IBM WebSphere Transformation Extender

Resource Adapters Introduction

Version 8.1

Note

Before using this information, be sure to read the general information in "Notices" on page 75.

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, email DTX_doc_feedback@us.ibm.com. We look forward to hearing from you.

Contents

Chapter 1. Resource adapters overview

WebSphere Transformation Extender resource adapters are used to retrieve and route data. They provide access to databases, files, messaging systems, enterprise applications and other data sources and targets. You can use adapters with the Command Server, Launcher, Software Development Kit, or with a map in a map rule.

Each adapter includes a set of adapter commands that can be used to customize its operation. Use the adapter commands to specify different queues and queue managers, specific messages by message ID, specific sets of messages with the same message ID, message descriptors in your data, and more.

Resource adapters answer the questions ″Where should the data come from?″ and ″Where should the data go?″

The resource adapters provided with the WebSphere Transformation Extender products enable data transformation and adapter-specific behavior recognition on different systems and data structures.

Source and target data may be in a file, from a specific application, in a database, from a message queue, from an archive file, or from other sources. The client components of the Design Studio,Integration Flow Designer, Map Designer, Type Designer, and Database Interface Designer are integral to using the resource adapters.

Each input of data and each output of data requires content definition settings. These card settings are specific to input and output cards, although some settings are common to both cards.

You can also develop your own adapters, enabling new types of data sources and targets, by using the open adapter interface provided in WebSphere Transformation Extender and in the WebSphere Transformation Extender Software Development Kit.

Importers

The Importer Wizard uses a series of maps to convert metadata into a type tree script file (**.mts**). The Type Tree Maker then processes this type tree script and generates a type tree that contains all the supported types that are defined in the imported metadata.

Many of the type trees that are generated by the Importer Wizard can be immediately used for map development. However, depending on the contents of the interface-specific metadata file, it might be necessary for the generated type tree to be modified using the Type Designer.

For information about importers corresponding to adapters that you have installed, see the specific resource adapter documentation.

System requirements

The minimum system requirements and operating system requirements for most Windows and UNIX-based adapters are listed in the release notes. It is assumed that a Command Server has already been installed on the computer where the utility adapters are to be installed for run-time purposes. Any additional requirements to install and run the messaging adapters vary with each adapter. Those differences are discussed in each adapter-specific documentation.

General rules for adapter commands

Following is a list of the general rules that apply when specifying adapter commands:

- Each command must begin with a hyphen.
- Command characters can be uppercase, lowercase, or mixed-case.
- Command strings are parsed in left-to-right order.
- The order of commands is not important.
- Command files can contain multiple lines. Line breaks can be used in place of the required space that must exist between commands. The line breaks are converted to spaces before processing.
- Space characters:
  - Spaces preceding the first command are ignored.
  - At least one space between commands is required (for example: **-KEEP -TRACE**).
  - A space is required preceding a variable value (for example: **-TRACE+S** *full_pathname*).
- The value of the **Command** setting (adapter-specific commands) in input and output cards and in a GET or PUT function within a map rule may not exceed 260 bytes. The expansion of the resource value is included in this limitation.

For platform-specific information about using specific adapter commands with execution commands, refer to the following:

- ″For Windows Sources and Targets on the Command Line″
- ″For UNIX Sources and Targets on the Command Line″

Command syntax summaries

A syntax summary is a comprehensive list of the adapter-specific commands organized for use with data sources and data targets. Syntax notation, as applied to all command documentation, indicates required and optional commands, as well as options available for each command.

Required commands appear at the top of the list. Optional commands appear in brackets [ ]. Command dependencies are also noted when commands appear on the same line.

The following is a portion of the syntax summary for the IBM MQSeries messaging adapter command available for use with data sources:

```
-QMN queue_manager_name
-QN queue_name
```

This example illustrates that both the Queue Manager Name adapter command (**-QMN**) and the Queue Name adapter command (**-QN**) are required for data sources. In this example, you may use the Queue Manager Name adapter command (**-QMN**) to specify the name of the queue manager on which the queue (specified by the -QN command) exists.

