IBM WebSphere Transformation Extender

IMS/DC Execution Option

Version 8.1

Note

Before using this information, be sure to read the general information in "Notices" on page 17.

October 2006

This edition of this document applies to IBM WebSphere Transformation Extender Version 8.1; and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, e-mail DTX_doc_feedback@us.ibm.com. We look forward to hearing from you.

Contents

Chapter 1. Introduction to the IMS/DC Execution Option

IMS on z/OS and MVS generally refers to a product providing two functions:

- Transaction management (IMS TM or IMS/DC)
- Database management (IMS/DB)

The IMS/DC Execution Option documentation refers to the transaction management function. Although often referred to as IMS TM, it will be referred to in this document as IMS/DC.

IMS/DC Execution Option overview

The IMS/DC Execution Option provides a means to use the platform API for z/OS and high-level language programs to programmatically call the WebSphere Transformation Extender execution environment. The high-level languages supported on z/OS are C/C++ and COBOL.

The platform API is available as part of the WebSphere Transformation Extender feature. The WebSphere Transformation Extender contains the platform API runtime and example programs for C and COBOL.

Chapter 2. Using the API on z/OS for IMS/DC

This is an introduction to the platform API for z/OS and one of the three supported execution environments, IMS/DC. For information about the other two execution environments, see the following documentation:

Execution environment	Documentation
Batch	Platform API
CICS	Launcher

Requirements

The use of the platform API on z/OS for IMS/DC requires the use of a Language Environment (LE) enabled C/C++ or COBOL compiler. The platform API works with all supported releases of the LE runtime environment.

Calling the Platform API from COBOL

The WebSphere Transformation Extender provides a way to call platform API from COBOL by calling the platform API functions as a DLL. This requires an IBM Host COBOL compiler that provides COBOL DLL support. IBM COBOL for z/OS and VM 2.1 is the earliest host compiler that supports COBOL DLLs.

Calling the Platform API from a COBOL DLL

From a programming perspective, calling a DLL from COBOL is not much different from a COBOL dynamic CALL. The major difference is that the DLL name can be up to 160 characters versus eight (8) for the traditional COBOL call statement. There are some restrictions, such as the program must be reentrant.

The following COBOL compiler options were used to compile all of the COBOL example programs:

```
CBL RENT,DLL,NOEXPORTALL,MAP,LIST,LIB,PGMNAME(M),OPT(STD)
```

For a description of these options, consult the IBM COBOL documentation for z/OS.

Passing function parameters

To call platform API functions as a DLL, you will set up function parameters to pass to the platform API.

The `RunMap` function takes only one parameter, the `ExitParam` structure address. When using the `RunMap` function, the pointer to the map file name and the pointer to the `DataFromApp` are contained in the `ExitParam` structure. `CALLRMAP` was designed to take the extra parameters as a way to minimize the need to manipulate and handle pointers. A major deficiency in COBOL is that the address of a working storage item cannot be assigned to a pointer. The item needs to be in the linkage section.

Setting the address of the map file name requires that either the map file name be passed through the `JCL PARM` statement (linkage section) or that a sub-program be created that takes the address of the map file name as one parameter and passes the address back, in the form of a pointer, in another parameter. It was felt that this sub-program deviation of the `CALLRMAP` function would simplify its use.

Cobol Copybooks

There are two copybooks that are members in the DTX.SDTXSAMP PDS included in the installation of the WebSphere Transformation Extender feature: DTXRMCOB and DTXMQCOB. Combined they make up the COBOL definition of platform API structures. The following list provides descriptions of each copybook:

Name	Description
DTXRMCOB	This copybook contains the COBOL definition for both the `ExitParam` and `Exeopts` structures.
DTXMQCOB	This copybook contains the COBOL definition for the `CARDINFO` structure.

The two copybooks are required because DTXMQCOB must be in the COBOL program's LINKAGE-SECTION and DTXRMCOB must reside in the WORKING-STORAGE section.

For content descriptions of these copybooks, see the Platform API documentation.

