MD0B: Exit Demonstrator
User Guide
Version 1.0

April 2003

Morag Hughson
WebSphere MQ Development
MP211,
IBM UK Laboratories Ltd.
Hursley
Winchester
Hants, SO21 2JN
United Kingdom

hughson@uk.ibm.com

Take Note!

Before using this User's Guide and the product it supports, be sure to read the general information under "Notices".

First Edition, April 2003

This edition applies to Version 1.0 of *Exit Demonstrator* and to all subsequent releases and modifications until otherwise indicated in new editions.

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

WebSphere MQ
IBM
The following terms are trademarks of other Microsoft Corporation in the United States and/or other countries:

Windows NT
Windows 2000

Contents

Preface

Teaching the concepts of Exits and how to program exits can be difficult without hands on programming, something that is clearly not available with a speaker - audience set-up. I created this demo to try to show the programming concepts of the exits in WebSphere MQ and to make talking about exit interfaces more interesting for the audience.

This Support Pac can be used for demonstration purposes or as a teaching aid. It should not be used in a production system since every exit call involves human interaction.

Chapter 1. Exit Demonstrator

This document describes the functions available in the Support Pac.

Overview

The *Exit Demonstrator* can be used to show the programming interfaces of various exits available in WebSphere MQ. This allows the user to interact with the exit and therefore with various functions in WebSphere MQ, e.g. the channels, without coding the exit from scratch. This could for example, allow the user to see what can be done using an exit, and to try out some ideas before actually writing the exit.

1. Channel exits.
   All the different channel exits can be shown through the *Exit Demonstrator* since the programming interface for all the channel exits, Security, Send, Receive, Message and Message Retry, are the same.

2. Channel Auto-definition exit.
   The Channel Auto-definition exit can be shown through the *Exit Demonstrator*. It's interface is very similar to that of the channel exits.

3. Cluster Workload exit.
   The Cluster Workload exit and all the data that is passed to it can be shown through the *Exit Demonstrator*.

Installation

This Support Pac provides a GUI and an exit DLL which provides entry points for each of the above exit types.

Create a directory, say EXITDEMO, and copy the following files into it :-

- exitdemo.exe - The GUI program
- demoexit.dll - The exit DLL from the platform directory (currently only available for Windows)

You can use the exit DLL from this directory by either changing the exit path of your queue manager to the directory you have created by setting ***ExitsDefaultPath*** in the registry or by using the full path name in the WebSphere MQ definitions (see *'Setting up the exits'* on page 2 for examples); or you can copy the exit DLL into your current exits path.

Once the GUI program is in the path it can be run just by typing 'EXITDEMO' on the command line.

Chapter 2. Getting Started

This chapter describes the steps required to run the *Exit Demonstrator*.

Setting up the exits

The exit DLL has several entry points, one for each exit type. The exit is available for Windows platforms. Future additions to this SupportPac will provide the exit for other platforms. Currently the exit and the GUI must run on the same machine.

Entry Points

Channel Exits	ChlExit
Channel Auto-Definition Exit	ChadExit
Cluster Workload Exit	ClusExit

Channel Exits

To use the channel exit, define your channels as in the following examples.

Sender Channel using a Message Exit and Full Path

```
DEF CHL(TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(QM2.MACH) XMITQ(QM2)
MSGEXIT('C:\EXITDEMO\DEMOEXIT(ChlExit)')
```

Receiver Channel using a Security Exit and Default Path

```
DEF CHL(TO.QM4) CHLTYPE(RCVR) TRPTYPE(TCP) SCYEXIT('DEMOEXIT(ChlExit)')
```

Channel Auto-Definition Exit

To use the channel auto-definition exit, alter your queue manager object as follows.

```
ALT QMGR CHADEXIT('DEMOEXIT(ChadExit)') CHAD(ENABLED)
```

