IBM WebSphere Transformation Extender

Command Server

Version 8.1

Note

Before using this information, be sure to read the general information in “Notices” on page 49.

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. Using a Command Server

To use the Command Server to execute a map, you will prepare the maps for execution.

For details about operating system requirements and installation information, see the Release Notes on the product Web site (www.ibm.com/software/integration/wtx). Before using the Command Server after you have installed it, test your installation by running an example map. To see how to test your installation on a specific platform, see the platform-specific topic.

Command Server Overview

The Command Server is used to develop, test, and execute maps in development environments. It can also be used to execute commands in production environments.

Maps are developed and tested using the Design Studio software on a Windows system-based development computer. When a map's development is complete, the compiled map file (**.mmc**) is, by default, ready to be executed by a Windows system-based Command Server. For example, you can run a map from the Map Designer interface or a Windows system-based Command Server, a Launcher, or the Platform API that has the same byte-order and character set.

The following list summarizes the steps you will do to run maps with the Command Server.

1. Create your map
2. Build or build for specific platform
3. Transfer if necessary
4. Execute with the Command Server

To account for environmental differences, some platforms require the map to be compiled or built for that specific platform. The Build for Specific Platform command is available from the **Map** menu in the Map Designer. After a map is built, it can be transferred to any destination computer hosting the Command Server. Using your preferred file transfer method, the file must be transferred in BINARY mode to ensure that the content of the compiled map arrives in its proper format on the destination platform.

Maps already transferred to their destination platform can be executed:

- from the command line
- as part of a series of maps in a command file
- from a program that runs an external program
- using a batch file or shell script
- as a task within a scheduler

Building a platform-specific map

After creating a map in Map Designer, you can build a map for a specific execution platform by selecting **Build for Specific Platform** from Map Designer's **Map** menu.

The Select Platform window opens, in which you to choose the destination execution platform for the map.

When a map is built for a specific platform, the name of the compiled map file has a platform-specific file name extension that indicates the execution platform for the map. The following table lists each platform and the corresponding platform-specific filename extension.

Platform	**Map File Name Extension**
HP-UX(TM) (Itanium processor)	.hpi
HP-UX(TM) (PA-RISC processor)	.hp
IBM AIX(TM) (RS/6000 processor)	.aix
IBM(TM) z/OS	.mvs
Linux (INTEL processor)	.lnx
Microsoft Windows	.mmc
Sun Solaris(TM) (SPARC processor)	.sun

After transferring a platform-specific built map, it is strongly recommended that you rename the compiled map by using the .mmc file extension. Renaming it is not necessary; however, it is helpful because an .mmc extension identifies the file name in mapping functions and the renamed map can be executed without specifying the extension.

Executing a map

After a map has been built for a specific execution platform and transferred, the map can be executed using a Command Server.

Using execution commands

Using the execution commands and options, you have precise control over how a map executes. For example, you can override sources and targets specified in the map, create trace and audit log files, and control a variety of characteristics of the run-time environment.

Some execution commands are not available on all platforms. For information about execution commands and their syntax, see the *Execution Commands* documentation.

Input data validation

When a map begins execution, the input data must first be validated to ensure that the data for each input conforms to the definition of the corresponding input card. If it does conform, the data is valid. If the data does not conform, it is invalid. If data is determined to be invalid, a run-time return code and corresponding message is issued.

If invalid data is encountered, you may still want the map to continue processing or you may even want to map the invalid data. This can be accomplished during map development by using the **Restart** attribute when defining the input data and using the `REJECT` function when mapping the output data. In addition, functions are available to test for the presence of errors. For information about using the **Restart** attribute, see the *Type Designer* documentation. For information about using the error-related functions, see the *Functions and Expressions* documentation.

The validation process assumes that source data is a data stream. To find data objects in the input stream, type tree definitions are used including information about syntax items, data patterns, restrictions, component rules, and size. To process your input, the data must be able to be accessed at a designated position, for a specified length.

As data is validated, information about the data is recorded into workspace in memory or work files that guide the output process. The workspace information controls the data objects appearing in the data stream, whether they are related to other data objects, and what (if any) the relationship is. This enables you to specify any input within a map rule regardless of where the data appears in the input stream. It also enables you to use rules about the presence of data so that, for example, you can count objects or test their presence.

Output objects built

After all the input data has been validated and the input data is valid, output objects are generated based on each output card specification, including the map rules. If the output has an initiator defined, it is built first. Then, each output is built, one rule at a time, beginning with the first rule of the first output card. The entire first rule is evaluated, even if it refers to other maps and functions. When the first output is built, it is a complete data object, identified and treated as any other input data object is identified and treated. This enables you to refer to earlier outputs in a map rule.

As outputs are built, syntax objects are inserted-such as a delimiter, initiator, terminator, or release character-according to the type definition, and whatever remaining bytes of a fixed-length item are filled with pad characters. The execution process continues as the rules are evaluated and output data is built through the completion of the last rule of the last card.

Troubleshooting tools

When a map begins to run, the first thing that occurs is the validation of the input data-performed by comparing the data to the type tree definition. After data validation completes, the map rules are evaluated for the output cards and output objects are built.

When a map executes, invalid or unknown input data may be encountered and you may want to analyze what happened during the validation process. You may also want to analyze the output data if the map did not produce the expected results.

Many debugging aids are available to assist you through the troubleshooting process including:

- A Command Server display of ongoing execution information
- Command Server return codes and messages
- Map audit log files, including data audit, execution audit, map settings, and card settings information
- Map trace files
- If an adapter is used, adapter-specific return codes and messages, audit log files, and trace files

Command Server display

Unless suppressed using execution commands, the Command Server displays several pieces of information when you run a map including the Command Server name, number of input objects found, number of output objects built, map execution time and a message indicating the map status. This information displays using a Command Server window on all Windows operating systems or using a console display on UNIX operating systems.

Ongoing feedback is shown during map execution and upon map completion. The messages that result (each message has a corresponding return code) indicate whether the map has completed and display the mapping results.

Command Server return codes and messages

After a map completes, a message indicates the execution results. These messages are also recorded in the appropriate audit logs.

Some return codes are only warnings indicating that the map did complete, but some problems occurred. You can change the default behavior for the warning codes by specifying **Fail on Warnings** or **Ignore Warnings** through the map settings or by using the `-F` or `-Z` execution command.

Map audit logs and trace files

Audit logs and trace files provide information that can be used effectively for debugging and troubleshooting purposes. There are many situations that occur during map execution in which you would want the activities associated with the processing of your data to be recorded in an audit log file or trace file. The auditing and tracing of the map execution process can be accomplished at various levels by specifying that the appropriate audit logs and trace files be enabled for maps, input cards, output cards, or particular adapters. These audit log and trace files may also be defined further to include summary or detail information about execution statistics, data information, or adapter activity.

For information about map audit logs and trace files, see the *Map Designer* documentation.

For information about audit log and trace files for an adapter, see the adapter-specific documentation about enabling and viewing these adapter-specific log files.

Map audit log

When the **MapAudit** Log **Switch** setting is set to **ON**, you can select to have either execution information or data information recorded. You can also specify to have both execution and data information recorded for a complete picture of what transpired during the execution of a map. The map audit logs can be enabled from the Map Settings window in the Map Designer, Integration Flow Designer, Command Server for Windows, or on any platform using execution commands on the command line.

The default name for the audit log is the full name of the map with a **.log** extension. By default, it is located in the same directory as the compiled map file.

The **MapAudit** log can contain three different sections: **BurstAudit**, **SummaryAudit**, and **SettingsAudit**. The sections produced depend on the settings for **MapAudit**.

BurstAudit

The information in the **BurstAudit** section of the **MapAudit** setting is configurable using the **Data Audit Settings** tab in the Organizer. See the *Map Designer* documentation for information about configuring these options and interpreting the information in the **BurstAudit** section.

SummaryAudit

When the **SummaryAudit's Execution** setting is **Always**, **OnError**, or **OnWarningsOrError**, an **ExecutionSummary** section is included in the log file. The **ExecutionSummary** section of the audit log file provides a summary of the return codes, sources, targets, and work areas for the map. This execution information also includes:
- The command string to use if you were running the map from the command prompt.
- The number of input and output objects.
- Information about data and work files.
- Execution time

See the *Map Designer* documentation for more information.

An ExecutionLog per burst

When the **BurstAudit Execution** setting is **Always**, **OnBurstError**, or **OnBurstWarningOrError**, the log file will contain a section for *each* burst within the map. If all inputs have a **CardMode** of **Integral**, there will be a single **Burst** section. The following excerpt is from the **tryaudit.log** file, created from running the **tryaudit.mmc** compiled map, which is included in the *install_dir***examples\general\audit** folder and presents the **Burst** section:

```
<Burst count="1">

<DataLog>

<input card="1">
```

```
   <object index="1" level="1"   size="10113" status="E00">
            DTX_MapAudit General Element
   </object>
   <object index="1" level="0" size="10113" status="V01">
            Audit_Log
   </object>
</input>

</DataLog>

<ExecutionLog   burstreturn="28" ElapsedSec="3.9280">

<inputstatus card="1" bytes="10113" adapterreturn="0" contentreturn="28"/>
<inputstatus card="2" bytes="1067"  adapterreturn="0" contentreturn="0"/>
<outputstatus card="1" bytes="610"  adapterreturn="0" contentreturn="0"/>
<outputstatus card="2" bytes="1055" adapterreturn="0" contentreturn="0"/>
<outputstatus card="3" bytes="10938" adapterreturn="0" contentreturn="0"/>
<outputstatus card="4" bytes="0" adapterreturn="0"  contentreturn="0"/>

</ExecutionLog>

</Burst>
```

The **ExecutionLog** section identifies the return code and elapsed time for the burst, as well as the status of each input or output, including both an adapter return code, if applicable, and a content return code.

