Middleware Approach for Configuration Monitoring of IBM MQ

Scripting in SetSphere's ASPL and Perl on UNIX Systems

SAMPLE BOOK PREVIEW

Bassem W. Jamaleddine

TCND Inc.
POBox 680
New York, NY 10156
ph. (212) 203 4379
fax (212) 208 3039
aspl@setsphere.com
www.setsphere.com
www.tcnd.com

Books for SetSphere Shops

SetSphere - ASPL User Manual
SetSphere - ASPL Operations Guide
SetSphere - ASPL USer Guide

This book was written, edited, designed, and produced by Bassem Jamaleddine.

SAMPLE BOOK PREVIEW

© 2026, Total Computing & Network Design, Inc.
All rights reserved.

ISBN 979-8-16-213826-6 (Sbk)
ISBN 979-8-16-112457-4 (Hbk)

Contents

Preface iii
Chapter 1 Introduction to IBM MQ Configuration Objects Monitoring with ASPL 3
1-1 Introduction to IBM MQ Configuration Objects Monitoring with ASPL 4
1-2 Parsing IBM MQ Queue Objects 17
1-3 Parsing IBM MQ Channel Objects 22
Chapter 2 Group Operations on IBM MQ Objects with ASPL 29
2-1 Grouping Class for IBM MQ Qmgr Attributes 30
2-2 Grouping Class for IBM MQ Queue Attributes 45
2-3 Grouping Class for IBM MQ Channel Attributes 61
2-4 Shallow Group Operations on IBM MQ Channel Attributes 76
Chapter 3 Monitoring Clustered IBM MQ 85
3-1 Monitoring Changes in Clustered IBM MQ Qmgrs and Sending Notifications 86
3-2 Monitoring Changes in Clustered IBM MQ Queues and Sending Notifications 97
3-3 Monitoring Changes in Clustered IBM MQ Channels and Sending Notifications 115
Chapter 4 ASPL MQ Writers 131
4-1 ASPL MQ Writers 132
Appendices
Appendix A The module MQutils::MQparser to Parse IBM MQ Objects 147
Appendix B Script to Display IBM MQ Qmgrs: qmgrsdisp.pl 165
Appendix C Script to Compare IBM MQ Qmgrs: qmgrscompare.pl 167
Appendix D Script to Display IBM MQ Queues: queuesdisp.pl 170
Appendix E Script to Compare IBM MQ Queues: queuescompare.pl 173
Appendix F Script to Display IBM MQ Channels: channelsdisp.pl 177
Appendix G Script to Compare IBM MQ Channels: channelscompare.pl 179
Appendix H Middleware MQ Monitoring at ACME Co. 183
Appendix I Adding MQ Scripting to ASPL 185
Index 191

SAMPLE BOOK PREVIEW
SAMPLE BOOK PREVIEW

Middleware Approach for Configuration Monitoring of IBM MQ
ASPLMQ 1.00

PREFACE

7 October 2026

This guide shows you how to use SetSphere's ASPL calculator to monitor and to detect changes in IBM MQ objects. You should have installed ASPL on your computer.

In this user guide we refer to the ASPL prompt with aspl> as shown here

aspl>

If the order of command is important, then we will number them as follows

aspl 1 > acmeny = ggqmgrchann(hostname,nynymq,qmgrname,ACMENYNY)

aspl 2 > v gad acmeny

aspl 3 > ? acmeny

We refer to the UNIX shell prompt with a hash as shown here

SAMPLE BOOK PREVIEW

aspl -v

SAMPLE BOOK PREVIEW

All commands issued on the command prompt are preceded by a hash #. What precedes the hash # can be informative on where the command was entered. For instance, since IBM MQ is typically hosted on a server under the UNIX user mqm, and for the purpose of testing, we need to issue some commands on the host serving the queue manager, for clarity we will use the following notation for the command prompt.

Say the qmgr ACMECO is running on the host server vavamq, we want to issue the command to list the queue of ACMECO, then we will use the following command:

(vavamq's qmgr:ACMECO) 18:23 mqm@vavamq: ~ `# echo "dis ql(*)" | runmqsc ACMECO`

Notice the shell prompt is always the # and *the host and the qmgr name* are shown in parenthesis at the beginning of the line and prior to the prompt. This command is being executed by the user mqm on the host vavamq.

If the order of commands execution is important, then the commands are numbered as follows:

1) - (vavamq's qmgr:ACMECO) 18:23 mqm@vavamq: ~ `# echo "dis QL(FOO) MAXMSGL" | runmqsc ACMECO`

2) - (vavamq's qmgr:ACMECO) 18:23 mqm@vavamq: ~ `# echo "echo "DEFINE QL(FOO) MAXMSGL(10111222) REPLACE" | runmqsc ACMECO`

3) - (vavamq's qmgr:ACMECO) 18:23 mqm@vavamq: ~ `# echo "dis QL(FOO) MAXMSGL" | runmqsc ACMECO`

Here, the first command displays the MAXMSGL of the local queue FOO, the second command changes the MAXMSGL to 10111222, and the third command displays the MAXMSGL again to make sure the change took place.

The formatting of the command was made possible with a specialized formatter PPAGES that I wrote to convey information in a distributed UNIX environment that depends on the shell command prompt.

The commands shown in this manual have been captured on one host where ASPL is installed, and two hosts: nynymq and vavamq where IBM MQ has been installed.

Throughout this text, commands issued at either the system shell prompt or the ASPL prompt are presented within scrollable boxes, each labeled with the word "Display" followed by an identification number. These display boxes illustrate the exact command-line entries; however, due to the scrollable container format, their complete contents are unexpanded within the printed edition of this manual. Full command-line listings can be viewed and expanded exclusively within the HTML distribution of the text.

The chapter "Introduction to IBM MQ Configuration Objects Monitoring with ASPL" is a brief description of what is being addressed in this text.
The MQ parser is the Perl module shown in appendix "The module MQutils::MQparser to Parse IBM MQ Objects".
The code distribution is in ASPLMQ-v1.tgz can be downloaded from ASPL's home site at TCND.COM. If you have installed ASPL, you can quickly install the code distribution as an add-on to your ASPL installation by going over appendix "Adding MQ Scripting to ASPL".

Bassem Jamaleddine
New York, NY
October 2026

SAMPLE BOOK PREVIEW
SAMPLE BOOK PREVIEW

```
def($_) } ($s =~ m{ DEFINE\s+(CHANNEL.*?) REPLACE$}gxms)

            Break the string into pieces where
            each piece starts with DEFINE CHANNEL
            and end with REPLACE.
() on each piece
 left attribue to
 value
```

SAMPLE BOOK PREVIEW

Chapter 1
Introduction to IBM MQ Configuration Objects Monitoring with ASPL

The ASPL interpreter is a containment set calculator that integrates natively with the UNIX system shell prompt. The first section of this chapter introduces the operational mechanics of ASPL, demonstrating how the interpreter unifies disparate IBM MQ group configurations using named identifiers within a centralized workspace to enable native group operations for analyzing and comparing configuration data.

Because many of the scripts provided within this text handle the parsing and comparison of IBM MQ objects deployed across mixed cloud host nodes, this chapter introduces core object parsing routines. The ASPL parsing module for IBM MQ objects categorizes configurations into three distinct object classes: queue managers (qmgr), queues, and channels. Each class features a regular structure, making it straightforward to parse them with short scripts utilizing Perl regular expressions and functional programming routines.

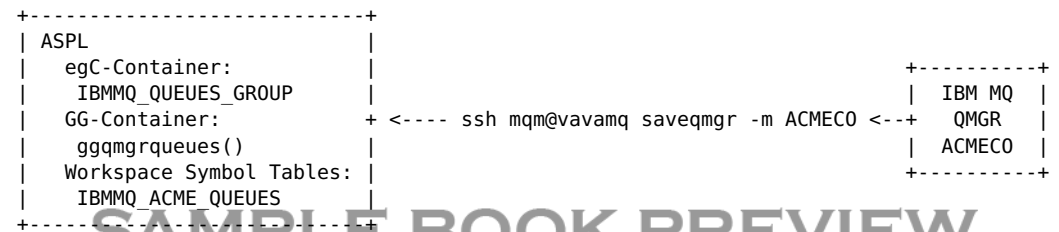
SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

1-1 Introduction to IBM MQ Configuration Objects Monitoring with ASPL

SetSphere's ASPL is a containment calculator that executes algebraic operations on set variables representing groups. Set variables are saved within distinct workspaces, where each variable name points to a GADg dataset object. This object is returned by a specialized global grouping function (GG-function) loaded in the GG-Container. Both the workspace and the grouping function must conform to the metadata specifications of the element grouping class managed by the egC-Container.

In the diagram below, the IBMMQ_QUEUES_GROUP class is defined in the egC-Container, and the ggqmgrqueues() function is loaded into the GG-Container. The IBMMQ_ACME_QUEUES workspace stores the set variables representing the queue groups for ACME Co.



SAMPLE BOOK PREVIEW

ASPL provides various operators to perform group operations on datasets saved within its symbol tables. Operators can refresh the dataset represented by a set variable by interrogating it: the interpreter evaluates the variable and checks for structural modifications within its constituent group elements. Upon detecting a change, ASPL processes an incremental update: the variable increments its historical archive index, move the previous data into that archive slot, and updates its internal state with the latest configuration data. Consequently, ASPL variables function natively as differential group variables.

Before we begin, we will demonstrate that ASPL grouping containment operates on par with the way IBM MQ structures its objects on the UNIX filesystem.

The QMGRLAB queue manager is created on two hosts: nynymq and vavamq.
First, create and start the qmgr QMGRLAB. Configure the QMGRLAB instance locally on host nynymq.

```
(nynymq's qmgr:QMGRLAB) 18:23 mqm@nynymq: ~ #  crtmqm QMGRLAB

(nynymq's qmgr:QMGRLAB) 18:23 mqm@nynymq: ~ #  strmqm QMGRLAB

(nynymq's qmgr:QMGRLAB) 18:23 mqm@nynymq: ~ #  runmqsc QMGRLAB < QMGRLAB.mqsc
```

Second, an identical creation, startup, and configuration is executed for the QMGRLAB instance running on host vavamq.

```
(vavamq's qmgr:QMGRLAB) 18:23 mqm@vavamq: ~ #  crtmqm QMGRLAB

(vavamq's qmgr:QMGRLAB) 18:23 mqm@vavamq: ~ #  runmqsc QMGRLAB < QMGRLAB.mqsc

(vavamq's qmgr:QMGRLAB) 18:23 mqm@vavamq: ~ #  strmqm QMGRLAB
```

IBM MQ stores its structured data on the filesystem relative to the mqm user's home directory, defaulting to the /home/mqm/qmgrs/ path. Figure 1.1.1 shows a split vi editor tree representation of the channels created on the filesystem for the users mqm@nynymq and mqm@vavamq. Since nynymq and vavamq are separate hosts servicing IBM MQ, the mqm user on each server possesses an independent /home/mqm/qmgrs/ directory.

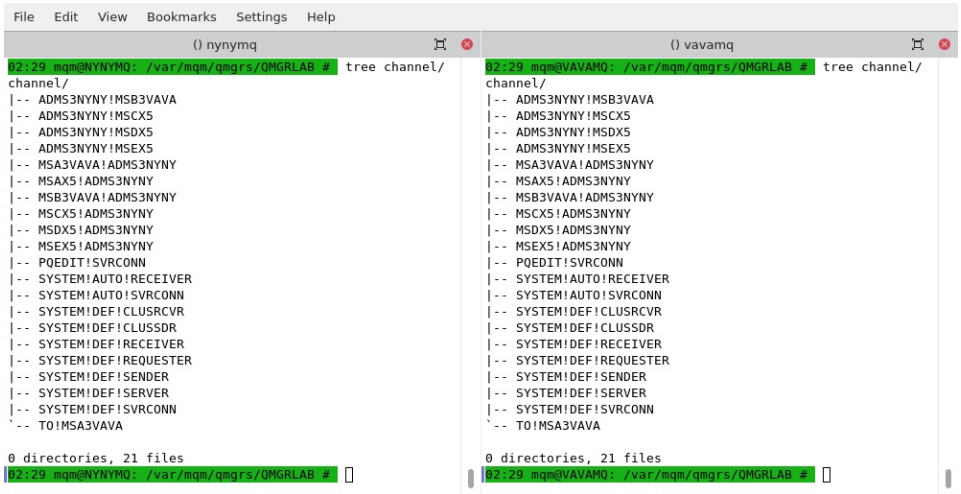


Fig. 1.1.1 [IBM MQ trees showing the channels on two different hosts]

ASPLMQ 2026 by Bassem Jamaleddine

As shown in the figure above, the channels created on both hosts share identical naming structures, yet they reside within entirely distinct filesystem directories.

The ASPL interpreter is strated by loading an arbitrary namedspace JUNKSPACE whose element grouping class is IBMMQ_CHANNELS_GROUP. The nyqmgrlab and vaqmgrlab set variables are then assigned the group of channels defined inside the QMGRLAB queue manager on hosts nynymq and vavamq, respectively. Figure 1.1.2 illustrates the operations required to create both variables and details their internal representation within ASPL symbol table.

SAMPLE BOOK PREVIEW

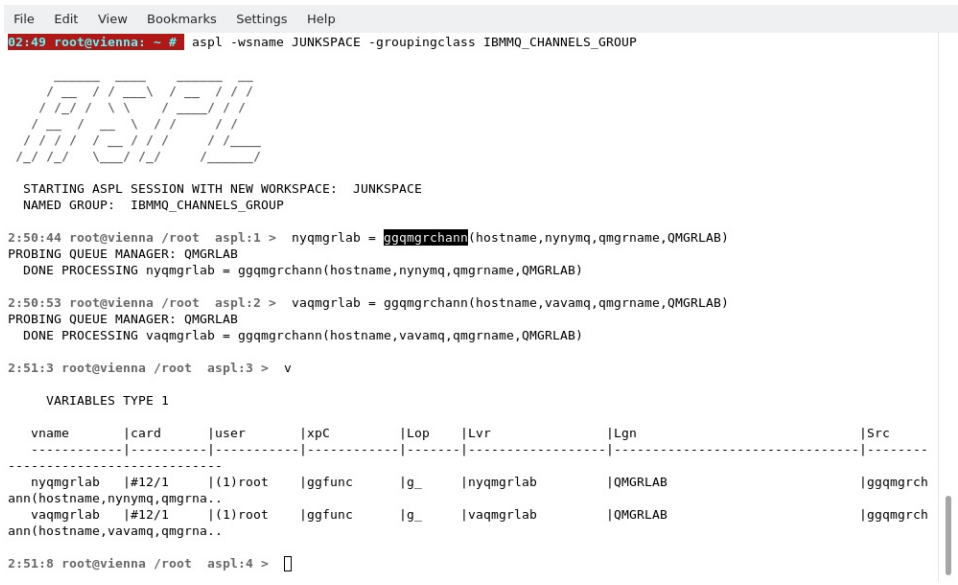
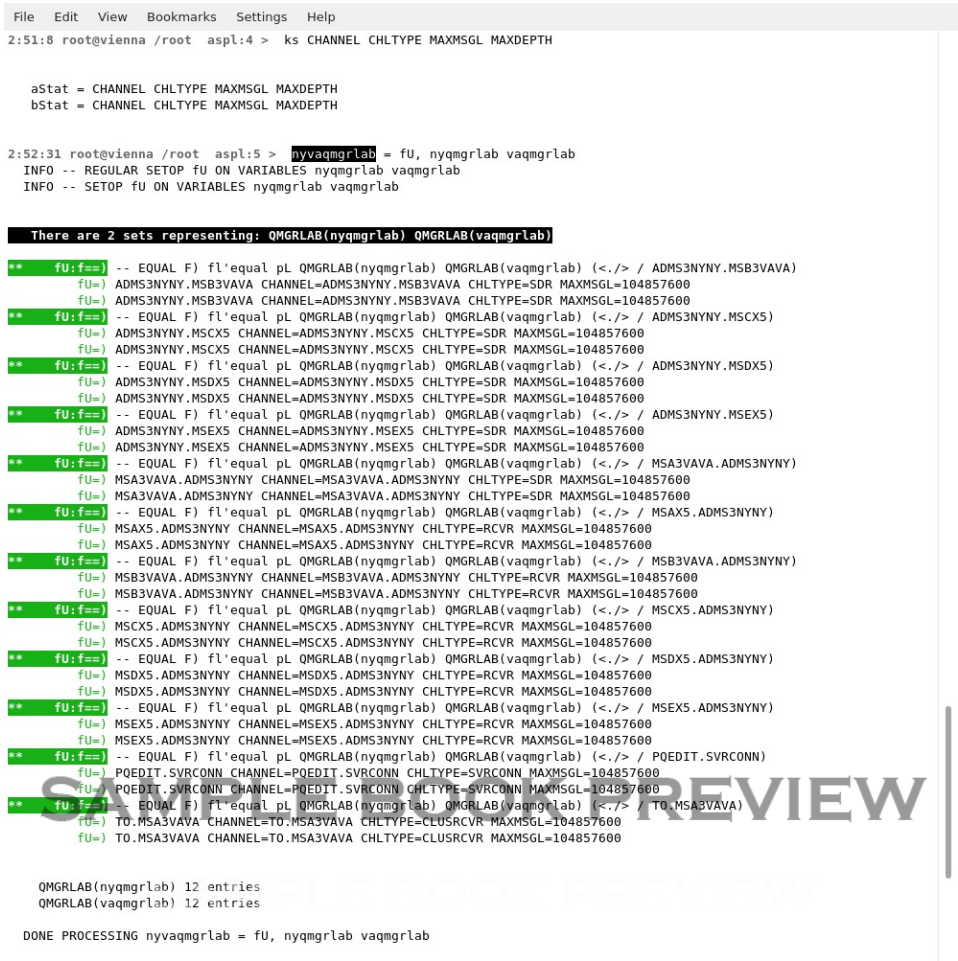


Fig. 1.1.2 [ASPL grouping variables for channels of two different qmgrs]

ASPLMQ 2026 by Bassem Jamaleddine

Comparing the QMGRLAB channels on both hosts is achieved by taking the union of the nyqmgrlab and vaqmgrlab set variables (see Figure 1.1.3).



