CICS Transaction Server for z/OS Version 2.1

CICS EJB™ Configuration Utility

Robert Harris,
CICS Technical Strategist,
IBM Hursley.

Issued:	01 August 2001
Revision Date:	01 August 2001
Previous Revision Date:	None
Review Date:	As Required

Take Note!

Before using this document be sure to read the general information under "Notices".

First Edition, August 2001.

Notices:

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

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

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

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

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

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

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

Trademarks:

The following are Trademarks of International Business Machines Corporation in the United States, in other countries, or both:

3090	ACF/VTAM	AD/Cycle
AFP	AIX	AnyNet
Application System/400	APPN	AS/400
AT	BookManager	C Set++
C/370	C/MVS	CBIPO
CBPDO	CICS	CICS/400
CICS/6000	CICS/ESA	CICS/MVS
CICS OS/2	CICS TS	CICS/VM
CICS/VSE	CICSPlex	CICSPlex SM
COBOL/370	COBOL/2	Common User Access
CUA	DATABASE 2	DB2
DFSMS	DFSMS/MVS	DFSMSdfp
DFSMSdss	DFSMShsm	DFSORT
DXT	eNetwork	Enterprise Systems Architecture/370
Enterprise Systems Architecture/390	ES/3090	ESA/370
ESA/390	ES/9000	ESCON
GDDM	HiperBatch	Hiperspace
InfoWindow	IBM	IBMLink
IMS	IMS/ESA	Language Environment
MQ	MQSeries	MVS
MVS/DFP	MVS/ESA	MVS Parallel Sysplex
MVS/SP	MVS/XA	Multiprise
NetView	OpenEdition	OS/2
OS/390	OS/400	Processor Resource/Systems Manager
Parallel Sysplex	PR/SM	Presentation Manager
RACF	Resource Measurement Facility	RETAIN
RISC System/6000	RMF	RT
S/370	S/390	SAA
SQL/DS	SP	System/36
System/38	System/360	System/370
System/390	SystemView	Systems Application Architecture
VisualAge	VSE/ESA	VTAM
WebExplorer	z/OS	

UNIX is a registered Trademark in the United States and other countries licensed exclusively through X/Open Company Limited

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

INTEL is a registered trademark of Intel Corporation, in the United States, or other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, or other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Summary of amendments

Date Of Change	Change made to document
01/08/2001	Document Creation in SupportPac CCE1

Reference Material and Bibliography:

This document uses a short reference to the following documentation:

Short reference	Book Title
EJB Book	Sun Microsystems: Enterprise JavaBeans™ Specification 2.0
CICS Java Book	Java™ applications in CICS SC34-5881
CICS RDO Book	CICS Resource Definition Guide SC34-5722
CICS SDG	CICS System Definition Guide SC34-5725
CICS CG	CICS Customization Guide SC34-5706
CICS APR	CICS Application Programming Reference SC34-5702
CICS SPR	CICS System Programming Reference SC34-5726
JVM Book	New IBM Technology featuring Persistent Reusable Java Virtual Machines SC34-5881
OE Command Book	OS/390 UNIX System Services Command Reference SC28-1892
Pipe Book	OS/390 Batch PipeWorks Users Guide SA22-7457

Other SupportPacs:

The following SupportPacs contain information relevant to this SupportPac:

SupportPac	Relevance
CDE1	Information on Enterprise Bean RDO Resources, how they interact and Enterprise Bean configuration information

Preface:

This SupportPac consists of a REXX Exec that displays information relating to the Enterprise JavaBeans™ (EJB™) configuration of a CICS Transaction Server for z/OS Version 2.1.

This SupportPac contains information on:

- CICS EJB related resources and their relationships
  - CorbaServer
  - DJar
  - RequestModel
  - TCPIPService
  - Transaction
  - Program
- EJB Configuration information
  - DFHJVM PDS
  - HFS definition files
- Accessing HFS files from within a Rexx/MVS Exec
- Running DFHCSDUP from within a Rexx Exec

The information and code in this SupportPac is **only** applicable to CICS TS 2.1. It is not applicable to other CICS releases.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1: The Concept

CICS Transaction Server for z/OS Version 2.1 introduced several new RDO definitions for use with Enterprise JavaBeans function. These types interact in a complex fashion and CICS does not provide any checking for these resources nor the ability to display the current EJB configuration.

This SupportPac provides a Rexx/MVS Exec that can display the Enterprise Bean Configuration for a CICS region. This display is **not** the configuration in a running CICS region but that at region startup. However, the output is good enough for a view of what is going on.

What you get is a listing showing each CorbaServer and the Resources derived from it:

- CorbaServer
  - The name of the Shelf
  - The TCPIPService definitions used for access
- DJar
  - The name of the HFS file
  - The contents of the HFS file represented by the DJar Resource
- Request Model
  - The criteria for Matching
  - The Transaction and Program which will run on a match
  - The DFHJVM member used to define the EJB properties
  - The HFS file representing additional properties for the CorbaServer

This SupportPac will **only** work for the CICS TS 2.1 environment: it will not work for any other release of CICS.

1.2: CICS Resource Definitions for EJBs

This section summarises the RDO definitions used by CICS for EJB access. See the *CICS RDO Book* for full details. SupportPac CDE1 contains an explanation of these relationships. A fuller description is given in *Section 3 ”Background on EJB and CICS Resource Definition” on page 8*.

