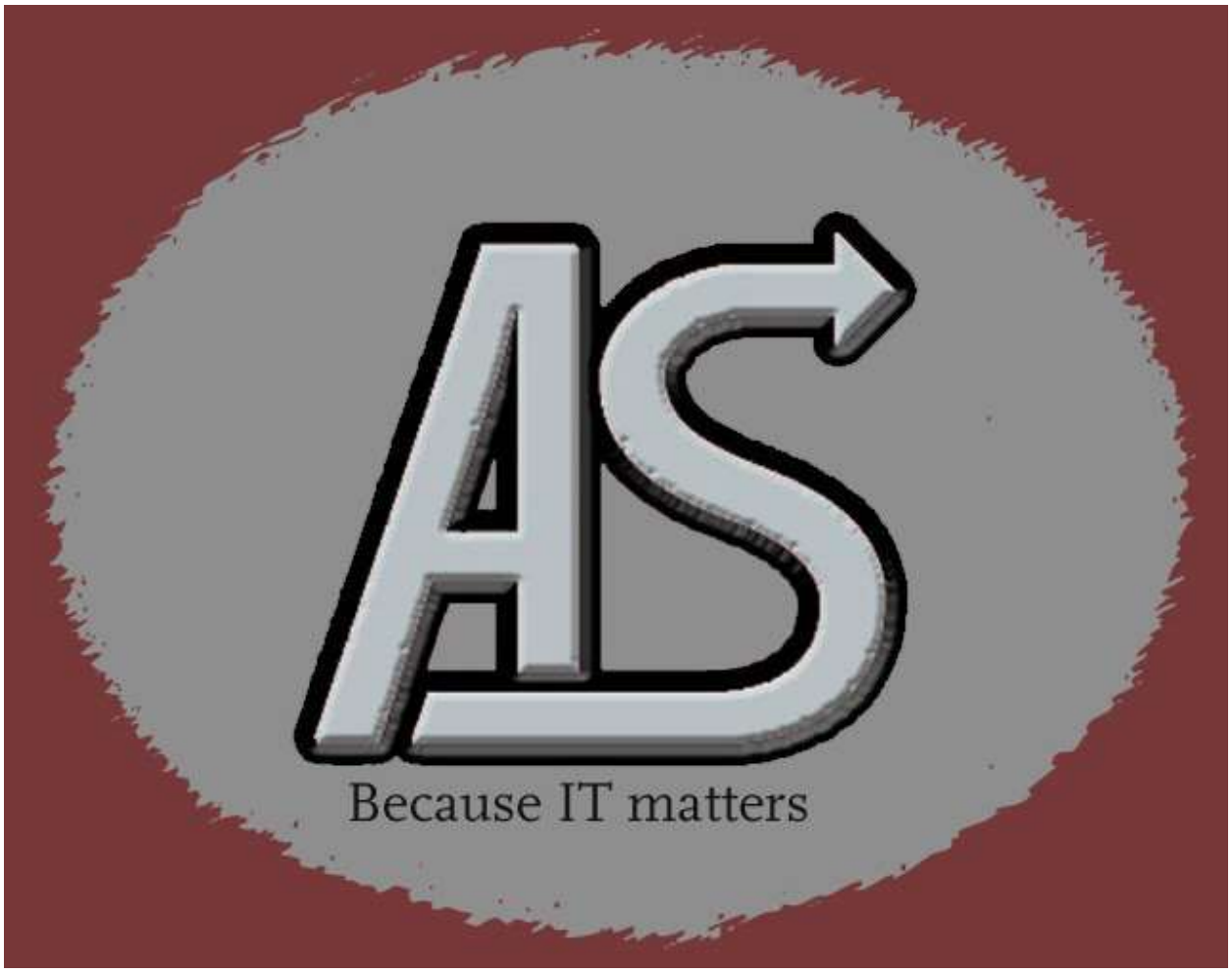
Abbydale Systems LLC.