SAMPLE BOOK PREVIEW

Fig. 1.1.3 [Grouping two qmgrs in an ASPL group variable]

ASPLMQ 2026 by Bassem Jamaleddine

Figure 1.1.4 illustrates the registration of channels within the nyvaqmgrlab variable, which represents the union of the nyqmgrlab and vaqmgrlab set variables. This centralized layout contrasts directly with the distributed filesystem hierarchy depicted in Figure 1.1.1. Within this GADg tree view of the nyvaqmgrlab variable, both groups of channels are managed simultaneously inside a single ASPL session. By using ASPL group containment functions, configuration data from a distributed cloud environment is aggregated into a common workspace, enabling the use of group operators to perform direct data comparisons.

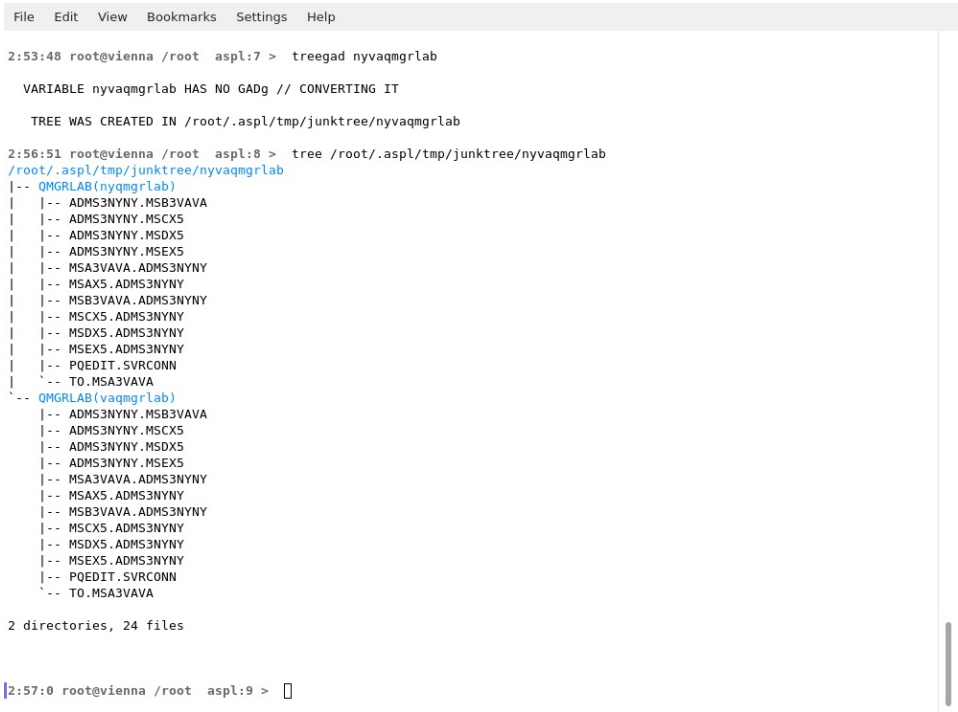


Fig. 1.1.4 [ASPL tree view of a GADg (Group Algebraic Data groups)]

ASPLMQ 2026 by Bassem Jamaleddine

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

The ASPL interpreter provides extended capabilities for targeted analysis; for example, specific attributes can be isolated by configuring the ks-vector prior to displaying the set union, as illustrated in Figure 1.1.5.



Fig. 1.1.5 [ASPL displaying the union with a ks-vector selection]

ASPLMQ 2026 by Bassem Jamaleddine

In a horizontal clustering topology where identical queue managers are replicated across multiple hosts, the similarity metric between the instances evaluates to 1.0, as illustrated in Figure 1.1.6.

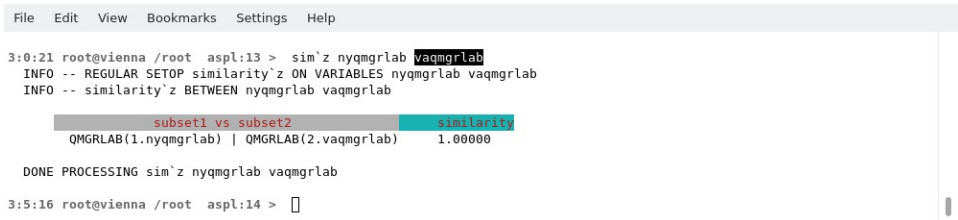
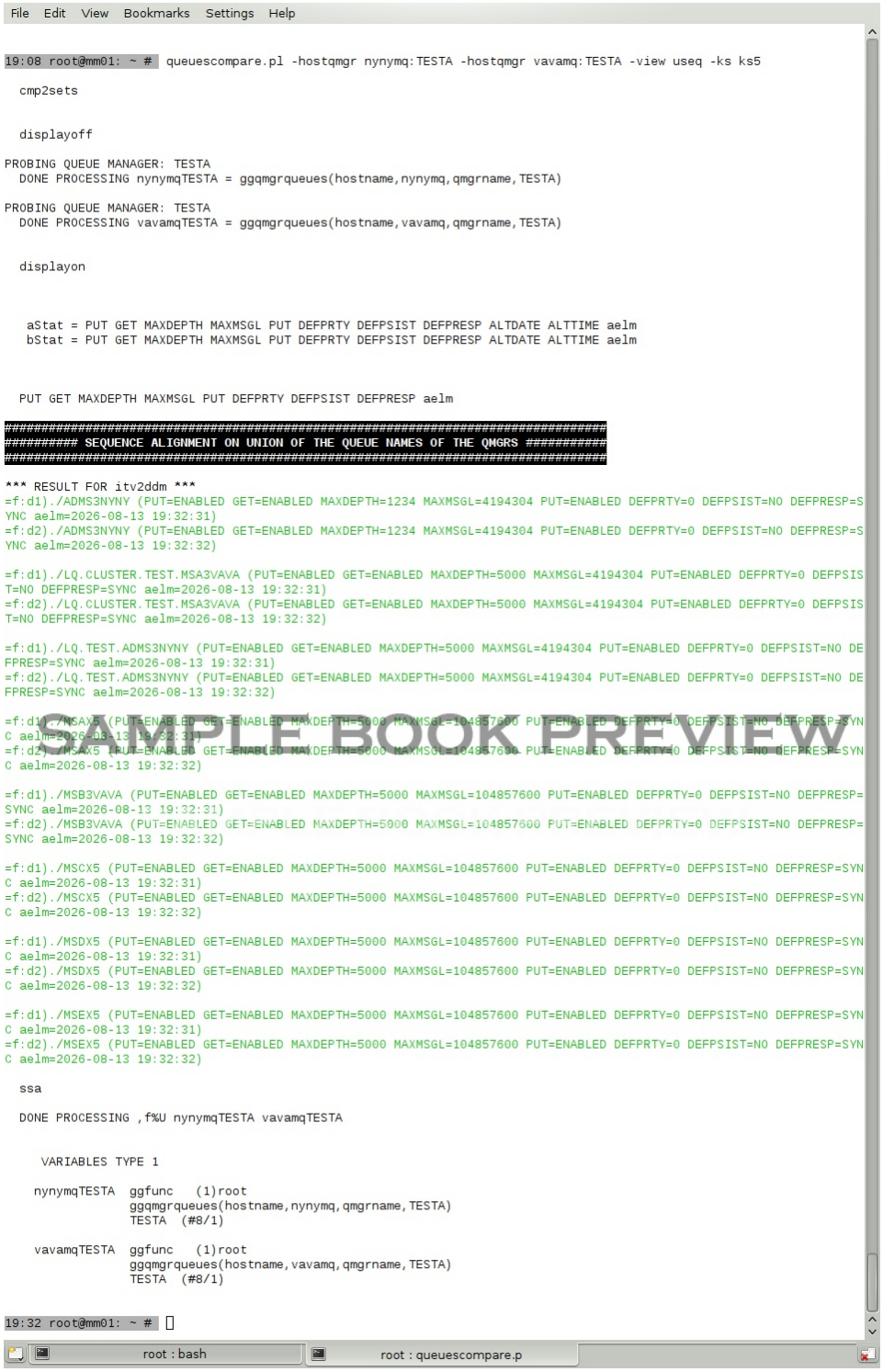


Fig. 1.1.6 [Similarity between the two clustered qmgrs]

ASPLMQ 2026 by Bassem Jamaleddine

ASPL alignment operators expose the exact differences between queue managers. Figure 1.1.7 shows that queue manager TESTA is in perfect alignment for its queue configurations under the ks5 ks-vector.

queuescompare.pl -hostqmgr nynymq:TESTA -hostqmgr vavamq:TESTA -view useq -ks ks5



SAMPLE BOOK PREVIEW

Fig. 1.1.7 [Comparing queues of two qmgrs using alignment]

ASPLMQ 2026 by Bassem Jamaleddine

After applying configuration changes to queue manager TESTA on host vavamq, the system alignment is verified once again. The commands are executed on vavamq by the mqm user: the first command creates the queue ABCDEF, and the second disables GET operations on the LQ.TEST.ADMS3NYNY queue.

(VAVAMQ's qmgr:TESTA) 19:32 mqm@VAVAMQ: ~ # **echo "DEFINE QLOCAL(ABCDEF) REPLACE" | runmqsc TESTA**

```
5724-H72 (C) Copyright IBM Corp. 1994, 2009.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager TESTA.
```

```
     1 : DEFINE  QLOCAL(ABCDEF)  REPLACE
AMQ8006: WebSphere MQ queue created.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

(VAVAMQ's qmgr:TESTA) 19:34 mqm@VAVAMQ: ~ # **echo "ALTER QLOCAL(LQ.TEST.ADMS3NYNY) GET(DISABLED)" | runmqsc TESTA**

```
5724-H72 (C) Copyright IBM Corp. 1994, 2009.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager TESTA.

     1 : ALTER  QLOCAL(LQ.TEST.ADMS3NYNY) GET(DISABLED)
AMQ8008: WebSphere MQ queue changed.
One MQSC command read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

Review the alignment (useq, union sequence) of queues in TESTA queue manager across both hosts by executing:

queuescompare.pl -hostqmgr nynymq:TESTA -hostqmgr vavamq:TESTA -view useq -ks ks5

Figure 1.1.8 reveals the changes, showing that the new ABCDEF queue is out of alignment because it is not defined on host nynymq. Aligned queues with differing attributes are displayed in yellow.

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW



BOOK PREVIEW

Fig. 1.1.8 [Difference in queues of two qmgrs using alignment]

ASPLMQ 2026 by Bassem Jamaleddine

The similarity coefficient falls from 1.0 after the queue manager TESTA attributes are modified on host vavamq

queuescompare.pl -hostqmgr nynymq:TESTA -hostqmgr vavamq:TESTA -view simz -ks ks5

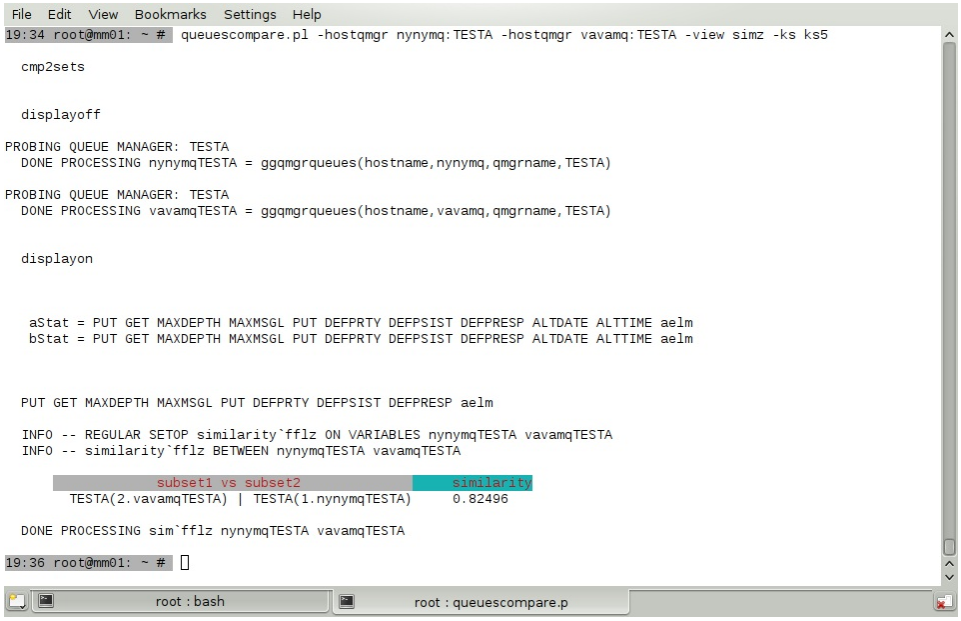


Fig. 1.1.9 [Similarity between the same named qmgr TESTA installed on two different hosts]

ASPLMQ 2026 by Bassem Jamaleddine

SAMPLE BOOK PREVIEW

The ASPL group union operator exposes the differences between queue manager configurations. Figure 1.1.10 shows the element union filtered by the ks5 ks-vector. The ABCDEF queue appears only in the vavamqTESTA set variable, and the LQ.TEST.ADMS3NYNY GET attribute is highlighted in red because its state differs across the variables.

queuescompare.pl -hostqmgr nynymq:TESTA -hostqmgr vavamq:TESTA -view funion -ks ks5



SAMPLE BOOK PREVIEW

Fig. 1.1.10 [Group union of the named qmgr TESTA installed on two different hosts]

ASPLMQ 2026 by Bassem Jamaleddine

The -report option can be added to output a tabulated report showing the changes in the TESTA queue managers.

queuescompare.pl -hostqmgr nynymq:TESTA -hostqmgr vavamq:TESTA -view funion -ks ks5 -report

Figure 1.1.11 shows the report generated for the changes in queue manager TESTA. The chapter ASPL MQ Writers explains how to code custom MQ writers in Perl and place them in the ASPL writers directory to print customized reports.

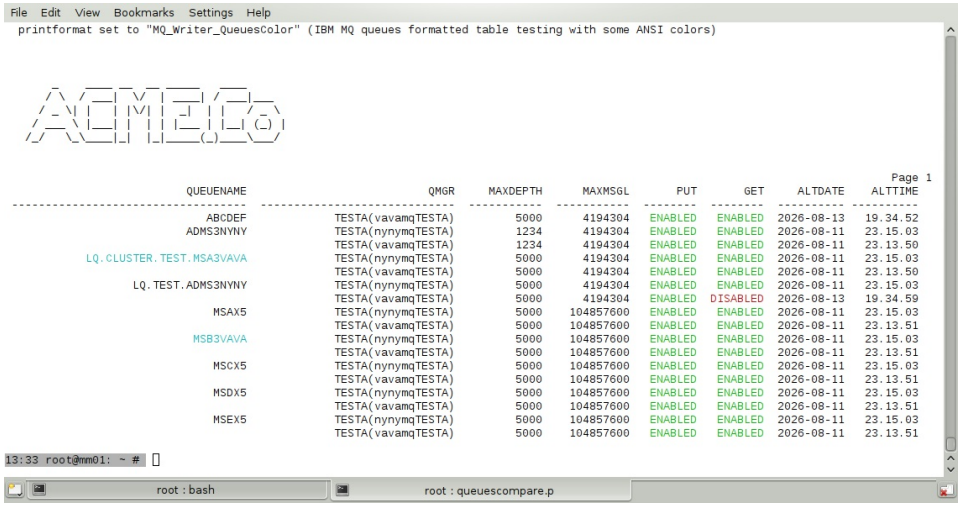


Fig. 1.1.11 [Example of report ouput by an MQ writer when comparing TESTA configured on two different hosts]

ASPLMQ 2026 by Bassem Jamaleddine

To wrap up this introduction, the following paragraph illustrates how ASPL sends email notifications when a group configuration change is detected.
Monitoring a group represented by an ASPL set variable is simple. An an example to demonstrate that ASPL set variables are self monitoring variables, and a group represented by an ASPL set variable can be simply monitored by toggling the variable's monitoring bits.

SAMPLE BOOK PREVIEW

Tracking a group represented by an ASPL set variable is simple. ASPL set variables are self-monitoring variables, meaning a group can be tracked by toggling the variable's monitoring bits. Each ASPL variable contains eight bits that are toggled to enable tracking for specific days of the week. Figure 1.1.13 shows the toggled monitoring bits for the nyvatesta set variable in the namedspace CLUSTERED_TESTA_QUEUES.