ExecutionSummary per map

The **ExecutionSummary** section provides information at the map level. The following excerpt presents the **ExecutionSummary** section:

```
<ExecutionSummary MapStatus="Valid" mapreturn="0" ElapsedSec="3.9610"
        BurstRestartCount="0">
   <Message>Input type contains errors</Message>

<CommandLine>'C:\install_dir\examples\general\audit\tryaudit.mmc'</CommandLine>
   <ObjectsFound>111</ObjectsFound>
   <ObjectsBuilt>10</ObjectsBuilt>

<SourceReport card="1" adapter="File" bytes="10113" adapterreturn="0">
   <Message>Data read successfully</Message>
   <Settings>C:\install_dir\examples\general\audit\goodaud.log</Settings>
   <TimeStamp>07:03:10 December 11, 2003</TimeStamp>
</SourceReport>

<SourceReport card="2" adapter="File" bytes="1067" adapterreturn="0">
   <Message>Data read successfully</Message>
   <Settings>C:\install_dir\examples\general\audit\maperrcd.dat</Settings>
   <TimeStamp>06:11:46 May 29, 2003</TimeStamp>
</SourceReport>

<TargetReport card="1" adapter="File" bytes="610" adapterreturn="0">
   <Message>Data written successfully</Message>
   <Settings>C:\install_dir\examples\general\audit\result.txt</Settings>
   <TimeStamp>09:27:11 January 23, 2004</TimeStamp>
</TargetReport>
.
.
.
<WorkArea type="File">
<inputarea  card="1"   Path="C:\install_dir\examples\general\audit\tryaudit.I01"
        TimeStamp="09:27:11 January 23, 2004" bytes="67522"/>
<inputarea  card="2"   Path="C:\install_dir\examples\general\audit\tryaudit.I02"
        TimeStamp="09:27:11 January 23, 2004" bytes="67522"/>
<outputarea card="1"   Path="C:\install_dir\examples\general\audit\tryaudit.O01"
```

```
        TimeStamp="09:27:11 January 23, 2004" bytes="65536"/>
<outputarea card="2"   Path="C:\install_dir\examples\general\audit\tryaudit.002"
        TimeStamp="09:27:11 January 23, 2004" bytes="65536"/>
<outputarea card="3"   Path="C:\install_dir\examples\general\audit\tryaudit.003"
        TimeStamp="09:27:11 January 23, 2004" bytes="65536"/>
<outputarea card="4"   Path="C:\install_dir\examples\general\audit\tryaudit.004"
        TimeStamp="09:27:11 January 23, 2004" bytes="65536"/>
</WorkArea>

</ExecutionSummary>
```

As shown above, the execution log can provide some high-level debugging information, such as:

- **Map return code and message.** The map return code and message indicate how the mapping operation completed and whether there were any problems. For example, a map return code of `0` and message of `Map completed successfully` indicate that no execution errors were encountered. This information helps in analyzing the source information in the log.
- **SourceReport and TargetReport.** For each source or target, the **ExecutionSummary** includes information indicating the adapter, the size of the data for the source or target, the adapter return code and message, and so on. For example, if using a database adapter, a return code of `-17` and a message of `Failed to parse SQL statement` for an input would indicate that some error occurred when the adapter attempted to execute the appropriate SQL statement to fetch the data from the database.
- **WorkArea.** For each input or output for which a WorkArea is created, the **ExecutionSummary** includes information, such as the location and size.

The **ExecutionSummary** is a good place to start when diagnosing map execution problems because you can quickly determine the sources or targets in error. Using the information in the **ExecutionSummary**, you can produce more detailed troubleshooting information for only those sources or targets that experienced problems.

MapSettings section

The **MapSettings** section contains a list of all map settings, including the settings for **MapAudit**, **MapTrace**, **WorkSpace**, **Century (SlidingCentury)**, **Validation (CustomValidation)**, **Retry** (**MapRetry)**, and **Warnings**. The following excerpt presents the **MapSettings** section:

```
<MapSettings>

<MapAudit switch="ON">
   <Log executionsummary="Always" databurst="Always"
           databurst_sizevalidation="WrongSize" executionburst="Always"
           mapsettings="Always" datasettings="OnError"/>
   <Location type="File" action="Create">
            <Directory type="Map">C:\install_dir\examples\general\audit</Directory>
            <FileName type="Default" prefix="MapName">tryaudit.log</FileName>
   </Location>
</MapAudit>

<MapTrace switch="OFF"></MapTrace>

<WorkSpace>
   <Location type="File" action="Delete">
            <Directory type="Map">C:\install_dir\examples\general\audit</Directory>
            <FileName type="Default" prefix="MapName"></FileName>
   </Location>
   <Paging>
```

```
            <PageSize>64</PageSize>
            <PageCount>8</PageCount>
   </Paging>
</WorkSpace>

<SlidingCentury switch="OFF"></SlidingCentury>

<CustomValidation switch="ON">
         <OnValidationError>Continue</OnValidationError>
         <RestrictionError>Validate</RestrictionError>
         <SizeError>Validate</SizeError>
         <PresentationError>Validate</PresentationError>
</CustomValidation>

<Retry switch="OFF"></Retry>

<BurstRestart switch="OFF"></BurstRestart>

<Warnings type="Every" action="Warn"></Warnings>

</MapSettings>
```

This information can be useful during debugging to determine why execution occurred in a certain way.

DataSettings section

The DataSettings section contains a list of all **InputData** and **OutputData** settings depending on the adapter used, including the **FetchAs** (**CardMode**), **WorkArea** , **Backup**, **Command** (**SourceAdapterCommand** or **TargetAdapterCommand**), **OnSuccess**, **OnFailure**, **Retry** (**AdapterRetry**), **Scope** (**AdapterScope**), and so on. The following excerpt presents the DataSettings section:

```
<DataSettings>

<InputData card="1"   CardMode="Integral"   WorkArea="!Reuse">
   <Backup switch="ON"   when="Always"   action="Create">
      <Path>C:\install_dir\examples\general\audit\
      Mer_tryaudit_10441074864820_5.I1.bak
</Path>
   </Backup>
   <Source adapter="File" OnSuccess="Keep" OnFailure="Rollback" Scope="Map"
            Warnings="Ignore">
      <Resource>C:\install_dir\examples\general\audit\goodaud.log</Resource>
      <Retry switch="OFF"></Retry>
   </Source>
</InputData>
.
.
.
<OutputData card="1">
   <Backup switch="OFF"></Backup>
   <Target adapter="File" OnSuccess="Create" OnFailure="Rollback" Scope="Map"
            Warnings="Ignore">
      <Resource>C:\install_dir\examples\general\audit\result.txt</Resource>
      <Retry switch="OFF"></Retry>
   </Target>
</OutputData>
.
.
.
</DataSettings>
```

This information can be useful during debugging to identify whether data should be copied to a backup file, whether the changes to a database target should be committed if the map fails, and so on.

Map trace files

Map trace files can be used for diagnosing invalid data or incorrect type definitions. When the **MapTrace Switch** setting is set to **ON**, you can specify the location and file name and select to have information captured for inputs and outputs including what level of detail you want recorded.

When you enable the **InputContentTrace** setting to trace your input data, the log file provides information about the data objects found, why data was found to be invalid, sizes and counts of data objects and their position in the data stream. When **OutputTrace** is enabled, the information recorded in the trace file reflects which output objects were built and which output objects evaluate to **NONE**.

You can choose to trace the input data, the output data or both. For additional information about map trace files, see the *Map Designer* documentation.

Note: You should enable the trace for debugging purposes only because enabling a map trace may affect execution performance.

External interfaces for applications, EXIT and RUN

Maps can interface with other maps, functions in libraries, or programs using the `EXIT` function, the `RUN` function or an adapter as a data source or target.

You can implement an adapter or `EXIT` function through a:

- library interface using a function in a library
- program passing a command line

Platform-specific limitations can determine the availability and usage of these interface methods on a particular platform.

The following table summarizes by platform the availability (Y) of the `RUN` function, the library interface, and program interface.

	Application and EXIT Interface		**RUN**
Platform	**Library**	**Program**	**Function**
HP-UX (PA-RISC)			Y
RS/6000 AIX	Y		Y
Windows			Y
z/OS Batch	Y	Y	Y
z/OS CICS		Y	Y

For information about the `RUN` and `EXIT` functions, see the *Functions and Expressions* documentation.

Chapter 2. Windows platforms

There are several activities you can do with the Command Server that you have installed on your Windows environment.

You can:

- test your Windows Command Server installation by running the sample map provided
- use the Command Server to run your maps

Running the map examples

After successfully installing the software, you can run the map examples included in the installation to test the installation to ensure the Command Server is functioning properly.

To run the map example

1. Run the setup script.
2. Execute the map:
   - Go to *install_dir***/examples/general/map/sinkmap**
   - From the command line, type:

     ```
     dtx sinkmap
     ```

     The Command Server updates the screen while the map executes. If the map ran successfully, the following message is displayed:

     ```
     Map completed successfully
     ```

     If the map does not complete successfully, see ″Using a Command Server″ for troubleshooting information.

Using the Command Server

The Command Server provides a window that enables you to control the way maps execute and show ongoing information during map execution.

To use the Command Server to run a map

When you open the Command Server application, a window opens indicating that the Command Server is ready to be run; however, no maps are currently loaded:

1. From the **Map** menu, select **Run**.
2. Choose a compiled map.

The Command Server window displays status information such as the number of objects found and built, as well as execution time.

Note: If the map does not complete successfully, see ″Using a Command Server″ for troubleshooting information.

To override map settings

1. From the **Map** menu, select **Setup** to browse your file structure.

2. Select the compiled map file for which you want to change one or more of the values of the map settings.

   The Map Settings window for the **tryaudit.mmc** file opens.
3. Change any value in the **Map Settings** window to override the settings compiled into the map for this execution only.
4. Click **OK**.

The map is ready to be run according to the newly entered override settings. The values entered in the Map Settings window do not update the compiled map file, but are only recognized for this one-time execution. To permanently change any values for map settings, use the Map Designer to modify the settings and recompile the map.

Help

Help is available for the Command Server to provide access to descriptions and procedures including information about execution commands, return codes, functions, and so on. You can access Help by selecting **Contents** from the **Help** menu.

When using the command line, Help for Command Server is available by typing:

```
DTX -?
```

Note: Ensure your current directory is where the Command Server is located or specify the relative path or full path as applicable when entering the Help command.

Chapter 3. z/OS Batch

There are several activities you can do with the Command Server that you have installed on your z/OS Batch environment.

You can:

- test your z/OS Batch Command Server installation by running the sample map provided
- use the Command Server to run your maps

Running the map examples

After successfully installing the software, you can run the map examples included in the installation to test the installation to ensure the Command Server is functioning properly.

For information about the examples, see the example readme files in the DTX.SDTXSAMP PDS, as well as the WebSphere Transformation Extender for z/OS Release Notes on the product Web site (www.ibm.com/software/integration/wtx).

Using the Command Server

The Command Server for z/OS Batch requires JCL coding to execute as a batch job.