QMGRWAIT

Program Information

E-mail: Support@abbydalesystems.com

Abbydale Systems LLC.
2925 Gulf Freeway South
Suite B #229
LEAGUE CITY
Texas USA
77573

ATTN: K.E.Ferguson
Legal@abbydalesystems.com

Disclaimer

This computer program and associated materials was developed by Kevin E. Ferguson of **Abbydale Systems LLC.**

This material has been used successfully by **Abbydale Systems LLC.** and to the best of our knowledge this material and any system(s) of which it is a part are operational as of the service level or date stated in the body of this material (if so stated). However, ***no warranty*** is given or implied as to the accuracy of this material or any related material or systems, and ***no responsibility*** is assumed for any effect or modification directly or indirectly caused by the use of this material.

It is the responsibility of any user of this material to evaluate its usefulness to the user's environment.

Abbydale Systems LLC. does not guarantee to keep this nor any related material current, nor does it guarantee to provide any corrections or extensions described by any users of this material or any corrections or extensions made in the future by **Abbydale Systems LLC.** itself.

Acknowledgements

This document refers to several software products that are produced by other companies. In most cases the names of these products are trademarks and/or copyright of those companies. It is not our intention to claim either the name of the trademark, nor the product itself, these remain solely the right of the owning companies.

CONTENTS

1 Overview

The program, **QMGRWAIT** is an assembler program that can be called whenever an MQ enabled program detects a MQ Reason code of 2161 (Queue Manager is Quiescing) is returned from a MQ API (Application Program Interface) call.

All well-behaving MQ enabled programs must handle return codes in a manner that applies to the application requirements. There are no particular 'industry standards' other than that they get handled.

All MQ program enabled programs should use the appropriate 'Fail-If-Quiescing' options. Failure to do so will cause the Queue Manager not to return a 2161 reason code.

QMGRWAIT addresses the situation where a shutdown is issued for a z/OS based queue manager, but it is being prevented from doing so by another program. Under these circumstances the second program will receive a MQ Reason code of 2161 on the MQ call. If the program detecting the 2161 reason code calls this program, then the ability is there to 'retry' the call, wait for a minute and retry the call, or cancel the job. By waiting it gives the system time to stop and restart the QMGR without having to cancel the job. Once the queue manager has been restarted then you can reply RETRY and, providing the program has been correctly coded, the program can then re-connect and continue processing.

When a 2161 is encountered any program calling **QMGRWAIT** has the chance to recover without having to be rerun. It also gives operations a good idea of those jobs that are using the queue manager that they are trying to stop.

Well behaving programs should disconnect from a queue manager if they get the 2161 return code.

The program is intended to be called by other programs although it can also be used 'stand-alone'.

Important note:

QMGRWAIT is <u>not</u> intended to be used by conversational (especially CICS) programs.

1.1 Passed Parameters

The program requires a single parameter passed to it. The parameter **must** be 4 characters in length. This parameter specifies the name of the queue manager to be used. i.e., MQXX

.

1.2 Called Programs

QMGRWAIT calls no other programs.

1.3 IBM macros used:

DEVTYPE	DOM	EXTRACT	IEZCIB
IEZCOM	QEDIT	STIMER	WAIT
WTO			

1.4 User macros used:

BEGIN	EOJ

1.5 Assembled User Values

There are no user values to be assembled before using **QMGRWAIT**, however, if you need to increase, or decrease the default wait interval from 60 seconds you will need to reassemble the source code.

2 Installation Procedure

2.1 From XMI File

The XMI (or XMIT) file is in IBM TSO TRANSMIT format and **must** be transferred to z/OS™ as a fixed blocked 80-byte BINARY file. The disk space requirement for the file is approximately 70 tracks of 3390 disk when blocked at 27920.

The FTP process (if performed in a 3270 emulator) must be performed in TSO READY mode or in option 6 of ISPF™.

The dataset name used as input for the TSO TRANSMIT was ABBYDALE.**QMGRWAIT**.PDS. Unless this is changed by the TSO RECEIVE (Please refer to the IBM RECEIVE command for details on the use of this TSO command) command it will be the name of the dataset created by the RECEIVE command.

