IMS Shared Message Queues
Implementation Considerations

Dave Viguers - IMS Performance
Silicon Valley Lab
5/1/2003
Purpose

The purpose of this document is to provide an overview of the implementation considerations that should be addressed for a successful migration to the IMS shared message queues function. This paper should be considered a practical guide to implementing IMS shared message queues and should not be considered a substitute for the IMS manuals or redbooks that might provide more detail. Shared message queues allow multiple IMS subsystems to share, and therefore process and deliver, messages from a common queue that is maintained in a Coupling Facility structure. There is one queue for full-function messages and another for Fast Path. The shared message queues function facilitates improved availability, workload balancing, and increased capacity. IMS subsystems may join and leave the shared message queues group independently thus also allowing for configuration flexibility. The shared message queues function has been available since IMS Version 6, was enhanced in IMS Version 7, and enhanced again in IMS Version 8, with other changes made via the service process along the way. In addition to IMS, many enhancements have also been made to OS/390, z/OS, and the Coupling Facility to improve availability, usability, and performance. These enhancements will also be taken into consideration. Additional detailed information can be found in the IMS manuals, OS/390 or z/OS manuals, redbooks, and education courses.

High level considerations and expectations

Every installation is unique and therefore it is not always possible to give specific recommendations regarding all of the possible variables. When specific guidelines are not appropriate we will attempt to show methods or formulas for you to determine the values for your installation. Implementing shared message queues reminds me of many years ago when I was performing a tune-up on my car. I followed the manual and made all of the exact adjustments as specified in the manual. At the very end of the procedure the manual stated,"now drive your car and readjust as necessary for best performance". This same philosophy can be applied to implementing and tuning IMS in general and specifically for shared message queues.
Implementing shared message queues for full-function messages or Fast Path can provide many benefits to your installation. As previously mentioned, increased availability, workload balancing, and extra capacity are some of those benefits. However, the shared message queues function, like many other major functional enhancements, has its associated costs. Most of this cost will be in additional CPU consumption as a direct result of the accesses to the Coupling Facility structures. This cost is variable, depending on your Sysplex configuration. Variables such as your central processor speed, Coupling Facility configuration, available storage and DASD can affect various components of overhead and performance. This paper describes the effect of those variables, as well as the various related IMS and operating system parameters.

It is strongly recommended that both functional and stress test environments be used to further evaluate the impact of shared message queues with respect to operational changes, backup and recovery procedures, and performance impact. In some cases, there may be impacts on the test environment that will not be the same in a production environment. Specific examples of these known differences will also be identified in subsequent sections of this article.
Implementation

The steps for implementation:
1. Prepare for implementation, Gather the data necessary as input to the CFSIZER tool.
2. Update the MVS Program Properties Table.
3. Make any necessary updates to your system proclib.
4. Update and add members to IMS.PROCLIB.
5. Add IMS shared message queues and MVS log structures to the CFRM policy.
6. Update the MVS Logger policy.
7. Update security profiles.
8. Allocate CQS data sets.
10. Test some more.

1) Prepare for implementation

So let’s look at the things we must consider before implementing the shared message queues function. There are some pieces of information we should gather in the local queue environment in order to make informed decisions regarding the setting of certain parameters for shared message queues. The following list contains the basic information we need to begin. This information will be used to help determine values for IMS and also as input to the CFSIZER tool available on the Web.

1. Normal and maximum LGMSG and SHMSG records in use.
2. Average input and output message size / SHMSG and LGMSG LRECL.
3. Is Fast Path being used?
4. Number of EMH messages in use.
5. Average size of EMH messages.

Now let’s see how to obtain the information.
1. The normal and maximum LGMSG and SHMSG records in use can be obtained from various online monitors or from the IMS log. Each time a simple (or statistics) checkpoint is taken IMS will write a x’4502’ log record containing statistics for the IMS queue manager. This log record, as well as the other statistics records, can be formatted with the IRUR report of IMSPA, or by using DFSERA10 to print the records and assembling the DSECT to determine the offset of the statistics within the record. The macro to assemble is ILOGREC RECID=45. Fields STSINUSR and STLINUSR in the x’4502’ log record show the current high record in use for the SHMSG and LGMSG data sets. You should gather sample statistics from several days and various times of the
day to determine if there are any trends to watch for, such as if the maximum record in use grows
during the week or month.

2. The second thing we must determine is the average input and output message size. You might
already have this information from various reports that you run on a regular basis. The basic source
of this information is again the IMS log and specifically the ‘01’ input message and ‘03’ output
message log records. IMSPA will provide you with the message sizes by running the Message
Queue Utilization report. You will get the average size of messages on both the SHMSG and
LGMSG data sets. Factor in the number of messages to each of the queue data sets to determine
the average size of all messages. You should run this report for many different intervals during the
day or week in case there are times when there is a significant deviation from the average. This
average message size will be used to determine the proper ratio of LE’s (List Entries) to Elements
for the shared message queues structures and also be used as a factor for sizing the structure.
NOTE: For the LE-to-Element ratio, you should always round down to the next lower multiple of
512. You might also consider just using the value of 512 to prevent running out of LE’s before data
elements since only the data elements are monitored for overflow processing. Also, it is less work
than trying to determine the average size.