The variable names contained in the copybooks are the same as the component names documented in the Platform API documentation, except that the COBOL names are uppercase. In addition, unlike C, a COBOL program cannot contain variables with the same name. Therefore, any variable name containing a hyphen (-) is already defined to the DTXRMCOB copybook.

IMSRUN DLL import definition

The DTXIAEXP file contains the DLL import statements for the platform API functions. The file is required for the LE prelinker step of any COBOL or C/C++ compile job that wants to link in the support for the platform API.

The DTXIAEXP file is a member in the DTX.SDTXSAMP PDS that is included in the installation of the WebSphere Transformation Extender feature. If you are using your own compile and link JCL, add the DTXIAEXP PDS member to the SYSIN DD definition of the LE prelinker step of the compile/link JCL.

IMS map and MDQ file preloader exit

The WebSphere Transformation Extender IMS map and mdq preloader exit is a user exit routine that provides a way for you to preload WebSphere Transformation Extender map and mdq files into memory.

All subsequent access to these maps and mdq files during map execution are from memory and therefore, will not require any file input and output (I/O) processing from these preloaded map and mdq files. This exit routine interacts with the IMSBATCH, DFSMPR, and IMSFP Dependent Region procedures provided in your

IMS installation. It is an optional feature that you can use and is included in the WebSphere Transformation Extender installation.

Before you use the IMS map and mdq file preloader exit, you must define it to IMS. The IMS Dependent Region Preinitialization exit routine, or program, provided with the WebSphere Transformation Extender installation as member DTXIMSIT in the DTX.SDTXLOAD PDS.

To activate IMS Dependent Region Preinitialization user exit routines, they must be identified by an 80-byte record in a DFSINT*xx* member of IMS.PROCLIB, where *xx* is a suffix specified by the PREINIT keyword of the IMSBATCH, DFSMPR, and IMSFP IMS Dependent Region procedures.

For more information about using IMS Dependent Region Preinitialization user exit routines, see the *IMS V9 Customization Guide SC18-7817.*

Activating the IMS map and mdq file preloader exit

This is a summary of the procedures that will activate the WebSphere Transformation Extender IMS map and mdq file preloader exit to preload map and mdq files.

To activate the exit, the following procedures must be completed:

- Create the DFSINT*xx* member in the IMS.PROCLIB, installed in your IMS environment, supplying a valid value to substitute for the *xx* suffix.
- Include an 80-byte record in the DFSINT*xx* member that you created, specifying the DTXIMSIT IMS Dependent Region Preinitialization program.
- Specify the values that you substituted for the *xx* suffix in the PREINIT keyword of the IMS JCL, which will be calling the DTXIMSIT exit.
- Add DD statements in the IMS startup JCL to reference your map and mdq files. You may have already added these DD statements in your testing environment during development.
- Create a file, which must be in fixed-block record format (RECFM=FB), that contains the keywords to specify the map and mdq files that you want IMS to preload. You create these entries as pairings of the keywords and the files that the DTXIMSIT program will read.
- Add a DD statement in the IMS JCL, which will be calling the DTXIMSIT exit, to reference the file you created containing the keyword and file pairings.

The IMS JCL will call the DTXIMSIT exit and preload the specified map and mdq files into memory. The result is that all subsequent access to the map and mdq files during map execution will be from memory, instead of from reading, again, the map and mdq files specified in the DD statement.

Specifying the keywords to preload map files

To identify the map files that you want IMS to preload using theWebSphere Transformation Extender IMS map and mdq file preloader exit, you need to specify keyword entries in the file you created, which the DTXIMSIT program will read.

You specify these entries as pairings of keywords and file references. To activate the exit to preload map files, the DTXIMSIT program reads the file that is associated with the data definition (DD) that you added to the IMS JCL.

Specify the keyword pairings for the map files that you want IMS to preload using the following syntax:

```
MAPNAME=map_to_be_loaded
```

The keyword must be in uppercase. You can specify multiple map files to be preloaded. You can specify one keyword and file pair on separate records or lines, or you can string them together in one record by using a space or comma delimiter between the entries.

There are three ways to specify a map file to be loaded. You must select the syntax that matches the way that your map is specified in the command line or RUNMAP command for the IMS JCL to run the same map file. The following list describes the three ways depending on how the map data set name (DSN) is specified on the data definition (DD) statement on the IMS JCL.