Once this dataset has been RECEIVEd you will need to execute the UNPACK member. This will unpack all the TRANSMITTED datasets. You will have the option of changing the high-level qualifier for the datasets to be created.

The UPACK member should unpack three datasets. These are:

Hlq.QMGRWAIT.JCL	Contains the JCL procedure for assembling both **QMGRWAIT** and **MQERROR** and for compiling the sample Cobol code. It also contains the JCL for running **QMGRWAIT** and the JCL for executing the sample programs.
Hlq.QMGRWAIT.LOADLIB	Contains the pre-assembled/compiled load modules. These are 'run ready'.
Hlq.QMGRWAIT.SOURCE	Contains the source code for **QMGRWAIT**, **MQERROR** and some sample Cobol programs

:
Once the ABBYDALE.**QMGRWAIT**.PDS dataset has been received please refer to the $$INSTAL member to complete the installation.

A copy of this document is also available in the **QMGRWAIT**.PDS dataset. This should be transferred to a Windows system as a binary file and saved as a .PDF file.

It should be noted that **MQERROR** comes as a part of **QMGRWAIT**. You do not need to install it separately.

This PDF document and the PDF document for **MQERROR** are contained in the installation dataset. They should be transferred to a Windows system as binary files and saved as a .PDF files.

3 Using QMGRWAIT

QMGRWAIT can be executed as a standalone program via JCL or called from another program.

QMGRWAIT is 31 bit.

QMGRWAIT has no required DD cards, but if you want it to issue messages then the pertinent DD cards need to be present. **QMGRWAIT** will, by default, issue all user message to the JOBLOG of the job executing it, however, by coding the optional DD card CONSOLE these can be made to be issued to a console. You must investigate how your site routing and descriptor codes are set up and make any changes to **QMGRWAIT** to match your site standards.

QMGRWAIT use the routing and descriptor codes as shipped by IBM as it's default settings.

3.1 Available Parameters

QMGRWAIT requires a 4-character Queue Manager name to be passed to it. If no parameter is passed an error message is displayed and the program will terminate with a reason code of 16..

3.2 Application Program Linkage Considerations

As **QMGRWAIT** can be called statically or dynamically the linkage requirements will vary depending on how it is being called. If a static call is requested (via the NODYNAM parameter on the COBOL compile, or by using the CALL macro in assembler) then the binder (LINKAGE editor) execution should include module **QMGRWAIT** from the pertinent load library

e.g.

```
//LINKSTEP EXEC PGM=IEWL,
//         PARM='MAP,XREF,LIST'
//SYSLIN   DD  DISP=(OLD<DELETE),DSN=&&OBJECT
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DISP=SHR,DSN=your.target.load.library
//QMGRLIB  DD  DISP=SHR,DSN=your.QMGRWAIT.load.library
//SYSUT1   DD  DSN=&&SYSUT1,UNIT=SYSDA,SPACE=(1024,(50,20))
//SYSPRINT DD  SYSOUT=*
//SYSLIB   DD  DISP=SHR,DSN=your.sceelked.library
//SYSIN    DD  *
   INCLUDE QMGRLIB(QMGRWAIT)
   NAME yourprog(R)
//
```

You will need to change this JCL according to your own specific requirements and site standards. The SYSLIN DD refers to the object deck as created by the compile/assembly step that should precede the binder (IEWL) step.

3.3 Required JCL for QMGRWAIT

```
1  //jobname  JOB
2  //stepname EXEC PGM=QMGRWAIT,PARM='qmgr'
3  //STEPLIB  DD  DISP=SHR,DSN=your.load.library
4  //NOWTOR   DD  DUMMY
```

JCL card	Required	Use
1	Yes	JOBCARD (Tailor to your site standards)
2	Yes	EXEC card.
3	Yes	This is the library where **QMGRWAIT** is located. The STEPLIB concatenation can only be removed if the module resides in a LNKLST library.
4	No	The NOWTOR DD card, if present, will tell **QMGRWAIT** to suppress the WTOR and issue the message as a non-delete WTO. If this DD card is specified, then communication to the program is via the operator console.