3. Is Fast Path being used? This decision is based on whether or not you have a FPCTRL macro in
your IMS definition. Even if you are not using EMH processing you will need to define an EMH
structure with Version 8 and prior releases if you are fast path capable since it is possible to
dynamically add fast path transactions. Of course you will only need to define a minimal structure,
so you can skip steps 4 and 5, in which you determine the number of EMH messages as well as the
size.

4. The number of EMH messages that are active in IMS at any given point may be difficult to
determine. Since there can only ever be 1 input or 1 output message on the queue per terminal at
any given time, the maximum number possible will be the number of terminals that can ever enter a
fast path transaction. Depending on certain terminal options such as FPACK it is possible that there
will virtually always be an output message on the queue until the next input is received, so using the
total number of terminals is probably the safest method to use.

5. The size of the EMH messages is used in much the same way as the full-function message size in
that it is used to determine the proper ratio of LE’s to data elements and also to determine the size
of the structure needed. Message sizes may be found in the IMSPA report ‘Fast Path (EMH)
Message Statistics’. Please see the note in step 2 above regarding the average message size. The
same applies to EMH.

6. Go to the Web and use the information you have gathered to estimate the structure sizes you will
need. The address on the Web is: www.ibm.com/servers/eserver/zseries/cfsizer. You also might
want to set lower sizes for your test environment in order to properly test and create operational
procedures for situations such as overflow processing, loss of path to a structure, etc.
2) Update the Program Properties Table in SYS1.PARMLIB

Add the following entry, if it is not already there, to the SCHEDxx member (Current z/OS releases should already have this entry included).

```
PPT PGMNAME(CQSINIT0)
   CANCEL
   KEY(7)
   NOSWAP
   NOPRIV
   SYST
   DSI
   PASS
   AFF(NONE)
   NOPREF
```

3) Update your system proclib

1. There are no changes necessary for your IMS procedures unless you choose to change the RGSUF value to use a different DFSPBxxx member. This method would allow you to keep your current DFSPBxxx member the same in the event there was a need to fall back.

2. Create the CQS (Common Queue Server) procedures. This is a new address space for shared message queues and will be started automatically by IMS similar to DLI/SAS and DBRC. In IMS Version 6, there must be one CQS address space for each IMS. However, with IMS Version 7 and later it is possible to have multiple IMS’s on the same MVS use a common CQS address space. The following is a sample procedure for CQS.

```csh
//*****************************************************************
//*     CQS Procedure
//*
//*     Parameters:
//*     BPECFG  - Name of BPE member
//*     CQSINIT - Suffix for your CQSIPxxx member
//*     PARM1   - Other override parameters (specified in proclib member CQSIPxxx)  //*
//*               ARMRST - ARM restart on CQS failure (Y/N)
//*               CQSGROUP - XCF Group for sharing CQSs
//*               SSN - CQS subsystem name
//*               STRDEFG - Suffix for CQSSGxxx member
//*               STRDEFL - Suffix for CQSSLxxx member
//*               example:
//*               PARM1='ARMRST=Y,SSN=CQSA,STRDEFG=001,STRDEFL=001'
//*//
//CQS      PROC RGN=3000K,SOUT=A,RESLIB='IMS.SDFSRESL',BPERES='IMS.SBPERESL',
//           BPECFG=BPECONFIG,CQSINIT=,PARM1=
//CQSPROC  EXEC PGM=CQSINIT0,REGION=&RGN,
//            PARAM='BPECFG=&BPECFG,CQSINIT=&CQSINIT,&PARM1'
//STEPLIB  DD  DSN=&RESLIB,DISP=SHR
//         DD  DSN=&BPERES,DISP=SHR
```
4) Update and add members to IMS.PROCLIB

**DFSPBxxx** - This member whose suffix is specified by the RGSUF=xxx value in the IMS procedure will need to be updated with the parameters for shared message queues. The following is a sample of shared message queues keywords and parameters in member DFSPBxxx.

Sample of shared queues values:

```
SHAREDQ=01Q,
QBUF=500,
QBUF=8192,
```

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHAREDQ=</td>
<td>Specifies the suffix of the new DFSSQxxx member. This keyword tells IMS to initialize with shared message queues instead of private queues.</td>
</tr>
<tr>
<td>QBUF=</td>
<td>This keyword is probably already specified, but when running in shared message queues mode, it specifies the initial pool size that can be expanded dynamically by IMS if necessary.</td>
</tr>
<tr>
<td>QBUFSZ=</td>
<td>Specifies the size of each queue buffer. Should be a multiple of LGMSGSZ and SHMSGSZ as stated below.</td>
</tr>
<tr>
<td>QBUFMAX=</td>
<td>Sets an upper limit on the number of buffers that IMS will allocate.</td>
</tr>
<tr>
<td>QBUFHIT=</td>
<td>Percentage of buffers in use that will trigger dynamic expansion.</td>
</tr>
<tr>
<td>QBUFLW=</td>
<td>Percentage of buffers in use that will trigger dynamic compression (but never below the QBUF= value).</td>
</tr>
<tr>
<td>QBUFPCTX=</td>
<td>Percentage of the originally allocated (QBUF) buffers to add when dynamically expanding the QBUF pool.</td>
</tr>
<tr>
<td>LMSGSZ=</td>
<td>Size of a long message buffer. Must be greater than or equal to SHMSGSZ. For efficient storage, this value should be equal to or evenly divisible into QBUFSZ.</td>
</tr>
<tr>
<td>SHMSGSZ=</td>
<td>Size of a short message buffer. Must be at least 1.5 times the maximum prefix size plus 4. For efficient storage, consider making this value evenly divisible into LGMSGSZ.</td>
</tr>
</tbody>
</table>

```
QBUFMAX=2000,
QBUFHITH=80,
QBUFLW=50,
QBUFPCTX=20,
LGMSGSZ=4096,
SHMSGSZ=512,
```