- DD statement specifies your map file as a PDS
  - Keyword pairing syntax: `MAPNAME=`*`ddname`*
  - Use this syntax if the DSN on the DD statement specifies your map file as a PDS.
  - *ddname* is the DDNAME on the DD statement that references your map file.
  - DD statement syntax: `//`*`DDNAME`*` DD DSN=`*`MY.MAP`*`,DISP=SHR`
  - In the following example, the DSN on the DD statement is referencing a map file set up as a PDS.

    ```
    //REVERSE DD DSN=MY.REVERSE.MAP,DISP=SHR
    ```

    You would specify the value for *map_to_be_loaded* as `REVERSE`, and therefore, the keyword pairing would be `MAPNAME=REVERSE`.
- DD statement specifies your map file as a member in a PDS
  - Keyword pairing syntax: `MAPNAME=`*`ddname`*
  - Use this syntax if the DSN on the DD statement specifies your map file as a member of a PDS.
  - *ddname* is the DDNAME on the DD statement that references your map file.
  - DD statement syntax: `//`*`DDNAME`*` DD DSN=`*`MY.MAPLIB`*`(`*`MEMBER`*`),DISP=SHR`
  - In the following example, the DSN on the DD statement is referencing a map file set up as a member of a PDS.

    ```
    //REVERSE DD DSN=MY.MAPLIB(REVERSE),DISP=SHR
    ```

    You would specify the value for *map_file_to_be_loaded* as `REVERSE`, and therefore, the keyword pairing would be `MAPNAME=REVERSE`.
- DD statement specifies your map file as a member in a PDS
  - Keyword pairing syntax: `MAPNAME=`*`ddname`*`(`*`membername`*`)`
  - Use this syntax if the DSN on the DD statement specifies your map file as a member of a PDS.
  - *ddname* is the DDNAME on the DD statement that references your map file.
  - DD statement syntax: `//`*`DDNAME`*` DD DSN=`*`MY.MAPLIB`*`,DISP=SHR`
  - In the following example, the DSN on the DD statement is referencing a map file set up as a member of a PDS.

    ```
    //MAPLIB DD DSN=MY.MAPLIB,DISP=SHR
    ```

    You would specify the value for *map_file_to_be_loaded* as `MAPLIB(REVERSE)`, and therefore, the keyword pairing would be `MAPNAME=MAPLIB(REVERSE)`.

Specifying the keywords to preload mdq files

To identify the mdq files that you want IMS to preload using theWebSphere Transformation Extender IMS map and mdq file preloader exit, you need to specify keyword entries in the file you created, which the DTXIMSIT program will read.

You specify these entries as pairings of keywords and file references. To activate the exit to preload mdq files, the DTXIMSIT program reads the file that is associated with the data definition (DD) that you added to the IMS JCL.

Specify the keyword pairings for the mdq files that you want IMS to preload using the following syntax:

```
MDQNAME=mdq_file_to_be_loaded
```

The keyword must be in uppercase. You can specify multiple mdq files to be preloaded. You can specify one keyword and file pair on separate records or lines, or you can string them together in one record by using a space or comma delimiter between the entries.

There are two ways to specify an mdq file to be loaded. You must select the syntax that matches the way that your mdq file is specified in the command line or RUNMAP command for the IMS JCL to run the same mdq file. The following list describes the two ways depending on how the mdq data set name (DSN) is specified on the data definition (DD) statement of the IMS JCL.

- DD statement specifies your mdq file as a PDS
  - Keyword pairing syntax: `MDQNAME=`*`ddname`*
  - Use this syntax if the DSN on the DD statement specifies your mdq file as a PDS.
  - *ddname* is the DDNAME on the DD statement that references your mdq file.
  - DD statement syntax: `//`*`DDNAME`*` DD DSN=`*`MY.MDQFILE`*`,DISP=SHR`
  - In the following example, the DSN on the DD statement is referencing an mdq file set up as a PDS.