Note: Maps must always be built for z/OS Batch and transferred as BINARY for execution on the z/OS platform. **The file transfer must not use the CRLF option because map corruption may result.**

Execution command line (JCL PARM statement)

The execution command line is coded in the PARM statement of the JCL. The command line comprises the DDNAME of at least one map followed by one or more commands that affect the execution of that same map. The map DDNAME is required; execution commands are optional.

The first argument in a command line string is always the DDNAME that identifies the map to be executed. The map DDNAME may indicate a sequential dataset that contains the z/OS Batch-specific map (Example 1), a PDS member that contains the z/OS Batch-specific map (Example 2), or a member name (Example 3).

Example 1

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MYMAP -TS'
//MYMAP DD DSN=USER.MAP1,DISP=SHR
```

Example 2

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MYMAPPDS(MAP1) -TS'
//MYMAPPDS DD DSN=USER.MAPPDS,DISP=SHR
```

Example 3

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MYMAP -TS'
//MYMAP DD DSN=USER.MAPPDS(MAP1),DISP=SHR
```

You can control the execution of a map by specifying one or more of the execution commands described below in ″z/OS Batch Execution Commands″ . These commands must follow the map whose execution they affect.

You can execute multiple maps in one job step. Example 4 shows the syntax.

Example 4

```
//STEP1 EXEC PGM=DTXCMDSV,
//  PARM='MAP1DD -<opt1> -<opt2> MAP2DD -<opt1> -<opt2>'
```

The PARM statement may be continued onto multiple lines; however JCL syntax limits it to a total of 100 characters. However, this JCL limitation can be overcome by using the Command Dataset execution command (`-@`).

During execution, the map is loaded once and is removed from storage only when the task ends or when the map being executed is different than the last executed map. This means that when maps are executed repeatedly, they are not reloaded each time.

Identifying data sources and targets

On z/OS, data sources and targets that exist as datasets are identified by DDNAME. Sources and targets must be defined by DD statements in the execution JCL before a map can be executed.

When you create a map to be run using the Command Server for z/OS Batch, define source and target names associated with input and output cards. If a map is tested on a Windows platform before being executed on a z/OS platform, data source and target names typically use the Windows file naming convention of drive:path\filename.extension. When a map executes on z/OS Batch, source and target names are converted to valid z/OS DDNAMES by removing *drive*, *path*, and *extension*, converting _ characters to # characters, and translating the remaining characters to uppercase.

Consider the following example. A map is to be executed on z/OS Batch. It has one input and one output-both of them sequential datasets. The map was developed on a Windows platform; for testing purposes the data source was initially defined as C:\My\Data\In_file.txt and the data target was defined as F:\Outfile.dat. When the map executes on z/OS Batch, source and target names are converted to DDNAMES. The data source name becomes IN#FILE and the data target name becomes OUTFILE. The execution JCL must contain the following DD statements:

```
//IN#FILE DD DSN=MY.INPUT,...
//OUTFILE DD DSN=MY.OUTFILE,...
```

The DDNAME associated with a data source or target may be changed at run time using a card override on the command line passed in the PARM statement. Input and output card override commands are discussed in ″z/OS Batch Execution Commands″ .

The use of instream datasets for data sources is supported. If an instream dataset is used, ensure that no sequence numbers exist in columns 73 through 80; sequence numbers will be interpreted as data by the map. The GETANDSET function may not be used with instream datasets.

Note: If you are testing a map on a computer that is targeted to z/OS Batch, the process can be simplified by specifying data source and target names that, when converted, will resolve to the expected z/OS DDNAMES.

Names to avoid

When defining data sources and targets in the Design Studio, use names that resolve to unique DDNAMES upon conversion. You should also ensure that a converted name is not reserved. The following DDNAMES are reserved for use on z/OS:

DDNAME	Description
DTXDEBG	Aid in problem resolution
DTXAUD	Audit Log
DTXLOG	Execution log
DTXTRCE	Trace output
DBTRACE	DB adapter trace output
MQTRACE	WebSphere MQ adapter trace output
DTXMRN	Fixed DDNAME for the **.mrn** file.
SYSTMP*nn*,	where *nn* is a number between 01 and 99. Reserved for statically allocated temporary datasets.
STEPLIB	Reserved by the operating system or Language Environment or both
SYSPRINT	Reserved by the operating system or Language Environment or both
SYSOUT	Reserved by the operating system or Language Environment or both
CEEDUMP	Reserved by the operating system or Language Environment or both

Temporary datasets

During mapping, temporary datasets are allocated as needed, depending on two factors:

- how the map is constructed
- the record format (RECFM) of the map's input and output files

One temporary dataset is allocated in all cases, which is a general use work file. One temporary dataset is also allocated for each input. Depending on how the map is constructed, it may create a temporary dataset for each output to serve as its work file.

In addition, one temporary dataset is always allocated for each input or output defined with an RECFM that is not F (fixed unblocked) or FBS (fixed blocked standard).

Because you can develop a map with no inputs, and output work files may not be used in some situations, the minimum number of possible temporary files for a map execution is 1. The maximum possible number is 1 + (2 * number of inputs) + (2 * number of outputs). Temporary files will be allocated using this formula where you have specified in the map to create output work files during its execution and all the inputs and outputs are formatted with a RECFM other than F or FBS.

By default, temporary datasets are dynamically allocated on disk. Dynamically allocated datasets require no DD statements in the JCL and can store up to approximately 3.1 MB of data. This is based on the settings specified in the **ALLOC*xx*** PARMLIB member. The default settings, as specified by IBM, will provide approximately 3.1 MB of space. These datasets are deleted at the end of mapping and have system-generated dataset names with low-level qualifiers of **CTMF*nnn***, where ***nnn*** is a number indicating the sequence in which the dataset was created. For example, the first temporary dataset created during a mapping has a qualifier of **CTMF000**; the second, **CTMF001**, and so on.

If the space allocation for a dynamic temporary dataset is insufficient, override it with a static allocation in the JCL. Use the following DD statement as a template:

```
//SYSTMP<nn> DD DSN=&&TEMP<nn>,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=<lrecl>),
//              UNIT=<unit>,
//              SPACE=(<space>)
```

The DDNAME for a statically allocated temporary dataset must use the form, **SYSTMP***nn*, where *nn* is a number between 01 and 99 corresponding to the sequence number of the temporary dataset. For example, if the third temporary dataset created by a map (the system-generated dataset name with a low-level qualifier of **CTMF002**) runs out of space (this will be indicated by an `IEC030I x37` message in JES log), override the dynamic allocation with a **SYSTMP03** DD statement that provides sufficient space.

The dataset name for a static allocation must begin with `&&` which indicates a temporary dataset. The disposition must be (NEW,DELETE,DELETE). The RECFM should be FBS (fixed blocked standard) although F (fixed unblocked) can be used. You can choose the logical record length, although larger record lengths may increase IO efficiency. Do not use the BUFNO parameter.

To statically allocate all the temporary datasets that can possibly be used by a map, code DD statements **SYSTMP01** through **SYSTMP***nn*, where *nn* is equal to 1 + (2 * number of inputs) + (2 * number of outputs). This will provide adequate space.

Note: Statically allocating temporary datasets may increase map performance.

Static temporary dataset allocations are reused when multiple maps are executed from the command line. Consider the following scenario: the command line executes MAPA followed by MAPB and MAPC. MAPA uses seven temporary files, MAPB nine, and MAPC four. In this example, you must statically allocate not 20 temporary datasets (7 + 9 + 4), but nine temporary datasets, the maximum number needed by any map in the series. Each successive map execution reuses the existing static temporary file allocations beginning with SYSTMP01.

The resulting execution log reports complete temporary dataset usage information including DDNAME, dataset name, DCB, and size.

JCL

The following sample JCL can be used as a guide to help develop JCL for your maps meeting your mapping requirements and the standards at your z/OS site.

The z/OS Batch Command Server is called DTXCMDSV.