1.2.1: CorbaServer

The Bean resides within the scope of a given CorbaServer definition. The `JNDIPrefix` is that quoted in the JNDI lookup step. The Port Number and the Protocol must be reflected in an installed TCPIPSERVICE definition.

1.2.2: DJar

The requested Bean must have been installed within the CorbaServer by a DJar definition that contains the actual DJar HFS file name.

1.2.3: TCPIPSERVICE

The port number used for Client access (which is returned within the JNDI lookup) must be defined in an installed TCPIPSERVICE definition.

1.2.4: RequestModel

The RequestModel definition ties up a Bean (and can be as specific as a method name) within a CorbaServer to a CICS TransactionID, thence to a CICS program (representing the JVM) and so to a JVMPROPS entry.

1.2.5: JVMPROPS

This defines the properties for the JVM associated with the CorbaServer and points to an additional HFS file containing further definitions.

Chapter 2: Installation

2.1: Pre-requisites

The code in this SupportPac uses various supplied CICS programs. These must have been installed and made available for use by TSO Batch.

2.1.1: DFHCSDUP

The Batch RDO program must be available for TSO Batch usage.

2.1.2: DFH£FORA

The `DFH£FORA` DFHCSDUP Batch exit program must have been compiled and be available for TSO Batch Usage. See the *CICS CG Book* under the Batch Exit section of the DFHCSDUP Chapter for compilation information.

If this program is **not** called `DFH£FORA` (for example, it might be called `DFH$FORA` in your installation) you will have to change the EJBLOOK to use the correct name as described in Section 2.3 ”Changing the Exec” on page 5.

2.2: Installing the SupportPac

2.2.1: Unpacking the ZIP file

Unzip `the CCE1.ZIP` file. It contains:

- `CCE1TXTC.BIN` the Rexx/MVS Exec that provides the function for this SupportPac in a source MVS PDS format

The SupportPac contains a single Rexx/MVS Exec called `EJBLOOK`. You may need to amend this Exec as described in Section 2.3 ”Changing the Exec” on page 5.

2.2.2: Copying the PDS to MVS

You should copy the unloaded MVS PDS **CCE1TXTC.BIN** to MVS.

You can do this operation via file transfer under TSO. Follow these instructions exactly or else the transfer will not work (It is assumed that IBM Personal Communications is being used as the 3270 emulator).

- Define (if not already present) a transfer type with the ASCII, CRLF and APPEND checkboxes unselected, the LRECL set to 80 with the transfer using FIXED length records; I call this the LOADLIB transfer type
- Transfer the `CCE1TXTC.BIN` file to MVS with a file name of CCE1TXTC using the LOADLIB transfer type. This will create a flat file called `CCE1TXTC` in MVS
- In TSO, issue a RECEIVE INDSN(CCE1TXTC); when prompted for a dataset name reply DSN(CCE1TXTP). A PDS called CCE1TXTP should be created in MVS containing the EJBLOOK Rexx/MVS Exec.

Alternatively, you can use FTP to transfer the PDS.

- Use the `FTP` command to start a session to the MVS region
  - Issue a `BINARY` command to prevent corruption of the data
  - Issue a `QUOTE SITE RECFM=FB` to transfer in the correct layout
  - Issue a `QUOTE SITE LRECL=80` to transfer in the correct layout
  - Send the file via `PUT CCE1TXTC.BIN CCE1TXTC`
- In TSO, issue a RECEIVE INDSN(CCE1TXTC); when prompted for a dataset name reply DSN(CCE1TXTP). A PDS called CCE1TXTP should be created in MVS containing the source of the EJBLOOK Rexx/MVS Exec

Once this `CCE1TXTP` PDS has been created, copy the `EJBLOOK` member to your CLIST library.

2.3: Changing the Exec

EJBLOOK uses the DFH£FORA DFHCSDUP Batch Exit program suppplied by CICS. If this program has been renamed (for example, to DFH$FORA) in your installation, you must change EJBLOOK to use the correctly named program.

In the **docsd** routine, there is a statement underneath a big comment assigning the variable wantexit. Change this assignment to the correctly named program.

```
docsd: procedure expose............
.......
/***********************************************/
/* Name the CSD Batch Exit Program             */
/*                                             */
/*      You may need to change this if it      */
/*      is not called DFH£FORA in your         */
/*      system.                                */
/*                                             */
/***********************************************/

wantexit = 'DFH£FORA'     <------Change this to the correct name for
                                 your region
```

2.4: Running the Exec

2.4.1: JCL

The JCL required to run EJBLOOK looks like:

```
//RUN      EXEC PGM=IKJEFT01,REGION=0M
//SYSPROC  DD  DSN=clist.library,DISP=SHR
//STEPLIB  DD  DSN=cics.loadlib,DISP=SHR
//DFHCSD   DD  DSN=RHARRI1.IYCKRAH5.CSD,DISP=SHR
//DFHJVM   DD  DSN=RHARRI1.IYCKRAH5.SDFHENV,DISP=SHR
//FOROUT   DD  DSN=&&FOROUT,DISP=(NEW,PASS),
//             UNIT=SYSDA,
//             SPACE=(CYL,(10,1)),
//             DCB=(RECFM=F,LRECL=1280,BLKSIZE=1280)
//SYSIN    DD  DSN=&&SYSIN,DISP=(NEW,PASS),
//             UNIT=SYSDA,
//             SPACE=(CYL,(10,1)),
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(100,10))
//SYSPRINT DD  DUMMY
//SYSUDUMP DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*,DCB=(RECFM=F,LRECL=132,BLKSIZE=132)
//SYSTSIN  DD  *
EJBLOOK rahlist rahlstej dfhlist
/*
```