    ```
    //REVMDQ DD DSN=MY.REVERSE.MDQFILE,DISP=SHR
    ```

    You would specify the value for *mdq_file_to_be_loaded* as `REVMDQ`, and therefore, the keyword pairing would be `MDQNAME=REVMDQ`.
- DD statement specifies your mdq file as a member in a PDS
  - Keyword pairing syntax: `MDQNAME=`*`ddname`*
  - Use this syntax if the DSN on the DD statement specifies your mdq file as a member of a PDS.
  - *ddname* is the DDNAME on the DD statement that references your mdq file.
  - DD statement syntax: `//`*`DDNAME`*` DD DSN=`*`MY.MDQLIB`*`(`*`MEMBER`*`),DISP=SHR`
  - In the following example, the DSN on the DD statement is referencing an mdq file set up as a member of a PDS.

    ```
    //REVMDQ DD DSN=MY.MDQLIB(REVERSE),DISP=SHR
    ```

    You would specify the value for *mdq_file_to_be_loaded* as `REVMDQ`, and therefore, the keyword pairing would be `MDQNAME=REVMDQ`.

Platform API examples

The WebSphere Transformation Extender installs a set of examples that demonstrate how to call the platform API. Specific to the platform API are three example programs and three example maps:

Example Programs	**Example Maps**
Example 1 (DTXPSB1)	readmap
Example 2 (DTXPSB2)	writemap
Example 3 (DTXTEST)	primemap

The readme files for each of the examples describe how to install and use the files for that example. Compiled versions of these programs are members of the load library included in theWebSphere Transformation Extender installation and can be run without building them. For more information about these examples, see ″About the examples″.

Chapter 3. Using the examples

This is a description of the sample programs that are installed with the WebSphere Transformation Extender feature that includes the IMS/DC Execution Option, which can be used to run maps.

Note: To run these examples, modifications to the WebSphere Transformation Extender system are required.

IMS/DC examples

During installation of the WebSphere Transformation Extender feature, files for the IMS/DC Execution Option examples are placed in the DTX.SDTXSAMP PDS and the DTX.SDTXMAPS PDS.

The files required to run the examples as well as the program source and the readme files are in the DTX.SDTXSAMP PDS, and the map and type tree source are in the DTX.SDTXMAPS PDS. The readme files (DTXRDRME for the readmap example, DTXWMRME for the writemap example, and DTXPMRME for the primemap example) describe how to use each of the examples. Compiled versions of the programs used in the examples are included in the z/OS IMSDC load library (DTX.SDTXLOAD PDS) and can be run without building them.

To use these examples, IBM WebSphere MQ for MVS/ESA and DB2 UDB for z/OS are required.

Setting up an example z/OS system

You might have already completed some of these modifications on your system outside of the IMS/DC Execution Option requirements. These topics are presented as a reference for those specific modifications required to make the examples work on your system.

The follow topics are presented, containing information about specific modifications that must be completed to make the provided examples work properly:

- ″Updating the IMS External Subsystem Attach Facility Table″
- ″Updating IMS PROCs and JCL″
- ″Making Additional IMS Modifications″
- ″Configuring IMS Transaction Names and DB2 Plans″
- ″Granting Appropriate DB2 Access Privileges to CSQQTRMN″
- ″Performing a PSBGEN″
- ″Defining IBM WebSphere MQ Queues″

Updating the IMS external subsystem attach facility table

IMS needs a list of the other subsystems with which it is to coordinate its actions. For these examples, this list is contained in **IMS.PROCLIB** and is shown below.

```
CSQ1,MQM1,CSQQESMT,,R,
SST=DB2,SSN=DSN6,LIT=SYS1,ESMT=DSNMIN10,RTT=DTXMNASM,REO=Q,CRC=!
```

In this code are entries for both the IBM WebSphere MQ subsystem and the DB2 subsystem. To make these changes, instructions from the *IBM WebSphere MQ System Management Guide* and the *DB2 UDB for z/OS Installation Guide* were referenced.