Explanation:
DFSSQxxx - This member is referenced in the SHAREDQ= keyword of the DFSPBxxx member. (In our example it is DFSSQ01Q). There is normally a different member for each IMS. There are only 5 values to be specified as follows:

Sample member:

CQS=IM1ACQS,
CQSSSN=CQ1A,
SQGROUP=IMS0G,
MSGQ=IM0A_MSGP,
EMHQ=IM0A_EMHP

Explanation:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CQS=</td>
<td>Tells IMS the name of the CQS started procedure. When in shared message queues mode, IMS will start this address space automatically if it is not already active.</td>
</tr>
<tr>
<td>CQSSSN=</td>
<td>Tells IMS with which CQS it will connect.</td>
</tr>
<tr>
<td>SQGROUP=</td>
<td>This is the XCF group name which each IMS in the shared message queues group will join. It must be the same name for all members of the group. The actual group name will be prefixed with DFS, In this example, the group name is DFSIMS0G.</td>
</tr>
<tr>
<td>MSGQ=</td>
<td>Defines the full function primary structure name and must be the same name defined in the CFRM policy, which will be described later.</td>
</tr>
<tr>
<td>EMHQ=</td>
<td>Defines the fast path EMH primary structure name which also must be the same name as in the CFRM policy. This value must be specified if a FPCTRL macro is defined in your IMS system definition, regardless of whether or not you use EMH processing.</td>
</tr>
</tbody>
</table>

**Note:** For further detail on the above proclib members, see IMS Installation Volume 2: System Definition and Tailoring manual.
The following members are referenced by CQS and are therefore prefixed with the letters CQS. They do not need to be in the same IMS.PROCLIB as the IMS members but it is probably easier to reference if they are.

**BPECONFG** - This is the member specified by the BPECFG= keyword in the CQS procedure. While defaults exist for this member, it is recommended that you code them even if you use the default values for ease of reference.

Sample member:

```
********************************************************************
* CONFIGURATION FILE FOR BPE WITH CQS                                  *
********************************************************************
LANG=ENU                /* LANGUAGE FOR MESSAGES     */
/* (ENU = U.S. ENGLISH)    */

* DEFINITIONS FOR BPE SYSTEM TRACES
*
TCLEV=(*,LOW,BPE)
* THE ABOVE SETS THE TCLEV TO LOW FOR ALL OF THE BPE SYSTEM TRACES
* THIS INCLUDES CBS, DISP, AWE, LATC, SSRV, and USRX
* INDIVIDUAL TRACES MAY BE OVERRIDDEN WITH SUBSEQUENT STATEMENTS IF DESIRED
*

* DEFINITIONS FOR CQS TRACES
*
TCLEV=(*,LOW,CQS)
* THE ABOVE SETS THE TCLEV TO LOW FOR ALL OF THE CQS TRACES
* THIS INCLUDES CQS, STR, and INTF
* INDIVIDUAL TRACES MAY BE OVERRIDDEN WITH SUBSEQUENT STATEMENTS IF DESIRED
*

* DEFINITIONS FOR USER EXIT PROCLIB MEMBER
*
* EXITMBR=(CQSEXIT0,CQS)
```

Explanation:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANG=</td>
<td>Specifies the language to use for BPE and IMS component messages. ENU (U.S. English) is currently the only value supported.</td>
</tr>
<tr>
<td>TRCLEV=</td>
<td>Specifies the level of detail for various traces. While ERROR would be the lowest cost for performance, a value of LOW is a good choice in order to provide internal traces in the event that problem diagnosis is necessary.</td>
</tr>
<tr>
<td>EXITMBR=</td>
<td>Specifies another proclib member containing an exit list for the named component. In this example, we are not using this function so it is commented out, but you should review the possible exit functions for use in your installation.</td>
</tr>
</tbody>
</table>
CQSIPxxx - This member is referenced in the CQS procedure described previously. There should be a different member for each CQS address space, and each will contain the following statements:

Sample member:

ARMRST=Y,
CQSGROUP=IMS0G,
SSN=CQ1A,
STRDEFL=01L,
STRDEFG=00G

Explanation:

<table>
<thead>
<tr>
<th>ARMRST=</th>
<th>Do you want the MVS Automatic Restart Manager to restart CQS in the event of a failure? You can choose to use your own automation, but ARM should be considered.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CQSGROUP=</td>
<td>The XCF group that CQS will join. This name is prefixed with CQS so even though the name specified here may be the same as in the DFSSQxxx member it is not the same XCF group.</td>
</tr>
<tr>
<td>SSN=</td>
<td>The CQS subsystem name, which must be the same as the CQSSSN value in the DFSSQxxx member for proper connection. This name will also be used as an identifier for CQS messages. This name does NOT need to be defined in the MVS SSN table.</td>
</tr>
<tr>
<td>STRDEFL=</td>
<td>Specifies the suffix for the CQSSLxxx member.</td>
</tr>
<tr>
<td>STRDEFG=</td>
<td>Specifies the suffix for the CQSSGxxx member.</td>
</tr>
</tbody>
</table>

CQSSLxxx - Pointed to by the CQSIPxxx member above and defines information local to a given CQS, with the exception of the primary structure name which is common.