SYSPROC	One of the CLIST libraries in this concatenation should contain the EJBLOOK exec
STEPLIB	One of the load libraries in this concatenation should contain DFHCSDUP and DFH£FORA
DFHCSD	This DDname should define the CSD being processed for the region
DFHJVM	This DDname should define the PDS being used to hold CICS EJB Configuration information for the region
FOROUT	This must be declared as a temporary file and must only have the given DCB (used by DFH£FORA)
SYSIN	This should be defined as a temporary file with the supplied DCB
SYSPRINT	should be dummied out (or output sent to SYSOUT=Z)

2.4.2: Parameters

`EJBLOOK` is called with the names of the RDO **Lists** used by the CICS region. Specify these lists in the same order as they are on the SIT `GRPLIST` setting (these list names are processed in Upper Case).

```
EJBLOOK rahlist rahlstej dfhlist
```

2.4.3: Authorisations

The Userid which runs this Exec must:

- be OE enabled
- have (R,W) access to the CSD
- have (R) access to the DFHJVM PDS
- have (R,W) access to the `/tmp/` directory
- have (R) access to hfs files named in DFHJVM

Chapter 3: Background on EJB and CICS Resource Definition

3.1: Concepts

The *CICS Java* and the *CICS RDO* books define the CICS Resources used by EJB processing.

RequestModels are used to set the properties of the execution EJB environment. In CICS terms, they set the CICS Transaction that will be used to execute EJB function. In other words, a RequestModel is used to set the Transaction that will be used to execute function within a JVM which has the required behaviour.

3.1.1: Transaction naming

The Transaction naming part of a RequestModel is not very important for the function of the Enterprise Bean - whatever transaction is used the Bean will be executed. What this Transaction naming provides is the ability to use standard CICS tuning, measurement and resource control on a JVM execution. This Transaction-setting is also used to route requests to CICS regions.

The upshot of this is that the CICS Transaction named in the RequestModel can be associated with a TCLASS and so the standard CICS scheduling mechanisms apply to limit activity.

3.1.2: JVM execution properties

The execution properties of the JVM used to execute the Bean are also set via the CICS TransId. This is important because it provides the ability to tune the JVM for best performance of the Bean.

In effect, the definitions required for the operation of the CorbaServer are keyed via a matching RequestModel. These definitions are contained in a member of the DFHJVM PDS and an HFS file.

3.2: RequestModels and Routing

A method call will arrive from the Client via a TCPIPSERVICE. The TCPIPSERVICE names a CICS Transaction that will receive the flow. This transaction, by default CIRR, uses the DFHIIRS program. This flow is called a **Request Stream**.

DFHIIRS searches through the RequestModels, matches on one and then uses the named CICS TransId (defaulting to CIRP) (which runs DFJIIRP) to access the Bean instance upon which the method is run.

The CIRR/DFHIIRS processing is referred to as **Request Stream Processing** and the CIRP/DFJIIRP processing as **Request Processing**.

Once the method request has been executed, the results are returned from the Request Processor to the Request Receiver via a **Response Stream**. The Request Receiver then sends this information to the client without using the TCPIPSERVICE. Figure 1, 'Request and Response Streams' shows these flows.