The following is an additional example of the syntax notation used by the IBM MQSeries messaging adapter:

```
-QMN queue_manager_name
-QN queue_name
[-MID message_ID|-HMID hex_message_ID]
```

This example illustrates the use of brackets to indicate optional message ID adapter commands. The brackets [ ] indicate that both of the message ID commands are optional. The pipe | indicates that you may choose one of these commands, but not both.

In this example, you may use the Message ID adapter command (**-MID**) to specify a particular message ID for a data source. Or you may choose to use the Hex Message ID adapter command (**-HMID**) to specify message identifiers using hexadecimal pair notation.

* The File and Sink adapters do not require any adapter commands for their operation. The File adapter uses only the full pathname and filename for a parameter.

Adapter command aliases

Specify the adapter commands using an execution command string on the command line. You can also create a command file that contains adapter commands dictating the desired execution settings.

Use the appropriate Input Source Override and Output Target Override execution commands with the adapter alias.

Refer to the specific adapter documentation for detailed information for each alias command.

Multiple adapter versions

For some adapters, multiple-versions are provided to support different releases of the resource with which the adapter is communicating. They are controlled by the **adapters.xml** file, which is provided in the default installation directory.

The **adapters.xml** file lists the multiple-version of the adapter, with the specified version uncommented. You must edit the **adapters.xml** file to comment out the adapter that you do not want to use. The XML comment syntax uses the exclamation point as a ″not this″ indicator:

```
<!-- <M4Adapter name="not_this_adapter"/> --
```

For example:

<M4Adapter name=″**TIBCO**″ alias=″**RV**″ id=″**109**″ type=″**msg**″ library=″**m4tibrv6**″ version=″**7.x**″ />

The syntax of the **adapters.xml** file shows that the **TIB/RV** adapter version is currently enabled. Ensure that only one version of an adapter is uncommented.

If more than one version of an adapter exists in the **adapters.xml** file, the file will fail processing. In addition, there will be no adapters listed in the **Type** setting in the Database Definition dialog box in the Database Interface Designer.

Editing the adapters.xml file

To comment out an adapter line in the **adapters.xml** file, you must insert the XML comment syntax:

<!-- .... **--**

After the less than sign (<) is the exclamation point (!) as a ″not this″ indicator, followed by two dashes (--). The adapter line that consists of the *adapter name, alias, id, type, library,* and *version* information for the specific adapter follows. The adapter line is then closed with the syntax of two dashes (--) and a greater than (>) sign.

To comment out an adapter

1. Open the **adapters.xml** file located in the default installation directory in an editing program such as UltraEdit or Visual Slick.
2. Locate the adapter line (name) that you want to comment out.
3. Type the XML comment syntax around the adapter line (name).

Example

In this example, the Tuxedo adapter version has been commented out and cannot be used as a source or target.

Before XML Syntax Added

```
<M4Adapter name="Tuxedo"     alias="TUX"     id="114" type="msg"
library="m4tux"    version="7.1"/>
```

After XML Syntax Added

```
<!-- <M4Adapter name="Tuxedo"     alias="TUX"     id="114" type="msg"
library="m4tux"    version="7.1"/>--
```

To un-comment an adapter

1. Delete the XML comment syntax for the specific adapter line (name). Make sure that you delete the comment syntax at both the beginning and end of the adapter line (name).

Configuring Java-based adapters

Jar files included with the software and installed in the **install_home** directory are normally added to the CLASSPATH environment variable automatically as in the case of the Command Server and the Launcher. You must modify the CLASSPATH environment variable for all Java-based adapters that rely on external .jar files in order to execute properly.