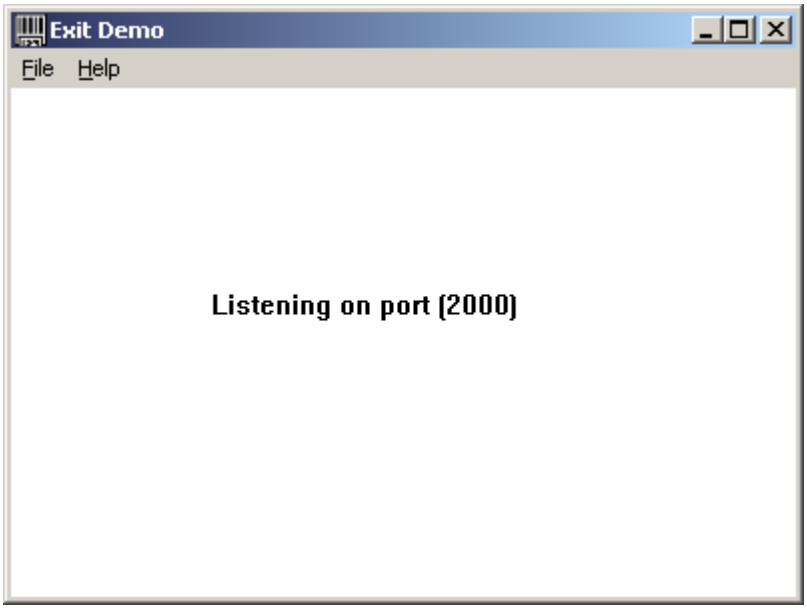
Cluster Workload Exit

To use the cluster workload exit, alter your queue manager object as follows. Run the cluster workload exit in SAFE mode. This is the default. See *WebSphere MQ System Administration Guide (SC34-6068)* for details on setting this property.

```
ALT QMGR CLWLEXIT('DEMOEXIT(ClusExit)')
```

Using the GUI

To run the Graphical display program type EXITDEMO on the command line. This will bring up a very simple window as shown.



The GUI will generate dialogs to display the exit interfaces as the exits are called. All the dialogs are different to reflect the differences in the interfaces for each of the exit types. There is one common point in all the dialogs and that is the buttons found on the bottom of each dialog.

Dialog Buttons

Pressing the OK button will return the data changed in the dialog back to the exit so that processing can continue. The dialog window will also be closed.

Pressing the Alter button will return the data changed in the dialog back to the exit so that processing can continue. The dialog window will remain and be reused by the same exit. When running several channel exits on the same channel, the dialog will be reused by all those exits since the interface to channels exits is the same and can therefore be shown using the same dialog.

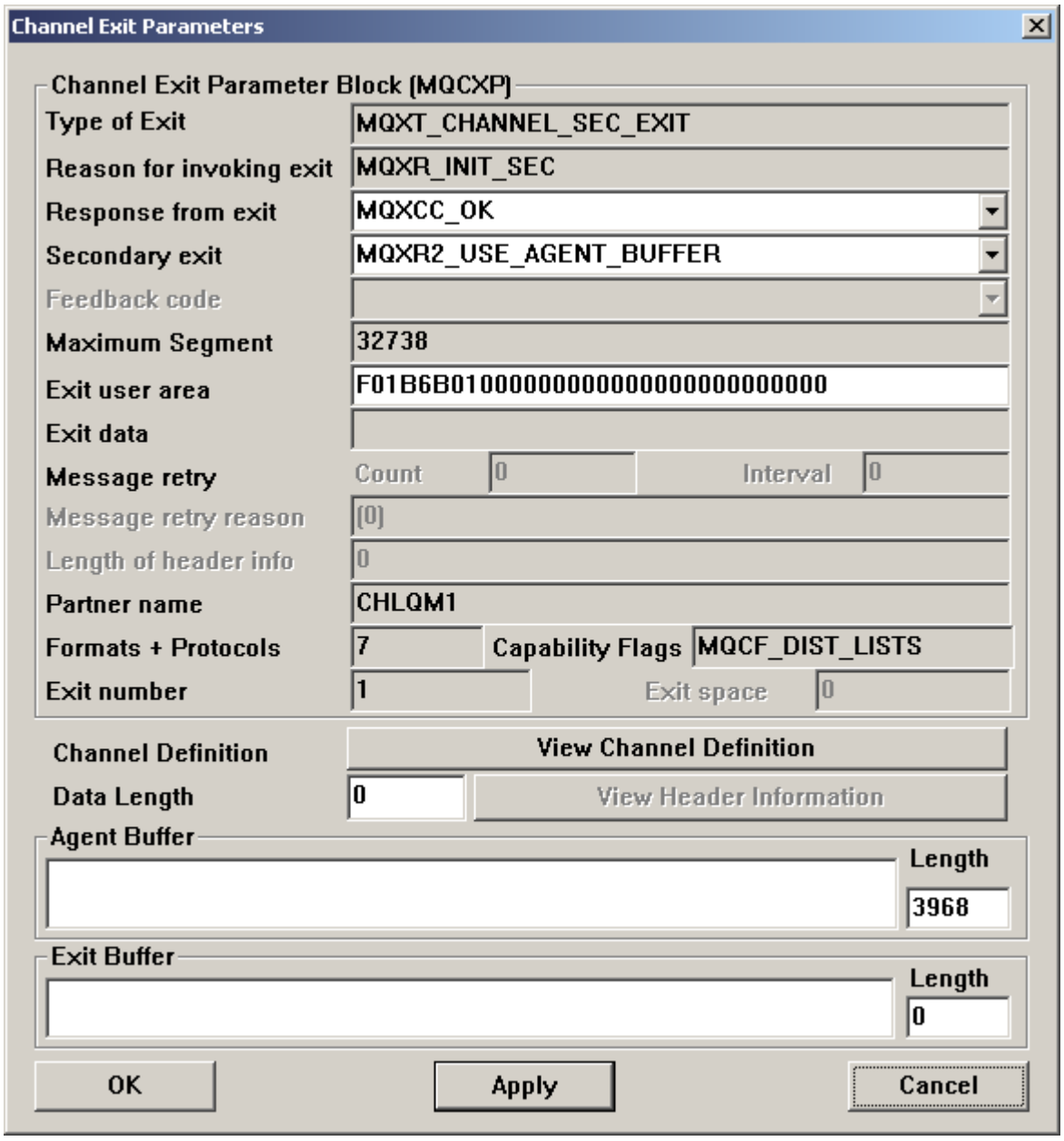
Pressing the Cancel button will close the dialog without sending the data changed in the dialog back to the exit.