To trigger automated email notifications for the nyvatesta monitoring variable, execute the ASPL varchangenotify command at the interpreter prompt followed by the variable name and target notification group: **varchangenotify nyvatesta emailgroup1**. The emailgroup1 tag identifies the distribution list in the ASPL emailsfile. Edit this configuration file by running the following command at the shell prompt:

asplcmd ~emailsfile

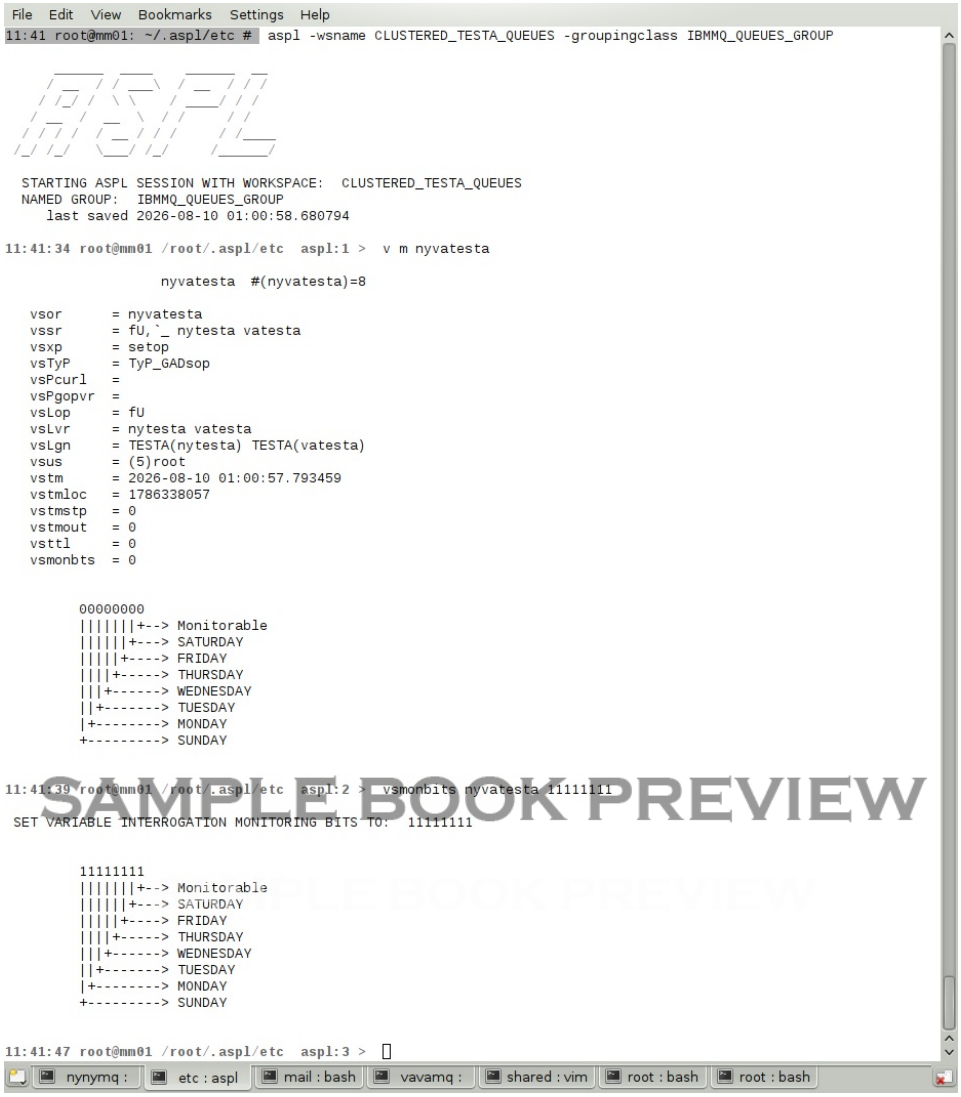
Figure 1.1.12 shows how the command opened the emailsfile in the vi editor. Figure 1.1.12 displays the emailsfile opened in the vi editor.



Fig. 1.1.12 [Editing ASPL ~emailsfile]

ASPLMQ 2026 by Bassem Jamaleddine

An email notification is sent to recipients when a monitoring variable detects a change in the group. Figure 1.1.14 displays the email sent due to the alterations made within queue manager TESTA.

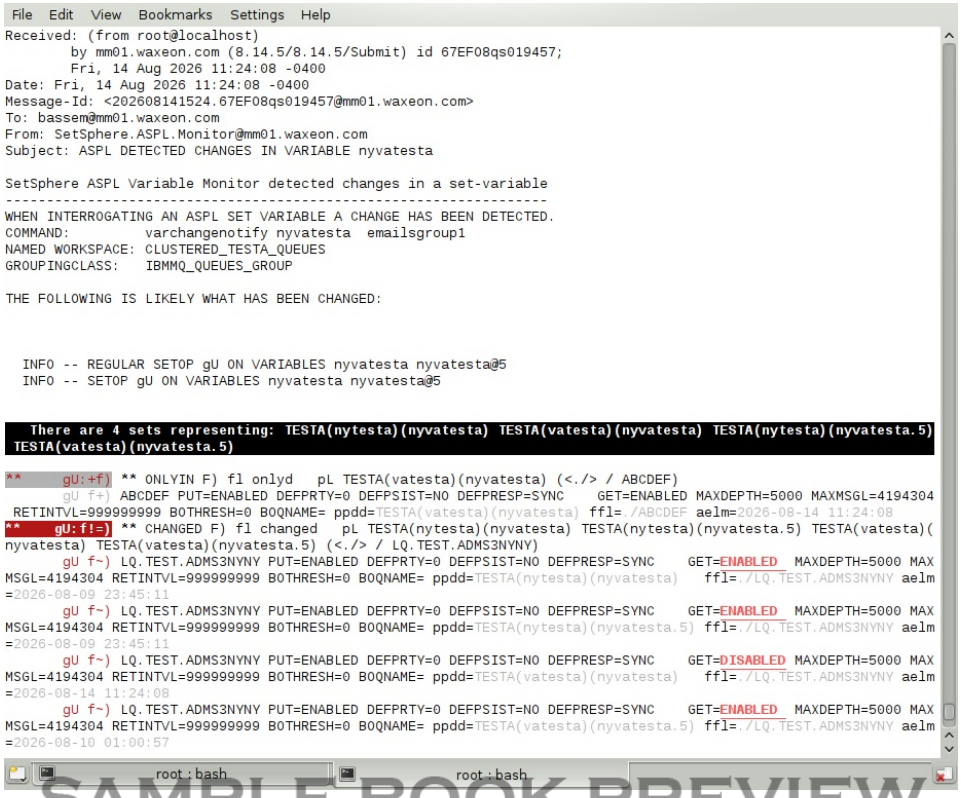


SAMPLE BOOK PREVIEW

Fig. 1.1.13 [ASPL set variables are self-monitoring variable]

ASPLMQ 2026 by Bassem Jamaleddine

ASPL allows groups represented by set variables to be compared with one another. Additionally, a set variable can be evaluated against its own historical states over time because ASPL set variables operate as differential group variables. Refer to the ASPL User Guide for further details on these variable structures.



SAMPLE BOOK PREVIEW

Fig. 1.1.14 [Example of the email notification sent due to change in TESTA configured on two different hosts]

SAMPLE BOOK PREVIEW

ASPLMQ 2026 by Bassem Jamaleddine

This text functions as a case study for using ASPL and Perl to manage middleware configurations in IBM MQ environments. The following sections explain how to parse IBM MQ objects using Perl to identify queues and channels, alongside short scripts for filtering and printing tabulated reports.

1-2 Parsing IBM MQ Queue Objects

Since IBM MQ queue objects are regularly structured, they can be parsed efficiently using regular expressions. Filtering these queue objects is achieved with the Perl code shown in Figure 1.2.1.

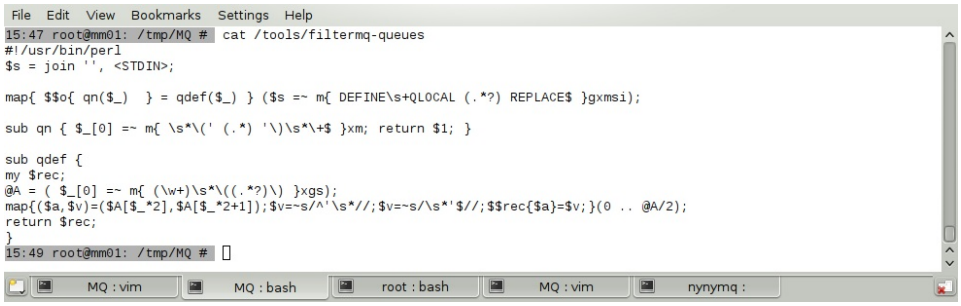


Fig. 1.2.1 [Filtering IBM MQ queues objects]

ASPLMQ 2026 by Bassem Jamaleddine

SAMPLE BOOK PREVIEW

The program reads STDIN into the scalar variable $s and splits the string into segments that start with DEFINE QLOCAL and end with REPLACE, converting each into an object reference $o. The object is constructed using the Perl map function, extracting each queue name with qn(). The bulk of the parsing work is done within this map statement; reading the expression from right to left details its functional transformation steps:

```
map{ $$o{ qn($_)  } = qdef($_) } ($s =~ m{ DEFINE\s+QLOCAL (.*?) REPLACE$ }gxmsi);

                                        Break the string into pieces where
                                        each piece starts with DEFINE QLOCAL
                                        and end with REPLACE.
                  Call qdef() on each piece
                  mapping a left attribue to
                  its right value
Aggregate each derived
queue name qn() into the
object $o and make the
$$o{ qn($_) points to its
derived structure
```

Listing 1.2.1 shows the filtermq-queues script. For clarity, this version spreads the regular expressions across multiple lines and expands the map function inside the qdef() routine into an explicit for loop.

-LIS- Listing. 1.2.1 [Listing of filtermq-queues]

```
1.      #!/usr/bin/perl
2.      use strict;
3.      use Data::Dumper;
4.
5.      my $s = join '', <STDIN>;
6.
7.      my(@QL) = ($s =~
8.              m{
9.                  DEFINE\s+QLOCAL
10.                     (.*?)
11.                 REPLACE$
12.             }gxmsi
```

```
13.        );
14.
15.    my $o;
16.    map{ $$o{ qn($_)  } = qdef($_) } @QL;
17.    print Dumper($o);
18.
19.    sub qn {
20.    my $s = shift;
21.    $s =~ m{
22.    \s*\('
23.    (.*)
24.    '\)\s*\+$
25.    }xm;
26.    return $1;
27.    }
28.
29.    sub qdef {
30.    my $s = shift;
31.    my $rec;
32.
33.    my (@A) = (
34.        $s =~
35.            m{
36.                 (\w+)\s*\((.*?)\)
37.            }xgs
38.    );
39.
40.    for (my $i=0; $i<@A; $i+=2) {
41.        my ($att,$v)=($A[$i],$A[$i+1]);
42.        $v =~ s/^'\s*//; $v =~ s/\s*'$//;
43.        $$rec{$att} = $v;
44.    }
45.    return $rec;
46.    }
47.
```

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

ASPLMQ 2026 Bassem Jamaleddine

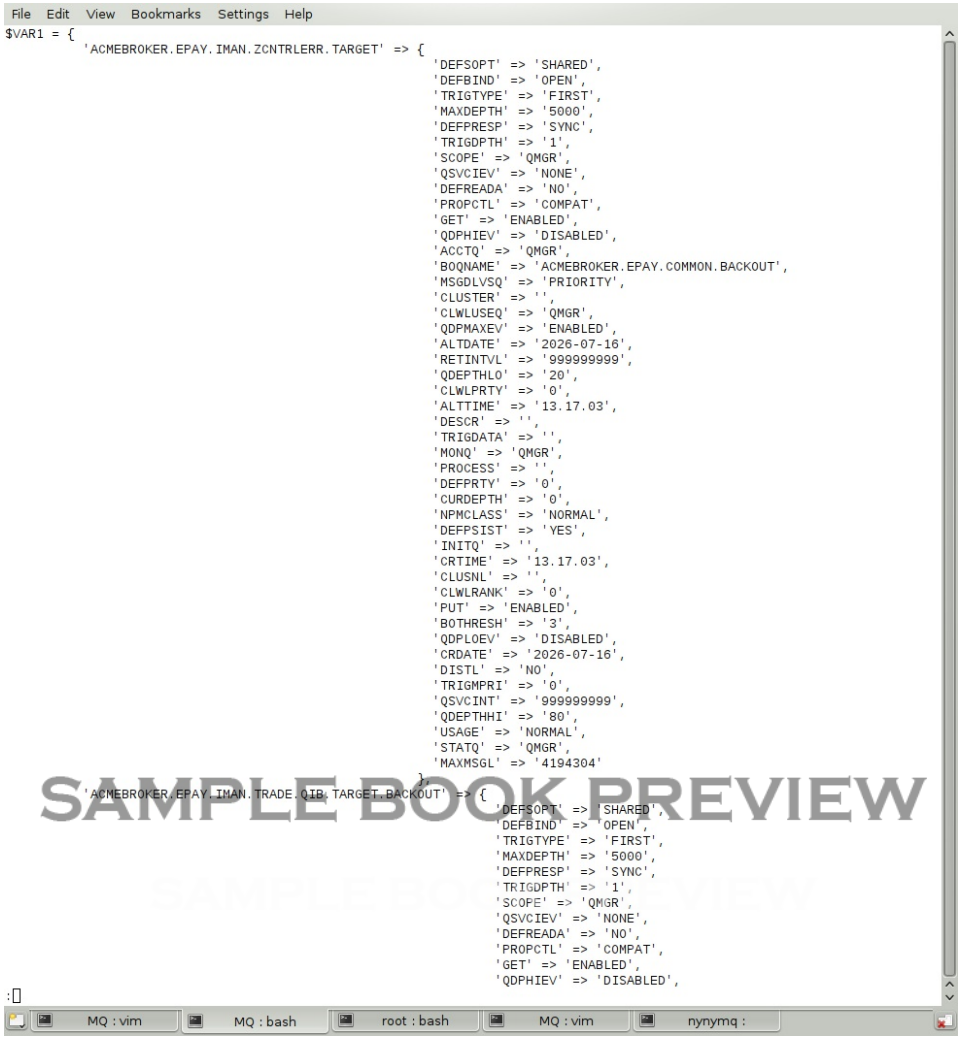
The ACMECO queue manager configuration can be dumped using saveqmgr and piped to filtermq-queues to aggregate the queue objects into the $o object reference.

```
# ssh -q mqm@vavamq /home/mqm/mqtools/saveqmgr -m ACMECO -s -c 2>&1 | filtermq-queues
```

Alternatively, the output of saveqmgr can be saved to a file named ACMECO.saveqmgr. This file can then be read and piped through the filtermq-queues filter script.

```
# cat ACMECO.saveqmgr | filtermq-queues
```

Figure 1.2.2 shows the structured data object generated for the parsed queues.



SAMPLE BOOK PREVIEW

Fig. 1.2.2 [Resulting object due to filtermq-queues]

ASPLMQ 2026 by Bassem Jamaleddine

■ Printing Formatted Reports for IBM MQ Queues

Adding a formatting function allows the script to print the queues in a tabulated report. Listing 1.2.2 shows the updated filtermq-queues script with this function included.