Sample member:

STRUCTURE
    (STRNAME=IM0A_MSGP,CHKPTDSN=IMSPSA.IM1A.MSG.CHKPT,SYSCHKPT=100000)
STRUCTURE
    (STRNAME=IM0A_EMHP,CHKPTDSN=IMSPSA.IM1A.EMH.CHKPT,SYSCHKPT=100000)

Explanation:
There will be a STRUCTURE statement for full function and another for fast path (if defined).
Each statement will have the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRNAME=</td>
<td>Name of the primary structure. This name must be the same as in the CQSSGxxx member below and also must exist in the active CFRM policy before it can be referenced.</td>
</tr>
<tr>
<td>CHKPTDSN=</td>
<td>Name of the CQS checkpoint data set. This is a relatively small data set used when a CQS checkpoint is taken. This is NOT to be confused with the structure checkpoint data sets mentioned below.</td>
</tr>
<tr>
<td>SYSCHKPT=</td>
<td>Number of CQS log records to be written before a CQS checkpoint is automatically initiated. This is similar in function to the CPLOG value in IMS. Set this value high if you wish to issue the checkpoint command through automation.</td>
</tr>
<tr>
<td>CQSSGxxx</td>
<td>Also pointed to by the CQSIPxxx member. Defines information common to the shared message queues group.</td>
</tr>
</tbody>
</table>

Sample member:

```plaintext
STRUCTURE
  (STRNAME=IM0A_MSGP,
   OVFLWSTR=IM0A_MSGO,
   OVFLWMAX=70,
   SRDSDSN1=IMSPSA.IM0A.MSG.SRDS1,
   SRDSDSN2=IMSPSA.IM0A.MSG.SRDS2,
   LOGNAME=IM0A.MSG.LOGSTR,
   OBJAVGSZ=512)

STRUCTURE
  (STRNAME=IM0A_EMHP,
   OVFLWSTR=IM0A_EMHO,
   OVFLWMAX=70,
   SRDSDSN1=IMSPSA.IM0A.EMH.SRDS1,
   SRDSDSN2=IMSPSA.IM0A.EMH.SRDS2,
   LOGNAME=IM0A.EMH.LOGSTR,
   OBJAVGSZ=512)
```

Explanation:
There will be a STRUCTURE statement for full function and another for fast path if defined. Each statement will have the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRNAME=</td>
<td>Name of the primary structure. This name must be the same as in the CQSSLxxx member above and also must exist in the active CFRM policy before it can be referenced.</td>
</tr>
<tr>
<td>OVFLWSTR=</td>
<td>Name of the optional overflow structure.</td>
</tr>
<tr>
<td>OVFLWMAX=</td>
<td>The percentage of data elements in use which will trigger overflow processing.</td>
</tr>
<tr>
<td>SRDSDSN1/2=</td>
<td>Name of the structure recovery data sets. These data sets are used when a</td>
</tr>
</tbody>
</table>
structure checkpoint is taken, which is similar to taking a checkpoint SNAPQ in the private queues environment. They are used alternately as structure checkpoints are taken.

<table>
<thead>
<tr>
<th>LOGNAME=</th>
<th>Name assigned to the MVS logstream. This name will also be specified in the policy for the MVS logger.</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJAVGSZ=</td>
<td>Average message size. This value will determine the ratio of List Elements (LE’s) to data elements. Each data element in the shared message queues structures is 512 bytes, so any number up to and including 512 will cause a 1:1 ratio. A value of 513 to 1024 will be a 1:2 ratio and so on. If you have any doubt about the average message size then ALWAYS round down to the next 512 byte increment to avoid running out of LE’s before data elements. To be safe with regards to detecting a possible structure full condition, consider just using the size of 512, which will insure that you do not run out of LE’s before data elements.</td>
</tr>
</tbody>
</table>

**Note:** For additional details about the above members, see IMS Version 6 or IMS Version 7 the Common Queue Server and Base Primitive Environment Guide and Reference. For Version 8 there are two separate manuals.
5) Add IMS shared message queues and MVS log structures to the CFRM (Coupling Facility Resource Manager) policy

This policy must include all of the Coupling Facilities to be used and all of the structures that can be allocated. Several different policies may be defined but only one can be active in the sysplex. The shared message queues structures must be defined, as well as the MVS logger structures, to support the log streams used by CQS.