Chapter 3. Channel Exit Displays

This chapter describes the displays which show the channel exit programming interface.

Channel Exit Interface Dialog

When the channel calls the exit, and the GUI is running, the following dialog will be shown. This dialog displays in a graphical format the programming interface to the channel exit, allowing the user to interact with the exit, and therefore with the channel. The fields are labelled with descriptive text rather than the actual field names from the MQCXP structure, but are shown in the order they appear in the structure and should be easy to map to the various structures as required.



Example Demonstrations

This section provides a few examples of what you might want to show with the *Exit Demonstrator* based on what I have used when presenting this as a demo.

Demonstrating the Security Exit

This example uses a sender channel and a receiver channel both defined with a security exit (see *'Setting up the exits'* on page 2 for example definitions).

1. Both ends of the channel will be called with *'Reason for invoking exit'* set to MQXR_INIT. Press the *'OK'* or *'Apply'* button (see *'Dialog Buttons'* on page 3 for details about the difference between these buttons).

2. Next the receiver end of the channel will be called with MQXR_INIT_SEC since responding channels are given the first opportunity to initialise the security on the channel.

3. Security exit messages are free format, they can be anything that the Security exit understands, the channel simply delivers the data to the other end of the channel and to the other Security exit. So in the *'Agent Buffer'* field on the dialog you can type a message to send to the other exit, for example, "Who are you?". Then select an appropriate *'Response from exit'* by using the pull-down list, for example MQXCC_SEND_SEC_MSG.

4. The message you typed in the *'Agent Buffer'* will be displayed in the dialog for the other end of the channel when it is called with MQXR_SEC_MSG.

5. You can send messages back and forth between the pair of security exits until you are happy that you have identified the caller and allow the channel to continue by selecting a *'Response from exit'* of MQXCC_OK, or if you are not happy you can close the channel by selecting a *'Response from exit'* of MQXCC_CLOSE_CHANNEL.

Demonstrating Send, Receive and Message Exits

This example uses a sender channel and a receiver channel both defined with a send, receive and message exit (see *'Setting up the exits'* on page 2 for example definitions). A message is MQPUT destined for a remote target queue to be delivered by this channel.

1. Both ends of the channel will be called with MQXR_INIT for the message, send and receive exit. As before press the *'OK'* or *'Apply'* button. You can see which exit is being called by looking in the *'Type of Exit'* field and examining the MQXT_* constant shown there.