```
//* The program to be executed is DTXCMDSV.  The command line is passed in the
//* PARM statement. The map to be executed is MYMAP, which has one input,
//* MYINPUT and one output, MYOUTPUT.  The map and all its datasets are
//* accessed through ddnames.
//*
//STEP1 EXEC PGM=DTXCMDSV,PARM='MAPPDS(MYMAP) /P256:12 /TS'
//*
//*  The ddname MAPPDS identifies the partitioned dataset
//*  where WebSphere Transformation Extender maps are stored.  MYMAP is the member.
//*
//MAPPDS DD DSN=MY.MAPPDS,DISP=SHR
//DTXLOG DD SYSOUT=*   Execution log
//DTXAUD DD SYSOUT=*   Audit
//DTXTRCE DD SYSOUT=*   Map Trace
//DTXDEBG DD SYSOUT=*   Map Debug listing
//DTXMRN DD DSN=USER.TEST(RES1MRN),DISP=SHR
//SYSPRINT DD SYSOUT=*   Used by LE
//SYSOUT DD SYSOUT=*   Used by LE
//CEEDUMP DD SYSOUT=*   Used by LE
//*
//* STEPLIB identifies the load library where WebSphere Transformation Extender was
//* installed.
//* It identifies the Language Environment library if it is not in the link list.
//*
//STEPLIB DD DSN=MY.LOADLIB,DISP=SHR
//        DD DSN=CEE.SCEERUN,DISP=SHR
//*
//* The map's output.
//*
//MYOUTPUT DD DSN=MY.OUTPUT,
//            DCB=(RECFM=VB,LRECL=256),
//            UNIT=SYSDA,SPACE=(TRK,(5,1)),
//            DISP=(NEW,CATLG,DELETE)
//*
//* The map's input.
//*
//MYINPUT DD DSN=MY.INPUT,DISP=SHR
//*
//* Static temporary dataset allocations.
//*
//SYSTMP01 DD DSN=&&TEMP01,DISP=(NEW,DELETE,DELETE),
//            DCB=(RECFM=FBS,LRECL=23476),
//            UNIT=VIO,SPACE=(TRK,(5,1))
//SYSTMP02 DD DSN=&&TEMP02,DISP=(NEW,DELETE,DELETE),
//            DCB=(RECFM=FBS,LRECL=23476),
//            UNIT=VIO,SPACE=(TRK,(5,1))
//SYSTMP03 DD DSN=&&TEMP03,DISP=(NEW,DELETE,DELETE),
//            DCB=(RECFM=FBS,LRECL=23476),
//            UNIT=VIO,SPACE=(TRK,(5,1))
//SYSTMP04 DD DSN=&&TEMP04,DISP=(NEW,DELETE,DELETE),
//            DCB=(RECFM=FBS,LRECL=23476),
//            UNIT=VIO,SPACE=(TRK,(5,1))
//SYSTMP05 DD DSN=&&TEMP05,DISP=(NEW,DELETE,DELETE),
//            DCB=(RECFM=FBS,LRECL=23476),
//            UNIT=VIO,SPACE=(TRK,(5,1))
```

z/OS Batch execution commands

To properly enter the z/OS Batch execution commands, you must observe the command syntax as well as the following general rules:

- Each command must begin with a hyphen (`-`) or a forward slash (`/`).
- At least one space between commands is required (for example: `-A -V`).
- Do not put spaces within commands, except for `-I`, `-O` and `-V`, which require a space before the source or target name.
- Execution commands are not case sensitive.
- When specifying commands, the entire command string (the map name and its associated commands and options) may be on one line or several lines if the `-@` command is used.

Only the execution commands listed in the following table are supported under z/OS Batch. The majority of these commands are described in the *Execution Commands* documentation. If there is a difference in how the command is used for z/OS Batch, the exception is noted in the Syntax column. Commands only used by z/OS Batch are marked ″Unique to z/OS Batch″ in the Syntax column. z/OS Batch-specific commands are not listed in the *Execution Commands* documentation, so are described here.

The Syntax displayed is for z/OS Batch usage.

Command
  Syntax

MapAudit
  `-A[D[R|W]][E[R|W]][B[R|W]][C[R|W]][P[R|W]][[M][S]]`

Sliding Century
  -D [*ccyy*|0]

  Exception: None

Fail on Warnings
  -F[!][warning_code[:warning_code...]]

  Exception: None

Ignore Validation Options
  `-G[R][S][P]`

  Exception: None

″Input Dataset Override (-I)″
  -I[F]*card_num*[W|!W][B] *ddname*

  Unique to z/OS Batch

Input Echo Override
  -IE*card_num*[S*size*]

  Exception: For z/OS Batch, this command cannot be used to designate the source (source) or change the specification for using workfiles (`W|!W`)

List `-L`

  Exception: None

No DTXLOG (-NL) or (/NL)
  -NL|/NL

Unique to z/OS Batch

"Output Dataset Override (-O)"
`-O[F]` *card_num* `[+|!+][B]` *ddname*

Unique to z/OS Batch

Output Echo Override
`-OE` *card_num* `[S]`

Exception: For z/OS Batch, the X option is not available to exclude this card from echo.

WorkSpace Paging
`-P`*size:count*

Exception: None

MapTrace
-T[I|IC*card_num*|I*from*[:*to*]][O|OC*card_num*][S]

Exception: For z/OS Batch, the `=dir` option is not available to change from the default the location in which the trace file is created.

Stop Validation
-V

Exception: None

"Workfiles in Memory (-WM)"
-WM

Exception: None

Ignore Warnings
-Z[!][*warning_code*[:*warning_code*...]]

Exception: None

"Print Uppercase (-+)"
-+

Unique to z/OS Batch

"Command Dataset (-@)"
-@*ddname*

Unique to z/OS Batch

"Problem Resolution (-$)"
-$

Unique to z/OS Batch

"Record Separators (/V)"
/V[*separator_list*...] *ddname*

Unique to z/OS Batch

The following commands are unique to z/OS Batch.

Input Dataset Override (-I)

Use the Input Dataset Override execution command (`-I`) to override the specified input card source with a dataset only for this execution.

`-I[F]`*`card_num`*`[W|!W][B]` *`ddname`*

Option	Description
`F`	Optional for compatibility with previous versions.
card_num	The number of the input card to override.
`W`	Reuse the existing work file for this data source. If the work file does not exist at execution time, create it.
`!W`	Do not use the existing work file, but create a new work file each time the map is run.
`B`	If map execution is **not** successful, roll back any changes made to the dataset and reinstate the dataset to the original state prior to map execution.
ddname	Specify the DDNAME of the dataset used as the input source.

No DTXLOG (-NL) or (/NL)

Use the No DTXLOG execution command (`-NL`) to suppress the creation of a **DTXLOG**. When you use it, no informational data will be produced from **DD DTXLOG**. If you pass the `-NL` command on the command line through the **RunMap()** function, **DD RUNLOG01** (**RUNLOG02** and so forth) information will be suppressed.

```
-NL|/NL
```

Output Dataset Override (-O)

Use the Output Dataset Override execution command (`-O`) to override the specified output card target with a dataset for this execution only.

`-O[F]`*`card_num`*`[+|!+][B]` *`ddname`*

Option	Description
`F`	Optional for compatibility with previous versions
card_num	The number of the output card to override
`+`	Append the data for the current map execution to an existing dataset.
`!+`	Do not append to an existing dataset, but create a dataset each time the map is run.
`B`	If map execution is **not** successful, rollback any changes made to the dataset and reinstate the dataset to the original state prior to map execution.
ddname	Specify the DDNAME of the dataset used as the output target.

Workfiles in Memory (-WM)

Use the Workfiles in Memory execution command (`-WM`) to override the default allocation of work files on disk and to allocate them as memory files.

```
-WM
```

Print Uppercase (-+)

Use the Print Uppercase execution command (`-+`) to cause all printed output, execution logs, audit logs, and trace logs to be written in uppercase.

```
-+
```

Command Dataset (-@)

Use the Command Dataset execution command (`-@`) to specify a DDNAME defining a dataset that contains map names and execution commands. This command is unique to z/OS Batch. If this command is used, it can only appear in the PARM statement and it must be the only command coded in the parm statement (for example: `EXEC PGM=DTXCMDSV,PARM='-@DTXPARM'`). Using the command dataset eliminates the 100-byte limitation of the JCL PARM statement. Within the command dataset, map names and execution commands may be coded free-form. The command dataset may reside on disk or it may be defined instream in the JCL. If defined instream, ensure that there are no sequence numbers in columns 73 through 80. A sequence number, because it does not begin with the switch indicator (`-` or `/`), is interpreted by the command parsing logic as a map name.

-@*ddname*

Option	Description
ddname	The DDNAME of the command file dataset.

Problem Resolution (-$)

Use the Problem Resolution execution command (`-$`) only under the direction of IBM Customer Support.

This command produces output that is unintelligible to the user but is required by Customer Support for debugging.

Record Separators (/V)

Use the Record Separators execution command (`/V`) to enable special handling of datasets with variable length records (datasets defined with `RECFM=V`).

`/V[`*separator_list*`...]` *ddname*

Option	Description
separator_list	Specify a comma-delimited string of separator bytes that can be expressed in hex format (for example: `/VX5B,X6C,X7C`) or character format (for example: `/V$,%,@`). The default record separator if none is indicated on the `/V` command as a single byte, hex15.
ddname	The DDNAME of the input or output dataset

This command functions differently for data sources and targets. If specified for an input dataset, the indicated (or default) record separator string is appended to each variable record after it has been read. If specified for an output dataset, the record separator string is interpreted as marking the end of a variable length record in the output.

For inputs, this command causes a string of one or more record separator bytes to be appended to each variable record to ensure that the data maps properly on z/OS Batch. For example, suppose you are mapping an input file on Windows that consists of lines of text delimited by carriage-return/line-feed (`0D 0A`) pairs. Even though you cannot see them in a text editor, these delimiters are embedded in the data and the map expects them to be there.

Now you need to run this same map on z/OS Batch. However, when you upload the input file to a z/OS dataset of variable length records using the `CRLF` option, the record separators embedded in the data are removed during the transfer. Consequently, when the file is read in for mapping under z/OS Batch, the record separators must be reinstated or the data will not map correctly. The solution is to use the command `/VX0D,X0A` to append a hex `0D 0A` record separator string to the end of each variable record.

If a dataset of variable length records already contains the separators defined to the map embedded in the data, this option should not be used.

For outputs, this command specifies that the indicated (or default) record separator byte or bytes are used to mark the ends of variable length records in the output. When this command is used, the map must embed record separator strings in the output data. These separator strings indicate the places where variable length record boundaries are to occur in the final output. The record separator strings are removed when the variable records are created. They do not appear in the output in its final form.

Output variable records wrap at the maximum record length defined for the dataset. If an output dataset is defined in the JCL with `DCB=(RECFM=VB,LRECL=80)` and either by accident or design a map produces an output record which is actually 120 bytes in length, two variable records are produced in the output dataset-one that is 80 bytes in length and one that is 40 bytes in length.

Troubleshooting

There are three files that are essential to troubleshooting efforts.

These files have DDNAMES as indicated in the following table:

DDNAME	File	Description
DTXTRCE	Trace File	Produced as a result of specifying the Trace execution command (`-T`)
DTXAUD	Map Audit Log	Produced as a result of enabling the Map Audit settings for a map and running the map with the Audit execution command (`-A`)
DTXDEBG	Problem Resolution	Produced as a result of specifying the Problem Resolution execution command (`-$`) as instructed by Customer Support.

VSAM considerations

The z/OS Batch Command Server supports the use of VSAM keyed-sequenced (KSDS), entry-sequenced (ESDS) and relative-record (RRDS) datasets for both inputs and outputs subject to the following conditions:

- VSAM datasets must be pre-defined and, if used for input, pre-loaded using IDCAMS or some other utility.
- The z/OS Batch Command Server opens input and output VSAM files for both reading and writing.
- When used as input, all VSAM datasets are processed sequentially; as output, KSDS files are updated and ESDS and RRDS files are appended.
- The use of SHAREOPTIONS 3 or 4 is not recommended. The library IO routines supplied by Language Environment do not check SHAREOPTIONS at dataset open time and do not provide support for read and write integrity for shared VSAM files.

Using the EXIT function

The `EXIT` function calls a user-written program to return data that is mapped to an item.

The `EXIT` function is documented in your *Functions and Expressions* documentation. However there are special considerations when using this function with the z/OS Batch Command Server.

A user exit is invoked by calling the `EXIT` function in a mapping rule. The `EXIT` function expects three arguments:

```
EXIT ("DLL_load_module_or_non_null",
"DLL_function_or_load_module", "parameters")
```

If the user exit program is a DLL, the first argument must contain the name of the DLL load module and the second argument must contain the name of the function within the DLL.

If the user exit program is not a DLL, the first argument must contain some value that is not a null string ("")-for example, one or more spaces-and the second argument must contain the name of the load module.

The third argument is a text item that contains parameter information passed to the exit program.

Your user exit programs must reside in one of the load libraries referenced by the STEPLIB DDNAME in the JCL used to run the z/OS Batch Command Server. Any static data areas declared within an exit program, such as COBOL working-storage or C static variables, persist across multiple calls to the exit.

For non-C language (for example, COBOL) exits, four arguments are passed that contain the address of:

- A full word that contains the length of the parameter text item
- A space-padded 64K buffer that contains the parameter text item
- A full word that will contain the length of the resulting text item
- A 64K buffer where the resulting text item will be placed

For C language exits, a single argument is passed, which is the address of the EXITPARAM structure as described in the *Functions and Expressions* documentation.

User exits must support the conventions of z/OS Batch standard linkage and be link-edited in 31-bit addressing mode. DLL exits must be compiled as DLL code.

For an example of a C language and COBOL user exit program, see DTXDLLSC and DTXTBPSA in the DTX.SDTXSAMP PDS included in the WebSphere Transformation Extender with Command Server installation.

Using the RUN function

The first argument to the RUN function is the name of the map to be executed.

On z/OS, this must be the DDNAME that identifies the deployed map. To ensure this, the z/OS Batch Command Server converts the map name to a legal z/OS DDNAME by removing drive, path, and file extension information if it exists, converting _ characters to # characters and translating the remaining characters to uppercase. For example, the map rule `RUN("` *install_dir* **\mymaps\map_10.mmc**`")` executes the map identified by the DDNAME **MAP#10**.

When using the `RUN` function, the z/OS Batch Command Server execution log, audit log, trace and problem resolution output must be written to individual datasets for each level of recursion; otherwise the output from different recursion levels are mixed together and will be unintelligible. By default, output from a map executed with the `RUN` function is directed to dynamically-allocated print datasets assigned to the default SYSOUT class for the job. This may be overridden by coding additional DD statements in the JCL. For each level of recursion, four DD statements should be added to your execution JCL:

- **RUNLOG** *nn* z/OS Batch execution log file
- **RUNAUD***nn* Data audit log file
- **RUNTRC***nn* Trace file
- **RUNDBG***nn* z/OS Batch problem resolution file (used with Customer Support)

where *nn* is the level of recursion (always two positions, zero-filled).

For example, if an executed map issues a `RUN` function and the map to be run does not execute any other maps, one level of recursion exists and the following statements should be added to the JCL step:

```
//RUNLOG01  DD  SYSOUT=B
//RUNAUD01  DD  SYSOUT=B
//RUNTRC01  DD  SYSOUT=B
//RUNDBG01  DD  SYSOUT=B
```

A map executed using the `RUN` function is loaded once and is removed from storage when the task ends or when the map executed is different from the last map executed. Maps repeatedly executed are not reloaded each time.

Statically allocated temporary datasets are reused when a map is executed repeatedly using the RUN function. For example, consider the following scenario: MAPA calls MAPB 100 times in succession using the RUN function. MAPA uses seven temporary files and MAPB uses three. In this example it is necessary to statically allocate, not 307 temporary datasets (7 + (100 * 3)), but 10 temporary datasets. MAPA will use SYSTMP01 through SYSTMP07 and MAPB will reuse SYSTMP08 through SYSTMP10 each time it is executed.

Reusing work files

To reuse a work file in the z/OS Batch Command Server, you must code a DD statement in the JCL, allocating the work file as a cataloged dataset. Only input work files can be reused.

The DDNAME for the work file to be reused must be a concatenation of up to five characters of the map name, followed by the `IW` string , followed by the number of the work file. The number of the work file corresponds to the number of the input card. For example, if you are executing a map called **MYMAP** and you want to reuse the work file for input card number three, code a DD statement with the following DDNAME:**MYMAPIW3**.

Work file DCB parameters should define a file with fixed-blocked standard (FBS) records. The disposition should be (MOD,CATLG,DELETE). Using MOD will ensure that if the file does not exist, it will be created. Using CATLG may cause the message `NOT RECATLGD 2` to appear in the output JES log, but this message can be safely ignored. The BUFNO parameter should not be used.

The following example DD statement demonstrates how you can reuse the work file for input card number 1 of a map named **MYMAP**:

```
//MYMAPIW1 DD  DSN=name,
//             DISP=(MOD,CATLG,DELETE),
//             DCB=(LRECL=23476,RECFM=FBS),
//             UNIT=unit,
//             SPACE=(space)
```

Output file same as input file

There are no special z/OS Batch considerations when the file named in the output card a map is the same as the file named in its input card. Code one DD statement for both files with a disposition of `OLD` or `SHR`. If you are mapping your output back to the input, be careful using card overrides.

The following example will not work:

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MAP -IF1 INFILE -OF1 OUTFILE'
//INFILE DD DSN=MY.IOFILE,DISP=SHR
//OUTFILE DD DSN=MY.IOFILE,DISP=SHR
```

Instead, the JCL should read:

```
//STEP1 EXEC PGM=DTXCMDSV,PARM='MAP -IF1 IOFILE -OF1 IOFILE'
//IOFILE DD DSN=MY.IOFILE,DISP=SHR
```

Appended output

When an output card for a map specifies that the output be appended to a file, the target file MUST previously exist. Use a disposition of `OLD` or `SHR`.

Resource names

For information about creating resource names for use on the Command Server for the z/OS Batch platform, see "Resource Names for z/OS Platform".

Chapter 4. z/OS CICS

There are several activities you can do with the CICS Execution Option that you have installed on your z/OS CICS environment.

You can:

- test your CICS Execution Option installation by running the sample maps provided in the installation
- use the CICS Execution Option to run your maps

Note: The WebSphere Transformation Extender includes the CICS Execution Option through which you can run maps in your CICS environments. Prior to WebSphere Transformation Extender version 8.1, this functionality was provided in the CICS Command Server product.

Running the map examples

After successfully installing the software, you can run the map examples included in the installation to test the installation to ensure the CICS Execution Option is functioning properly.

For information about the examples, see the example readme files in the DTX.SDTXSAMP PDS, as well as the WebSphere Transformation Extender for z/OS Release Notes on the product Web site (www.ibm.com/software/integration/wtx).

Using the CICS Execution Option

The CICS Execution Option can be invoked in several ways.

It can be invoked:

- From a clear terminal, enter the **DTXI** transaction and follow it with a command string containing the name of at least one map and any execution commands required for the execution of that map. Command arguments must be separated by spaces. For example, to run the map **MYMAP** from a terminal with commands specifying input and output card overrides, type:

  ```
  DTXI MYMAP -I1 MYINPUT -O1 MYOUTPUT
  ```

  Note: The **MYMAP** file must have been built for z/OS CICS, file transferred to z/OS, and loaded into the z/OS CICS map **KSDS**.
- Start the **DTXI** transaction from a user program by passing the transaction name **DTXI** followed by the command string containing the map name and any required execution commands in the **FROM** option of the EXEC CICS START command.
- Automatically initiate the **DTXI** transaction from a trigger transient data queue. The first record in the queue must contain the transaction name **DTXI** followed by the command string.
- Invoke the **DTXI** program using an **EXEC CICS LINK** and pass the link control blocks. (Use the **DTXSCTST** sample program as a guide.) These control blocks define areas for the command string, optional data buffers, and a return code and message.

Note: When **DTXI** is invoked using an **EXEC CICS LINK,** the command string supplied must be null terminated. If the calling program is COBOL, this means that the command string must be followed, in working storage, by at least one byte containing LOW-VALUES.

To run a map using the CICS Execution Option for z/OS CICS

1. Build the map for z/OS CICS. From the Map Designer select **Map** → **Build for Specific Platform** and select **IBM(TM) z/OS** from the drop-down list.
2. File transfer the platform-specific map to z/OS CICS as BINARY with no ASCII to EBCDIC translation and no elimination of carriage returns (CRs) and line feeds (LFs).

   Note: Transfer the file exactly as it is, without any changes.
3. Modify and execute the **DTXCMJCL** JCL to load the map into the z/OS CICS VSAM map dataset see ″Running the Map Examples″ .

   Note: It is possible to execute multiple maps from one command line. The syntax is:

map1 map1_execution_commands map2 map2_execution_commands

... Maps are loaded once and removed from storage when the task ends or when the map executed is different from the last map executed. This means that maps repeatedly executed are not reloaded each time.

z/OS CICS execution commands

To properly enter the z/OS CICS execution commands, you must observe the command syntax as well as the following general rules:

- Each command must begin with a hyphen (-) or a forward slash (/).
- At least one space between commands is required (for example: -**A** -**V**).
- Do not put spaces within commands, except for `-I`, `-O` and `-V`, which require a space before the source or target name.
- Execution commands are not case sensitive.

Only the execution commands listed in the following table are supported under z/OS CICS. The majority of these commands are described in the *Execution Commands* documentation. If there is a difference in how the command is used for z/OS CICS, the exception is noted in the Syntax column. Commands only used by z/OS CICS are marked ″Unique to z/OS CICS″ in the Syntax column. z/OS CICS-specific commands are not listed in the *Execution Commands* documentation, so are described in this section. The Syntax displayed is for z/OS CICS usage.

Command
  Syntax

″**Audit (-A)**″

  -A [D[R|W]][E[R|W]][B[R|W]][C[R|W]][P[R|W]][[M][S]] [@*storage_codeName*]

SlidingCentury
  -D [*ccyy*|0]

  Exception: None.

Fail on Warnings
-F[!][*warning_code*[:*warning_code*...]]

Exception: None.

Ignore Validation Options
-G[R][S][P]

Exception: None.

″**Input Dataset Override (-I)**″

-I[F]*card_num*[W|!W][B] [[@*storage_code*]
[:{W|P[*delimiter_list*]}]] [*name*]

Unique to z/OS CICS.

Input Echo Override
-IE*card_num*[S*size*]

Exception: For z/OS CICS, this command cannot be used to designate the source (source) or change the specification for using workfiles (W|!W).

″**Execution Log (-L)**″
-L[@*storage_codeName*]

Unique to z/OS CICS.

″**Output Dataset Override (-O)**″

-O[F]*card_num*[+|!+][B][[@*storage_code*]
[#*record_length*][:{W|P[*delimiter_list*]}]] [*name*]

Unique to z/OS CICS.

Output Echo Override
-OE*card_num*[S]

Exception: For z/OS CICS, the X option is not available to exclude this card from echo.

WorkSpace Paging
-P*size:count*

Exception: None.

″**Enqueuing on Data Files (-Q)**″
-Q

Unique to z/OS CICS.