3.4 Responding to QMGRWAIT

The method of responding to **QMGRWAIT** depends on if a NOWTOR DD card was specified or not.

An operator modify (F) command is available regardless of the presence of the NOWTOR DD card. If there is no NOWTOR DD card, then a Write to Operator with Reply (WTOR) will be issued and communication to the program can be via the reply to this message.

In all cases a STOP (P) command is allowed and will have the same effect as if CANCEL was the response to the WTOR.

Valid commands/responses are:

RETRY

F jobname,RETRY

This can be issued as either F *jobname*,RETRY or, if the NOWTOR DD card is omitted, by replying R *nn*,RETRY

Issuing the response RETRY will result in an immediate return to the calling program with a condition code of 0.

WAIT

F jobname,WAIT

This can be issued as either F *jobname*,WAIT or, if the NOWTOR DD card is omitted, by replying R *nn*,WAIT

Issuing the response WAIT will result in the program waiting for the default number of seconds (60) before re-issuing the QMGRW04A or QMGRW04I message.

WAIT *nnn*

F jobname,WAIT nnn

Where *nnn* is the number of seconds that the program is to wait before re-issuing the QMGRW04A or QMGRW04I message.

The value of *nnn* must be 999 or less.

This can be issued as either F *jobname*,WAIT *nnn* or, if the NOWTOR DD card is Omitted, by replying R *nn*,WAIT *nnn*

Issuing the response WAIT *nnn* will result in the program waiting for supplied number of seconds before re-issuing the QMGRW04A or QMGRW04I message.

CANCEL

F jobname,CANCEL

This can be issued as either F *jobname*,CANCEL or, if the NOWTOR DD card is omitted, by replying R *nn*,RETRY. The P *jobname* command has the same effect as issuing this command.

Issuing the response CANCEL (Or issuing the stop (P) command) will result in an immediate return to the calling program with a condition code of 8.

3.5 Return (Condition) Codes

It is recommended that all calling application programs check for condition codes 0, 8 and 16 being returned by **QMGRWAIT.**

Possible condition codes and their meanings and suggested responses are:

Condition Code 0

When a value of zero is returned by **QMGRWAIT** it is an indication that the operator has replied 'RETRY' to QMGRW04A.

It is up to the calling program to determine what is involved in a retry and this would depend on what MQ function determined that it needed to call **QMGRWAIT**.

If the failing action was a MQCONN then simply retrying the MQCONN should suffice.

If the failing action was a MQGET then it may be more involved. For example: you will probably need to reconnect, re-open the queue and retry the MQGET.

Condition Code 8

When a condition code of eight is returned, this is an indication that the operator has either replied 'CANCEL' to QMGRW04A or has issued the STOP jobname command.

It is up to the calling application program to determine the ramifications of this action and how to respond.

Condition Code 16

A condition code of sixteen should always be considered as a calling program logic problem.

It is an indication that either no parameter was passed, or the parameter passed was invalid.

The calling program should be changed to correctly call **QMGRWAIT** and the failing job should be re-run if required.

4 Calling QMGRWAIT

QMGRWAIT has been specifically designed to be called by other programs. This was done to help provide a standard look and feel interface for the operators in the event that the MQ queue manager address space is being stopped while jobs are still connected to it.

The program will allow the calling program to make choices based on the operator's responses by means of the return code.

4.1 Coding Examples

The following code snippets demonstrate how to call **QMGRWAIT** from another program. They should be considered just as an example of how to use **QMGRWAIT** within an application programming language and not as fully functional code. The examples are simply to demonstrate how to call **QMGRWAIT**. There is no error recovery coded within the examples

For examples of how **QMGRWAIT** can be used within an application please refer to the sample programs provided later in this document.