2. The sender channel's message exit will be called with MQXR_MSG. Note that the *'Length of header info'* field contains a value. This is provided to allow an exit to skip over the header information and find the message data. This message data is shown in the *'Agent Buffer'* field. If you wish to view the header information, press the *'View Header Information'* button and this will bring up another dialog with tabs to show the *'Transmission Header'* information and the *'MQMD'*.

3. The sender channel's send exit will be called with MQXR_XMIT. Now the information shown in the 'Agent Buffer' field is not readable text. This is because there are WebSphere MQ headers here and we are shown the data that is about to be sent down the wire.

4. The receiver channel's receive exit will be called with MQXR_XMIT with the buffer of data we have just seen going through the send exit on the sender side. We are shown the data that has just been received off the wire.

5. The receiver channel's message exit will be called with our message data and we can look at similar information as we saw on the sender channel. We are called just before the message is put to the target queue, so we could, for example, change it's destination.

6. We have only sent one message down the channel, so when it discovers that there are no more messages to send it will end the batch. We will see the sender channel's send exit called again with MQXR_XMIT with a much shorted length of data. Again with is internal WebSphere MQ data and is not readable text and we are being show the data that is about to be sent down the wire.

7. The receiver channel's receive exit will be called with this internal buffer and then you will see the receiver channel's send exit be called with another internal buffer and then the sender channel's receive exit called with this internal buffer. This illustrates that receive exits are used on sender channels and send exits are used on receiver channels. Although your messages only flow in one direction, data will flow in both directions.

Demonstrating the Message Retry Exit

This example uses a sender channel and a receiver channel where the receiver channel is defined with a message retry exit (see *'Setting up the exits'* on page 2 for example definitions). A message is MQPUT destined for a remote target queue to be delivered by this channel, but the remote target queue is full.

1. The Message Retry exit will be called with MQXR_INIT. As before press the *'OK'* or *'Apply'* button.

2. As with the message exit, the *'Length of header info'* field will be filled in, your message data can be seen in the *'Agent Buffer'* field and if you press the *'View Header Information'* button you can see the *'Transmission Header'* and the *'MQMD'*. You can change this information to redirect the message to a different queue.

There are other interesting pieces of information to look at with a Message Retry Exit which might help you to decide what to do with this message.

- The *'Message retry reason'* field contains the MQRC reason code from the channel's MQPUT to the target queue. The display also provides a text description of the MQRC.
- The *'Message retry count'* field contains the number of retry attempts the channel has made with this message.

Other things to look at

The above examples demonstrate particular points about each channel exit type. With all of the above exit types you can also look at the channel definition by clicking on the *'View Channel Definition'* button. This brings up a separate dialog to show the channel definition.

Once the channel has connected to it's partner the *'Partner name'* field will contain the name of the partner queue manager.