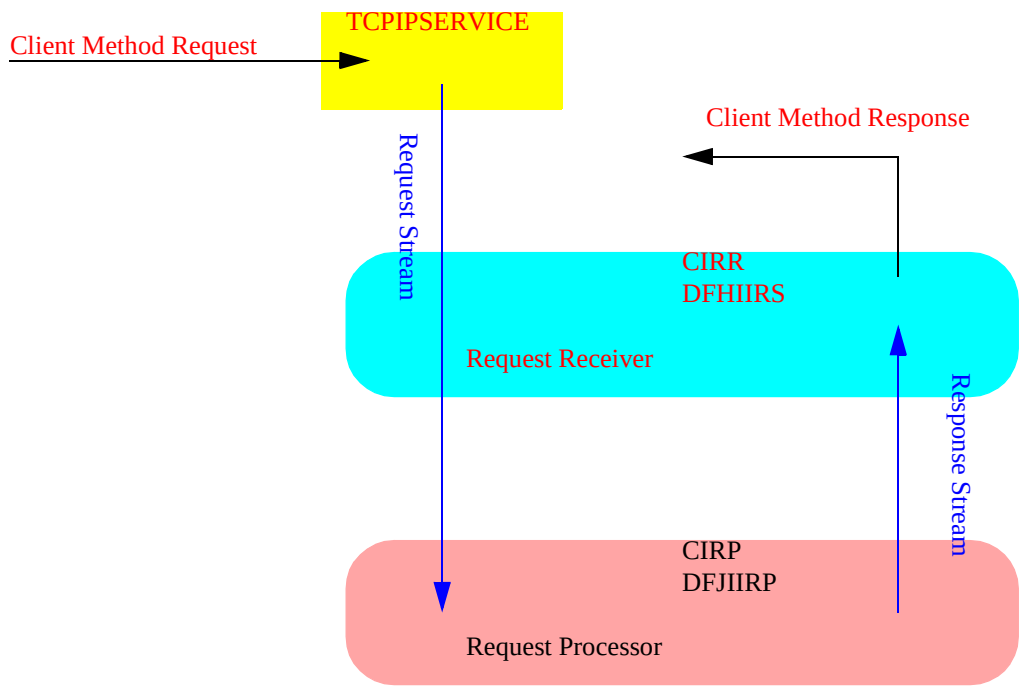

Figure 1: Request and Response Streams

The TCPIPSERVICE and the Request Receiver are in the same CICS region, but the Request Processor can reside in another CICS region (within the same Plex). These Request Processor regions are linked to the Request Receiver regions via standard MRO Connection definitions, and standard CICS load balancing techniques are used to select which region is most appropriate for execution (if there is a choice).

The CICS TransId selected via the RequestModel controls this routing. Figure 2, 'Routing and Request Models' shows how this function works if different CorbaServers control the routing.

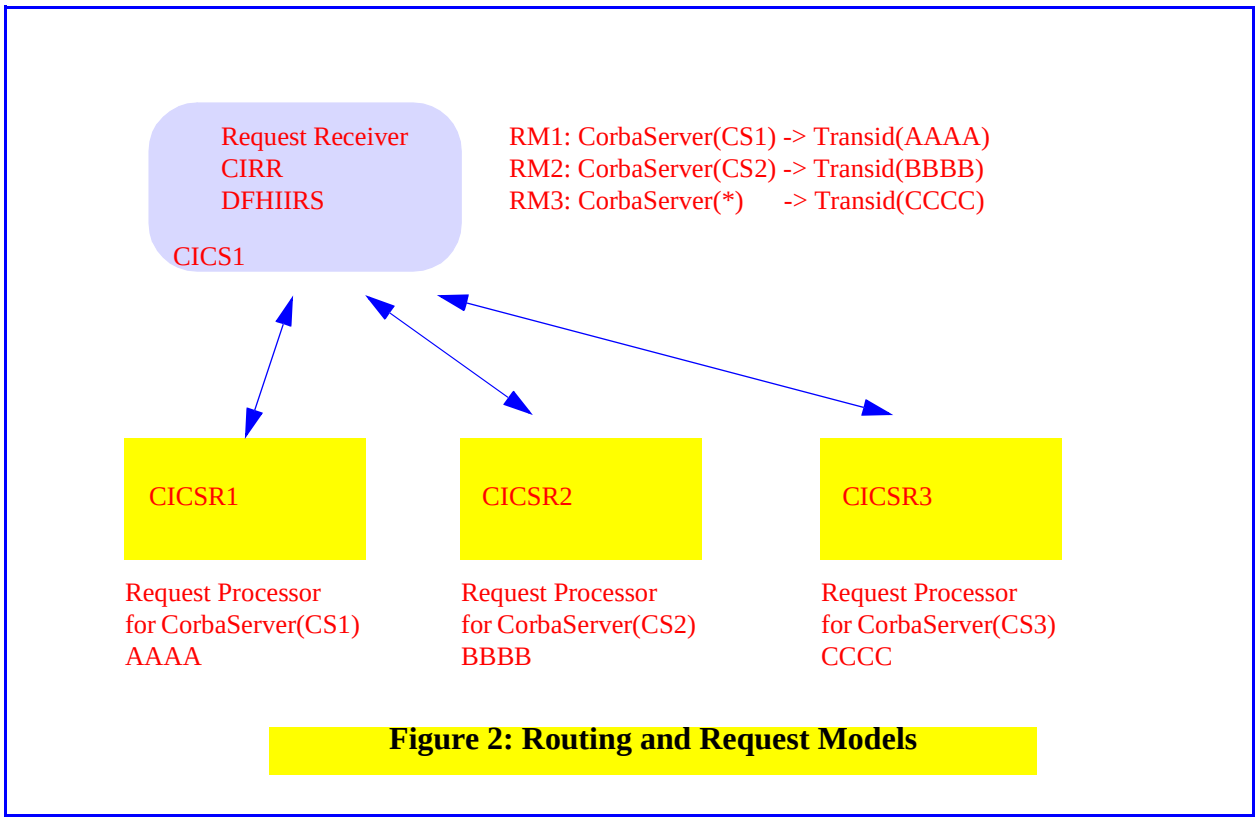

Figure 2: Routing and Request Models

To enable this routing, in addition to the RequestModel definitions, Transactions AAAA, BBBB and CCCC were defined with DFJIIRP as the executing program. The RequestModels look like:

```
RequestModel    : RM1
CorbaServer     : CS1
TYpe            : Ejb               Corba | Ejb | Generic
Beanname        : *
INTFacetype     : Both              Both | Home | Remote
OPeration       : *
TRansid         : AAAA
```

The actual region selected for execution is selected via the DFHDSRP user-replaceable module. See *Routing Method Requests for Enterprise Beans and CORBA Stateless objects* in the *CICS CG* for information on routing selection.