4.1.1 From Assembler

```
         MVC   TESTNAME,=C'MQAD'    Move in qmgr name
*
TRY_AGAIN DS    0H
*
*
         LINK EP=QMGRWAIT,PARAM=(PARMS_FOR_QMGRWAIT)
         LTR  R15,R15       Return code 0?
         BZ   TRY_AGAIN     Yes - Retry
*
*
**   Storage Definitions
*
PARMS_FOR_QMGRWAIT DS  0F
         DC   H'6'        Length of parms
TESTNAME DS   CL4
```

Obviously, the code is only for demonstration purposes. Hardcoding the queue manager name is not good coding practice.

In this case the program will put out the following message (if there is no NOWTOR DD).

xx **QMGRW04A :** *jobname* **has detected that** *qmgr* **is quiescing. Reply 'RETRY','WAIT' or 'CANCEL'**

If a NOWTOR is present, then the message will read:

QMGRW04I : *jobname* **has detected that** *qmgr* **is quiescing**

4.1.2 From COBOL

```
* ------------------------------------------------------------ *
 WORKING-STORAGE SECTION.
* ------------------------------------------------------------ *

 01  W00-RC                    PIC S9(04) BINARY  VALUE ZERO.
 01  PARM-WTOR.
     05  PARM-WTOR-LEN         PIC S9(03) BINARY VALUE +8.
     05  PARM-WTOR-MSG         PIC X(4).

* ------------------------------------------------------------ *
 PROCEDURE DIVISION.
* ------------------------------------------------------------ *
     MOVE 'MQAD' to PARM-WTOR-MSG.

 A-CALL-QMGRWAIT.

     CALL 'QMGRWAIT' USING PARM-WTOR.
     IF RETURN-CODE = 0 THEN
     DISPLAY 'Return code=' RETURN-CODE '. Retry connection'
        GO TO A-CALL-QMGRWAIT
     END-IF.

     DISPLAY 'Back from QMGRWAIT. Return code=' RETURN-CODE.

 A-MAIN-END.
```

Obviously, the code is only for demonstration purposes. Hardcoding the queue manager name is not good coding practice.

In this case the program will put out the following message (if there is no NOWTOR DD).

xx **QMGRW04A :** *jobname* **has detected that** *qmgr* **is quiescing. Reply 'RETRY','WAIT' or 'CANCEL'**

If a NOWTOR is present, then the message will read:

QMGRW04I : *jobname* **has detected that** *qmgr* **is quiescing**

4.1.3 From REXX

```
Do until WAITRC = 0
      "call 'your.QMGRWAIT.loadlIB(QMGRWAIT)' 'MQAD'"
      WAITRC = RC
End
```

Obviously, the code is only for demonstration purposes. Hardcoding the queue manager name is not good coding practice.

In this case the program will put out the following message (if there is no NOWTOR DD allocated to the TSO session or job).

xx **QMGRW04A :** *jobname* **has detected that** *qmgr* **is quiescing. Reply 'RETRY','WAIT' or 'CANCEL'**

If a NOWTOR is present, then the message will read:

QMGRW04I : *jobname* **has detected that** *qmgr* **is quiescing**

Replying ‘CANCEL’ will not cancel the session (or batch execution) but it will cancel the REXX Exec being executed.

The TSO session will be held in a wait state until the reply has been entered.

QMGRWAIT is **not** recommended for conversational programs

5 Coding Considerations For a 2161 Reason Code

It is incumbent on the programmer to decide the business needs of an application program and code that strategy accordingly, however, all well behaving MQ enabled programs should be coded to handle a Queue Manager Quiescing situation. (MQ reason Code 2161 MQRC-Q-MGR-QUIESCING

Regardless of the failure reason code, applications must consider how they will deal with an error situation. **QMGRWAIT** is intended to specifically handle the MQRC-Q-MGR-QUIESCING reason code (MQ reason code 2161).

Obviously, application programs can choose whether to make use of **QMGRWAIT** and if they do then they must also decide what action they want operations to take if the situation arises. Once again, it is a business-driven decision as to how to handle these situations. This document simply gives some suggestion as to how to handle the 2161 situation.

5.1 On a MQCONN

Most MQ Programs will be required to establish a connection to at least one queue manager if they are going to use MQ.