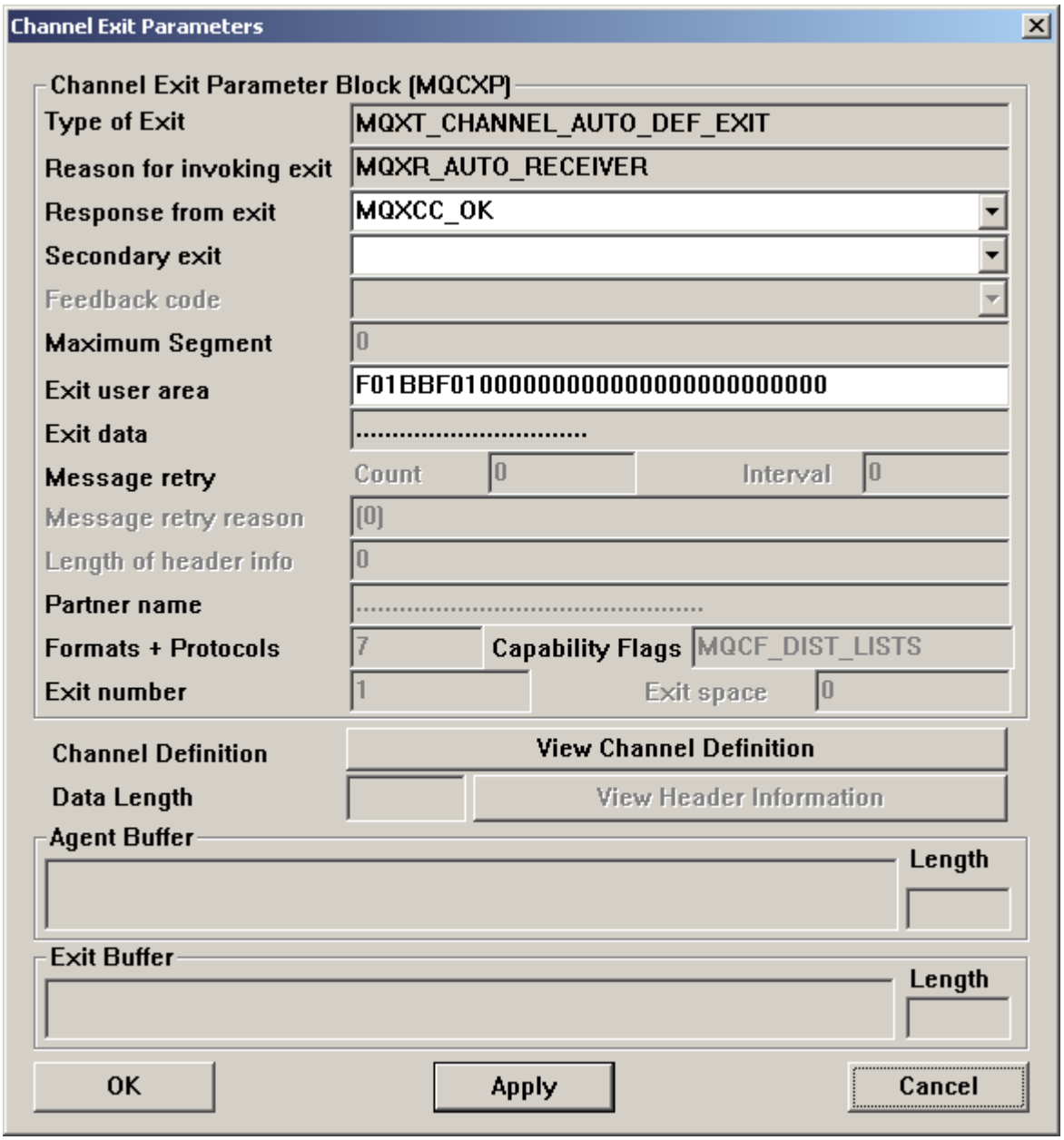
There are a variety of *'Response from exit'* values. They can be selected from the pull down list. The uses of the various response values are documented in *"WebSphere MQ Intercommunication (SC34-6059)"*.

Chapter 4. Channel Auto-Definition Exit Displays

This chapter describes the displays which show the channel auto-definition exit programming interface.

Channel Auto-Definition Exit Interface Dialog

When the channel auto-definion exit is called, and the GUI is running, the following dialog will be shown. This dialog displays in a graphical format the programming interface to the channel auto-definition exit, allowing the user to interact with the exit, and therefore with the definition of the channel. The fields are labelled with descriptive text rather than the actual field names from the MQCXP structure, but are shown in the order they appear in the structure and should be easy to map to the various structures as required.



Note that the dialog used is exactly the same as that used for the channel exits, but the buffer fields at the bottom are greyed out because they are not relevant for this type of exit.

Example Demonstration

This section provides an example of what you might want to show with the *Exit Demonstrator* based on what I have used when presenting this as a demo.

Demonstrating the Channel Auto-Definition Exit

This example uses a sender channel which has no partner receiver channel defined. The receiving end queue manager has the auto-definition exit specified and enabled (see *'Setting up the exits'* on page 2 for example definitions).

3. The channel auto-definition exit will be called with *'Reason for invoking exit'* set to MQXR_INIT. Press the *'OK'* or *'Apply'* button (see *'Dialog Buttons'* on page 3 for details about the difference between these buttons).

4. Next the channel auto-definition exit will be called with the *'Reason for invoking exit'* field set to a MQXT_* constant which shows the channel type being automatically defined. In this example that is MQXT_AUTO_RECEIVER.

5. Now you can choose to either

    a. Disallow the automatic definition of this channel by selecting a response of MQXCC_SUPPRESS_FUNCTION from the *'Response from exit'* pull down list.