3.3: Transaction and Program definitions

The RequestModel selects a CICS TransId under which the JVM will run. The Transaction definition points to a Program Definition which names the Java Class that is to be run and a name used to determine the properties of the JVM. Figure 3, 'RequestModels and RDO definitions' shows the relationships between the CICS RDO definitions

.

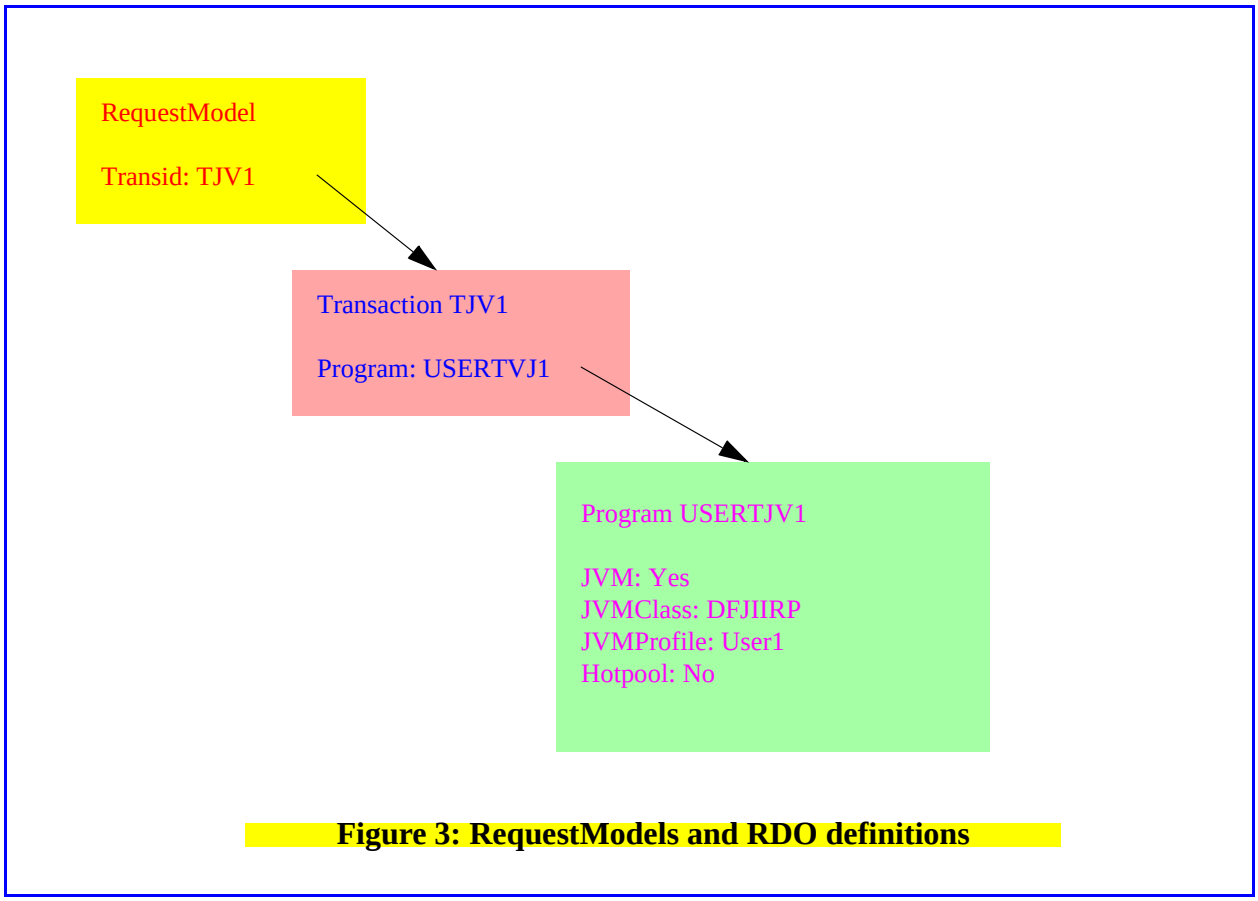

Figure 3: RequestModels and RDO definitions

The JVM setting is `YES` and the JVMClass name is always `DFJIIRP` (The CICS Request Processor function).

The JVM Profile setting of the PROGRAM definition is the name of a member in the DFHJVM PDS. This member contains parameters used to initialise the JVM. One of these, in turn, this points to an HFS file which contains the properties of the JVM.