Updating IMS PROCs and JCL

The steps needed to update PROCS and JCL are found in the *IBM WebSphere MQ System Management Guide* and the *DB2 UDB for z/OS Installation Guide*. In addition, you must add the Platform API library to the STEPLIB of the message processing region where you intend to use WebSphere Transformation Extender in your transactions. The Platform API should not be APF authorized. It is recommended that you have a DFSESL DD statement with authorized libraries and leave the Platform API in the STEPLIB, unauthorized. You must also add DD statements for the WebSphere Transformation Extender work files and trace files if your maps require them. WebSphere Transformation Extender uses the IBM Language Environment runtime library (**CEE.SCEERUN**). If this library is not on your linklist, you must add it to the STEPLIB in your MPR.

The following is a section of the DFSMPR proc.

Note: On line 6 in this example, the placeholder text `<YOUR_IMS/DC_LOADLIBRARY>` is listed. Replace this text with the location of your load library as defined during installation.

```
//STEPLIB  DD DSN=IMS.&SYS2.PGMLIB,DISP=SHR
//         DD DSN=IMS.&SYS2.RESLIB,DISP=SHR
//         DD DSN=MQM.CSQQDEFV,DISP=SHR
//         DD DSN=MQM.SCSQAUTH,DISP=SHR
//         DD DSN=MQM.SCSQANLE,DISP=SHR
//         DD DSN=<YOUR_IMS/DC_LOADLIBRARY>,DISP=SHR
//*
//* ALL OF THE LIBRARIES ON DFSESL ARE APF AUTHORIZED
//*
//DFSESL   DD DSN=IMS.&SYS2.RESLIB,DISP=SHR
//         DD DSN=MQM.CSQQDEFV,DISP=SHR
//         DD DSN=MQM.SCSQAUTH,DISP=SHR
//         DD DSN=MQM.SCSQANLE,DISP=SHR
//         DD DSN=DSN610.SDSNLOAD,DISP=SHR
```

The following excerpt shows the other required DD statements in the MPR job.

On lines 1-3 in this example, the placeholder text `<YOUR_PRODUCT_MAPLIBRARY>` is listed. Replace this text with the location of your map library.

```
//MQSAPIRD DD  DISP=SHR,DSN=<YOUR_PRODUCT_MAPLIBRARY>(MQSAPIRD)
//MQSAPIWR DD  DISP=SHR,DSN=<YOUR_PRODUCT_MAPLIBRARY>(MQSAPIWR)
//DTXMAPS  DD   DSN=<YOUR_PRODUCT_MAPLIBRARY>,DISP=SHR
//DTXLOG   DD  SYSOUT=*
//DTXDEBG  DD  SYSOUT=*
//DTXTRCE  DD  SYSOUT=*
//DTXAUD   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//RUNLOG   DD  SYSOUT=*
//RUNDBG   DD   SYSOUT=*
//*
```

Making additional IMS modifications

Using the instructions in the *IBM WebSphere MQ System Management Guide* as a guide, CSQQDEFV was linked and the CSQQTRMN batch messaging program (BMP) JCL was created. The program specification block (PSB) and application control block (ACB) for CSQQTRMN were generated. Also, modifications were made to the IMSGEN to include the CSQQTRMN transaction. A few transactions for the program were also added to call the Platform API. This information is reflected in the code that follows.

```
*---------------------------------------------------------------------
* IMS Trigger monitor for MQSeries
*---------------------------------------------------------------------
          SPACE 2
          APPLCTN PSB=CSQQTRMN,PGMTYPE=BATCH,SCHDTYP=PARALLEL
          SPACE 2
*--------------------------------------------------------------------
* Sample Applications for WebSphere Transformation Extender and MQSeries
*--------------------------------------------------------------------
          SPACE 2
          APPLCTN  DOPT,FPATH=NO,PGMTYPE=TP,PSB=DTXPSB1
          TRANSACT CODE=DTXMQS1,MODE=SNGL
          SPACE 2
          APPLCTN  DOPT,FPATH=NO,PGMTYPE=TP,PSB=DTXPSB2
          TRANSACT CODE=DTXMQS2,MODE=SNGL
          SPACE 2
          APPLCTN  DOPT,FPATH=NO,PGMTYPE=TP,PSB=DTXTEST
          TRANSACT CODE=DTXTEST,MODE=SNGL,INQUIRY=YES
```

Configuring IMS transaction names and DB2 plans

WebSphere Transformation Extender uses a single DB2 plan for all access to DB2. The DB2 IMS Attachment Facility assumes a one-to-one correspondence between IMS applications and DB2 plans. The instructions in ″Defining DB2 Plans for IMS Applications″ in the *DB2 UDB for z/OS Installation Guide* were used to create a resource translation table. The name of the runtime translation table is **DTXMNASM** and its source follows.