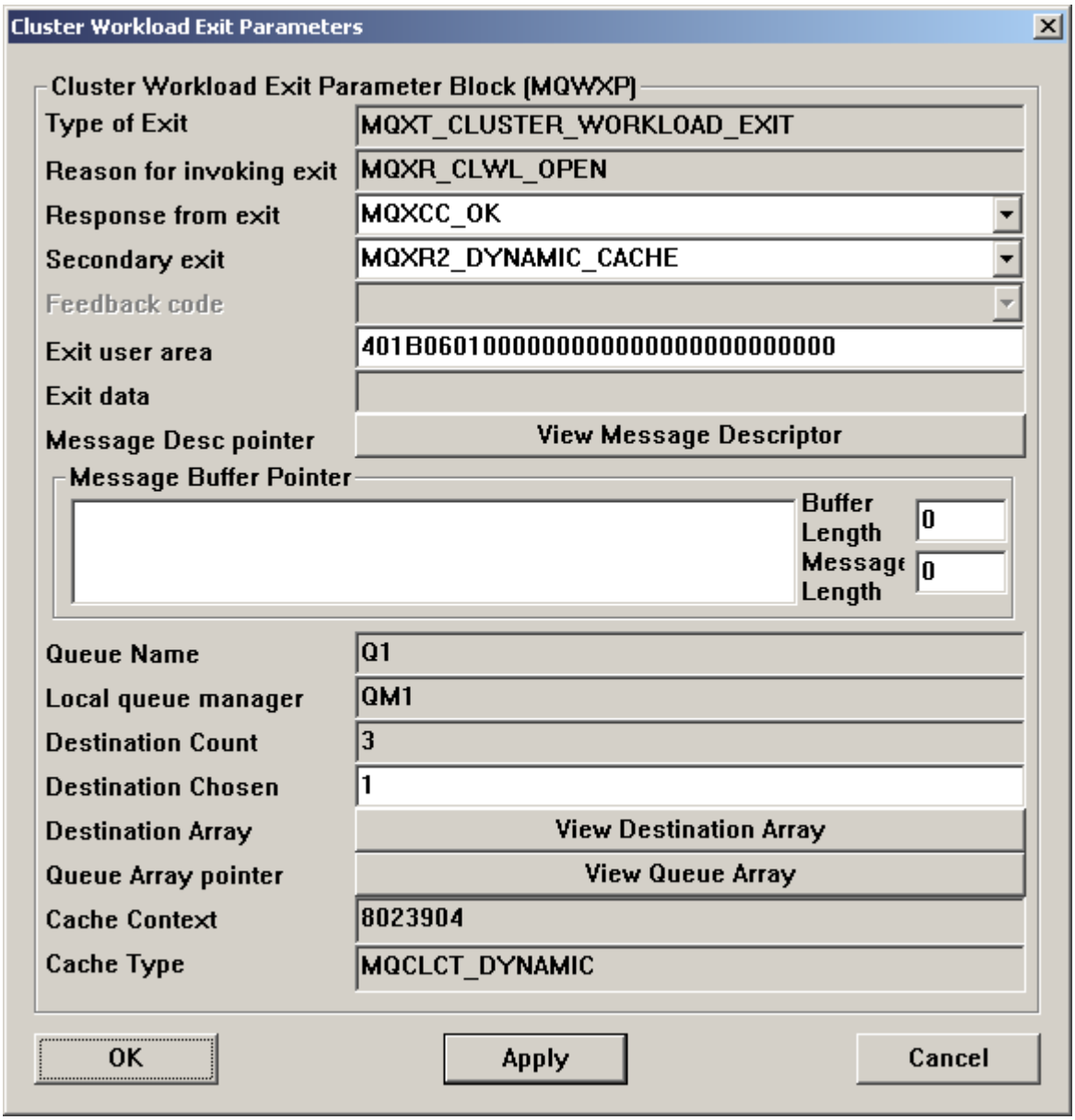
    b. Allow the automatic definition of this channel and make changes to the channel definition by pressing the *'View Channel Definition'* button and editing the channel definition in the dialog presented to you.

Chapter 5. Cluster Workload Exit Displays

This chapter describes the displays which show the cluster workload exit programming interface.

Cluster Workload Exit Interface Dialog

When the cluster workload exit is called, and the GUI is running, the following dialog will be shown. This dialog displays in a graphical format the programming interface to the cluster workload exit, allowing the user to interact with the exit, and therefore with the destination chosen for the message. The fields are labelled with descriptive text rather than the actual field names from the MQWXP structure, but are shown in the order they appear in the structure and should be easy to map to the various structures as required.