-LIS- Listing. 1.2.2 [Listing of filtermq-queues with formatting function]

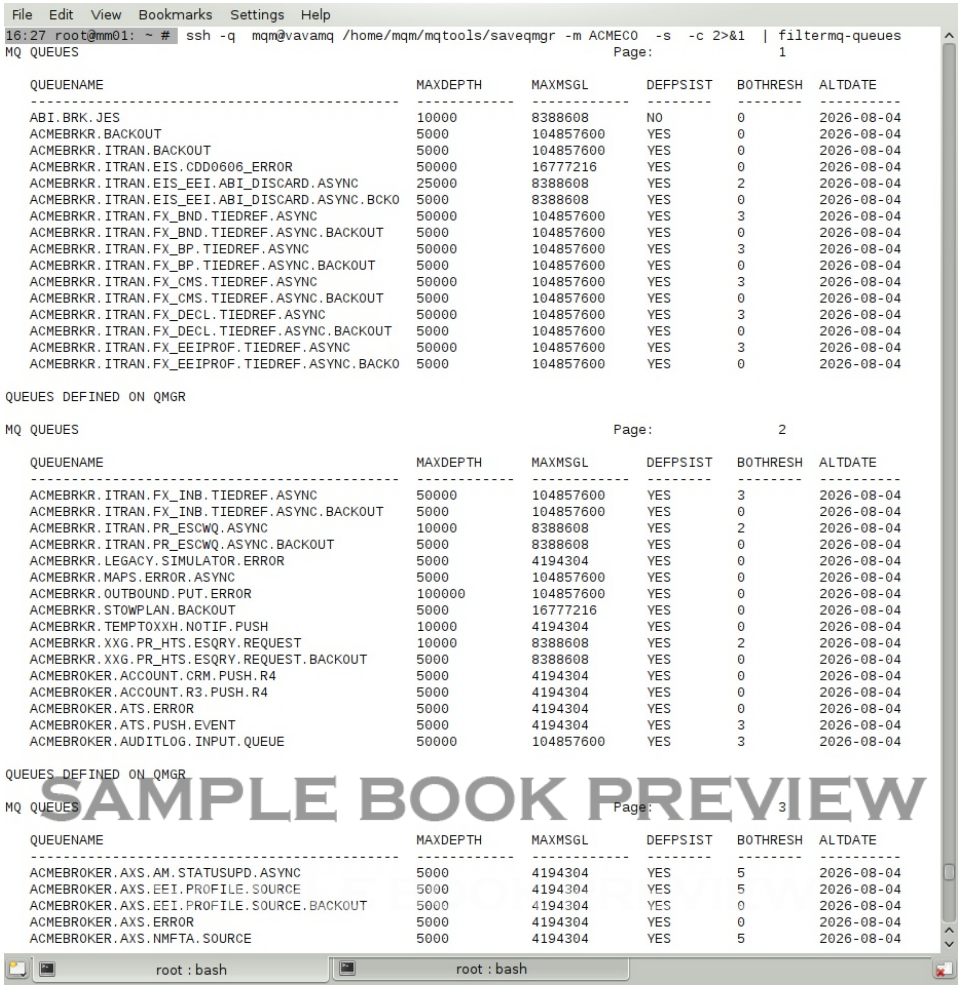
```
1.     #!/usr/bin/perl
2.     use strict;
3.
4.     my $s = join '', <STDIN>;
5.     my(@QL) = ($s =~
6.             m{
7.                 DEFINE\s+QLOCAL
8.                     (.*?)
9.                 REPLACE$
10.            }gxmsi
11.        );
12.
13.    my $o;
14.    map{ $$o{ qn($_)  } = qdef($_) } @QL;
15.    qout($o);
```

```
16.
17.    sub qn {
18.    my $s = shift;
19.    $s =~ m{
20.    \s*\('
21.    (.*)
22.    '\)\s*\+$
23.    }xm;
24.    return $1;
25.    }
26.
27.    sub qdef {
28.    my $s = shift;
29.    my $rec;
30.
31.    my @A = ( $s =~ m{ (\w+)\s*\((.*?)\) }xgs);
32.
33.    for (my $i=0; $i<@A; $i+=2) {
34.        my ($att,$v)=($A[$i],$A[$i+1]);
35.        $v =~ s/^'//; $v =~ s/'$//;
36.        $v =~ s/^\s+//; $v =~ s/\s+$//;
37.        $$rec{$att} = $v;
38.    }
39.    return $rec;
40.    }
41.
42.    ####################
43.    sub qout {
44.    my ($o) = @_;
45.
     my($qn,$maxdepth,$maxmsgl,$defpsist,$bothresh,$boqname,$cluster,$crdate,$altdate);
46.    format STDOUT_TOP =
47.    MQ QUEUES
       Page: @>>>>>>>>>>>>>>>
48.          $%
49.
50.       QUEUENAME                        MAXDEPTH      MAXMSGL       DEFPSIST
        BOTHRESH  CRTDATE     ALTDATE
51.       --------------------------------  ------------  ------------  --------
        --------  ----------  ----------
52.    .
53.    format STDOUT =
54.       @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<  @<<<<<<<<<<<  @<<<<<<<<<<<
      @<<<<<<<<  @<<<<<<<  @<<<<<<<<<  @<<<<<<<<<
55.    $qn,$maxdepth,$maxmsgl,$defpsist,$bothresh,$crdate,$altdate
56.    .
57.    $== 20;
58.    $-= 0;
59.    $^L= "\nQUEUES DEFINED ON QMGR\n\n";
60.    foreach my $qname (sort keys %$o) {
61.        $qn = $qname;
62.        ($maxdepth,$maxmsgl,$defpsist,$bothresh,$boqname,$crdate,$altdate) =
63.            map($$o{$qn}{$_}, qw(MAXDEPTH  MAXMSGL  DEFPSIST  BOTHRESH
      BOQNAME  CRDATE  ALTDATE));
64.        write;
65.    }
66.    $- = 0;
67.    }
```

SAMPLE BOOK PREVIEW

ASPLMQ 2026 Bassem Jamaleddine

The resulting tabulated report displaying the queue configurations is illustrated in Figure 1.2.3.



```
File  Edit  View  Bookmarks  Settings  Help
16:27 root@mm01: ~ # ssh -q  mqm@vavamq /home/mqm/mqtools/saveqmgr -m ACMECO  -s  -c 2>&1  | filtermq-queues
MQ QUEUES                                                    Page:              1

   QUEUENAME                                    MAXDEPTH     MAXMSGL      DEFPSIST   BOTHRESH  ALTDATE
   -------------------------------------------  -----------  -----------  --------   --------  ----------
   ABI.BRK.JES                                  10000        8388608      NO         0         2026-08-04
   ACMEBRKR.BACKOUT                             5000         104857600    YES        0         2026-08-04
   ACMEBRKR.ITRAN.BACKOUT                       5000         104857600    YES        0         2026-08-04
   ACMEBRKR.ITRAN.EIS.CDD0606_ERROR             50000        16777216     YES        0         2026-08-04
   ACMEBRKR.ITRAN.EIS_EEI.ABI_DISCARD.ASYNC     25000        8388608      YES        2         2026-08-04
   ACMEBRKR.ITRAN.EIS_EEI.ABI_DISCARD.ASYNC.BCKO 5000        8388608      YES        0         2026-08-04
   ACMEBRKR.ITRAN.FX_BND.TIEDREF.ASYNC          50000        104857600    YES        3         2026-08-04
   ACMEBRKR.ITRAN.FX_BND.TIEDREF.ASYNC.BACKOUT  5000         104857600    YES        0         2026-08-04
   ACMEBRKR.ITRAN.FX_BP.TIEDREF.ASYNC           50000        104857600    YES        3         2026-08-04
   ACMEBRKR.ITRAN.FX_BP.TIEDREF.ASYNC.BACKOUT   5000         104857600    YES        0         2026-08-04
   ACMEBRKR.ITRAN.FX_CMS.TIEDREF.ASYNC          50000        104857600    YES        3         2026-08-04
   ACMEBRKR.ITRAN.FX_CMS.TIEDREF.ASYNC.BACKOUT  5000         104857600    YES        0         2026-08-04
   ACMEBRKR.ITRAN.FX_DECL.TIEDREF.ASYNC         50000        104857600    YES        3         2026-08-04
   ACMEBRKR.ITRAN.FX_DECL.TIEDREF.ASYNC.BACKOUT 5000         104857600    YES        0         2026-08-04
   ACMEBRKR.ITRAN.FX_EEIPROF.TIEDREF.ASYNC      50000        104857600    YES        3         2026-08-04
   ACMEBRKR.ITRAN.FX_EEIPROF.TIEDREF.ASYNC.BACKO 5000        104857600    YES        0         2026-08-04

QUEUES DEFINED ON QMGR

MQ QUEUES                                                    Page:              2

   QUEUENAME                                    MAXDEPTH     MAXMSGL      DEFPSIST   BOTHRESH  ALTDATE
   -------------------------------------------  -----------  -----------  --------   --------  ----------
   ACMEBRKR.ITRAN.FX_INB.TIEDREF.ASYNC          50000        104857600    YES        3         2026-08-04
   ACMEBRKR.ITRAN.FX_INB.TIEDREF.ASYNC.BACKOUT  5000         104857600    YES        0         2026-08-04
   ACMEBRKR.ITRAN.PR_ESCWQ.ASYNC                10000        8388608      YES        2         2026-08-04
   ACMEBRKR.ITRAN.PR_ESCWQ.ASYNC.BACKOUT        5000         8388608      YES        0         2026-08-04
   ACMEBRKR.LEGACY.SIMULATOR.ERROR              5000         4194304      YES        0         2026-08-04
   ACMEBRKR.MAPS.ERROR.ASYNC                    5000         104857600    YES        0         2026-08-04
   ACMEBRKR.OUTBOUND.PUT.ERROR                  100000       104857600    YES        0         2026-08-04
   ACMEBRKR.STOWPLAN.BACKOUT                    5000         16777216     YES        0         2026-08-04
   ACMEBRKR.TEMPTOXXH.NOTIF.PUSH                10000        4194304      YES        0         2026-08-04
   ACMEBRKR.XXG.PR_HTS.ESQRY.REQUEST            10000        8388608      YES        2         2026-08-04
   ACMEBRKR.XXG.PR_HTS.ESQRY.REQUEST.BACKOUT    5000         8388608      YES        0         2026-08-04
   ACMEBROKER.ACCOUNT.CRM.PUSH.R4               5000         4194304      YES        0         2026-08-04
   ACMEBROKER.ACCOUNT.R3.PUSH.R4                5000         4194304      YES        0         2026-08-04
   ACMEBROKER.ATS.ERROR                         5000         4194304      YES        0         2026-08-04
   ACMEBROKER.ATS.PUSH.EVENT                    5000         4194304      YES        3         2026-08-04
   ACMEBROKER.AUDITLOG.INPUT.QUEUE              50000        104857600    YES        3         2026-08-04

QUEUES DEFINED ON QMGR

MQ QUEUES                                                    Page:              3

   QUEUENAME                                    MAXDEPTH     MAXMSGL      DEFPSIST   BOTHRESH  ALTDATE
   -------------------------------------------  -----------  -----------  --------   --------  ----------
   ACMEBROKER.AXS.AM.STATUSUPD.ASYNC            5000         4194304      YES        5         2026-08-04
   ACMEBROKER.AXS.EEI.PROFILE.SOURCE            5000         4194304      YES        5         2026-08-04
   ACMEBROKER.AXS.EEI.PROFILE.SOURCE.BACKOUT    5000         4194304      YES        0         2026-08-04
   ACMEBROKER.AXS.ERROR                         5000         4194304      YES        0         2026-08-04
   ACMEBROKER.AXS.NMFTA.SOURCE                  5000         4194304      YES        5         2026-08-04
root : bash   root : bash
```

SAMPLE BOOK PREVIEW

Fig. 1.2.3 [Resulting object due to filtermq-queues with formatting]

ASPLMQ 2026 by Bassem Jamaleddine

1-3 Parsing IBM MQ Channel Objects

Similar to IBM MQ queue configurations, channel objects share a regular structure that can be parsed efficiently using regular expressions. Filtering these channel objects is achieved with the Perl code shown in Figure 1.3.1.

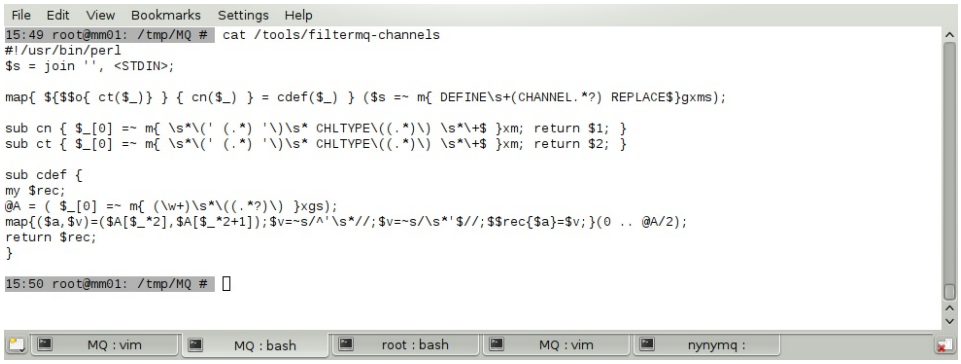


Fig. 1.3.1 [Filtering IBM MQ channels objects]

SAMPLE BOOK PREVIEW

ASPLMQ 2026 by Bassem Jamaleddine

The program reads STDIN into the scalar variable $s and splits the string into segments that start with DEFINE CHANNEL and end with REPLACE, converting each into an object reference $o. The object is constructed using the Perl map function, extracting each channel name with cn(). The bulk of the parsing work is done within this map statement; reading the expression from x right to left details its functional transformation steps:

```
map{ ${$$o{ ct($_)}} {cn($_)} = cdef($_) } ($s =~ m{ DEFINE\s+(CHANNEL.*?) REPLACE$}gxms);

                                          Break the string into pieces where
                                          each piece starts with DEFINE CHANNEL
                                          and end with REPLACE.
                       Call cdef() on each piece
                       mapping a left attribue to
                       its right value
Aggregate each derived
channel type ct() and
channel name cn() into the
object $o and make the
${$$o{ ct($_)}} {cn($_)}
points to its derived structure
```

Listing 1.3.1 shows the filtermq-channels script. For clarity, this version spreads the regular expressions across multiple lines and expands the map function inside the cdef() routine into an explicit for loop.

-LIS- Listing. 1.3.1 [Listing of filtermq-channels]

```
1.     #!/usr/bin/perl
2.     use strict;
3.
4.     my $s = join '', <STDIN>;
5.
6.     my(@QC) = ($s =~
7.             m{
8.                 DEFINE\s+(CHANNEL.*?)
9.                 REPLACE$
```

```
10.            }gxms
11.            );
12.
13.    my $o;
14.    map { ${ $$o{ ct($_) } } { cn($_) } =  cdef($_) } @QC;
15.
16.    sub cn {
17.    my $s = shift;
18.    $s =~ m{
19.    \s*\('
20.    (.*)
21.    '\)\s*
22.    CHLTYPE\((.*)\)
23.    \s*\+$
24.    }xm;
25.    return $1;
26.    }
27.
28.    sub ct {
29.    my $s = shift;
30.    $s =~ m{
31.    \s*\('
32.    (.*)
33.    '\)\s*
34.    CHLTYPE\((.*)\)
35.    \s*\+$
36.    }xm;
37.    return $2;
38.    }
39.
40.
41.    sub cdef {
42.    my $s = shift;
43.    my $rec;
44.
45.    my (@A) = (
46.        $s =~
47.            m{
48.                 (\w+)\s*\((.*?)\)
49.            }xgs
50.    );
51.
52.    for (my $i=0; $i<@A; $i+=2) {
53.        my ($att,$v)=($A[$i],$A[$i+1]);
54.        $v =~ s/^'\s*//; $v =~ s/\s*'$//;
55.        $$rec{$att} = $v;
56.    }
57.    return $rec;
58.    }
59.
```

SAMPLE BOOK PREVIEW

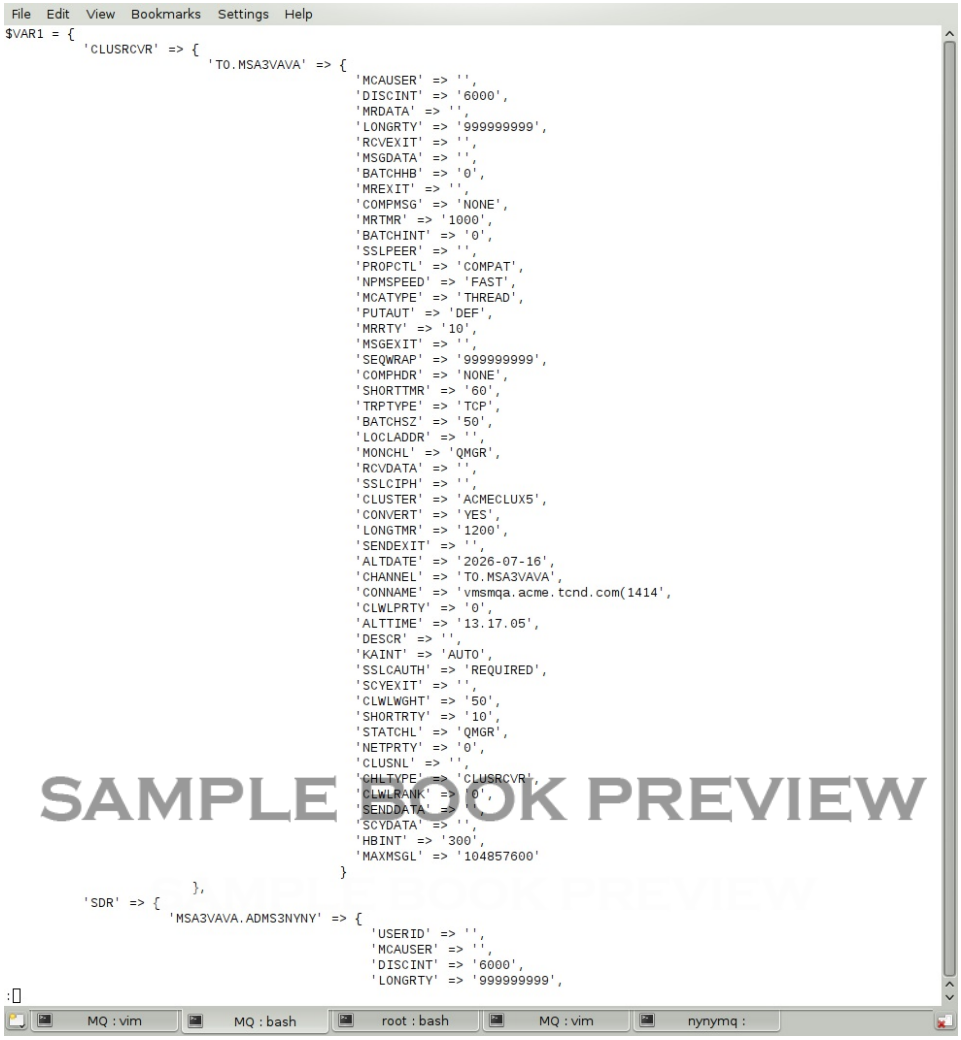
The ACMECO queue manager configuration can be dumped using saveqmgr and piped to filtermq-channels to aggregate the channel objects into the $o object reference.

```
# ssh -q mqm@vavamq /home/mqm/mqtools/saveqmgr -m ACMECO -s -c 2>&1 | filtermq-channels
```

Alternatively, the output of saveqmgr can be saved to a file named ACMECO.saveqmgr. This file can then be read and piped through the filtermq-channels filter script.

```
# cat ACMECO.saveqmgr | filtermq-channels
```

Figure 1.3.2 shows the structured data object generated for the parsed channels.



SAMPLE BOOK PREVIEW

Fig. 1.3.2 [Resulting object due to filtermq-channels]

ASPLMQ 2026 by Bassem Jamaleddine

■ Printing Formatted Reports for IBM MQ Channels

Adding a formatting function allows the script to print the channels in a tabulated report. Listing 1.3.2 shows the updated filtermq-channels script with this function included.