A sample of how to specify the structures is shown below:

```c
/*--------------------------------------------------------------*/
/* IMS SHARED MESSAGE QUEUES STRUCTURES                        */
/*---------------------------------------------------------------*/

STRUCTURE NAME(IM0A_MSGP)
   SIZE(30000) INITSIZE(20000)
   PREFLIST(CF11,CF12)
   REBUILDPERCENT(1)

STRUCTURE NAME(IM0A_MSGO)
   SIZE(30000)
   PREFLIST(CF12,CF11)
   REBUILDPERCENT(1)

STRUCTURE NAME(IM0A_EMHP)
   SIZE(20000)
   PREFLIST(CF12,CF11)
   REBUILDPERCENT(1)

STRUCTURE NAME(IM0A_EMHO)
   SIZE(20000)
   PREFLIST(CF11,CF12)
   REBUILDPERCENT(1)

/*---------------------------------------------------------------*/
/* IMS SHARED MESSAGE QUEUES LOGGER STRUCTURES */
/*---------------------------------------------------------------*/

STRUCTURE NAME(IM0A_LOGM)
   SIZE(200000) INITSIZE(50000)
   PREFLIST(CF12,CF11)
   REBUILDPERCENT(1)

STRUCTURE NAME(IM0A_LOGE)
   SIZE(200000) INITSIZE(50000)
   PREFLIST(CF11,CF12)
   REBUILDPERCENT(1)
```
### Explanation:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>For the shared message queues structures, name must be the same as the name specified in the CQSSGxxx, CQSSLxxx, and DFSSQxxx members. The overflow structure names are only referenced in the CQSSGxxx member. For the logger structures, the names must match the names in the logger policy.</td>
</tr>
<tr>
<td>SIZE</td>
<td>The maximum size that this structure may be (in thousands). Use the CFSIZER tool to estimate the appropriate starting size. For the overflow structures you should make the INITSIZE (or SIZE if no INITSIZE is specified) slightly larger than the percentage specified in the OVFLWMAX in the CQSSGxxx times the SIZE of the primary structure at a minimum. This allows for the possibility of one destination name causing overflow processing, in which case all of that queue must be able to be moved to the overflow structure. <strong>NOTE:</strong> To be safe, consider making the size of the overflow structure the same as your primary structure. For the logger structures, the CFSIZER tool always returns a value of 50MB. This is probably a bit on the small size but of course it depends on the amount of data being logged per second and the desired frequency of offload to DASD. See the discussion about the MVS Logger in the performance section of this document.</td>
</tr>
<tr>
<td>INITSIZE</td>
<td>The requested initial size to build this structure when it is first allocated (in thousands). If this value smaller than the SIZE value, the structure may be dynamically altered to a larger value if necessary.</td>
</tr>
<tr>
<td>PREFLIST</td>
<td>The available CF’s for this structure and the preferred order of CF’s in which this structure should and can be allocated. This order is really only a ‘suggestion’ to the system since the actual allocation depends on the CF connectivity, available storage, etc.</td>
</tr>
<tr>
<td>REBUILDPERCENT</td>
<td>Specifies at what ‘loss of connectivity’ percent a rebuild should be attempted in order to restore full connectivity. A low value is probably best.</td>
</tr>
</tbody>
</table>
Additional parameters that can be specified:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOWAUTOALT</td>
<td>It is recommended that you not specify this parameter since allowing the system to automatically alter the structure may have undesired results such as freeing space for some other structure.</td>
</tr>
<tr>
<td>FULLTHRESHOLD</td>
<td>This value goes with the ALLOWAUTOALT value and should therefore not be specified.</td>
</tr>
<tr>
<td>DUPLEX</td>
<td>There are 3 possible values for DUPLEX: DISABLED, ALLOWED, ENABLED. Since the SMQ structures are not duplexed by IMS and neither are the MVS logger structures, you might want to investigate whether or not this function is necessary for your installation. Even without duplexing it would take multiple failures to cause loss of the queues, so the DUPLEX function might or might not be worth the cost. When you specify the use of duplexing, response times to the CF will certainly increase so make sure that this parameter setting is providing you with the results you need.</td>
</tr>
</tbody>
</table>

CFRM Format Utility parameters:

In order to use the DUPLEX function mentioned above you must specify SMDUPELEX when you format the CFRM Couple data sets. You can also specify SMREBLD (it is implied if SMDUPELEX is coded) to allow a rebuild of a structure when no connectors are active, this would allow movement of a structure to a different CF even if CQS were not up at the time.
6) Update the MVS logger policy