As MQCONN is the single most ‘costly’ (in terms of CPU cycles and response) MQAPI call that a program will make programs should only connect to a queue manager once. This is true regardless of how many queues it is going to open and regardless of the types of messages it will read or write. The exception to this rule is if the connection is broken or if the program detects that the Queue Manager is attempting to close down.

As MQCONN is usually the very first MQ API call then if the return code from the MQ connection is 2161 the only program action that may be needed is simply to wait for the queue manager to become active again. **QMGRWAIT** delivers this ability without having to cancel and restart the program.

5.2 On a MQOPEN

Once a program has successfully connected to a queue manager the next MQ API call is usually MQOPEN to open the target queue. If a request to stop the queue manager is issued between the MQCONN and the MQOPEN then the MQOPEN will get a 2161 reason code returned to it.

At this point we are in somewhat of a ‘deadly embrace’ situation. The queue manager can’t end because the job still holds a connection, and the job can’t open the queue because the queue manager is trying to close down.

What well behaving applications should do is:

- Assess the need to roll back already processed data.
- Disconnect from the Queue manager.
- Fail the code or call the **QMGRWAIT** program.
- If the code is failed, then normal abnormal end processing can take place
- If **QMGRWAIT** is called, then when the queue manager returns to active state the program can reconnect to the queue manager and then retry the open.

Obviously, these decisions are business driven and those decisions should be taken with the business unit.

It should be noted that using an MQPUT1 would still have the same issues as an MQPUT1 still attempts to open a queue.

5.3 On a MQGET

If the application detects a 2161 (Queue manager quiescing) while attempting to perform an MQGET then the action to be performed **may** simply be to close the queue, disconnect from the queue manager and wait until the queue manager is restarted. However, as always it is a business decision, so you need to review the options together with the business unit.

It is important to note that if the MQGET is for a reply and that reply is to a temporary dynamic queue then the reply message will not be returned, thus the program will have to re-issue the original request when the queue manager is restarted.

Application programmers should decide what actions to take for any other reason codes that are returned.

Some of the more common reasons include:

- 2016 – Queue is GET inhibited.
- 2033 – No message available.
- 2035 – Security authorization failed
- 2037 – Queue is not open for input.
- 2079 – Truncated message accepted.
- 2080 – Truncated message failed.

It is incumbent of the application program to decide which, if any, of these conditions should be handled within the program.

It is especially important for programs to handle the truncated message scenario. A truncated message will not be removed from the queue by the queue manager and hence the potential exists for a program to loop trying to get a message that is too big for the area provided by the application program.

5.4 On a MQPUT

Once a program has successfully connected to the queue manager and if the program opened a queue for output then the next action is usually to PUT a message onto that queue using MQPUT.

One of the possible reasons for failure from the MQPUT is a 2161 (Queue manager quiescing).

This is just one of many reasons for failure of an MQPUT (or MQPUT1).

Application programmers should decide what actions to take for any other reason codes that are returned.

Some of the more common reasons include:

- 2035 – Security authorization failed
- 2039 – Queue is not open for output.
- 2051 – Queue is PUT inhibited.
- 2054 – Queue full

It is incumbent of the application program to decide which, if any, of these conditions should be handled within the program. For example, if the queue becomes full the application may decide to write the remaining records to a different queue (or a different Queue Manager), or it may decide to write to a flat file, or it may decide to terminate abnormally.

Obviously, these decisions are business driven and those decisions should be taken with the business unit.

5.5 On a MQPUT1

When only one message needs to be written to a queue then the most efficient method of doing this is to use MQPUT1.

Using MQPUT1 eliminates the need to issue separate MQOPEN, MQPUT and MQCLOSE API's.

As with MQPUT there are many other reason codes that may be issued and it is the application programs responsibility to decide what recovery actions, if any, to take when these are issued.

Obviously, these decisions are business driven and those decisions should be taken with the business unit.

6 Messages

QMGRW01E

QMGRW01E : NO PARM PASSED. RC=16

Meaning

The program was called without a parameter being passed to it. **QMGRWAIT** will end with a return code of 16.