-LIS- Listing. 1.3.2 [Listing of filtermq-channels with formatting function]

```
1.     #!/usr/bin/perl
2.     use strict;
3.
4.     my $s = join '', <STDIN>;
5.
6.     my(@QC) = ($s =~
7.             m{
8.                 DEFINE\s+(CHANNEL.*?)
9.                 REPLACE$
10.            }gxms
11.            );
12.    my $o;
13.    map { ${ $$o{ ct($_) } } { cn($_) } =  cdef($_) } @QC;
14.
15.    chout($o);
16.
17.    sub cn {
```

```
18.    my $s = shift;
19.    $s =~ m{
20.    \s*\('
21.    (.*)
22.    '\)\s*
23.    CHLTYPE\((.*)\)
24.    \s*\+$
25.    }xm;
26.    return $1;
27.    }
28.
29.    sub ct {
30.    my $s = shift;
31.    $s =~ m{
32.    \s*\('
33.    (.*)
34.    '\)\s*
35.    CHLTYPE\((.*)\)
36.    \s*\+$
37.    }xm;
38.    return $2;
39.    }
40.
41.    sub cdef {
42.    my $s = shift;
43.    my $rec;
44.
45.    my (@A) = (
46.        $s =~
47.            m{
48.                (\w+)\s*\((.*?)\)
49.            }xgs
50.    );
51.
52.    for (my $i=0; $i<@A; $i+=2) {
53.        my ($att,$v)=($A[$i],$A[$i+1]);
54.        $v =~ s/^'\s*//; $v =~ s/\s*'$//;
55.        $$rec{$att} = $v;
56.    }
57.    return $rec;
58.    }
59.
60.    ################################
61.    sub chout {
62.    my ($chrec) = @_;
63.    my($CHANNEL,$CHLTYPE,$MAXMSGL,$ALTDATE,$ALTTIME,$DESCRIPTION);
64.    format STDOUT_TOP =
65.    MQ CHANNELS
           Page: @>>>>>>>>>>>>>>
66.
           $%
67.
68.       CHANNEL                      CHLTYPE      MAXMSGL     ALTDATE    ALTTIME
       DESCRIPTION
69.       --------------------------  -----------  ----------  ---------- --------
       ----------------------------------------
70.    .
71.    format STDOUT =
72.       @<<<<<<<<<<<<<<<<<<<<<<<<<  @<<<<<<<<<<  @<<<<<<<<<  @<<<<<<<<<
       @<<<<<<<< @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
73.    $CHANNEL,$CHLTYPE,$MAXMSGL,$ALTDATE,$ALTTIME,$DESCRIPTION
74.    .
75.
76.    $== 30;
77.    $-= 0;
78.    $^L= "\nCHANNELS DEFINED ON QMGR\n\n";
79.
80.
81.    foreach my $ct (sort keys %$chrec) {
82.        foreach my $cn (sort keys %{$chrec->{$ct}}) {
83.            $CHANNEL = $cn;
84.            ($CHANNEL, $CHLTYPE,$MAXMSGL, $ALTDATE, $ALTTIME, $DESCRIPTION) =
85.                map(${${$$chrec{$ct}}{$cn}}{$_}, qw(CHANNEL CHLTYPE MAXMSGL
       ALTDATE  ALTTIME DESCR));
```

SAMPLE BOOK PREVIEW

```
86.             $DESCRIPTION //= "";
87.             write;
88.         }
89.     }
90.     $- = 0;
91.     }
```

ASPLMQ 2026 Bassem Jamaleddine

The resulting tabulated report displaying the channel configurations is illustrated in Figure 1.3.3.

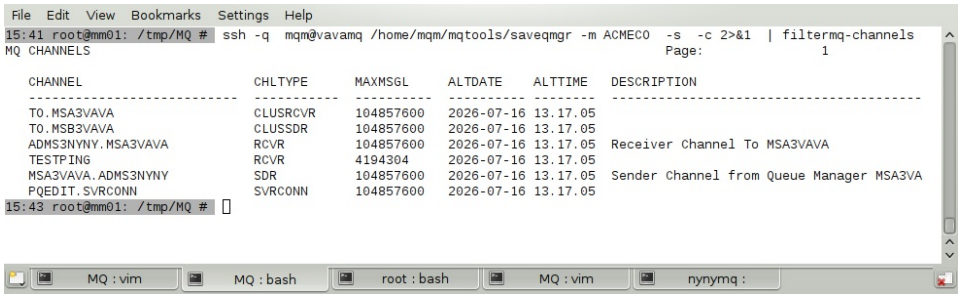


Fig. 1.3.3 [Resulting object due to filtermq-queues with formatting]

ASPLMQ 2026 by Bassem Jamaleddine

SAMPLE BOOK PREVIEW

Parsing IBM MQ queue and channel configurations is handled by the MQparser.pm Perl module. To monitor modifications within an IBM MQ topology, the ASPL interpreter loads MQutils::MQparser as part of its library. Refer to the appendix **The module MQutils::MQparser to Parse IBM MQ Objects** for the full source file.

SAMPLE BOOK PREVIEW

```
#!/usr/bin/env aspl
#ENVARG= -wsname TRANSIENT -
```

SAMPLE BOOK PREVIEW

Chapter 2
Group Operations on IBM MQ Objects with ASPL

Scripting with ASPL to detect changes in IBM MQ queue managers and their objects deployed over distributed systems in a farm requires the creation of three element grouping classes: IBMMQ_QMGRS_GROUP, IBMMQ_QUEUES_GROUP, and IBMMQ_CHANNELS_GROUP. An ASPL workspace has its metadata defined per element grouping class; this metadata configuration applies to all symbols stored within that workspace's symbol table. Once the element grouping classes are created, multiple workspaces can be initialized per class to aggregate IBM MQ objects into set variables and save them to the symbol table. The initial three sections of this chapter detail the creation of the aforementioned element grouping classes and global grouping functions. These sections also introduce utilities to display and compare queue managers, queues, and channels. The fourth section addresses approximate matching of group elements on IBM MQ channels - a topic executed using ASPL shallow matching operations.

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

2-1 Grouping Class for IBM MQ Qmgr Attributes

This section explains how to use ASPL to detect changes in IBM MQ queue manager attributes. The scripting provided here enables visual comparisons that contrast attribute modifications, whether they occur within the same queue manager or across distributed queue managers in a cloud environment. ASPL scripting requires the following prerequisites:

- Define the Grouping Class: Create the IBMMQ_QMGR_GROUP element grouping class to describe the queue manager attributes.
- Code the Global Grouping Function: Code the ggibmqmgrs() global grouping function to return the GADg pointer to the queue manager group (with one queue manager per group).
- Create the Workspace: Establish the ACME_QMGRS workspace to save the set variables pointing to the baseline grouped queue managers.
- Develop the Main Tracking Script: Code the acme_qmgrs.aspl script to interrogate identifiers representing queue manager configurations, and expose configuration drift across the cluster.
- Prepare the Initial Set Variables: Configure the initial set variables within the acme_qmgrs.aspl script; these parameters are interrogated as differential group variables to reveal changes across the queue managers.

Creating the element grouping class IBMMQ_QMGR_GROUP

SAMPLE BOOK PREVIEW

The following metagrouping command creates the IBMMQ_QMGR_GROUP element grouping class within ASPL egC-Container.

```
11:08 root@mm01: ~ # metagrouping -creategrouping IBMMQ_QMGR_GROUP -displayphrase "GROUP
OF ATTRIBUTES FOR IBM MQ QMGR" -attributes "PSRTYCNT,i" "LOCALEV,s" "LOGGEREV,s"
"SCHINIT,s" "PSSYNCPT,s" "PSMODE,s" "MAXUMSGS,i" "TREELIFE,i" "SSLCRLNL,s" "ACCTCONO,s"
"ACTIVREC,s" "MONCHL,s" "ACCTQ,s" "REPOSNL,s" "MAXPRTY,i" "MONACLS,s" "CHADEXIT,s"
"CONFIGEV,s" "CHADEV,s" "ALTTIME,s" "ACCTMQI,s" "STATACLS,s" "SSLRKEYC,i" "SSLFIPS,s"
"MAXPROPL,s" "MARKINT,i" "CLWLLEN,i" "STATINT,i" "CRTIME,s" "STATCHL,s" "CLWLEXIT,s"
"INHIBTEV,s" "CHAD,s" "CRDATE,s" "PLATFORM,s" "CHLEV,s" "DISTL,s" "CMDEV,s" "STATQ,s"
"MAXMSGL,i" "STATMQI,s" "ACCTINT,i" "CLWLDATA,s" "DEADQ,s" "CCSID,i" "STRSTPEV,s"
"CMDLEVEL,i" "PSNPRES,s" "SSLCRYP,s" "SSLEV,s" "QMNAME,s" "CLWLUSEQ,s" "TRIGINT,i"
"ALTDATE,s" "REPOS,s" "QMID,s" "DESCR,s" "REMOTEEV,s" "MONQ,s" "COMMANDQ,s" "PARENT,s"
"MAXHANDS,i" "SCMDSERV,s" "IPADDRV,s" "PERFMEV,s" "ROUTEREC,s" "SSLKEYR,s"
"CLWLMRUC,i" "AUTHOREV,s" "DEFXMITQ,s" "PSNPMSG,s"
```

Figure 2.1.1 shows the execution of metagrouping for the element grouping class IBMMQ_QMGR_GROUP.

SAMPLE BOOK PREVIEW

2-2 Grouping Class for IBM MQ Queue Attributes

This section explains how to use ASPL to detect changes in IBM MQ queue attributes. The scripting provided here enables visual comparisons that contrast queue attribute modifications, whether they occur within the same queue manager or across distributed queue managers in a cloud environment.

ASPL scripting requires the following prerequisites:
- Define the Element Grouping Class: Create the IBMMQ_QUEUES_GROUP class to describe the queue attributes.
- Code the Global Grouping Function: Code the ggqmgrqueues() function to return the GADg pointer to a queue manager's queue group.
- Create the Workspace: Establish the ACME_QUEUES workspace to save the set variables pointing to the baseline grouped queue managers.
- Prepare the Initial Set Variables: Configure the initial set variables in the acme_queues.aspl script; these parameters are interrogated as differential group variables to expose configuration changes across the queue managers.

Creating the element grouping class IBMMQ_QUEUES_GROUP

The following metagrouping command creates the IBMMQ_QUEUES_GROUP element grouping class within ASPL egC-Container.

SAMPLE BOOK PREVIEW

```
17:50 root@mm01: ~ # metagrouping -creategrouping IBMMQ_QUEUES_GROUP -displayphrase
"GROUPING CLASS FOR QUEUES ATTRIBUTES OF IBM MQ QUEUES" -attributes "DESCR,s" "PUT,s"
"DEFPRTY,i" "DEFPSIST,s" "DEFPRESP,s" "DEFREADA,s" "PROPCTL,s" "LIKE,s" "CLWLUSEQ,s"
"SCOPE,s" "GET,s" "MAXDEPTH,i" "MAXMSGL,i" "DEFSOPT,s" "MSGDLVSQ,s" "USAGE,s"
"TRIGTYPE,s" "TRIGDPTH,i" "TRIGMPRI,i" "TRIGDATA,s" "PROCESS,s" "INITQ,s" "RETINTVL,i"
"BOTHRESH,i" "BOQNAME,s" "QDEPTHHI,i" "QDEPTHLO,i" "QDPMAXEV,s" "QDPHIEV,s" "QDPLOEV,s"
"QSVCINT,i" "QSVCIEV,s" "DISTL,s" "NPMCLASS,s" "STATQ,s" "MONQ,s" "ACCTQ,s" "CLUSTER,s"
"CLUSNL,s" "DEFBIND,s" "CLWLRANK,i" "CLWLPRTY,i" "TARGET,s" "ALTDATE,s" "ALTTIME,s" -
heterogeneousrecords
```

The command creates the element grouping class IBMMQ_QUEUES_GROUP in ASPL egC-Container.
Make sure to run the following substitution command on the newly created IBMMQ_QUEUES_GROUP.ELMD file.

```
11:02 root@mm01: ~ # sed -i s/default=ABC/default=/ `asplcmd
~elmddir`/IBMMQ_QUEUES_GROUP.ELMD
```

Edit the file to apply the required modifications to the ks-vector mappings.

```
# vi `asplcmd ~elmddir`/IBMMQ_QUEUES_GROUP.ELMD
```

These mapping vectors utilize short mnemonics, such as ks1 and ks2, to define the specific list of attributes to evaluate. All designated attributes must belong to the EnodeBaseAttributes set, which is specified via the -attributes option in the metagrouping command. Refer to the ASPL User Guide for instructions on defining numbered ks-vectors within ELMD files. Figure 2.2.1 shows the metagrouping execution used to define the IBMMQ_QUEUES_GROUP element grouping class.

SAMPLE BOOK PREVIEW

2-3 Grouping Class for IBM MQ Channel Attributes

This section explains how to use ASPL to detect changes in IBM MQ channel attributes. The scripting provided here enables visual comparisons that contrast channel attribute modifications, whether they occur within the same queue manager or across distributed queue managers in a cloud environment.

ASPL scripting requires the following prerequisites:
- Define the Element Grouping Class: Create the IBMMQ_CHANNELS_GROUP class to describe the channel attributes.
- Code the Global Grouping Function: Code the ggqmgrchann() function to return the GADg pointer to a queue manager's channel group.
- Create the Workspace: Establish the ACME_CHANNELS workspace to save group variables pointing to the baseline channel configurations of a queue manager.
- Prepare the Initial Set Variables: Configure the initial set variables in the acme_channels.aspl script; these parameters are interrogated as differential group variables to expose configuration changes across the queue managers.

Creating the element grouping class IBMMQ_CHANNELS_GROUP

The following metagrouping command creates the IBMMQ_CHANNELS_GROUP element grouping class within ASPL egC-Container.

```
# metagrouping -creategrouping IBMMQ_CHANNELS_GROUP -displayphrase "GROUPING CLASS FOR CHANNEL ATTRIBUTES OF IBM MQ CHANNELS" -attributes "CHANNEL,s" "CHLTYPE,s" "CLUSTER,s" "CONNAME,s" "CONVERT,s" "DESCR,s" "MCATYPE,s" "MCAUSER,s" "MAXMSGL,i" "XMITQ,s" "LONGTMR,i" "SHORTTMR,i" "MRRTY,i" "SHORTRTY,i" "SSLCAUTH,s" "HBINT,i" "COMPMSG,s" "COMPHDR,s" "ALTDATE,s" "ALTTIME,s" -heterogeneousrecords
```

SAMPLE BOOK PREVIEW

Figure 2.3.1 shows the metagrouping execution used to define the IBMMQ_CHANNELS_GROUP element grouping class.

SAMPLE BOOK PREVIEW

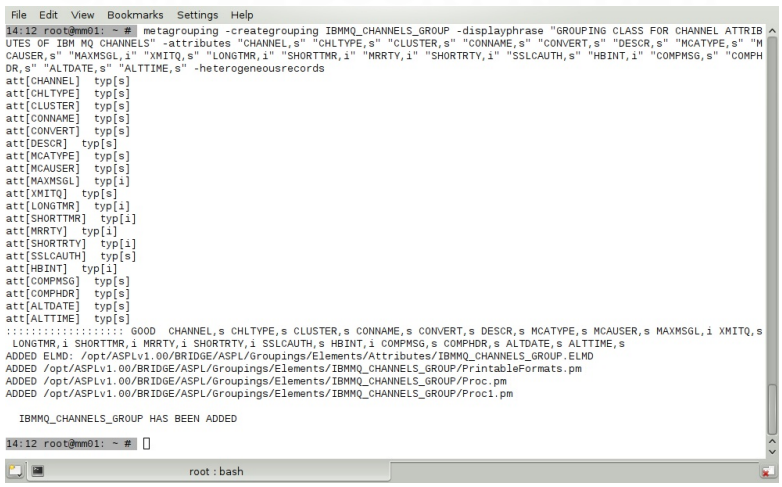


Fig. 2.3.1 [Create the grouping class IBMMQ_CHANNELS_GROUP]

ASPLMQ 2026 by Bassem Jamaleddine

IBM MQ channel objects are further classified by subtypes, such as SDR (Sender) and RCVR (Receiver). The common attributes shared across all subtypes are defined within the $MQutils::MQparser::INTERSECT_CHANNELS_ATTRIBUTES variable. Because channels of differing types do not share an identical attribute set, the -heterogeneousrecords option must be appended to the metagrouping creation command. The command creates the element grouping class IBMMQ_CHANNELS_GROUP in ASPL egC-Container.
Apply this substitution command to the new IBMMQ_QUEUES_GROUP.ELMD file.

```
# sed -i s/default=ABC/default=/ `asplcmd ~elmddir`/IBMMQ_CHANNELS_GROUP.ELMD
```

Edit the file to apply the required modifications to the ks-vector mappings.

```
# vi `asplcmd ~elmddir`/IBMMQ_CHANNELS_GROUP.ELMD
```

These mapping vectors utilize short mnemonics, such as ks1 and ks2, to define the specific list of attributes to evaluate. All designated attributes must belong to the EnodeBaseAttributes set, which is specified via the -attributes option in the metagrouping command. Refer to the ASPL User Guide for instructions on defining numbered ks-vectors within ELMD files. To create a group whose

SAMPLE BOOK PREVIEW

2-4 Shallow Group Operations on IBM MQ Channel Attributes

Channel names typically describe the data flow direction between environments. For example, queue manager A defines a channel named FROMSOURCE.TODESTINATION, while queue manager B defines a matching channel named TODESTINATION.FROMSOURCE. Alternatively, queue manager A defines a channel named FOO1.TO.FOO2, while the corresponding channel FOO2.TO.FOO1 resides on queue manager B. Regardless of the middleware engineer's naming intent, ASPL performs group operations on these channels. Within ASPL, regular expressions match element names in a group using equality filters coded in the ShallowMatches.pm shallow module. These processes are known as shallow group operations.