This policy defines the structure names and the logstream names that CQS references. The running of the logger policy definition job is cumulative in that these definitions are added to whatever definitions already exist for other logstreams. Also, unlike the CFRM policy, this policy becomes active as soon as the job completes successfully.

```plaintext
//DEFLOGR JOB ,MSGCLASS=H,NOTIFY=&SYSUID,
//         CLASS=A,MSGLEVEL=(1,1),REGION=4096K
//STEP1 EXEC PGM=IXCMIAPU
//SYSPRINT DD SYOUT=* 
//SYSIN DD * 
DATA TYPE(LOGR) REPORT(YES)

DEFINE STRUCTURE NAME(IM0A_LOGM)
   LOGSNUM(1)
   MAXBUFSIZE(65272)
   AVGBUFSIZE(4096)

DEFINE LOGSTREAM NAME(IM0A.MSG.LOGSTR)
   STG_DUPLEX(NO)
   STRUCTNAME(IM0A_LOGM)
   HLQ(IXGLOGR)
   LS_SIZE(524000)
   HIGHOFFLOAD(50)
   LOWOFFLOAD(0)
   LS_DATACLAS(LOGGER)
   RETPD(0)
   AUTODELETE(NO)

DEFINE STRUCTURE NAME(IM0A_LOGE)
   LOGSNUM(1)
   MAXBUFSIZE(65272)
   AVGBUFSIZE(4096)

DEFINE LOGSTREAM NAME(IM0A.EMH.LOGSTR)
   STG_DUPLEX(NO)
   STRUCTNAME(IM0A_LOGE)
   HLQ(IXGLOGR)
   LS_SIZE(524000)
   HIGHOFFLOAD(50)
   LOWOFFLOAD(0)
   LS_DATACLAS(LOGGER)
   RETPD(0)
   AUTODELETE(NO)
```
**Explanation:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRUCTURE NAME</td>
<td>Associates the structure with the logstream.</td>
</tr>
<tr>
<td>MAXBUFSIZE</td>
<td>Must be large enough to hold the largest CQS log record. It is recommended that you specify the size shown in the example (65272).</td>
</tr>
<tr>
<td>AVGBUFSIZE</td>
<td>Can be specified as shown in this example and is only used to initially establish a directory-to-element ratio. The logger will manage this ratio dynamically.</td>
</tr>
<tr>
<td>LOGSNUM</td>
<td>The number of logstreams that can be managed in this structure. It is recommended that you specify a value of 1.</td>
</tr>
<tr>
<td>LOGSTREAM NAME</td>
<td>These names must match the LOGNAME= values that you specified in the CQSSGxxx member.</td>
</tr>
<tr>
<td>STG_DUPLEX</td>
<td>Specifies whether or not (YES or NO) you want to use DASD staging data sets for duplexing. It is recommended that you specify NO, which will cause data to be duplexed in a data space. This is due to performance concerns since access to the coupling facility is much faster than to DASD.</td>
</tr>
<tr>
<td>HLQ</td>
<td>This is the high-level qualifier assigned to the offload data sets when they are allocated.</td>
</tr>
<tr>
<td>LS_SIZE</td>
<td>The size of the offload data sets that are allocated. By specifying a large number (which is multiplied by 4K) up to 2GB, you minimize the number of times a new data set must be allocated.</td>
</tr>
<tr>
<td>HIGHOFFLOAD</td>
<td>Specifies what percentage full the structure will be before offload processing is invoked. This, together with the size of the structure (in the CFRM policy) and the amount of data being written to the logstream per second, will determine how often the offload process gets driven and how much space is available for log writes while the offload is taking place. 50 percent is a reasonable value.</td>
</tr>
<tr>
<td>LOWOFFLOAD</td>
<td>The point at which the logger will stop offloading. 0 is a good value that allows all of the data to be offloaded.</td>
</tr>
<tr>
<td>LS_DATAACLAS</td>
<td>The SMS data class of the logger offloads.</td>
</tr>
<tr>
<td>RETPD</td>
<td>The number of days to retain this data. CQS does not require anything other than the default of 0 since CQS will tell the logger when the data is no longer needed for recovery and can be deleted.</td>
</tr>
<tr>
<td>AUTODELETE</td>
<td>CQS will notify the logger that data older than the oldest structure checkpoint may be deleted, so you can let this parameter default to NO.</td>
</tr>
</tbody>
</table>
7) Update security profiles

Security, who needs security? Unfortunately, it is a way of life these days, Here is an example of the security definitions you might need at your installation:

Examples:

To secure access to CQS:

RDEFINE FACILITY CQS.cqsid UACC(NONE)
PERMIT CQS.cqsid CLASS(FACILITY) ID(cqsuser1) ACCESS(UPDATE)
.....and so forth for all connectors to CQS
SETROPTS CLASSACT(FACILITY)

To secure access to CQS structures:

RDEFINE FACILITY CQSSTR.strname UACC(NONE)
PERMIT CQSSTR.strname CLASS(FACILITY) ID(cqsid1) ACCESS(UPDATE)
.....and so forth for all of the CQS tasks
SETROPTS CLASSACT(FACILITY)

To secure access to CQS logstreams:

RDEFINE FACILITY LOGSTRM log.stream.name UACC(NONE)
PERMIT log.stream.name CLASS(LOGSTRM) ID(cqsid1) ACCESS(UPDATE)
.....and so forth for all of the CQS tasks
SETROPTS CLASSACT(LOGSTRM)
8) Allocate CQS data sets.

We must now allocate the data sets needed for CQS checkpoint and also the Structure Recovery Data Sets (SRDS’s). The following is an example of the job to perform this function.