The name also appears on the ″RTT=″ parameter in the IMS External Subsystem Attach Facility Entry example in Updating the IMS External Subsystem Attach Facility Table.

```
TITLE 'WebSphere Transformation Extender IMS attach facility table'
         ACONTROL COMPAT(MACROCASE)
DTXMNASM AMODE    ANY
DTXMNASM RMODE    24
DTXMNASM DSNMAPN  APN=DSTXPSB1,PLAN=DBUTILE
         DSNMAPN  APN=DSTXPSB2,PLAN=DBUTILE
         DSNMAPN  APN=DSTXPSB3,PLAN=DBUTILE,END=YES
         END    DTXMNASM
```

Granting appropriate DB2 access privileges to CSQQTRMN

When CSQQTRMN inserts a message into the IMS message queue, the USERID associated with the message is **CSQQTRMN**. This USERID is used as the key for DB2 access. To retrieve information from the DB2 databases described in the map, you must grant the appropriate access to user CSQQTRMN. For information about the GRANT SQL statement, see the *DB2 UDB for z/OS v8 Application Programming and SQL Guide*.

Performing a PSBGEN

Before you execute an application program, a program specification block generation (PSBGEN) must be performed to create the PSB for the program. The PSB contains one PCB for each DL/I database (logical or physical) the application program will access. The PCBs specify those segments the program uses and the kind of access (retrieve, update, insert, delete) for which the program is authorized. The PSBs are maintained in one or more IMS system libraries called a PSBLIB library. The PSBs for these examples were very simple. They refer to one of the IMS IVP databases.

The following example is for PSB 1.

```
          TITLE 'WebSphere Transformation Extender Sample PSB 1'
         ACONTROL COMPAT(MACROCASE)
* Procopt on all PCBs referring to sample databases is set
* to "G" to prevent an accidental overwrite of the sample DB.
          PCB    TYPE=DB,DBDNAME=IVPDB1,PROCOPT=G,PCBNAME=DTXPCB1,KEYL1
               EN=10
          SENSEG NAME=A1111111,PARENT=0
          PSBGEN PSBNAME=DTXPSB1,CMPAT=YES,LANG=ASSEM
          END
```

The following example is for PSB 2.

```
          TITLE 'WebSphere Transformation Extender Sample PSB 2'
          ACONTROL COMPAT(MACROCASE)
* Procopt on all PCBs referring to sample databases is set
* to "G" to prevent an accidental overwrite of the sample DB.
          PCB    TYPE=DB,DBDNAME=IVPDB2,PROCOPT=G,PCBNAME=DTXPCB1,KEYL1
               EN=10
          SENSEG NAME=A1111111,PARENT=0
          PSBGEN PSBNAME=DTXPSB2,cmpat=yes,lang=assem
          END
```

The following example is for DTXTEST.

```
          TITLE 'WebSphere Transformation Extender Sample DTXTEST'
          ACONTROL COMPAT(MACROCASE)
* Procopt on all PCBs referring to sample databases is set
* to "G" to prevent an accidental overwrite of the sample DB.
          PCB    TYPE=DB,DBDNAME=IVPDB2,PROCOPT=G,PCBNAME=DTXTEST,KEYL1
               EN=10
          SENSEG NAME=A1111111,PARENT=0
          PSBGEN PSBNAME=DTXTEST,CMPAT=YES,LANG=ASSEM
          END
```