Corrective Action

Call **QMGRWAIT** with the required parameter.

.

QMGRW02E

QMGRW02E : INVALID PARM PASSED. RC=16

Meaning

The program was called with an invalid parameter passed to it. **QMGRWAIT** will end with a return code of 16.

Corrective Action

Call **QMGRWAIT** with the required parameter.

QMGRW03I

QMGR03I : FREE OF START CIB UNSUCCESSFUL

Meaning

QMGRWAIT issued a QEDIT to free the start CIB for the job but it got a return code other than 0

.

Corrective Action

This is an informational message only and it may be ignored.

QMGRW04A

QMGRW04A : *jobname* ***has detected that*** *qmgr* ***is quiescing. Reply 'RETRY', 'WAIT' or 'CANCEL'***

Meaning

This message is generated by the program to give the operators an indication that the queue manager name (*qmgr*) passed as a parameter to the program is quiescing.

'*jobname*' will be replaced by the name of the job that called **QMGRWAIT**.

Corrective Action

Check the JOB log for any accompanying error messages.

Respond to this message as directed by the application programmer.

.

QMGRW04I

QMGRW04I : *jobname* ***has detected that*** *qmgr* ***is quiescing.***

Meaning

This message is generated by the program to give the operators an indication that the queue manager name (*qmgr*) passed as a parameter to the program is quiescing.

'*jobname*' will be replaced by the name of the job that called **QMGRWAIT**.

Corrective Action

Check the JOB log for any accompanying error messages.

Respond to this message as directed by the application programmer.

QMGRW05I

QMGRW05I : RETRY COMMAND ACCEPTED

Meaning

This message is issued in when the reply to the WTOR from the operator was RETRY or when the F *jobname*,RETRY command is entered.

Corrective Action

This is an informational message only and it may be ignored.

.

QMGRW06E

QMGRW06E : INVALID COMMAND

Meaning

The command entered via the F *jobname*,xxxx method is invalid and is ignored.

Corrective Action

Issue a correct command. Valid commands are RETRY, CANCEL, WAIT or WAIT *sss*. Where *sss* is the number of seconds to wait.

QMGRW07E

QMGRW07E : INVALID WAIT VALUE. SET TO 60 SECONDS

Meaning

The value passed via either the F *jobname*,WAIT sss command or via the WAIT sss reply to the WTOR is invalid.

The default value will be reset to 60 seconds.

Corrective Action

Re-issue the reply/command specifying the correct value for the wait duration.

QMGRW08I

QMGRW08I : *jobname* **JOB CANCELLED**

Meaning

This message is issued in when the reply to the WTOR from the operator was CANCEL or when the F *jobname*,CANCEL command is entered.

Corrective Action

This is an informational message only and it may be ignored.

.

7 Summary of Amendments

Date	Version	Fix Id.	Comment
4th June 2022	6.1	n/a	Release version.
24th October 2017	4.0	n/a	Ownership transferred to Abbydale Systems LLC.
7th July 2006	2.2	FIX001	Message ID for QMGRW04A changed to QMGRW04**I** when NOWTOR card is used.
5th July 2006	2.1	n/a	Support for NOWTOR added
28th September 2005	2.0	n/a	Added command interface support
26th September 2005	1.0	n/a	Initial Program written

Obtaining Support

Support for, comments about and suggestions for enhancements for this product can be obtained from our website :

www.abbydalesystems.com

or by emailing us at

support@abbydalesystems.com

In order to assist us in filtering support emails please specify in the heading of the email the name of the product that you require support on.

Spam will not be tolerated at this email address.

Where source code is provided for the product, support will be on a 'best efforts' basis. Where the user site has modified the source code, support may entail requesting copies of that sites source code and may result in support being withdrawn if this is not provided.

Abbydale Systems LLC. reserves the right to any code modifications that may have been undertaken at the user site.

Any alteration of the copyright information contained in the original source code is an infringement of the copyright of this and any other Abbydale Systems product and may result in legal action being taken against the perpetrator.