Example:

```plaintext
//DEFCQSDS JOB MSGCLASS=H,MSGLEVEL=(1,1),REGION=640K,
//STEP01 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
DEFINE CLUSTER                                            -
  (NAME (IMSPSA.IM1A.MSG.CHKPT)                         -
   CYL(1,1) VOL(IMS001) NONINDEXED SHAREOPTIONS (2,3)  -
   RECSZ(505,505) REUSE CISZ (512))
DEFINE CLUSTER                                            -
  (NAME (IMSPSA.IM1A.EMH.CHKPT)                         -
   CYL(1,1) VOL(IMS001) NONINDEXED SHAREOPTIONS (2,3)  -
   RECSZ(505,505) REUSE CISZ (512))
DEFINE CLUSTER                                            -
  (NAME (IMSPSA.IM2A.MSG.CHKPT)                         -
   CYL(1,1) VOL(IMS002) NONINDEXED SHAREOPTIONS (2,3)  -
   RECSZ(505,505) REUSE CISZ (512))
DEFINE CLUSTER                                            -
  (NAME (IMSPSA.IM2A.EMH.CHKPT)                         -
   CYL(1,1) VOL(IMS002) NONINDEXED SHAREOPTIONS (2,3)  -
   RECSZ(505,505) REUSE CISZ (512))
DEFINE CLUSTER                                            -
  (NAME (IMSPSA.IM0A.MSG.SRDS1)                         -
   CYL(100,20) VOL(IMS003) NONINDEXED SHAREOPTIONS (2,3) -
   RECSZ(32761,32761) REUSE CISZ (32768))
DEFINE CLUSTER                                            -
  (NAME (IMSPSA.IM0A.MSG.SRDS2)                         -
   CYL(100,20) VOL(IMS004) NONINDEXED SHAREOPTIONS (2,3) -
   RECSZ(32761,32761) REUSE CISZ (32768))
DEFINE CLUSTER                                            -
  (NAME (IMSPSA.IM0A.EMH.SRDS1)                         -
   CYL(50,10) VOL(IMS003) NONINDEXED SHAREOPTIONS (2,3) -
   RECSZ(32761,32761) REUSE CISZ (32768))
DEFINE CLUSTER                                            -
  (NAME (IMSPSA.IM0A.EMH.SRDS2)                         -
   CYL(50,10) VOL(IMS004) NONINDEXED SHAREOPTIONS (2,3) -
   RECSZ(32761,32761) REUSE CISZ (32768))
```

The names of the checkpoint data sets must match those specified in the CQSSLxxx members as defined previously. The names of the SRDS’s must match those specified in the CQSSGxxx member. Except for the size and volume (and the name of course) the other values should be specified as shown in the example.
Performance Considerations

This section addresses some of the performance questions that might arise before, during, and after implementation of the shared message queues function.

Testing
The value of testing cannot be over emphasized. Functional and recovery scenarios should be tested as well as stress testing to help wring out any problems that might occur in your specific environment. There is one known issue that has been encountered during the testing with shared message queues that you should understand. For message integrity reasons there is an additional logger check-write and a wait-for-write in the transaction path. In a production environment the impact of these will likely not be noticed. However, in a test environment where you might setup input from a single source and process through a single region, you might see significant throughput degradation compared to non-shared message queues due to these logger functions. This degradation is to be expected in this specific environment and can be minimized by creating a test with multiple input sources and multiple regions to process the transactions. Another situation to be careful of in a test environment is the possibility of having the CF LPAR using dynamic dispatching. This can cause very long CF response times, making overhead and response times seem much worse than normal.

CPU Impact
The CPU overhead of shared message queues is also quite variable depending on the level of software and hardware that you are using. For best overall performance, be sure that your CF hardware is the same level or higher than the LPAR running your OS/390 or z/OS in order to minimize the impact to host CPU time. CF accesses may be done synchronously where the host CPU is considered busy for the duration of the process or asynchronous, such that the host CP is made available for other work. Most IMS requests are issued as synchronous but the XES component of MVS will change or convert some of these requests based on a number of internal algorithms that are constantly changing. In any case, you can estimate the CPU overhead by looking at the synchronous CF response time to a LIST structure and multiplying this value (add 30% or so for actual CPU processing of the request in the host) times the number of requests per transaction. The number of requests per transaction will vary somewhat depending on a number of factors but a safe number to use is 12. This includes 8 to the shared message queue structure and 4 to the MVS logger structure. So take this result and add it to your total IMS CPU per transaction to estimate the additional cost.

Sysplex Processing Option
This option applies only to Fast Path EMH processing. It is implemented with the Fast Path input edit routine DBFHAGU0 (see the IMS Customization Guide for more information). There are 3 possible Sysplex Processing Codes that you may specify:
1. **Local first** - This is the default, it specifies that IMS should try to process the transaction on the inputting system first without placing it on the shared message queue, but if the transaction can not be processed immediately, then IMS should place it on the shared queue for any system to process. This option is probably the best selection to optimize capacity, performance, and availability.
2. **Local Only** - This option tells IMS to allow only local processing for this transaction. If resources (such as an IFP region), are not available, then the normal ‘routing code not available’ message is issued. Local Only effectively bypasses the use of shared message queues for this transaction.

3. **Global Only** - This option tells IMS to place all messages on the shared message queue regardless of whether or not they can be processed locally. Note that the application that processes this transaction must be active on at least one of the shared message queues systems for this option. Otherwise, the message is discarded and an error message is sent to the user.

**False Scheduling**
False scheduling may occur in a shared message queues environment. A false schedule is when IMS is notified that there is work to do and tries to schedule a region to handle it, but finds that the transaction is no longer on the queue because some other IMS has already retrieved it. Prior to some changes in the scheduling algorithms, it was possible for false scheduling to be fairly disruptive. However, with current maintenance this should no longer be a serious problem. False scheduling will still occur occasionally but should not be disruptive to the system. In shared message queues mode the use of PWFI and class scheduling may reduce any impact of false scheduling as well as improve overall performance. Remember, the best schedule is the one that never happens.