-T[I|IC*card_num*|I*from*[:*to*]][O|OC*card_num*][S]
[@*storage_codeName*]

Unique to z/OS CICS.

Stop Validation
-V

Exception: None.

″**Workfiles in Memory (-WM)**″
-WM

Exception: None.

Ignore Warnings
-Z[!][*warning_code*[:*warning_code*...]]

Exception: None.

″**Problem Resolution (-$)**″
-$

Unique to z/OS CICS.

″**Print Uppercase (-+)**″
-+

Unique to z/OS CICS.

″**Eliminate AICA (ICVR Timeout) Abends (-!n)**″
-!n

Unique to z/OS CICS.

The following commands are unique to z/OS CICS.

Audit (-A)

Use the Audit execution command (`-A`) to control the creation of audit log information. When used *with* other specified options, the **Audit** settings specified with the Audit command (`-A`) override *all* **Audit** settings compiled into the map. When used *without* other specified options, no audit log is produced. For information about the audit log files, see the *Map Designer* documentation. The default is to write the audit information to a temporary storage queue named **DTXAUD.** You can override the default by specifying the storage medium to be used and the name. See ″Specifying Storage Medium″ for details.

```
-A [D[R|W]][E[R|W]][B[R|W]][C[R|W]][P[R|W]][[M][S]]
   [@storage_codeName]
```

Options
Description

@storage_code
Specify one of the following codes, which represents the storage medium to be used for the audit report

T = Specify a temporary storage queue
D = Specify a transient data queue
R = Specify VSAM relative-record dataset
K = Specify VSAM key-sequenced dataset
E = Specify VSAM entry-sequenced dataset

Name Specify the queue name or VSAM dataset name for the audit report

Name must not be preceded by any spaces.

Input Dataset Override (-I)

Use the Input Dataset Override execution command (`-I`) to override the specified input card source with a dataset for this execution only. Use this command to override data source specifications that are in the compiled map file for a single execution of a map.

Use this command to override the name for a specific input data source, work file behavior, or rollback.

You can also specify storage medium, record separator delimiters, behavior for data with variable length records, and so on. See ″Specifying Storage Medium″ and ″Variable Length Records″ for more information about specifying these options.

```
-I[F]card_num[W|!W][B][[@storage_code][:{W|P[delimiter_list]}]] [name]
```

Option	Description
F	Optional for compatibility with previous versions
card_num	The number of the input card to override
W	Reuse the existing work file for this data source. If the work file does not exist at execution time, a work file is created.
!W	Do not use the existing work file but create a new work file each time the map is run.
B	If map execution is **not** successful, roll back any changes made to the file and reinstate the file to the original state prior to map execution.
storage_code	Specify one of the following codes, which represent the type of storage medium to be used for input data. The default is a temporary storage queue with the name defined for the input card with the path and file extension removed. `T` = Specify a temporary storage queue. `D` = Specify a transient data queue. `R` = Specify VSAM relative-record dataset. `K` = Specify VSAM key-sequenced dataset. `E` = Specify VSAM entry-sequenced dataset. `B` = Specify storage buffer. **Note:** If the storage buffer option (`B`) is selected, then specifying an override name is not applicable.
:W	Specify for a file consisting of variable length records to copy data from the input file into fixed format workspace before mapping begins. The map will take its input not from the file but from workspace. For details see ″The :W Option″ .
:P	Specify that the data will be *profiled*, that is, a table will be built in memory which cross-references record identifiers to corresponding file offsets. For details, see ″The :P Option″ .
Delimiter_list	Specify a comma-delimited list of character or hex values. (X*nn* is the syntax for indicating a hexadecimal value.) The default delimiter is a single byte with the value hex 15, decimal 21.
name	Specify the name of the storage entity (source) containing the data for the specified input card. If `name` is omitted, the name used is the name defined for the input card with the PC-based path information and file extension removed. **Note:** `name` must be preceded by one or more spaces.

Execution Log (-L)

Use the Execution Log execution command (`-L`) to specify that an execution log is produced. The default is to write the execution information to a temporary storage queue name **DTXLOG**. You can override the default by specifying the storage medium to be used and the name of the log. See ″Specifying Storage Medium″ for details.

```
-L[@storage_codeName]
```

Options
Description

storage_code
One of the following codes, which represent the type of storage medium used for the execution log

`T` = Specify a temporary storage queue.
`D` = Specify a transient data queue.
`R` = Specify VSAM relative-record dataset.
`K` = Specify VSAM key-sequenced dataset.
`E` = Specify VSAM entry-sequenced dataset.

Name The name for the execution log

Note: *name* must not be preceded by any spaces.

Output Dataset Override (-O)

Use the Output Dataset Override (`-O`) execution command to override the specified output card target with a dataset for this execution of the map only. For example, use this command to override the name for a specific output data target or to specify the appending of data files, file deletion, or rollback.

You can also specify the storage medium, record length, delimiters to use for record separators, and so on. See ″Specifying Storage Medium″ , ″Specifying Output Fixed Format Record Length″ and ″Variable Length Records″ for more information about specifying these options.

When using an input or output override execution command (-I) or (-O), the source or target name must be preceded by one or more spaces. If the override specifies a storage buffer, using an override name is not applicable. Names for the data and execution audit, trace, and problem resolution information are not separated from the command by spaces.

```
-O[F]card_num[+|!+][B][[@storage_code][#record_length]
[:{W|P[delimiter_list]}]] [name]
```

Option	Description
F	Optional for compatibility with previous versions.
card_num	The number of the output card to override.
+	Append the current map execution data to an existing data file.
!+	Do not append the data to an existing data file, but create a file each time the map is run.
B	If map execution is not successful, rollback any changes made to the file and reinstate the file to the original state prior to map execution.

@storage_code
: One of the following codes that represent the type of storage medium to be used for output data. The default is a temporary storage queue with the output card name defined without its path and file extension.

T = Specify a temporary storage queue.
D = Specify a transient data queue.
R = Specify VSAM relative-record dataset.
K = Specify VSAM key-sequenced dataset.
E = Specify VSAM entry-sequenced dataset.
B = Specify storage buffer.

If the storage buffer option (B) is specified, then specifying an override name is not applicable.

record_length
: The fixed output record length specified for writing the data that is less than or equal to 32768. The default is to write the data as 4096-byte fixed length records.

For VSAM, the record length is the record length defined for the dataset.

:W The data is mapped into workspace and then copied from workspace to its final target after mapping completes. Variable length records are separated out during the copy operation based on a defined record separator. See "The :W Option" .

:P The data *profiled*, that is, a table is built in memory that cross-references record identifiers to corresponding file offsets. For details, see "The :P Option" .

delimiter_list
: A comma-delimited list of character or hex values. (X*nn* is the syntax for indicating a hexadecimal value.) The default delimiter is a single byte with the value hex 15, decimal 21.

name The name for the temporary storage entity (target) where data for the specified output card will be written. If *name* is omitted, the name used will be the name defined for the output card without the PC-based path information and file extension.

name must be preceded by one or more spaces.

Enqueuing on Data Files (-Q)

Use the Enqueuing on Data Files execution command (-Q) to specify this execution command to enable enqueuing on data sources and targets for maps that run concurrently in a z/OS CICS region and share inputs or outputs or both inputs and outputs. By default, I/O calls to and from input and output files are not serialized.

```
-Q
```

Trace (-T)

Use the Trace execution command (-T) to turn off tracing or to specify a particular card or a range of input or output data objects or both input and output data objects to trace. When the -T command is used without other options specified, no trace log is produced. You can also produce a trace summary and specify the storage medium to use for the trace report. The default is to write the trace information to temporary storage queue DTXTRCE. You can override the default by specifying the storage medium to be used and the name. See "Specifying Storage Medium" for details.

```
-T[I|ICcard_num|Ifrom[:to]]
 [O|OCcard_num][S][@storage_codeName]
```

Option	Description
I	Produce detailed trace information for all output cards
IC*card_num*	Produce detailed trace information for a single input card. For example, the command `-TIC3` only produces detailed trace information for input card 3.
I*from:to*	Produce detailed trace information for a range of input objects. For example, to produce trace information for objects `1000` to `1320`, the option could be specified as `-TI1000:1320`. The upper limit of the object range is optional such that to trace from input object `1000` to the end of the file, the option could be specified as `-TI1000`.
O	Produces detailed trace information for all output cards
OC*card_num*	Produce detailed trace information for a single output card. For example, the command `-TOC2` produces detailed trace information only for output card `2`.
S	Produce summary trace information for input and output cards as specified, providing a message for each card. For example, the message for a particular card may be `Input 1 was valid, but 34 bytes of unknown data found` or `Output 2 built successfully`.

storage_code	One of the following codes that represent the type of storage medium used for the trace report. The default storage medium is transient data queue CSML.
	
	T = Specify a temporary storage queue.
	D = Specify a transient data queue.
	R = Specify VSAM relative-record dataset.
	K = Specify VSAM key-sequenced dataset.
	E = Specify VSAM entry-sequenced dataset.

name	The name for the trace report. *name* must not be preceded by any spaces.

Workfiles in Memory (-WM)

Use the Workfiles in Memory execution command (`-WM`) to override the default allocation of work files on disk and allocate them as memory files.

```
-WM
```

Problem Resolution (-$)

Use the Problem Resolution execution command (`-$`) only under the direction of Customer Support.

This command produces output that is unintelligible to the user but is required by Customer Support for debugging.

```
-$
```

Print Uppercase (-+)

Use the Print Output to Uppercase execution command (`-+`) to cause all printed output, execution logs, audit logs, and trace logs to be written in uppercase.

```
-+
```

Eliminate AICA (ICVR Timeout) Abends (-!n)

Use the Eliminate AICA (ICVR Timeout) Abends execution command (`-!n`) to issue an EXEC CICS SUSPEND being issued after every *n* objects found or built by the map. The EXEC CICS SUSPEND resets the ICVR timer and eliminates AICA abends caused by long-running maps that do a large amount of processing in CPU. Be aware that using this command could adversely affect map performance if other tasks with higher priority are executing in the z/OS CICS region at the same time as the map.

```
-!n
```

Specifying storage medium

The z/OS CICS Execution Option can map from or to any of six different storage media.

You would specify the storage media by using the following codes:

- `T` = Specify a temporary storage queue.
- `D` = Specify a transient data queue.
- `R` = Specify VSAM pre-defined relative-record dataset.
- `K` = Specify VSAM pre-defined key-sequenced dataset.
- `E` = Specify VSAM pre-defined entry-sequenced dataset.
- `B` = Specify storage buffer.