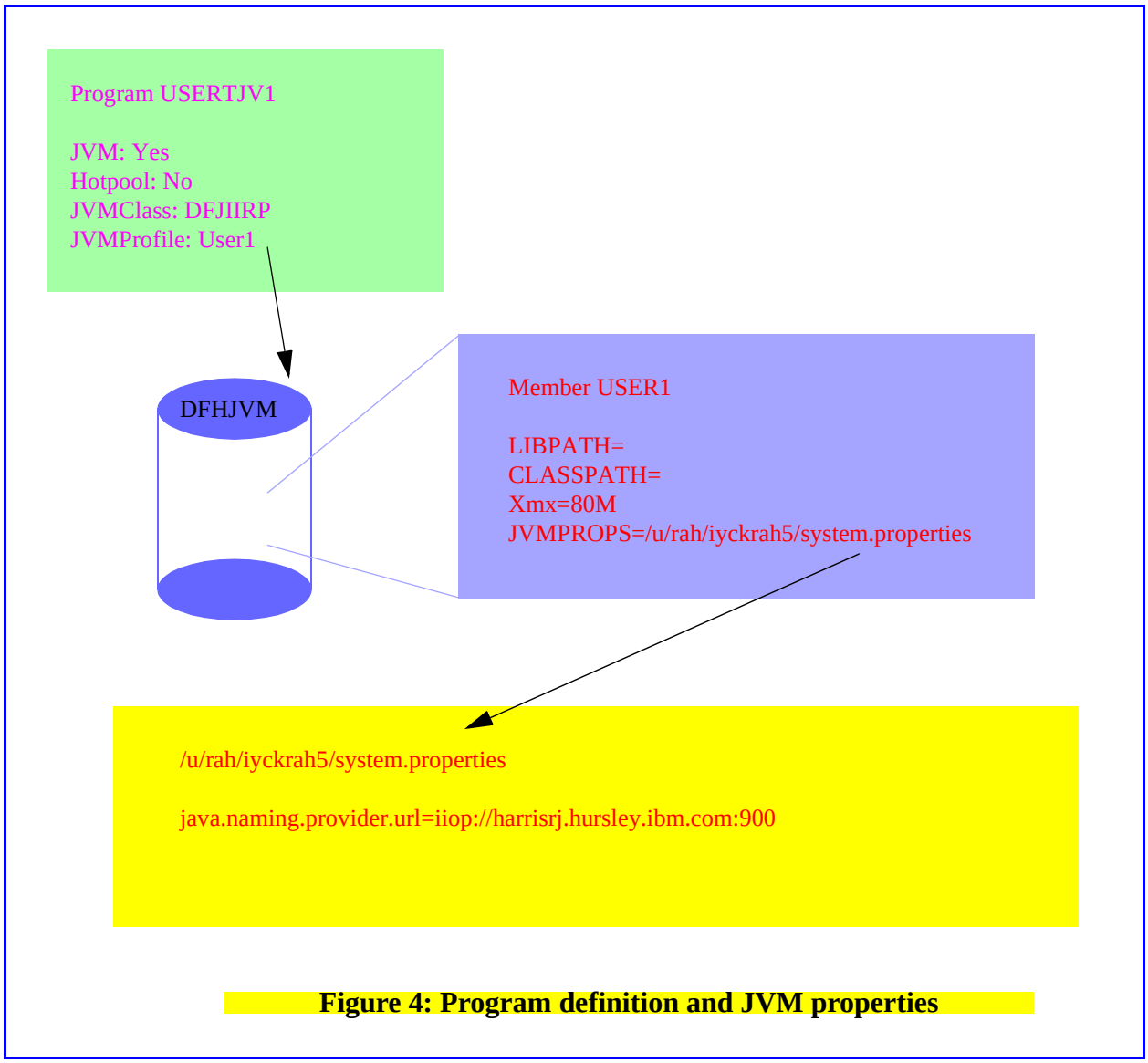

Figure 4: Program definition and JVM properties

In effect, the definitions required for the operation of the CorbaServer are keyed via a matching RequestModel. These definitions are contained in a member of the DFHJVM PDS and an HFS file.

3.4: The DFHJVM member

The DFHJVM Member contains definitions for the JVM that CICS will use. They control the operation of the JVM both from a CICS viewpoint and the view of the JVM itself.

These settings are described in the JVM Initialisation section of the *CICS Java Book* and the values passed to the JVM in the *JVM Book*.

You can have multiple members in the `DFHJVM` PDS, each of which sets different characteristics. The only **required** member is DFHJVMPR which acts as the default JVMProfile.

Each application suite running within a JVM will have its own Storage characteristics. Use of a different DFHJVM member provides the ability to tune these suites.

It is **strongly** suggested that you define a `DFHJVMPR` exclusively for the use of CICS-supplied activities. It is further recommended that you have different DFHJVM members for EJB activity and for native Java code function.

3.4.1: Required DFHJVMPR entry

DFHJVMPR must be defined as a member in the `DFHJVM` PDS. This is because this profile is used by all CICS functions that use Java processing. These operations do not go through RequestModel processing, so there is no opportunity to alter this name.

Similarly, `DFHJVMPR` is used as the default profile if no RequestModel processing defines a member-name or if no JVMPROFILE definition is supplied on the RDO Program definition for the Java program. This is probably not what is required for use. Therefore, one should always code an installation-specific JVMPROFILE for all Java programs.

3.5: RequestModels and JVM Tuning

The RequestModel, via the indirections described above, sets the properties of a JVM. These properties can, therefore, be managed to provide the best execution environment for the Bean instance.