Defining IBM WebSphere MQ queues

A process was defined for each IBM WebSphere MQ trigger queue. The process definition contains the name of the IMS transaction that will be queued by CSQQTRMN when it reads the initiation queue. You must have an IBM WebSphere MQ process definition for each IMS transaction to be initiated, but not for each trigger queue. An example of a process definition follows.

```
Display a Process
Press Enter to refresh details.
Process name  . . . . . . . . DTX.DB2MAP.PROCESS
Description . . . . . . . . : process for running DB2 maps
                               using dtxpsb1
Application type  . . . . . : IMS
Application ID  . . . . . . : DTXMQS1
User data . . . . . . . . . :
```

```
Environment data  . . . . . :

Last alteration time  . . . :,2001-09-28 11.03.42,
```

Next, an IBM WebSphere MQ trigger queue must be defined. If you use the ISPF window, there are several panels that you must complete before pressing **Enter**. The first panel follows.

```
Display a Local Queue
Press F8 to see further fields  or Enter to refresh details.

More:    +

Queue name  . . . . . . . . . DTX.TRIGGER.DB2MAP
Description . . . . . . . . : Trigger queue for the DTXPSB1
                              IMS transaction
Put enabled . . . . . . . . : Y  Y=Yes N=No
Get enabled . . . . . . . . : Y  Y=Yes N=No
Usage . . . . . . . . . . . : N  N=Normal X=XmitQ
Storage class . . . . . . . : DEFAULT
Creation method . . . . . . : PREDEFINED
Output use count  . . . . . : 0
Input use count . . . . . . : 0
Current queue depth . . . . : 6
Command ===> ___________________________
```

This is an example of Panel 3.

```
Display a Local Queue
Press F7 or F8 to see other fields  or Enter to refresh details.

More:  - +

Trigger Definition

Trigger type  . . . . . . . : E  F=First E=Every D=Depth N=None

   Trigger set  . . . . . . : Y  Y=Yes N=No
   Trigger message priority : 0  0 - 9
   Trigger depth  . . . . . : 1          1 - 999999999
   Trigger data . . . . . . : SQLEMP2F

   Process name . . . . . . : DTX.DB2MAP.PROCESS
   Initiation queue . . . . : DTX.DB2MAP.INITQUEUE
```

Using the general COBOL examples

The COBOL example programs are supplied as separate source files. Rather than reproduce the code in this document, refer to each file to view its contents.

COBOL for z/OS and VM Version 2 Release 1 was used. This version supports direct calls to a DLL. If you have never called a DLL from COBOL before, read the creating and called DLLs sections of the *COBOL for z/OS and OS/390 Programming Guide* for background information about these examples.

Earlier versions of COBOL could not call DLLs. Examples of a C language interface routine are supplied as part of WebSphere Transformation Extender for z/OS. Both example programs call a subroutine named initep, included as member DTXIECOB in the DTX.SDTXSAMP PDS. The initep subroutine is present to circumvent the restriction on the SET verb that requires the target of the ″address of″ phrase to be specified in the Linkage Section. This restriction is removed in COBOL for z/OS and VM Version 2 Release 2.

Additional environmental configuration issues

To use the examples included with the IMS/DC Execution Option included in the WebSphere Transformation Extender feature, certain additional steps need to be taken.

Your systems might already be properly configured. This procedure is provided as a reference to make sure that these configurations have been completed.

To configure the environment for using the examples:

1. On the IMS side, in **IMS.JOBS**, create the MQSQMON member to run the IBM WebSphere MQ monitor BMP, CSQQTRMN. This job can be started from the IMS console or submitted as a batch job.
2. Also in **IMS.JOBS**, create DTXMPR2, the message processing program (MPP) job that actually runs the IMS transactions. This job defines the following locations:
   - the WebSphere Transformation Extender map library
   - the WebSphere Transformation Extender work files
   - other files required for logging, tracing, auditing, and so on

   For more detailed information, see the sample JCL in ″Updating IMS PROCs and JCL″ .
3. On the IBM WebSphere MQ side, define the IBM WebSphere MQ process and queues using the IBM WebSphere ISPF window.

   Note: Make sure you define the process *before* defining the trigger queue.
4. Returning to the IMS side, create and assemble a transaction name to plan name cross-reference. Name it DTXMNASM. For more detailed information, see the ″Configuring IMS Transaction Names and DB2 Plans″ .
5. Link DTXMNASM into an authorized library.