This chapter explains how to use shallow group operations to compare IBM MQ channels.

■ Shallow group operation on channels with mqcha2elem

Consider the following example comparing channel names across two queue managers: ACMENYNY and ACMEVAVA. The figure below illustrates two set variables, nyAcme and vaAcme, which contain the captured channels from queue manager ACMENYNY (hosted on NYNYQM) and queue manager ACMEVAVA (hosted on VAVAQM), respectively. The figure also displays the group union operation, **fU, nyAcme vaAcme**, executed between the two set variables.

SAMPLE BOOK PREVIEW

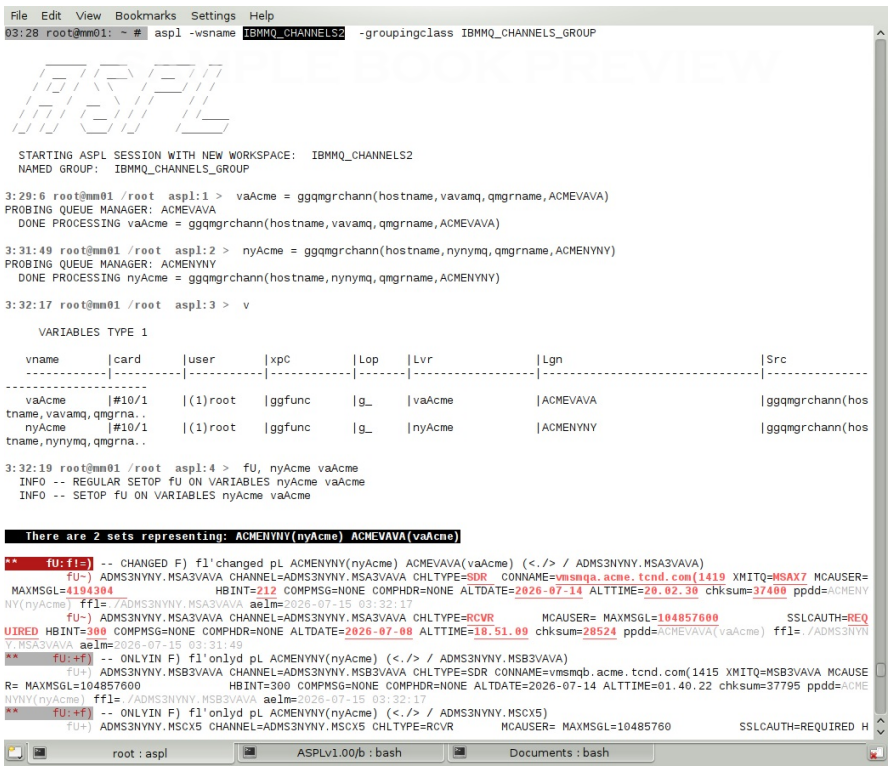


Fig. 2.4.1 [ASPL group variables nyAcme and vaAcme]

ASPLMQ 2026 by Bassem Jamaleddine

Set the shallow matching to mqcha2elem, then perform a shallow union group operation using the group operator **f*U**. At the ASPL prompt, execute the **shallowed** command to print the currently preset processor.

SAMPLE BOOK PREVIEW

```
eceived: (from root@localhost)
        by mm01.waxeon.com (8.14.5/8.14.5/Submit) id 67EF08qs
        Fri, 14 Aug 2026 11:24:08 -0400
ate: Fri, 14 Aug 2026 11:24:08 -0400
essage-Id: <202608141524.67EF08qs019457@mm01.waxeon.com>
o: bassem@mm01.waxeon.com
rom: SetSphere.ASPL.Monitor@mm01.waxeon.com
ubject: ASPL DETECTED CHANGES IN VARIABLE nyvatesta

etSphere ASPL Variable Monitor detected changes in a set-va
----------------------------------------------------------
HEN INTERROGATING AN ASPL SET VARIABLE A CHANGE HAS BEEN DE
OMMAND:          varchangenotify nyvatesta  emailsgroup1
AMED WORKSPACE: CLUSTERED_TESTA_QUEUES
ROUPINGCLASS:    IBMMQ_QUEUES_GROUP

HE FOLLOWING IS LIKELY WHAT HAS BEEN CHANGED:
```

SAMPLE BOOK PREVIEW

```
 INFO -- REGULAR SETOP gU ON VARIABLES nyvatesta nyvatesta@
 INFO -- SETOP gU ON VARIABLES nyvatesta nyvatesta@5

  There are 4 sets representing: TESTA(nytesta)(nyvatesta)
TESTA(vatesta)(nyvatesta.5)

*     gU:+f) ** ONLYIN F) fl onlyd   pL TESTA(vatesta)(nyva
      gU f+) ABCDEF PUT=ENABLED DEFPRTY=0 DEFPSIST=NO DEFPRE
RETINTVL=999999999 BOTHRESH=0 BOQNAME= ppdd=TESTA(vatesta)(
*     gU:f!=) ** CHANGED F) fl changed   pL TESTA(nytesta)(ny
yvatesta) TESTA(vatesta)(nyvatesta.5) (<./> / LQ.TEST.ADMS3N
      gU f~) LQ.TEST.ADMS3NYNY PUT=ENABLED DEFPRTY=0 DEFPSIS
SGL=4194304 RETINTVL=999999999 BOTHRESH=0 BOQNAME= ppdd=TEST
2026-08-09 23:45:11
      gU f~) LQ.TEST.ADMS3NYNY PUT=ENABLED DEFPRTY=0 DEFPSIS
SGL=4194304 RETINTVL=999999999 BOTHRESH=0 BOQNAME= ppdd=TEST
2026-08-09 23:45:11
      gU f~) LQ.TEST.ADMS3NYNY PUT=ENABLED DEFPRTY=0 DEFPSIS
SGL=4194304 RETINTVL=999999999 BOTHRESH=0 BOQNAME= ppdd=TEST
2026-08-14 11:24:08
```

Chapter 3
Monitoring Clustered IBM MQ

When a group represents configuration data, any change to the configuration implies a change to the group, and vice versa. Therefore, detecting a change within the group representing MQ configuration objects is sufficient to verify that the underlying MQ configuration has changed.

This chapter covers scripts for detecting changes in IBM MQ objects. Vertical changes in a queue manager's configurataions mirror the variations within its corresponding **differential group variable**. The **playchanges** operator displays these incremental changes in IBM MQ queue manager configuration.

ASPL variables are self-monitoring variables; toggling their monitoring bits directs the ASPL interpreter to track group modifications. Upon detecting a change during interrogation, the system transmits an email notification alert to specified recipients. This chapter covers monitoring scripts designed to track clustered queue managers for a hypothetical company, ACME Co.

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

3-1 Monitoring Changes in Clustered IBM MQ Qmgrs and Sending Notifications

Visual comparisons of IBM MQ queue manager attribute drift are rendered by executing ASPL operators against the differential group variables representing the queue manager configuration. Additionally, ASPL monitoring operators can dispatch real-time email alerts to notify designated recipients immediately when changes are detected. In this section we show how to use ASPL to detect changes in clustered IBM MQ qmgrs. In the first section, we introduce the script **clustered_delta_qmgrs.aspl** that detects changes in the clustered qmgr TESTA installed at ACME Co. on hosts nynymq and vavamq. In the second part, we show the usage of ASPL powerful **playchanges** on differential group variables to visualiase the changes in the qmgrs configuration that occured over time. In the third section, we introduce the script **monitor_clustered_delta_qmgrs.aspl** that does the same as the previous script, but it also send email notifications to recipients reporting what has changed in the qmgrs.

In the previous section "Grouping Class for IBM MQ Qmgr Attributes" we already created the **element grouping class (metagrouping) IBMMQ_QMGR_GROUP** that we will use in here.

■ Detecting Configuration Change in a Clustered Qmgr

SAMPLE BOOK PREVIEW

We need to to detect changes in the clustered qmgr TESTA. For instance, we will make some changes in the qmgr TESTA on host vavamq, then we will run the script clustered_delta_qmgrs.aspl to detect the changes. Before making any change, let's start coding the script **clustered_delta_qmgrs.aspl**
We edit the script clustered_delta_qmgrs.aspl in ASPL's shared directory.

```
# vi `asplcmd ~shared`/clustered_delta_qmgrs.aspl
```

The script clustered_delta_qmgrs.aspl is shown in Listing 3.1.1.

-LIS- Listing. 3.1.1 [Listing of clustered_delta_qmgrs.aspl]

```
1.    #!/usr/bin/env  aspl
2.    #ENVARG= -wsname CLUSTERED_TESTA_QMGRS -groupingclass IBMMQ_QMGR_GROUP
3.
4.    ;;*************************************************************************
5.    ;;   clustered_delta_qmgrs.aspl
6.    ;;   detects changes in IBM MQ queues attributes in a cluster of qmgrs.
7.    ;;   Assuming the qmgr TESTA is installed on NYNYMQ and VAVAMQ hosts.
8.    ;;
9.    ;;   Copyright Â© 2021-2026 Bassem W. Jamaleddine
10.   ;;   All rights reserved.
11.   ;;
12.   ;;*************************************************************************
13.
14.   ;; Before running the script, a first initialization of group variable
15.   ;; captures is required. The script interrogates the set variables
16.   ;; detecting changes in an incremental change of data configuration.
17.   ;; # aspl -wsname CLUSTERED_TESTA_QMGRS -groupingclass IBMMQ_QMGR_GROUP
18.   ;;     aspl > nytesta = ggibmqmgrs(hostname,nynymq,qmgrname,TESTA)
19.   ;;     aspl > vatesta = ggibmqmgrs(hostname,vavamq,qmgrname,TESTA)
20.   ;; NOTE: below fU used as there is no need to labinate with fU, since
21.   ;;       grp1 is uc(hostname) in ggibmqmgrs(), and is not the qmgr name.
22.   ;;       However, if grp1 is uc(qmgrname),then we need to labinate
23.   ;;       since a qmgrname can be the same on multiple hosts, and
24.   ;;       labeling multiple groups with same grp1 label is possible
25.   ;;       with labination.
26.   ;;     aspl > nyvatesta = fU nytesta vatesta
27.
28.   endScriptIfShellArgsLessThan 1
```

SAMPLE BOOK PREVIEW

3-2 Monitoring Changes in Clustered IBM MQ Queues and Sending Notifications

Visual comparisons for IBM MQ queue configuration drift are rendered by applying ASPL operators to the differential group variables representing the queues. Additionally, ASPL monitoring operators can trigger real-time email alerts to notify designated recipients. This section explains how to use ASPL to detect queue changes within a clustered IBM MQ qmgr. The first part introduces the **clustered_delta_queues.aspl** script, which detects changes in the queues across the group union of the clustered TESTA qmgr installed on hosts nynymq and vavamq. The second part details the usage of the ASPL **playchanges** command with differential group variables to visualize queue configuration changes over time. The third part introduces the **monitor_clustered_delta_queues.aspl** script, which performs the same change detection but also sends automated email reports to designated recipients.

The scripts and ASPL sessions discussed in this section depend on the element grouping class IBMMQ_QUEUES_GROUP created in section "Grouping Class for IBM MQ Queue Attributes".

ACME Co. configured the TESTA queue manager on hosts nynymq and vavamq. Initially, the queue objects on both hosts must be identical. Validate this configuration by displaying the similarity score and any configuration drift between the queue objects on both hosts:

```
# queuescompare.pl -hostqmgr nynymq:TESTA -hostqmgr vavamq:TESTA -view simz
```

SAMPLE BOOK PREVIEW

Adding the ALTTIME attribute to the ks-vector causes the similarity score to drop to 0.0, as all queues were configured at different times across the hosts.

```
# queuescompare.pl -hostqmgr nynymq:TESTA -hostqmgr vavamq:TESTA -view simz -ks+ ALTTIME
```

As shown in Figure 3.2.1, comparing the TESTA queue objects across both hosts yields a similarity score of 1.0. However, including the ALTTIME attribute drops the similarity score to 0.0 because it causes every element in the ks-vector to mismatch. For this reason, the queuescompare.pl script excludes the ALTDATE, ALTTIME, and chksum attributes from ASPL operations by default.

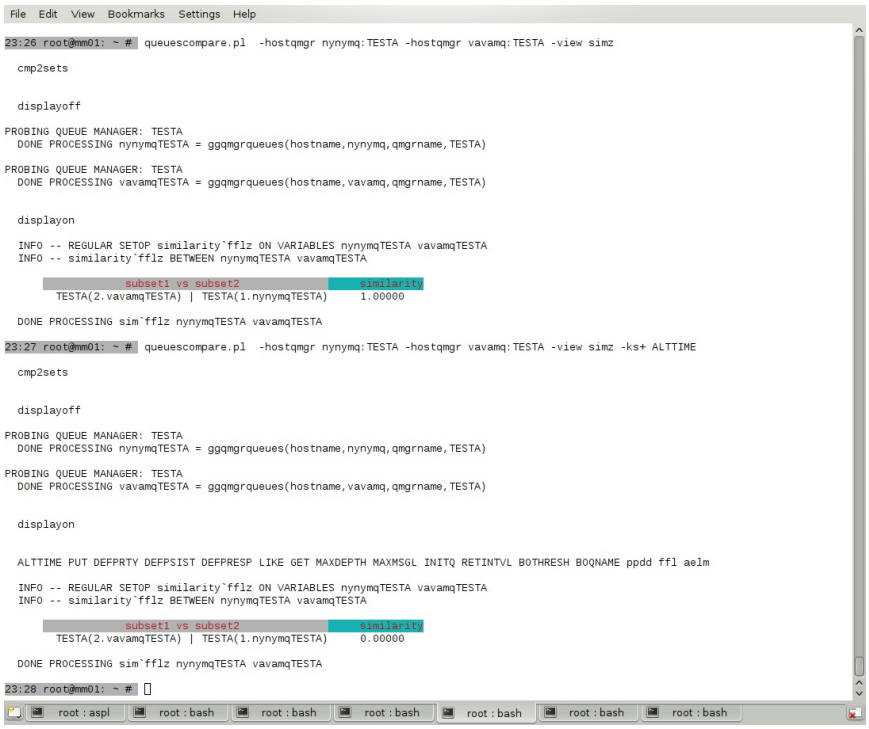


Fig. 3.2.1 [The similarity between qmgr TESTA installed on two hosts]

SAMPLE BOOK PREVIEW

3-3 Monitoring Changes in Clustered IBM MQ Channels and Sending Notifications

Visual comparisons for IBM MQ channel configuration drift are rendered by applying ASPL operators to the differential group variables representing the channels. Additionally, ASPL monitoring operators can trigger real-time email alerts to notify designated recipients immediately when a change is detected. This section explains how to use ASPL to detect channel changes in a clustered IBM MQ qmgr. The first part introduces the **clustered_delta_channels.aspl** script, which detects channel changes across the group union of the queues in the clustered TESTA qmgr installed on hosts nynymq and vavamq. The second part explains how to use the ASPL **playchanges** command with differential group variables to visualize channel configuration changes over time. The third part introduces **monitor_clustered_delta_channels.aspl**, which performs the same change detection but appends a feature to email change reports to specified recipients. The scripts and ASPL sessions discussed in this section depend on the IBMMQ_CHANNELS_GROUP metagrouping class created in Section "Grouping Class for IBM MQ Channel Attributes".

ACME Co. configured the TESTA qmgr on hosts nynymq and vavamq. Initially, the channel objects on both hosts must be identical. Validate this configuration using the following command:

```
# channelscompare.pl -hostqmgr nynymq:TESTA -hostqmgr vavamq:TESTA -view simz
```

As shown in Figure 3.3.1, comparing the TESTA channel objects across both hosts yields a similarity score of 1.0.

SAMPLE BOOK PREVIEW

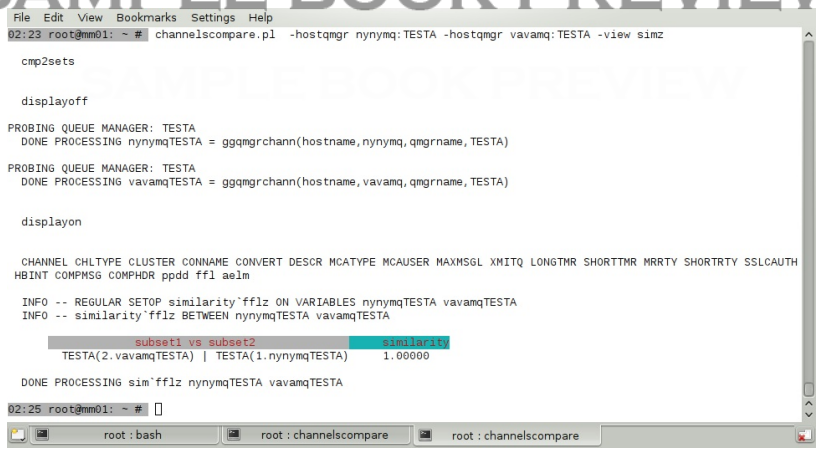


Fig. 3.3.1

ASPLMQ 2026 by Bassem Jamaleddine

Run the channelscompare.pl script (viewing the union of elements) to verify that the channel configurations match on both hosts.

```
# channelscompare.pl -hostqmgr nynymq:TESTA -hostqmgr vavamq:TESTA -view funion
```

All elements in the group union shown in Figure 3.3.2 are marked in green.