Example Demonstration

This section provides an example of what you might want to show with the *Exit Demonstrator* based on what I have used when presenting this as a demo.

Demonstrating the Cluster Workload Exit

This example uses four queue managers in a cluster where the queue manager that is connected to, to do the MQPUT to a cluster queue has the cluster workload exit specified (see *'Setting up the exits'* on page 2 for example definitions). Each queue manager in the cluster hosts an instance of a cluster queue, including the local one. This can help to demonstrate how you can use the exit to override the default behaviour of putting to a local queue if there is one.

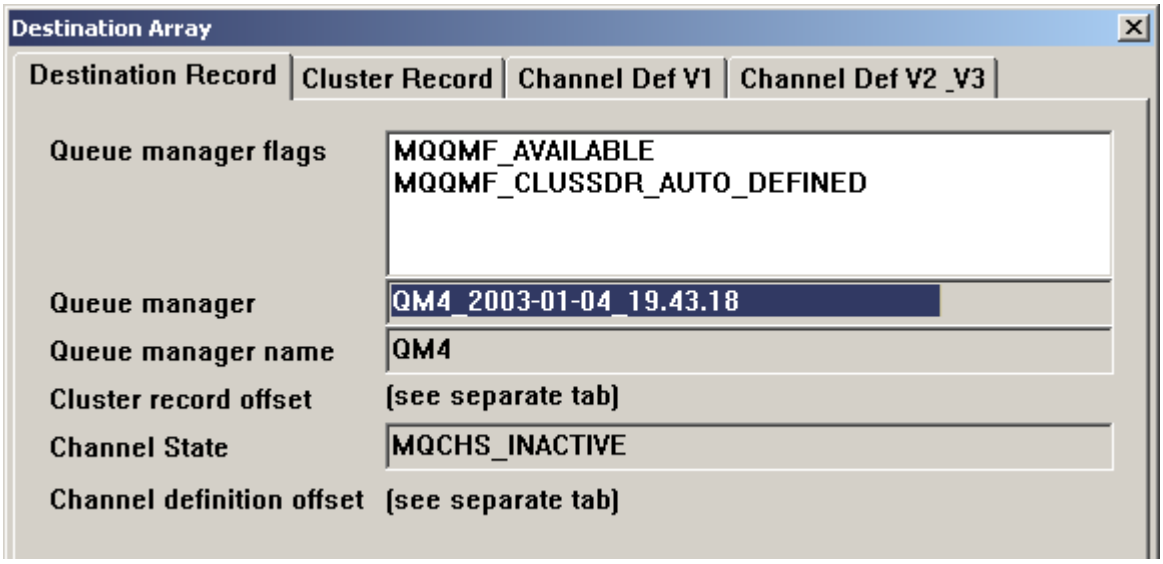
When the queue manager is first started after altering the queue manager object to specify the exit name, if the GUI is running, it will show the exit being called for MQXR_INIT and also perhaps some other reasons to do with moving repository messages. Press the *'OK'* or *'Apply'* button on these (see *'Dialog Buttons'* on page 3 for details about the difference between these buttons).

When a cluster queue is opened with MQOO_BIND_ON_OPEN, or a message is MQPUT if the cluster queue was opened with MQOO_BIND_NOT_FIXED, then the cluster workload exit is called. The cluster workload exit doesn't provide too many things that can be changed, the usual response fields and the *'Destination Chosen'* field. Most of the other thing you can see using the GUI allow you to make a decision about what number to put into that field. So we will look at what the exit shows you.

The 'Reason for invoking exit' will show for example that we used MQOO_BIND_ON_OPEN, by giving MQXR_CLWL_OPEN.

We can look at the destination array, in other words, the set of cluster queue managers that host this cluster queue, by pressing the *'View Destination Array'* button. This brings up a dialog with several tabs.