3.5.1: CorbaServers

The RequestModel mechanism sets the properties of an individual CorbaServer. A given CorbaServer can have identical properties to another CorbaServer. The lookup arrangement is:

```
CorbaServer ->
              RequestModel ->
                             Transaction ->
                                           Program
                                             JVMProfile ->
                                                         DFHJVM
                                                           JVMPROPS->HFS
                                                           settings
```

The best way of achieving a distinct set of properties for a set of CorbaServers is to associate a different RequestModel with each CorbaServer and then use distinct DFHJVM members and HFS files to set the properties. Doing things this will permit an easy change to be made if the CorbaServers are subsequently to behave differently.

```
CorbaServer(RH1)                :
       RequestModel(RAH1)       : CorbaServer(RH1) Transid(TRH1)
           Transaction(TRH1)    : Program(PRAH1)
               Program(PRAH1)   : JVMProfile(RAH1)

CorbaServer(RH2)                :
       RequestModel(RAH2)       : CorbaServer(RH2) Transid(TRH2)
           Transaction(TRH2)    : Program(PRAH2)
               Program(PRAH2)   : JVMProfile(RAH2)

DFHJVM(RAH1) : JVMPROPS=/u/rah/props/RH1/system.properties  ... ... ...
DFHJVM(RAH2) : JVMPROPS=/u/rah/props/RH2/system.properties  ... ... ...

/u/rah/props/RH1/system.properties : ... ... ... ...
/u/rah/props/RH2/system.properties : ... ... ... ...
```

3.6: The JVMPROPS file

The JVMPROPs file is an HFS file that contains items passed to the JVM so they can be accessed via the standard environmental mechanisms. The following items are defined in this HFS file:

- Identification of the JNDI Server
- Naming the JNDI Factory
- Setting the JDBC Drivers
- Defining the Loader Security policy

Chapter 4: What the Exec produces

You get a listing showing the relationships associated with all CorbaServers. Each entry will look something like:

```
Extracting EJB Information from Lists: RAHLIST RAHLSTEJ DFHLIST

Corb RAH1 in group RAHEJB
     saves items on shelf /u/rharri1/shelf
     with a JNDI prefix of rah
     access via HTTP on port  15001 defined via RAHEJB in group RAHEJB
     access via SSL  on port  15002 defined via RAHEJBS1 in group RAHEJB
     DJar HELLO1 defined in RAHEJB is file /u/rharri1/java/Hello1_GEN.jar
          which contains
           com/ibm/hursley/harris/package1/Hello1.class
           com/ibm/hursley/harris/package1/Hello1Bean.class
           com/ibm/hursley/harris/package1/Hello1Home.class
           META-INF/ejb-jar.xml
           META-INF/ibm-ejb-jar-bnd.xmi
           META-INF/ibm-ejb-jar-ext.xmi
           META-INF/cics-ejb-jar-ext.xmi
           com/ibm/cics/portable/CICSEJBMetaData.class
           com/ibm/cics/portable/CICSSessionHandle.class
           com/ibm/cics/portable/CICSSessionHomeHandle.class
           com/ibm/hursley/harris/package1/EJSHello1HomeBean.class
           com/ibm/hursley/harris/package1/EJSHello1HomeBean.java
           com/ibm/hursley/harris/package1/EJSRemoteHello1.class
           com/ibm/hursley/harris/package1/EJSRemoteHello1.java
           com/ibm/hursley/harris/package1/EJSRemoteHello1Home.class
           com/ibm/hursley/harris/package1/EJSRemoteHello1Home.java
           com/ibm/ejs/container/_EJSWrapper_Stub.class
           com/ibm/ejs/container/_EJSWrapper_Stub.java
           com/ibm/ejs/container/_EJSWrapper_Tie.class
           com/ibm/ejs/container/_EJSWrapper_Tie.java
           com/ibm/hursley/harris/package1/_EJSRemoteHello1Home_Tie.class
           com/ibm/hursley/harris/package1/_EJSRemoteHello1Home_Tie.java
           com/ibm/hursley/harris/package1/_EJSRemoteHello1_Tie.class
           com/ibm/hursley/harris/package1/_EJSRemoteHello1_Tie.java
           com/ibm/hursley/harris/package1/_Hello1Home_Stub.class
           com/ibm/hursley/harris/package1/_Hello1Home_Stub.java
           com/ibm/hursley/harris/package1/_Hello1_Stub.class
           com/ibm/hursley/harris/package1/_Hello1_Stub.java
           com/ibm/websphere/csi/_CSIServant_Stub.class
           com/ibm/websphere/csi/_CSIServant_Stub.java
           com/ibm/websphere/csi/_TransactionalObject_Stub.class
```

```
        com/ibm/websphere/csi/_TransactionalObject_Stub.class
        com/ibm/websphere/csi/_TransactionalObject_Stub.java
        org/omg/stub/javax/ejb/_EJBHome_Stub.class
        org/omg/stub/javax/ejb/_EJBHome_Stub.java
        org/omg/stub/javax/ejb/_EJBObject_Stub.class
        org/omg/stub/javax/ejb/_EJBObject_Stub.java
        org/omg/stub/javax/ejb/_Handle_Stub.class
        org/omg/stub/javax/ejb/_Handle_Stub.java
        org/omg/stub/javax/ejb/_HomeHandle_Stub.class
        org/omg/stub/javax/ejb/_HomeHandle_Stub.java
   ReqM HELLO1 defined in RAHEJB
        matches Bean * Method * to tran RIRP in group RAHSYS
        which is program RAHIIRP in group RAHEJB
        which uses DFHJVM member DFHJVMPR
        #
        # CICS/RAH settings
        #
        #
        # CICS things
        #
        INVOKE_DFHJVMAT=NO
        CICS_DIRECTORY=/usr/lpp/cicsts/satbsf
        JAVA_HOME=/usr/lpp/java130s/J1.3
        WORK_DIR=/u/rharri1/java/log
        JVMPROPS=/u/rharri1/java/system.properties
        LIBPATH=/usr/lpp/cicsts/satbsf/lib\
                :/usr/lpp/cicsts/satbsf/ctg\
                :/usr/lpp/java130s/J1.3\
                :/usr/lpp/java130s/J1.3/bin\
                :/usr/lpp/java130s/J1.3/bin/classic\
                :/usr/lpp/db2610/db2710/lib
        STDIN=/u/rharri1/java/log/dfhjvmin
        STDOUT=/u/rharri1/java/log/dfhjvmout
        STDERR=/u/rharri1/java/log/dfhjvmerr
        TMPREFIX=.:
        TMSUFFIX=/usr/lpp/db2610/db2710/classes/db2sqljruntime.zip:
```

```
#
# Java Standard options
#
CLASSPATH=/u/rharri1/java\
         :/u/rharri1/java/Hello1_GEN.jar\
         :/usr/lpp/cicsts/satbsf/lib\
         :/usr/lpp/cicsts/satbsf/ctg\
         :/usr/lpp/java130s/J1.3\
         :/usr/lpp/java130s/J1.3/bin\
         :/usr/lpp/java130s/J1.3/bin/classic\
         :/usr/lpp/db2610/db2710/lib\
         :.
SHOWVERSION=YES
VERBOSE=YES
#
# Java strange options
#
Xcheck=NO
Xdebug=NO
Xinitacsh=1M
Xinitsh=1M
Xinitsh=1M
Xinitth=1M
Xmaxe=1M
Xmine=1M
Xmaxf=0.6
Xminf=0.3
Xms=10M
Xmx=100M
Xnoagent=NO
Xnoclassgc=NO
Xoss=10M
Xresettable=YES
Xrs=NO
Xss=5M
Xverify=none
#
# End of Settings
#
```

```
which uses system.properties of /u/rharri1/java/system.properties
 #
 # RAH.properties file
 #
 #
 ibm.jvm.shareable.application.class.path=/u/rharri1/java
 ibm.jvm.events.output=ure.log
 ibm.jvm.resettrace.events=on
 ibm.jvm.crossheap.events=true
 ibm.jvm.unresettable.events.level=max
 java.naming.provider.url=iiop://harrisrj.hursley.ibm.com:900
 java.naming.factory.initial=
      com.ibm.ejs.ns.jndi.CNInitialContextFactory
```