SAMPLE BOOK PREVIEW

```
18.             SSLPEER KAINT MONCHL STATCHL COMPMSG COMPHDR
19.             TPNAME PROPCTL PASSWORD MODENAME MCANAME LIKE
20.         )],
21.         SVR => [qw(
22.             CRDATE CRTIME ALTDATE ALTTIME
23.             CHANNEL CHLTYPE
24.             TRPTYPE BATCHINT BATCHHB BATCHSZ CONNAME LOCLADDR (
    LONGRTY LONGTMR SHORTRTY SHORTTMR MAXMSGL
25.             MCATYPE MCAUSER MSGDATA MSGEXIT NPMSPEED RCVDATA RC
    SENDDATA SENDEXIT SEQWRAP USERID XMITQ SSLCAUTH
26.             SSLCIPH SSLPEER KAINT MONCHL STATCHL COMPMSG COMPHD
27.             TPNAME PROPCTL PASSWORD MODENAME MCANAME LIKE
28.         )],
29.         RCVR => [qw(
30.             CRDATE CRTIME ALTDATE ALTTIME
```

SAMPLE BOOK REVIEW

Chapter 4
ASPL MQ Writers

ASPL set variables are identifiers that map directly to structured group data, where each variable represents a Group Algebraic Data group (GADg). In addition, a GADg can exist as an anonymous group generated dynamically from ASPL set operations. These GADg structures represent valuable datasets that record tracking variations over time within a target group's content. To visualize these configurations, ASPL provides writer functions that ingest a GADg (or its transformed counterpart, GADsop) and print a tabulated, pretty-printed report exposing the underlying datasets. ASPL users can code their own Perl modules and drop them in ASPL writers directory (~writerdir), hence allowing the users to have their own customized report. This chapter explains how to write your own writer functions to pretty print your reports.

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

4-1 ASPL MQ Writers

ASPL writers use Perl modules to process GADg or GADsop objects into formatted reports. An ASPL variable is a named set variable pointing to a GADg (Group Algebraic Data group). The GADsop is a transform of the GADg structure that internally maps each element's containment path to its group label. At the ASPL prompt, running the writeformat command by itself invokes a preselected writer against the grouping object at the top of the answer stack. The writer then loops through the group represented by that object, reading the attributes of each element to generate a formatted report. This workflow best demonstrated through a practical example.

The following four ASPL commands manage report generation via custom writer modules:
- ~**writersdir** â€“ Specifies the directory containing the active writer modules.
- **showwriteformat** â€“ Displays all writer modules currently available from within the ASPL session.
- **setwriteformat** â€“ Selects a specific writer module by name for the active session.
- **writeformat** â€“ Prints the tabulated report output using the currently selected writer.

The ASPL interpreter globs the ~**writersdir** directory and loads all contained writer modules. To list the available writer modules, execute either of the following commands:

```
# ls -l `asplcmd ~writersdir`/*.pm

# file `asplcmd ~writersdir`/*.pm
```

SAMPLE BOOK PREVIEW

When the ~writersdir directory is empty, issuing the following command returns the nothing handle.

```
# asplcmd showwriteformat
```

The output shows that the only active writer handler defaults to "nothing":

```
  THE WRITER FORMAT IDENTIFIER TO SELECT THE FORMATTING ROUTINE:

                    IDENTIFIER              GROUPINGCLASS   DESCRIPTION
                    ----------  ------------------------    --------------------------
   >                   nothing                          *   format nothing at all

   CURRENTLY WRITER IS POINTING TO  nothing
   TO SET THE WRITER TO A SPECIFIC HANDLER USE setwriteformat FOLLOWED BY ONE OF THE IDENTIFIERS
```

The MQ_Writer_*.pm files in the source code distribution are shown below:

```
# ls -1 MQ_Writer_*.pm

MQ_Writer_ChannelsCmpQmgr.pm
MQ_Writer_ChannelsColor.pm
MQ_Writer_ChannelsColorTest.pm
MQ_Writer_Channels.pm
MQ_Writer_QmgrsColor.pm
MQ_Writer_QmgrsColorTest.pm
MQ_Writer_Qmgrs.pm
MQ_Writer_QmgrsPrint.pm
MQ_Writer_QueuesCmpQmgr.pm
MQ_Writer_QueuesColor.pm
MQ_Writer_QueuesColorTest.pm
MQ_Writer_QueuesLogos.pm
MQ_Writer_Queues.pm
```

These writer modules are specifically designed to format raw IBM MQ objects into pretty-printed tabulated reports. Copy these files to the ASPL ~writersdir directory by executing the following command:

```
# cp MQ_Writer_*.pm `asplcmd ~writersdir`
```

Once copied, these modules are immediately available within the ASPL interpreter. To display them, execute the **showwriteformat** command:

```
# asplcmd showwriteformat


  THE WRITER FORMAT IDENTIFIER TO SELECT THE FORMATTING ROUTINE:
```

SAMPLE BOOK PREVIEW

A The module MQutils::MQparser to Parse IBM MQ Objects

The ASPL interpreter loads the MQutils::MQparser module as part of its runtime library. The complete implementation listing for MQparser.pm is detailed within this appendix. This parsing module is used by the ASPL global grouping functions ggqmgrqueues() and ggqmgrchannels().

Check for the module within the ASPL lib directory by executing:

```
# ls -l `asplcmd ~home`/lib/MQutils
```

If MQparser.pm is missing, copy it from the distribution code to the ASPL lib directory by running:

```
# mkdir `asplcmd ~home`/lib/MQutils
```
create MQutils directory relative to lib

```
# cp MQparser.pm `asplcmd ~home`/lib/MQutils
```
copy the MQparser.pm into ASPL lib

```
# ls -l `asplcmd ~home`/lib/MQutils
```

MQparser.pm

```
1.     package MQutils::MQparser;
2.
3.     # Author: Bassem W Jamaleddine
4.     # August 2026
5.
6.     use strict;
7.     use warnings;
8.
9.     our $verbose = 0;
10.
11.    # channels attributes per type of IBM MQ channel
12.    my $CHANNELS_ATTRIBUTES = {
13.        SDR => [qw(
14.            CRDATE CRTIME ALTDATE ALTTIME
15.            CHANNEL CHLTYPE
16.            TRPTYPE BATCHINT BATCHHB BATCHSZ CONNAME LOCLADDR CONVERT DESCR DISCINT HBINT
       LONGRTY LONGTMR SHORTRTY SHORTTMR MAXMSGL
17.            MCATYPE MCAUSER MSGDATA MSGEXIT NPMSPEED RCVDATA RCVEXIT SCYDATA SCYEXIT
       SENDDATA SENDEXIT SEQWRAP USERID XMITQ SSLCIPH
18.            SSLPEER KAINT MONCHL STATCHL COMPMSG COMPHDR
19.            TPNAME PROPCTL PASSWORD MODENAME MCANAME LIKE
20.        )],
21.        SVR => [qw(
22.            CRDATE CRTIME ALTDATE ALTTIME
23.            CHANNEL CHLTYPE
24.            TRPTYPE BATCHINT BATCHHB BATCHSZ CONNAME LOCLADDR CONVERT DESCR DISCINT HBINT
       LONGRTY LONGTMR SHORTRTY SHORTTMR MAXMSGL
25.            MCATYPE MCAUSER MSGDATA MSGEXIT NPMSPEED RCVDATA RCVEXIT SCYDATA SCYEXIT
       SENDDATA SENDEXIT SEQWRAP USERID XMITQ SSLCAUTH
26.            SSLCIPH SSLPEER KAINT MONCHL STATCHL COMPMSG COMPHDR
27.            TPNAME PROPCTL PASSWORD MODENAME MCANAME LIKE
28.        )],
29.        RCVR => [qw(
30.            CRDATE CRTIME ALTDATE ALTTIME
31.            CHANNEL CHLTYPE
32.            TRPTYPE BATCHSZ DESCR HBINT MAXMSGL MCAUSER MRDATA MREXIT MRRTY MRTMR MSGDATA
       MSGEXIT NPMSPEED PUTAUT RCVDATA RCVEXIT
33.            SCYDATA SCYEXIT SENDDATA SENDEXIT SEQWRAP SSLCAUTH SSLCIPH SSLPEER KAINT
       MONCHL STATCHL COMPMSG COMPHDR
34.            LIKE
35.        )],
36.        SVRCONN => [qw(
37.            CRDATE CRTIME ALTDATE ALTTIME
38.            CHANNEL CHLTYPE
```

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

B Script to Display IBM MQ Qmgrs: qmgrsdisp.pl

The listing of **qmgrsdisp.pl** is shown below. The **qmgrsdisp.pl** script displays the IBM MQ queue managers running in the cloud environment. For usage examples, see the chapter **Grouping Class for IBM MQ Qmgrs Attributes**. The listing for qmgrsdisp.pl is shown below:

```
1.     #!/usr/bin/perl
2.
3.     # FILE: qmgrsdisp.pl
4.     # DESC: compares the qmgrs of IBM MQ
5.     # Author: Bassem Jamaleddine
6.     # Date: August 2026
7.
8.     use strict;
9.     use warnings;
10.
11.    usage() unless @ARGV >= 1;
12.    sub usage { print join "",<DATA>; exit; }
13.
14.    my @hosted;
15.    my $dbg = 0;
16.    my $ks;
17.    my $kslessen;
18.    my $ksadden;
19.
20.    while (@ARGV) {
21.        if ($ARGV[0] eq '-hostqmgr') {
22.            shift @ARGV;
23.            my $hostqmgr = shift;
24.            push(@hosted,$hostqmgr);
25.        }
26.        elsif ($ARGV[0] eq '-ks') {
27.            shift @ARGV;
28.            $ks = shift;
29.        }
30.        elsif ($ARGV[0] eq '-ks-') {
31.            shift @ARGV;
32.            $kslessen = shift;
33.        }
34.        elsif ($ARGV[0] eq '-ks+') {
35.            shift @ARGV;
36.            $ksadden = shift;
37.        }
38.        elsif ($ARGV[0] eq '-dbg') {
39.            shift @ARGV;
40.            $dbg = 1;
41.        }
42.        elsif ($ARGV[0] eq '-help') {
43.            usage();
44.        }
45.        else {
46.            die "UNKNOWN OPTION $ARGV[0]\n\n";
47.        }
48.    }
49.
50.    usage() unless @hosted >=1;
51.
52.    my $interpreter = "aspl -groupingclass IBMMQ_QMGR_GROUP -wsname TRANSIENT
      -singlepass -dm 3";
53.    my $s = "\n";
54.    # The host name and the qmgr name are appended as one symbol to represent the
      grouping
55.    # variable in ASPL.  It is also being used as the label name of the ASPL
      variable.
56.    # We will create ASPL variables using the host name as the grp1 label.
```

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

C Script to Compare IBM MQ Qmgrs: qmgrscompare.pl

The **qmgrscompare.pl** script compares the IBM MQ queue managers running in the cloud environment. For usage examples, see the chapter **Grouping Class for IBM MQ Qmgr Attributes**. The listing for qmgrscompare.pl is shown below:

```
1.    #!/usr/bin/perl
2.
3.    # FILE: qmgrscompare.pl
4.    # DESC: compares the qmgrs of IBM MQ
5.    # Author: Bassem Jamaleddine
6.    # Date: August 2026
7.
8.    use strict;
9.    use warnings;
10.
11.   usage() unless @ARGV > 1;
12.   sub usage { print join "",<DATA>; exit; }
13.
14.   my @hosted;
15.   #my $view = "funion";
16.   my $view = "changed";
17.   my $dbg = 0;
18.   my $ks;
19.   my $kslessen;
20.   my $ksadden;
21.
22.   while (@ARGV) {
23.       if ($ARGV[0] eq '-hostqmgr') {
24.           shift @ARGV;
25.           my $hostqmgr = shift;
26.           push(@hosted,$hostqmgr);
27.       }
28.       elsif ($ARGV[0] eq '-view') {
29.           shift @ARGV;
30.           $view = shift;
31.       }
32.       elsif ($ARGV[0] eq '-ks') {
33.           shift @ARGV;
34.           $ks = shift;
35.       }
36.       elsif ($ARGV[0] eq '-ks-') {
37.           shift @ARGV;
38.           $kslessen = shift;
39.       }
40.       elsif ($ARGV[0] eq '-ks+') {
41.           shift @ARGV;
42.           $ksadden = shift;
43.       }
44.       elsif ($ARGV[0] eq '-dbg') {
45.           shift @ARGV;
46.           $dbg = 1;
47.       }
48.       elsif ($ARGV[0] eq '-help') {
49.           usage();
50.       }
51.       else {
52.           die "UNKNOWN OPTION $ARGV[0]\n\n";
53.       }
54.   }
55.
56.   usage() unless @hosted >=2;
57.
58.   my $interpreter = "aspl -groupingclass IBMMQ_QMGR_GROUP -wsname TRANSIENT
      -singlepass -dm 3";
```

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

D Script to Display IBM MQ Queues: queuesdisp.pl

The queuesdisp.pl script displays the queues across IBM MQ queue managers running in the cloud environment. For usage examples, see the chapter **Grouping Class for IBM MQ Queue Attributes**. The listing for queuesdisp.pl is shown below.

```
1.     #!/usr/bin/perl
2.
3.     # FILE: queuesdisp.pl
4.     # DESC: compares the queues of IBM MQ
5.     # Author: Bassem Jamaleddine
6.     # Date: August 2026
7.
8.     use strict;
9.     use warnings;
10.
11.    usage() unless @ARGV >= 1;
12.    sub usage { print join "",<DATA>; exit; }
13.
14.    my @hosted;
15.    my $dbg = 0;
16.    my $ks;
17.    my $kslessen;
18.    my $ksadden;
19.    my $report = "";
20.
21.    while (@ARGV) {
22.        if ($ARGV[0] eq '-hostqmgr') {
23.            shift @ARGV;
24.            my $hostqmgr = shift;
25.            push(@hosted,$hostqmgr);
26.        }
27.        elsif ($ARGV[0] eq '-ks') {
28.            shift @ARGV;
29.            $ks = shift;
30.        }
31.        elsif ($ARGV[0] eq '-ks-') {
32.            shift @ARGV;
33.            $kslessen = shift;
34.        }
35.        elsif ($ARGV[0] eq '-ks+') {
36.            shift @ARGV;
37.            $ksadden = shift;
38.        }
39.        elsif ($ARGV[0] eq '-dbg') {
40.            shift @ARGV;
41.            $dbg = 1;
42.        }
43.        elsif ($ARGV[0] eq '-report') {
44.            shift @ARGV;
45.            $report = "\nsetwriteformat MQ_Writer_QueuesColor\nwriteformat\n";
46.        }
47.        elsif ($ARGV[0] eq '-help') {
48.            usage();
49.        }
50.        else {
51.            die "UNKNOWN OPTION $ARGV[0]\n\n";
52.        }
53.    }
54.
55.    usage() unless @hosted >=1;
56.
57.    my $interpreter = "aspl -groupingclass IBMMQ_QUEUES_GROUP -wsname TRANSIENT -singlepass
      -dm 3";
58.
```

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

E Script to Compare IBM MQ Queues: queuescompare.pl

The **queuescompare.pl** script compares the queues across IBM MQ queue managers running in the cloud environment. For usage examples, see the chapter **Grouping Class for IBM MQ Queue Attributes**. The listing for queuescompare.pl is shown below.

```
1.     #!/usr/bin/perl
2.
3.     # FILE: queuescompare.pl
4.     # DESC: compares the queues of IBM MQ
5.     # Author: Bassem Jamaleddine
6.     # Date: August 2026
7.
8.     use strict;
9.     use warnings;
10.
11.    usage() unless @ARGV > 1;
12.    sub usage { print join "",<DATA>; exit; }
13.
14.    my @hosted;
15.    my $view = "changed";
16.    my $dbg = 0;
17.    my $ks;
18.    my $kslessen;
19.    my $ksadden;
20.    my $report = "";
21.
22.    while (@ARGV) {
23.        if ($ARGV[0] eq '-hostqmgr') {
24.            shift @ARGV;
25.            my $hostqmgr = shift;
26.            push(@hosted,$hostqmgr);
27.        }
28.        elsif ($ARGV[0] eq '-view') {
29.            shift @ARGV;
30.            $view = shift;
31.        }
32.        elsif ($ARGV[0] eq '-ks') {
33.            shift @ARGV;
34.            $ks = shift;
35.        }
36.        elsif ($ARGV[0] eq '-ks-') {
37.            shift @ARGV;
38.            $kslessen = shift;
39.        }
40.        elsif ($ARGV[0] eq '-ks+') {
41.            shift @ARGV;
42.            $ksadden = shift;
43.        }
44.        elsif ($ARGV[0] eq '-dbg') {
45.            shift @ARGV;
46.            $dbg = 1;
47.        }
48.        elsif ($ARGV[0] eq '-report') {
49.            shift @ARGV;
50.            $report = "\nsetwriteformat MQ_Writer_QueuesColor\nwriteformat\n";
51.        }
52.        elsif ($ARGV[0] eq '-help') {
53.            usage();
54.        }
55.        else {
56.            die "UNKNOWN OPTION $ARGV[0]\n\n";
57.        }
58.    }
59.
```