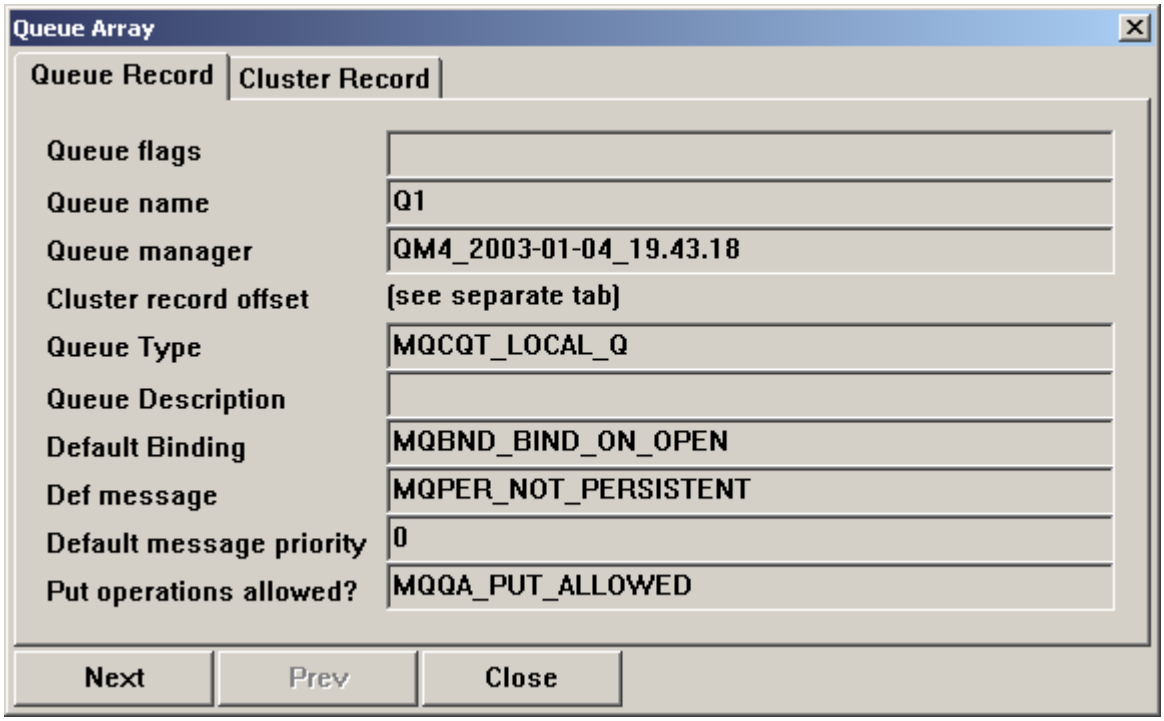
The *'Destination Record'* tab shows the fields from an MQWDR structure as shown below. This allows you to check for attributes of the queue manager and the channel to that queue manager. For



example you can see whether the queue manager has been suspended from the cluster, whether there is a manually defined channel to that queue manager, whether the queue manager is also a full repository and what the status of the channel to that queue manager is. The *'Cluster Record'* tab shows which cluster the queue manager is in, and the *'Channel Def'* tabs allow you to inspect the channel definition to this queue manager. You can press the *'Next'* and *'Prev'* buttons to view all the queue managers in the array.

We can look at the queue array, in other words, the set of cluster queues, by pressing the *'View Queue Array'* button. This brings up a dialog with several tabs.

The *'Queue Record'* tab shows the fields from an MQWQR structure as shown below. This allows you



to check attributes of each instance of the queue. For example, whether the queue has been disabled for put, or whether it is hosted locally. The *'Cluster Record'* tab shows which cluster the queue is in. You can press the *'Next'* and *'Prev'* buttons to view all the queues in the array.

Appendix A: Current Restrictions

This appendix details the main restrictions on use of the *Exit Demonstrator* and also acts as something of a to-do list for the author.

- Exit Buffer is not used on channel exits.
- Only the first cluster is shown in the cluster workload exit if a Cluster Queue Manager or Cluster Queue is in more than one cluster.
- Exit and GUI must run on the same machine.

Appendix B: Future Additions

This appendix details some of the additions that are likely to be made to future versions of this SupportPac.

- API Exit support
- Exit provided for other platforms as well as Windows

Appendix C: Bibliography

This appendix details relevant manuals to reference for more information.

- WebSphere MQ Intercommunication (SC34-6059)
  For information on channel exits and the channel auto-definition exit
- WebSphere MQ Queue Manager Clusters (SC34-6061)
  For information on the cluster workload exit
- WebSphere MQ Application Programming Guide (SC34-6064)
  For information on using and writing API exits
- WebSphere MQ Application Programming Reference (SC34-6062)
  For information on the MQI calls
- WebSphere MQ System Administration Guide (SC34-6068)
  For information on configuring the API exit, and reference material on the various structures used. Also for information on WebSphere MQ configuration information.