Storage media can be specified for data sources and targets as well as execution log, trace, and audit output. For a data source or target, the default storage medium is a temporary storage queue with the input or output card name defined without the PC-based path and file extension.

A particular storage medium can also be specified for data audit, execution log, trace, and problem resolution information:

This temporary storage queue
  is the default medium for the...

DTXLOG
  execution log

DTXAUD
  audit log

DTXTRCE
  trace

DTXDEBG
  debug listing

The following examples demonstrate how to specify the storage medium option with the applicable execution commands. In these examples, it is assumed that the storage medium for input and output data files comprises one or more records of a consistent, fixed length.

Example
Description

-I1@D INPQ
Specifies data for input card 1 resides in a TDQ called **INPQ**

-O1 OUTPUTQ
Specifies data for output card 1 is written to a TSQ named **OUTPUTQ**

-I3@B Specifies data for input card 3 resides in a storage buffer

-I2@K MYKSDSIN
Specifies input 2 is a VSAM KSDS named **MYKSDSIN**

-O2@E MYESDSOU
Specifies output 2 is a VSAM ESDS named **MYESDSOU**

-O2@R
Specifies output 2 is written to a VSAM RRDS named without a PC path or file extension

-L@TDTXLOG
Specifies that an execution log is written to a TSQ named **DTXLOG**

-A@DDTXAUD
Specifies that an audit log file is written to a TDQ named **DTXAUD**

-TI@TDTXTRCE
Specifies that an input trace file is written to a TSQ named **DTXTRCE**

-$@TDTXDEBG
Specifies that a debug listing is written to a TSQ named **DTXDEBG**

Note: When using an input or output override execution command (`-I`) or (`-O`), the source or target name must be preceded by one or more spaces. If the override specifies a storage buffer, using an override name is not applicable. Names for the data and execution audit, trace, and problem resolution information are not separated from the command by spaces.

Specifying output fixed format record length

Output data to transient data queues and temporary storage queues is, by default, written as 80-byte fixed length records. Use the Output File Override execution command (`-O`) to override the default and specify a different fixed output record length that is less than or equal to 32768.

The following examples demonstrate how to specify the fixed output record length using the Output File Override (`-O`) execution command:

Example
Description

-O2@D#72 OUTQ
Data is written to a transient data queue named **OUTQ** in 72-byte fixed-length records.

-O1#256 MYTSQ
Data is written to a temporary storage queue named **MYTSQ** in 256-byte fixed-length records.

Note: Fixed format record length cannot be specified for input cards and storage buffers. For VSAM, the record length is the record length defined for the dataset.

Variable length records

The z/OS CICS Execution Option views inputs and outputs as byte stream files rather than as collections of records.

The CICS Execution Option requires support for random positioning to any byte offset within an input or output file. Random positioning is easily implemented for fixed length records, because any file offset can be easily converted to a record number and a byte offset from the start of the record; however, it is more difficult for variable length records.

In the CICS Execution Option, variable length records are handled either by copying the data to or from fixed format workspace or by building a table to cross-reference record identifiers to file offsets. You can designate the method to be used by specifying the `:W` or `:P` options when using the Input File Override (`-I`) and Output File Override (`-O`) execution commands.

The :W option

When the `:W` option is specified using the Input File Override (`-I`) execution command, variable length records are reformatted by copying them from the input file into fixed format workspace before mapping begins and the map will take its input not from the input file but from workspace.

When this option is specified using the Output File Override (`-O`) execution command, data is mapped into fixed format workspace and then copied to its final target after mapping completes. Variable length records are separated out during the copy operation based on a defined record separator.

The following examples demonstrate how to specify the `:W` option using the Input File Override (`-I`) and Output File Override (`-O`) execution commands:

Example	Description
-I1:W MYINPUT	Input 1, **MYINPUT**, is a TSQ of variable length records; copy it to fixed format workspace before mapping it.
-O1:W MYOUTPUT	Output 1, **MYOUTPUT**, is a TSQ that will contain variable length records. Data is mapped to fixed format workspace and after mapping completes, variable length records are separated out and written to the queue.

Output files of variable length records must have a delimiter or delimiters at the end of each record so that the copy operation can detect where one record ends and the next begins. Output record separators are defined in and built by the map. The default delimiter expected is a single byte with the value hex 15, decimal 21. Optionally, you can use different delimiters by specifying them as a comma-delimited list of character or hex values. (X*nn* is the syntax for indicating a hexadecimal value.)

The following example demonstrates how to specify a delimiter list using the Input File Override (`-I`) and Output File Override (`-O`) execution commands:

Example
  Description

-O1:WX0D,X0A
  Output data is mapped to fixed format workspace and records are copied to a TSQ of variable length records using the delimiter list of hex 0D 0A as record separators. In other words, when a 0D 0A byte pair is sensed in the output stream, it is understood that it indicates the end of a record. Record separators are removed.

Because reads against transient data queues are destructive and multiple passes may need to be made against them, data mapped to and from TDQs take place in workspace, regardless of the record format.

The :P option

When the `:P` option is specified using the Input File Override (`-I`) or Output File Override (`-O`) execution command, the data is *profiled*, that is, a table will be built in memory that cross-references record identifiers to corresponding file offsets. It is the responsibility of the map to place record separators in the output data so that the profiling operation can interpret where one record ends and the next begins.

The following examples demonstrate how to specify the `:P` option using the Input File Override (`-I`) and Output File Override (`-O`) execution commands:

Example
  Description

-I1:P MYINPUT
  Input 1 is a TSQ of variable length records; it will be profiled.

-O1@K:PX25 MYKSDS
  Output 1, VSAM KSDS **MYKSDS**, contains variable length records; build a profile of the output using hex 25 as a record separator.

When using the `:W` or `:P` options, a delimiter list can also be specified for record separators to be used for input cards.

This can be useful in a situation in which you defined a map on the PC that reads a text file as input. A text file, by definition, consists of lines delimited by carriage-return/line-feed pairs (0D 0A). If you transfer this data to a z/OS environment as TEXT, the record separators are removed and variable length records are created. Mapping this data is a problem because the map as defined expects each record to be terminated by a carriage-return/line-feed pair that no longer exists. You can specify the following command string:

```
-I1:PX0D,X0A MYINPUT
```

This example command string would result in the bytes 0D 0A being added to the end of each record at the time it is read in.

Using EXEC CICS LINK passing storage buffers

When the z/OS CICS Execution Option is called by an EXEC CICS LINK, it is often because you want to pass to the CICS Execution Option the address of one or more storage buffers containing input data or the address of one or more storage buffers that will contain mapped output data or both address types.

Suppose that you want to call the CICS Execution Option from a z/OS CICS COBOL program using an EXEC CICS LINK and execute a map named **MAPA**. This map has one input and one output, both of which exist as storage buffers. To do this, the COMMAREA passed to the CICS Execution Option should be defined as:

```
01 DTX-LINK-BLOCK-B.
   05 MRLB-VERSION                             PIC X(04) VALUE 'B002'.
   05 MRLB-PARM-COUNT                          PIC 9(08) COMP VALUE 0.
   05 MRLB-PARM-LIST-ADDRESS                       POINTER VALUE NULL.
   05 MRLB-RETURN-CODE                         PIC 9(04) COMP VALUE 0.
      88 MRLB-NORMAL-RETURN                        VALUE 0.
   05 MRLB-RETURN-MESSAGE                      PIC X(80) VALUE SPACE.
   05 MRLB-MAPPING-RETURN-CODE                 PIC 9(04) COMP VALUE 0.
      88 MRLB-NORMAL-MAP-RETURN                    VALUE 0.
```

Next, code a working-storage area similar to the following example to hold the command line. The command line has card overrides for input 1 and output 1 indicates that the source and target exist as storage buffers.

```
01 MAP-COMMAND-LINE
   05 MAP-COMMAND-STRING   PIC X(16) VALUE 'MAPA -I1@B -O1@B'.
   05 FILLER               PIC X(01) VALUE LOW-VALUE.
```

MRLB-PARM-LIST-ADDRESS in DTX-LINK-BLOCK-B should contain the address of a parameter list such as the following example. The parameters need not be in this exact order but the command line must be the first parameter in the list.

```
01 PL00-PARM-LIST.
   05  PL01-MAP-COMMAND           POINTER.
   05  PL02-I1-BUFFER-PARM        POINTER.
   05  PL03-O1-BUFFER-PARM        POINTER.
```

PL01-MAP-COMMAND should contain the address of the null-terminated Command Line, MAP-COMMAND-LINE, defined above. PL02-I1-BUFFER-PARM should contain the address of the following working storage area:

```
01    DTX-DATA-BUFFER.
      05  MRDB-DATA-LENGTH                        PIC S9(8) COMP.
      05  MRDB-DATA-ADDRESS                                POINTER.
      05  MRDB-DATA-NAME-AREA                     PIC X(04).
      05  FILLER REDEFINES MRDB-DATA-NAME-AREA.
          10  MRDB-DATA-NAME-L2                   PIC X(02).
          10  MRDB-NULL-TERMINATOR-L2             PIC X(01).
          10  FILLER                              PIC X(01).
      05  FILLER REDEFINES MRDB-DATA-NAME-AREA.
          10  MRDB-DATA-NAME-L3                   PIC X(03).
          10  MRDB-NULL-TERMINATOR-L3             PIC X(01).
```

MRDB-DATA-LENGTH should contain the actual length of the data in the buffer for input 1. MRDB-DATA-ADDRESS should contain the address of the buffer for input 1. MRDB-DATA-NAME-AREA should contain the null-terminated string ″I1″, indicating that this buffer is to be used as input 1.

PL02-O1-BUFFER-PARM should contain the address of an identical working storage area initialized as follows: MRDB-DATA-LENGTH should contain the full size of the buffer allocated for output 1. MRDB-DATA-ADDRESS should contain the address of the output buffer. MRDB-DATA-NAME-AREA should contain the null-terminated string ″O1″, indicating that this buffer will hold the mapped data for output 1.

MAPLB-PARM-COUNT should be set to 3 (1 for the command line + 1 for the input buffer + 1 for the output buffer). If more or fewer parameters are to be passed, make sure that you adjust the value of MAPLB-PARM-COUNT accordingly. See the **DTXSCTST** sample program.

Using the EXIT function

The `EXIT` function is available when entering mapping rules for fields using the Map Designer on your PC.