Chapter 5: Writing your own functions

You are completely free to use the techniques in the `EJBLOAD` Exec to write your own Execs and display items according to your preferences. You might care to provide a listing showing which members of DFHJVM are used by which CorbaServers or which TCPIPServices do what function.

The *OE Command Book* and *Pipe Book* publications contain information relevant to these routines as well as the *CICS CG*.

5.1: docsd

This routine uses the batch extract functions of DFHCSDUP to get a file which is then parsed for CICS resources. Individual resources are returned in their own stem. variables. It is called with parameters naming the RDO Lists to be processed.

You can add to this routine by defining additional Stem. variables (and exposing them) for the relevant CICS resources. The columns required for the parsing instructions are found at the bottom of the DFH£FORA sample program - and are simply the start columns for the resource.

DFHCSDUP sends a listing out to SYSPRINT which will contain an error relating to the contents of the PARM parameter. This should be ignored.

5.2: dounix

This procedure executes an UNIX command via the `BPXBATCH` TSO routine. It returns the results in the `oe.` stem variable. You should `drop oe.` before calling the routine.

- `oe.out.0` contains the number of lines written to STDOUT, with each line appearing in `oe.out.n`
- `oe.err.0` contains the number of lines written to STDERR, with each line appearing in `oe.err.n`
- Some error messages appear in `oe.zerr` and a return code in `oe.zrc`

This is routine is called with a single Parm of the function to execute (case dependant) prefixed by SH or PGM as appropriate.

Note that this routine allocates temporary HFS files for its function in the `/tmp/` directory and allocates STDIN, STDOUT and STDERR.

5.3: doconfig

This procedure reads all members from the `DFHJVM` DDname. It then parses each one to extract the relevant system.properties file, and then reads this as well. It returns the results in the `iconf.` stem variable. You should `drop iconf.` before calling the routine.

- `iconf.0` contains the number of members in DFHJVM
- `iconf.members` contains the member names as words
- Each member is then returned (n is from 1 to `iconf.0` in the usual Rexx fashion)
  - `iconf.name.n` is the member name
  - `iconf.file.n` is the associated HFS system.properties file name
  - `iconf.mem.n.0` contains the number of lines in the member, with each line appearing in `iconf.mem.n.m`
  - `iconf.hfs.n.0` contains the number of lines in the HFS system.properties file named in the member, with each line appearing in `iconf.hfs.n.m`

This does not get called with any parameter. The interesting thing to note is that a separate test (via dounix) has to be done to determine the existence of the HFS file before it is read to avoid Pipe error messages.

>>>>>>>>>>>>>>>>END of document<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<