   Upon linking, all of the transactions defined in the IMSGEN are mapped to the DBUTILE plan. For more detailed information, see ″Configuring IMS Transaction Names and DB2 Plans″ .
6. On the DB2 side, grant access to CSQQTRMN. CSQQTRMN is the name of the IBM WebSphere MQ-supplied monitor program.

   CSQQTRMN becomes the user ID for the transaction scheduled in the IMS MPP region that will run the map. This user ID also appears in the terminal PCB and is used for database access authorization. For more detailed information, see ″Configuring IMS Transaction Names and DB2 Plans″ .

Chapter 4. About the examples

There are three sample programs included with this installation that can assist you in using the IMS/DC Execution Option to run maps.

- dtxpsb1 (″Example 1 (dtxpsb1)″)
- dtxpsb2 (″Example 2 (dtxpsb2)″)
- dtxtest (″Example 3 (dtxtest)″)

Note: Information about how to use IMS is an assumed prerequisite for using these examples. The information provided deals only with the high-level functionality of each example.

Example 1 (dtxpsb1)

1. A simple map, ″PrimeIMS″ inserts an equally simple message on an IBM WebSphere MQ trigger queue.
2. A message is placed on an IBM WebSphere MQ initiation queue by IBM WebSphere MQ, using information provided when the trigger queue was created.
3. The IBM WebSphere MQ initiation queue is read by a waiting IMS Batch Message Processing program, CSQQTRMN, provided as a part of IBM WebSphere MQ. The BMP is started when IMS is started. A message is not delivered by IBM WebSphere MQ to the initiation queue until CSQQTRMN has tried to read the IBM WebSphere MQ Init queue.
4. Using information read from the IBM WebSphere MQ init queue, the CSQQTRMN program creates and inserts a transaction on an IMS message queue.
5. IMS schedules the transaction.
6. The transaction, written in COBOL, reads the IMS message queue and obtains information about the transaction. For Example 1, this is the name of the IBM WebSphere MQ queue that is to receive the transaction output.
7. The transaction calls the platform API and runs a map that reads the IBM WebSphere MQ trigger queue and returns information back to the COBOL program.
8. The COBOL program then reads information from an IMS database using the key retrieved from the transaction by the map.
9. The COBOL program then calls the platform API, providing the name of the output IBM WebSphere MQ queue, and the result of the database read.
10. The COBOL transaction terminates, causing the IMS monitor to ″commit″ all of the parts of this transaction.
11. To verify the output (data retrieved from the IMS database), use the MQSAPIRD map to read the IBM WebSphere MQ queue.

Example 2 (dtxpsb2)

The second example, dtxpsb2, builds on the first example. More of the work done by the COBOL program is moved to the map. The data flow is essentially the same as Example 1, except for what happens in the COBOL program. The COBOL program only makes one call to the platform API using the parameters read from the terminal PCB. If the map succeeds and the platform API returns **0**, the COBOL program completes, implicitly committing all actions. If the map fails, the COBOL program writes error messages and calls a routine to cause an ABEND to ensure rollback of all IBM WebSphere MQ and database actions.

Example 3 (dtxtest)

The third example, dtxtest, provides a transaction that can test a map. This program reads a command line from the terminal PCB and directly calls the platform API. It then inserts a message with the return code using the terminal PCB.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other 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 NON-INFRINGEMENT, 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.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

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

AIX
AIX 5L
AS/400
Ascential
Ascential DataStage
Ascential Enterprise Integration Suite
Ascential QualityStage
Ascential RTI
Ascential Software
Ascential
CICS
DataStage
DB2
DB2 Universal Database
developerWorks
Footprint
Hiperspace
IBM
the IBM logo
ibm.com
IMS
Informix
Lotus
Lotus Notes
MQSeries
MVS
OS/390
OS/400
Passport Advantage
Redbooks
RISC System/6000
Roma
S/390
System z
Trading Partner
Tivoli

WebSphere
z/Architecture
z/OS
zSeries

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

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

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

IBM WebSphere Transformation Extender, Version 8.1

Index

Printed in USA