Use this function to exit to a user-written program to get data for mapping to a field. See the *Functions and Expressions* documentation. However, there are special considerations when using this function in conjunction with the z/OS CICS Execution Option.

A user exit is invoked by calling the `EXIT` function in a mapping rule using the Map Designer. The `EXIT` function expects three arguments, as seen in the following example:

```
EXIT ("non_null_value", "load_module_name", "parameters")
```

When using the CICS Execution Option:

- The first argument contains a value that is not a null string (a single space is sufficient).
- The second argument contains the name of the exit program (the load module name).
- The third argument is a text item containing parameter information for the exit program. This parameter text item may be up to 64K bytes in length, and the `EXIT` function returns a resulting text item, which also may be up to 64K bytes in length. Execution of this mapping rule causes the text item returned by the program *exit-name* to be placed in the field to which you are mapping data.

Exit programs must be defined as z/OS CICS programs. Any static data areas declared within an exit program, such as COBOL working storage, do not persist across multiple calls to the exit. Four arguments are passed to the user exit program in the COMMAREA:

- The address of a full word that contains the length of the parameter text item.
- The address of a space-padded buffer of up to 64K bytes that contains the parameter text item.
- The address of a full word that contains the length of the resulting text item.
- The address of a buffer of up to 64K bytes into which the resulting text item will be placed.

For an example of source code for a COBOL user exit program, see DTXSCXIT in the DTX.SDTXSAMP PDS included in the WebSphere Transformation Extender installation.

Using the RUN function

The first argument of the **RUN** function is the name of the map to be executed. Under z/OS CICS this must be the name of the map as it exists in the map KSDS. At run-time, the CICS Execution Option converts the map name specified in the **RUN** function by removing any drive, path, and file extension information if it exists. For example, the following map rule will execute the map **MAP_10**:

```
RUN("install_dir\mymaps\map_10.mmc")
```

When using the `RUN` function, by default, the CICS Execution Option execution log, audit log, trace, and problem resolution output are written to temporary storage queues **RUNLOG***nn*, **RUNAUD***nn*, **RUNTRC***nn*, and **RUNDBG***nn*, respectively, where *nn* is the recursion level of the map greater than or equal to 01. You can override these targets as described above.

A map executed using the `RUN` function is loaded once and is removed from storage when the task ends or when the map executed is different from the last map executed. This means that maps repeatedly executed are not reloaded each time.

Abnormal termination

If a task terminates abnormally with an internal error, the return code 16 and an error message are passed back to the calling program.

Diagnostic information is written to the execution log if one is being produced or to the system transient data queue **CSML** if there is no execution log. Diagnostic information for a task can be identified by the task ID that occupies the first eight bytes of each line of output. All temporary files in workspace are deleted before the task ends.

The DTXMPULD utility

In addition to loading maps into the KSDS, where maps are stored for use in the z/OS CICS region as described above, the DTXMPULD utility has the capability of listing and deleting maps from the **KSDS** map.

The command syntax is as follows:

```
DELETE=map_name_or_mask
LIST=map_name_or_mask
```

Either or both of these can be used in combination with the INPUT command that specifies maps to be loaded into the VSAM **KSDS**.

The parameter *map_name_or_mask* can be used to specify a single map or a collection of maps if you use the wildcard character *. For example, DELETE=MAPA will delete only MAPA from the **KSDS** map. DELETE=MAPA* deletes all maps with names beginning with MAPA. LIST=* lists all maps stored in the VSAM dataset. DELETE=* deletes all maps.

To use either the DELETE or LIST, the DDNAME MAPKSDS must identify the map **KSDS**:

```
//MAPKSDS DD DSN=map_KSDS,DISP=SHR
```

If the INPUT command is not used, no output is generated for loading into the map **KSDS**. Consequently, the SORT and REPRO steps of the **DTXCI** JCL should not be executed.

Resource names

For information about creating resource names for use on the CICS Execution Option for the z/OS CICS platform, see ″Resource Names for z/OS Platform″.

Multiple WebSphere Transformation Extender versions

This section presents information about running multiple versions of the WebSphere Transformation Extender CICS Execution Option concurrently in a single CICS region.

This feature will help you migrate to additional releases of the CICS Execution Option without the need to define and configure additional CICS regions. You will also be able to do a staged migration, or phase-approach, to a new release while your existing release is running in that same CICS region.

Enabling the multiple WebSphere Transformation Extender feature

This feature is enabled by making the names of the files and programs you are currently installing unique to the names used in other releases you have installed previously.

It is done by embedding the product version number, *nnn*, in the resource name, D*nnn*MAP. An example is to specify D810MAP as the resource name you are currently installing, while MERCMAP is the resource name used in your previous install.

With this resource naming convention, you must update the CICS resource definitions for each release of CICS Execution Option and duplicate the associated VSAM datasets.

The result is that a completely separate copy of the product is installed and none of its resources are shared with any other version of the product that exists in the same CICS region. The user Dynamic Storage Area used by CICS Execution Option components will effectively be doubled.

Using the multiple installations

After the new release of the CICS Execution Option has been installed, you need to be able to distinguish between the multiple installations and then invoke them.

There are two ways to identify and run multiple installations of the CICS Execution Option depending on which method you use to invoke the CICS Execution Option.

- If you invoke the CICS Execution Option through the EXEC CICS START command, Transient Data Queue trigger, or by entering the transaction ID at a terminal
  - use the transaction ID D*nnn* (where *nnn* is the version of the release you have installed) and MERC or DSTX depending on the specific release you currently have installed.

- If you invoke the CICS Execution Option through the EXEC CICS LINK command
  - use the DTX*nnn*KI (used with XPLINK) or DTX*nnn*CI program names (where *nnn* is the version of the release you have installed) for the new version of the CICS Execution Option and MERCCICS or DSTXCICS depending on the specific release you currently have installed.

    Note: If you do not need to use the multiple CICS Execution Option feature, you can use the alias name, **MERCCICS** for the program name for backward compatibility.

Chapter 5. Resource names for z/OS platform

These are the details for creating resource names for use on z/OS Batch and CICS platforms.

Resource names created in the Resource Registry can be used in maps running on z/OS platforms. However, there are a few things to take into consideration:

- Resource values must conform to z/OS rules for their types, such as DDNAMES and Database names.
- Virtual server defined for the z/OS platform must be named **MVSSERVER**
- Resource name file (**.mrn**) must be copied to z/OS platform
- Name and location of resource name file are specified in either **DTXMRN** DDNAME for Batch region or **DTXMRN** VSAM dataset definition for CICS region
- Resource configuration file (**.mrc**) is not required

Resource name aliases

Because the z/OS platform does not recognize file path information, the WebSphere Transformation Extender automatically truncates the path specified in input and output cards when the map is executed.

Therefore, it is not necessary to create a resource name that has a value defined for a file path.

Resource name limitations for files and DDNAMES

To conform to z/OS conventions, the WebSphere Transformation Extender truncates file and resource names to 8 characters when a map is executed.

Therefore, resource names and their values have an 8-character limitation. Because this includes the percent (%) symbols that enclose the resource name when using it in a source or target resource, the resource name must not be longer than 6 characters. For example, the resource name **MyFile** is specified as **%MyFile%** when used in the **Command** setting of an input or output card.

The resolved resource value is actually a DDNAME that is defined on the z/OS Batch platform. The resolved value must conform to the rules for DDNAMES.

See ″DDNAMES″ for more information about defining DDNAMES for resolved resource values.

Virtual server names

When creating virtual servers to represent the z/OS deployment environment, **MVSSERVER** (case-sensitive) must be defined as the virtual server name.

MVSSERVER is the only server name the WebSphere Transformation Extender recognizes.

Even if other virtual servers are also defined in the same **.mrn** file, **MVSSERVER** is the only name recognized by the z/OS version of the WebSphere Transformation Extender as being the active server.

Specifying resource name file for batch

When associating a resource name file for the z/OS Batch version of the Command Server, the name and location of the **.mrn** file are specified in the **DTXMRN** DDNAME definition in a JCL (Job Control Language) file.

DDNAMES

DTXMRN is the only DDNAME recognized by the Command Server that contains the **.mrn** file specifications. A DD statement specifies the file name or path of the **.mrn** file.

To specify a resource name file for Batch
1. Create the resource name **.mrn** file in the Resource Registry.
2. Copy the **.mrn** file to the z/OS platform.
3. Update the JCL file.
   - Specify the **DTXMRN** DDNAME definition.
   - Include a DDNAME definition for each resolved resource value used.

Specifying resource name file for CICS

When associating a resource name file for the z/OS CICS version of the WebSphere Transformation Extender, the name and location of the resource name file are specified in the **DTXMRN** Virtual Storage Access Method (VSAM) dataset.

VSAM

JCL files are used when executing the WebSphere Transformation Extender in the CICS region just as they are for Batch. However, instead of specifying the **.mrn** file in a **DTXMRN** DDNAME definition in the same JCL file in which the job steps that invoke the WebSphere Transformation Extender are also specified, the **.mrn** file is specified in a **DTXMRN** VSAM dataset definition submitted in a separate JCL file. The **DTXMRN** VSAM dataset definition includes the name and location of the **.mrn** file.

To specify a resource name file for CICS
1. Create the resource name .mrn file in the Resource Registry.
2. Copy the **.mrn** file to the z/OS CICS platform and then load it into a VSAM dataset.
3. Update the separate JCL file.
   - Specify the **DTXMRN** DDNAME definition to define this VSAM dataset to the CICS region.
   - Include a CICS resource definition for each resolved resource value used.

Only one **DTXMRN** VSAM dataset can be used while the CICS region is running, therefore, only one resource name file can be referenced during this duration. Because of this, it is suggested that you define all resource names and their associated values that will be used on the z/OS CICS version of the WebSphere Transformation Extender in one resource name file.

Resource configuration files

A resource configuration file (**.mrc**) is not used by the z/OS version of WebSphere Transformation Extender to determine which **.mrn** file to use and what server is active. Instead, the **.mrn** file is specified in either a DD statement for the **DTXMRN** DDNAME or in the **DTXMRN** VSAM dataset definition, and **MVSSERVER** is recognized as the only active server for the z/OS version of WebSphere Transformation Extender.

Because a resource configuration file is not required, it is not necessary to specify an .mrc file in the Map Designer or the **dtx.ini** file.

Notices

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

IBM WebSphere Transformation Extender, Version 8.1

Index

Printed in USA