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

F Script to Display IBM MQ Channels: channelsdisp.pl

The channelsdisp.pl script displays the channels across IBM MQ channel managers running in the cloud environment. For usage examples, see the chapter **Grouping Class for IBM MQ Channel Attributes**. The listing for channelsdisp.pl is shown below.

```
1.     #!/usr/bin/perl
2.
3.     # FILE: channelsdisp.pl
4.     # DESC: compares the channels of IBM MQ
5.     # Author: Bassem Jamaleddine
6.     # Date: August 2026
7.
8.     use strict;
9.     use warnings;
10.
11.    usage() unless @ARGV > 1;
12.    sub usage { print join "",<DATA>; exit; }
13.
14.    my @hosted;
15.    my $dbg = 0;
16.    my $ks;
17.    my $kslessen;
18.    my $ksadden;
19.    my $report = "";
20.
21.    while (@ARGV) {
22.        if ($ARGV[0] eq '-hostqmgr') {
23.            shift @ARGV;
24.            my $hostqmgr = shift;
25.            push(@hosted,$hostqmgr);
26.        }
27.        elsif ($ARGV[0] eq '-ks') {
28.            shift @ARGV;
29.            $ks = shift;
30.        }
31.        elsif ($ARGV[0] eq '-ks-') {
32.            shift @ARGV;
33.            $kslessen = shift;
34.        }
35.        elsif ($ARGV[0] eq '-ks+') {
36.            shift @ARGV;
37.            $ksadden = shift;
38.        }
39.        elsif ($ARGV[0] eq '-dbg') {
40.            shift @ARGV;
41.            $dbg = 1;
42.        }
43.        elsif ($ARGV[0] eq '-report') {
44.            shift @ARGV;
45.            $report = "\nsetwriteformat MQ_Writer_ChannelsColor\nwriteformat\n";
46.        }
47.        elsif ($ARGV[0] eq '-help') {
48.            usage();
49.        }
50.        else {
51.            die "UNKNOWN OPTION $ARGV[0]\n\n";
52.        }
53.    }
54.
55.    usage() unless @hosted >=1;
56.
57.    my $interpreter = "aspl -groupingclass IBMMQ_CHANNELS_GROUP -wsname TRANSIENT
      -singlepass -dm 3";
58.
```

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

G Script to Compare IBM MQ Channels: channelscompare.pl

The **channelscompare.pl** script compares the channels across IBM MQ channel managers running in the cloud environment. For usage examples, see the chapter **Grouping Class for IBM MQ Channel Attributes**. The listing for channelscompare.pl is shown below.

```
1.     #!/usr/bin/perl
2.
3.     # FILE: channelscompare.pl
4.     # DESC: compares the channels of IBM MQ
5.     # Author: Bassem Jamaleddine
6.     # Date: August 2026
7.
8.     use strict;
9.     use warnings;
10.
11.    usage() unless @ARGV > 1;
12.    sub usage { print join "",<DATA>; exit; }
13.
14.    my @hosted;
15.    my $view = "changed";
16.    my $dbg = 0;
17.    my $ks;
18.    my $kslessen;
19.    my $ksadden;
20.    my $report = "";
21.
22.    while (@ARGV) {
23.        if ($ARGV[0] eq '-hostqmgr') {
24.            shift @ARGV;
25.            my $hostqmgr = shift;
26.            push(@hosted,$hostqmgr);
27.        }
28.        elsif ($ARGV[0] eq '-view') {
29.            shift @ARGV;
30.            $view = shift;
31.        }
32.        elsif ($ARGV[0] eq '-ks') {
33.            shift @ARGV;
34.            $ks = shift;
35.        }
36.        elsif ($ARGV[0] eq '-ks-') {
37.            shift @ARGV;
38.            $kslessen = shift;
39.        }
40.        elsif ($ARGV[0] eq '-ks+') {
41.            shift @ARGV;
42.            $ksadden = shift;
43.        }
44.        elsif ($ARGV[0] eq '-dbg') {
45.            shift @ARGV;
46.            $dbg = 1;
47.        }
48.        elsif ($ARGV[0] eq '-report') {
49.            shift @ARGV;
50.            $report = "\nsetwriteformat MQ_Writer_ChannelsColor\nwriteformat\n";
51.        }
52.        elsif ($ARGV[0] eq '-help') {
53.            usage();
54.        }
55.        else {
56.            die "UNKNOWN OPTION $ARGV[0]\n\n";
57.        }
58.    }
59.
```

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

H Middleware's MQ Monitoring at ACME Co.

At ACME Co., the middleware engineering team deployed IBM MQ across two distinct hosts: nynymq and vavamq. To monitor the active infrastructure, the team requires real-time tracking of configuration modifications applied to the queue managers, including their underlying queue and channel objects. The team also needs to compare disparate queue managers and their associated objects, with the capability to dispatch automated email notifications reporting any detected anomalies. To implement this solution, the team deployed ASPL and integrated the utility scripts outlined in the following tables. Refer to the appendix Adding MQ Scripting to ASPL for instructions on provisioning these scripts within a local environment.

At ACME Co., the middleware team deployed IBM MQ on hosts nynymq and vavamq. To track and compare configuration changes across queue managers, queues, and channelsâ€”and to automate email alerts when drift occursâ€”the team deployed ASPL and implemented the scripts listed in the following tables. See the appendix Adding MQ Scripting to ASPL to make these scripts available on a local target system.

The scripts to compare qmgrs are shown in Table H.1. Each namedspace listed in the table uses the IBMMQ_QMGR_GROUP element grouping class.

Script Name	Namedspace	Description
qmgrsdisp.pl	TRANSIENT	Display Qmgr Attributes
qmgrscompare.pl	TRANSIENT	Compare Qmgrs
acme_qmgrs.aspl	IBMMQ_QMGRS	Display incremental changes in a qmgr
acme_delta_qmgrs.aspl	IBMMQ_DELTA_QMGRS	Record and display incremental changes in a qmgr
clustered_delta_qmgrs.aspl	CLUSTERED_TESTA_QMGRS	Compare clustered qmgr TESTA
monitor_clustered_delta_qmgrs.aspl	CLUSTERED_TESTA_QMGRS	Notify via email on changes in clustered qmgr TESTA

SAMPLE BOOK PREVIEW

Tab. H.1 [Scripts to Display and Compare IBM MQ QMGRs]

The scripts to compare queues are shown in Table H.2. Each namedspace listed in the table uses the IBMMQ_QUEUES_GROUP element grouping class.

Script Name	Namedspace	Description
queuesdisp.pl	TRANSIENT	Display queue attributes
queuescompare.pl	TRANSIENT	Compare queues based on their attributes
acme_queues.aspl	IBMMQ_QUEUES	Display incremental changes in queues
acme_delta_queues.aspl	IBMMQ_QUEUES	Record and display incremental changes in queues
clustered_delta_queues.aspl	CLUSTERED_TESTA_QUEUES	Compare clustered queue

SAMPLE BOOK PREVIEW

I Adding MQ Scripting to ASPL

Extract the code distribution ASPLMQ-v1.tgz in /tmp.

```
# cd /tmp; tar xzvf ASPLMQ-v1.tgz
```

This will create the directory /tmp/ASPLMQ-v1/ where you can find the code that we used in this text.
Under the directory ASPLMQ-v1/ you can locate the following files:
- README.TXT describes the content of the directory ASPLMQ-v1.
- filtermq-queues is the parsing script for IBM MQ queues, it includes the formatting function to tabulate the queues.
- filtermq-channels is the parsing script for IBM MQ channels, it includes the formatting function to tabulate the channels.
- ACMECO.mqsc is a stringified IBM MQ listing of objects for some hypothetical company ACME Co. that we use for testing the parser, it can also be used by runmqsc to create ACMECO qmgr.
- ShallowMatches.pm contains the preprocessors for shallow matching operations in ASPL.
- HOWTO-CREATE-THE-GROUPING-CLASSES.TXT contains the metagrouping commands to create the grouping classes in ASPL's egC-Container used in this text.

Under the directory ASPLMQ-v1/ you can locate the following subdirectories.

```
ASPLMQ-v1/
|-- ggfunc        <--- contains *.pm to be copied to GG-Container
|-- junkwr        <--- contains *.pl to test writers in standalone
|-- MQutils       <--- contains the MQparser.pm
|-- MQ_Writers    <--- contains the Perl writers modules
`-- shared        <--- contains the shared scripts
```

SAMPLE BOOK PREVIEW

SAMPLE BOOK PREVIEW

COPYING THE FILTERS TO ASPL SHARED DIRECTORY

```
# cp filtermq-queues `asplcmd ~shared`

# cp filtermq-channels `asplcmd ~shared`

# chmod 755 `asplcmd ~shared`/filtermq-queues

# chmod 755 `asplcmd ~shared`/filtermq-channels
```

TESTING THE FILTERS ON DIRECT OUTPUT OF ACMECO.mqsc

```
# cat ACMECO.mqsc | filtermq-queues

# cat ACMECO.mqsc | filtermq-channels
```

TESTING THE FILTERS ON OBJECTS RETRIEVED FROM LIVE QMGR

The ACMECO.mqsc can be used to create the the qmgr ACMECO of some hypothetical company ACME Co.
If you have MQ installed on some host vavamq, create the qmgr ACMECO using ACMECO.mqsc.

```
(vavamq's qmgr:ACMECO) 18:23 mqm@vavamq: ~ # crtmqm -ll -lf 2048 -lp 10 -ls 12 -u
SYSTEM.DEAD.LETTER.QUEUE ACMECO

(vavamq's qmgr:ACMECO) 18:23 mqm@vavamq: ~ # strmqm ACMECO

(vavamq's qmgr:ACMECO) 18:23 mqm@vavamq: ~ # runmqsc ACMECO < ACMECO.mqsc

(vavamq's qmgr:ACMECO) 18:23 mqm@vavamq: ~ # endmqm ACMECO

(vavamq's qmgr:ACMECO) 18:23 mqm@vavamq: ~ # strmqm ACMECO
```

Test the filter by getting ACMECO objects using saveqmgr.

```
# ssh -q mqm@vavamq /home/mqm/mqtools/saveqmgr -m ACMECO -s -c 2>&1 | filtermq-
queues

# ssh -q mqm@vavamq /home/mqm/mqtools/saveqmgr -m ACMECO -s -c 2>&1 | filtermq-
channels
```

SAMPLE BOOK PREVIEW

Index

#!/usr/bin/env, 36, 38-39, 50, 54, 66, 69, 86, 93, 101, 111, 117, 126
? $1, (interrogation operator ? on ASPL variable placeholder), 36, 39, 51, 55, 67, 69, 87, 102, 118. *See also* @; playchanges.
? nyvatesta, 90, 105, 108, 121, 123
@, (operator @), 65, 93, 111, 126. *See also* interrogate; ?; playchanges.*See related* ? $1;?.
@ nyvatesta, 93, 126
acme_channels.aspl, 61, 66-68, 184, 188
acme_delta_channels.aspl, 68-70, 184, 188
acme_delta_qmgrs.aspl, 38, 183, 188
acme_delta_qmgrs_notify.aspl, 39, 188
acme_delta_queues.aspl, 54, 183, 188. *See related* script.
acme_qmgrs.aspl, 30, 35-38, 183, 188
acme_queues.aspl, 36, 38, 45, 50-54, 183, 188. *See related* script.
align, 8-11, 59-60, 73-75, 100, 175-176, 181-182
answer stack, 67, 87, 90-91, 102, 108-109, 117-118, 121, 123-124, 132-133, 143
ASPL prompt, 34, 49, 64, 76-77, 132
asplcmd, 14, 31-35, 45-46, 48-50, 61-64, 66, 81, 86, 90-91, 101, 110, 112, 117, 124, 132, 135-137, 143-144, 147, 185-188
channelscompare.pl, 71-74, 115, 117, 121-122, 179, 181-182, 184, 188
channelsdisp.pl, 70-72, 177-178, 184, 188
clustered_delta_channels.aspl, 115, 117-124, 126, 184, 188. *See related* script.
clustered_delta_queues.aspl, 95, 97, 101-105, 109, 111, 135, 184, 188
Data group, 7, 131-132. *See related* dataset.
dataset,, 4, 131, 135. *See also* Group Algebraic Data groups.*See related* GADg;SHAPES.
DIFFERENTIAL GROUP VARIABLE, 4, 15, 30, 39, 45, 55, 61, 65-67, 69, 86, 89-93, 97, 102, 104-106, 109-111, 115, 121, 123-124, 126, 143. *See related* playchanges.
display mode, 50, 65, 93, 111, 126, 133
dumpgad, 135-136, 143
dumphash, 135-136, 143
egC, 4, 30-31, 45, 61, 185-186
egC-Container, (Element Grouping Class Container), 4, 30, 45, 61, 185-186. *See also* pings the egC-Container.
ELEMENT GROUPING CLASS, 4, 29-31, 34, 40, 45, 49, 56, 61, 64, 70, 86, 97, 102, 119, 133-134, 183-184, 186. *See also* ELMD.
ELMD, (Element Metadata Descriptor), 31, 45, 61-62. *See also* Element Grouping Class; egC-Container.
Enode, 45, 61
fU, 33, 35-36, 39, 48, 51-52, 55, 63, 65, 67, 69, 76, 86-88, 90-91, 93-94, 100-103, 105, 107-109, 111-112, 117-119, 121, 123-124, 126-127, 143, 166, 168, 171, 174, 178, 180
GADg, (Group Algebraic Data groups), 4, 6-7, 30-33, 45-48, 61-63, 131-139, 141, 143, 159-160, 162-163. *See also* dataset; GG-Container; GG-function.
GG-Container, 4, 32-33, 46, 49, 62-63, 185, 187. *See also* GADg.
GG-function, (Global grouping function), 4, 46, 187. *See also* GG-Container.
Group Algebraic Data groups, 7. *See also* dataset.
intermittence, 39, 55, 67
intermittentarc, 39, 55, 69, 87, 90-91, 93, 102, 105, 108-111, 118, 121, 123-124, 126. *See also* playchanges.
INTERROGATE,, 30, 35-36, 39-40, 45, 50-51, 55, 61, 65-67, 69, 86, 89-91, 101, 104, 108-109, 111-112, 118, 121, 123-124, 126, 142. *See related* Differential Group Operators.
Interrogating, 4, 65-66, 79, 90, 121, 143
interrogation operations, 143
metagrouping, 30, 45, 61, 86, 115, 185-187
monitor_clustered_delta_channels.aspl, 115, 126, 184, 188
monitor_clustered_delta_qmgrs.aspl, 86, 93-95, 111, 183, 188
monitor_clustered_delta_queues.aspl, 95, 97, 111, 184, 188
Monitoring Changes, 39, 86, 97, 115

SAMPLE BOOK REVIEW

MQ COMMANDS,
runmqsc, 4, 9-10, 33, 35, 56, 65, 69, 78, 88-89, 95, 104, 119-120, 151, 185
saveqmgr, 4, 18, 23, 48, 63, 152, 185-186
DEFINE QL, 9-10, 17, 56, 104
dis QMGR, 151
ALTER QMGR, 35, 88-89, 95
ALTER QLOCAL, 10
ALTER CHANNEL, 119
crtmqm, 4, 185
strmqm, 4, 185
endmqm, 185
mqcha2elem, 76-77, 79, 81, 187
mqcha3elem, 78, 80-82, 187
MQparser, 26, 32-33, 47-48, 61-63, 147, 185, 187
MQutils::MQparser, 26, 32-33, 47-48, 61-63, 147
Namedspace, 5, 14, 34-35, 40, 49-50, 55, 64, 66, 70, 88, 92, 102-104, 111, 123, 133, 135, 142, 183-184
Playchanges,, 85-86, 91-93, 97, 110-111, 115, 124-126. *See also* intermittentcontinuous; intermittentarc; intermittent changes; interrogate; ?; playsim; playsimilarity; playop; rate of change; rate of dissimilarity.
qmgrscompare.pl, 42-44, 167-169, 183, 188
qmgrsdisp.pl, 40-42, 165-166, 183, 188
queuescompare.pl, 8, 10-13, 58-60, 97-101, 173, 175-176, 183, 188
queuesdisp.pl, 55-58, 170-171, 183, 188
SEQUENCE ALIGNMENT, 59, 73-75, 175-176, 181-182
setshallowed, 77. *See related* shallow matching.
setwriteformat, 36, 39, 51, 55, 67, 69, 87, 90-91, 102, 105, 108-109, 117-118, 121, 123-124, 132-133, 166, 168, 170, 173, 177, 179, 186
Shallow matching, 29, 76-77, 79, 81, 185, 187
shallowed module, 187. *See related* shallow matching.
SHAPE, 134. *Same as* SHAPES.*See also* syntax box in ASPL Operations Guide.
showwriteformat, 132-133, 186
similarity, 8, 11, 74, 97-98, 115, 175-176, 181-182
Symbol Table, 4-5, 29, 35, 40, 50, 65, 88, 93, 111, 117, 123, 126, 133
TRANSIENT namedspace, 40, 55, 70. *See related* namedspace.
writeformat, 36, 39, 51, 55, 67, 69, 87, 90-91, 102, 105, 108-109, 117-118, 121, 123-124, 132-134, 142-144, 166, 168, 170, 173, 177, 179, 186
~shallowed, 81, 187. *See related* shallow matching.
~writersdir, 39, 55, 132, 134, 136-137, 140, 186

SAMPLE BOOK PREVIEW