**Parallel Scheduling**
Parallel Scheduling is performed in the SMQ environment but is handled slightly differently. Since IMS does not know the number of messages which exist on the shared message queues, the standard algorithm will not have any effect. Instead, with shared message queues, IMS attempts to perform this function by using the number of consecutive successful GU’s to the queue instead of the number of messages on the queue. The end result should be similar to non-shared message queues but beware of this difference, especially in a test environment where you may set up artificial test cases.

**Structure Checkpoints**
Structure checkpoints are functionally equivalent to SNAPQ checkpoints in non-shared message queues. The data currently in the message queue is read from the coupling facility structure and written to the Structure Recovery Data Sets (SRDS). There is a pair of these data sets as mentioned before and they are written to in alternating sequence. CQS log data will be kept from the oldest structure checkpoint to facilitate structure recovery in the event of a failure. Data older than the oldest checkpoint will be marked for deletion from the CQS logstream. This minimizes the amount of data which must be kept on DASD. As you can see, the frequency of a structure checkpoint has a direct relation to the amount of CQS logstream data that must be kept. When a structure checkpoint is initiated, the CQS’s must all be quiesced and there will be activity to the CFRM couple data sets to accommodate this function. This DASD activity normally constitutes most of the delay during these checkpoints. As such, these checkpoints should be considered performance critical and allocated on appropriate DASD. Since all IMS (CQS) subsystems will be quiesced for as much as a few seconds, you should consider this in selecting your structure checkpoint frequency and also consider staggering the checkpoints for full function and fast path.
Structure Duplexing
This function is implemented on a structure by structure basis as defined in the CFRM policy described earlier in this document. Shared message queues structures including the MVS logger structures, can take advantage of this function. While it would take multiple failures to cause loss of shared message queues data even without duplexing you might want to implement this function as an additional safeguard. Duplexing is not without cost, however. In addition to the hardware costs there is the additional response time associated with each request to the shared message queues structures. The additional time for each request could be considerable and therefore have an impact on both transaction response times and host CPU cost. As with any new function, you should evaluate the need for this function carefully and understand the impact to your environment by performing tests and measuring the impact.

MVS Logger
Various parameters have been discussed already regarding specification of structure size, logstream size, etc. Additional considerations are discussed in this section. First of all, remember that the structure size has a relationship to data space usage on the host CPU. The larger the structure, the larger the amount of storage on the host required. For this reason you should carefully consider the amount of data being logged per second, the frequency of offloads, the structure size, and the host storage demand, since all of these functions are interrelated. If you allocate too large a structure size, then you could cause excessive demands that result in paging and subsequent poor performance. As mentioned in the CFRM Policy section, the CFSIZER always returns a size of 50MB for the logger structures. This allows for 35MB of data to be written before offload will be invoked (assuming a 70% threshold) which means that at total logging rate for all shared message queues members of 5MB per second, the offloads would occur about every 7 seconds. This is not necessarily bad, but allows little time for delay before the structure becomes full and processing is stopped. In this example, you might want to consider a structure size somewhat larger to allow more flexibility, but remember to make sure that you have sufficient real memory to avoid paging.

COMMANDS and Monitors
Some IMS commands, especially those that use the keyword MSGAGE, can have a significant impact on activity to the SMQ structures. This impact will be greater if there are large numbers of messages on the queues. While these commands might be necessary at times to keep track of the status of messages and the queues, you should keep in mind that the more frequently you use these commands, the more likely that they could cause interference with normal message traffic. Also be aware that some online monitors will either issue these commands or invoke the code to perform the function. Care should be taken when specifying options in those monitors to avoid the over use of these commands.
Monitoring

RMF provides valuable information about the number of accesses and response times for requests to the shared message queues structures. Here is a sample RMF CF report.

--- COUPLING FACILITY NAME = CF1
TOTAL SAMPLES(AVG) = 898 (MAX) = 899 (MIN) = 896
---

--- COUPLING FACILITY USAGE SUMMARY
---
--- PROCESSOR SUMMARY
---
--- COUPLING FACILITY STRUCTURE ACTIVITY
---

The Coupling Facility Structure Activity report above shows the average service times for both SYNC and ASYNC requests and the numbers of requests of each type per second. While not shown here another report for a time with less activity shows much better service times. Also, you can see in the Coupling Facility Usage Summary report that the average CF utilization is 40%. This is fairly high for a CF processor and is probably the cause of the high service times. Either moving some structures to another CF or adding a processor to this CF will most likely lower the service times and result in better IMS performance.
Summary

Well, at this point you can see that the actual implementation steps to migrate to shared message queues are fairly easy to follow. The hardest part is figuring out the various parameter specifications but hopefully, you now know where to find the information you need to make decisions and have an idea of how those decisions might impact the performance and recoverability of the shared message queues environment. The good news is that many of the variables such as size of structures, duplexing support, and others may be changed dynamically while the system is running, so even if you misjudge some value it may not require an outage to change. Just remember to test thoroughly and document recovery procedures. Once implemented, monitor on a regular basis to understand changes in your workload and plan for any system changes before they become a problem.