You can do this directly, or by adding the **.jar** files to the **dtx.ini** file. You must use a full path in the **dtx.ini** file. To do so, add entries to the [External Jar Files] section. Refer to the *Launcher documentation* for more information on the **dtx.ini** file.

For example:

```
[External Jar Files]
jar1=c:\J2EE\lib\j2ee.jar
jar2=c:\mypath\myjar.jar
```

UNIX only

For all Java components the directories containing the Java Virtual Machine (JVM) libraries need to be in the library path environment variable. Typically, this is already defined in the environment, however if it is not, then it must be manually added to the library path environment variable. The following environment variables exist for:

UNIX Platform	Environment Variable
AIX	LIBPATH
Solaris	LD_LIBRARY_PATH
HP-UX	SHLIB_PATH
Linux	LD_LIBRARY_PATH

API usage

When a map is started using the Java or CORBA API, where the main process starts with the java command, all necessary **.jar** files from the install_home\ directory and external **.jar** files related to the adapter must be added to the CLASSPATH environment variable manually.

Encoding and decoding data

There are a number of adapters which only manipulate data (for example, Base64, SOAP), while others provide transport (for example, HTTP). Chaining these adapters together provides greatly enhanced functionality.

Using the `-ENCODE` and `-DECODE` commands in combination with the `-TRANSPORT` command allows you to effectively combine the encoding/decoding capabilities of one adapter with the getting or sending of data of another adapter.

Supported adapters

Almost any adapter can be used as a transport adapter, although it must be used in its supported contexts. The only adapter that cannot be a transport adapter is the File adapter because it is the only adapter that is not started from the Resource Manager.

The following adapters can be used as encoding or decoding adapters:

- Base64
- Quoted-Printable
- SOAP

Limitations

The following limitations exist:

- There can be only ONE transport adapter in a card. Since the transport generally represents the end point or start point in a process, it does not make sense to permit more than one transport adapter.
- Encode/decode adapters can be nested. However, the total length of the command line cannot exceed 2000 bytes.

Encoding

Data can be encoded using a specified adapter. The encoding adapter is called before the transport adapter in an output card, PUT or GET where request data is passed as the third parameter.

The following characteristics exist for the `-ENCODE` command:

- The `-ENCODE` command is used within the command line of a transport adapter, or another encoding adapter.
- The `-ENCODE` command cannot be used in an input card.

For example:

```
-ENCODE 'encoding_adapter (command_line)'
```

The `-ENCODE` command is not required if the `-TRANSPORT` command is used. Encoding is implied by using `-TRANSPORT` in an output card, PUT or GET where request data is passed as the third parameter.

See ″Encode/Decode Scenarios″ for more information.

Decoding

You can also decode the data using a specified adapter. The decoding adapter is called after the transport adapter in an input card or GET.

The following characteristics exist for the `-DECODE` command:

- The `-DECODE` command is used within the command line of a transport adapter, or another decoding adapter.
- The `-DECODE` command cannot be used in an output card or a PUT function.

For example:

```
-DECODE 'decoding_adapter (command_line)'
```

The `-DECODE` command is not required if the `-TRANSPORT` command is used. Decoding is implied by using `-TRANSPORT` in an input card, or GET function.

See ″Encode/Decode Scenarios″ for more information.

Transporting

After specifying either an encode or decode command, data can be transported using a specified adapter. The transport adapter is called:

- Before the decoding adapter in an input card or GET
- After the encoding adapter in an output card or PUT
- Between the encoding and decoding adapters in a GET where request data is passed as the third parameter.

The -TRANSPORT command is used in the command line of an encoding or decoding adapter. If used from a GET, the *transport_adapter* must support request-reply and the adapter it is used with the first parameter of the GET must support both encode and decode functions.

For example:

```
-TRANSPORT 'transport_adapter (command_line)'
```

When reversing the chained adapters and using the -ENCODE or -DECODE commands rather than the -TRANSPORT command the -TRANSPORT command the behavior is unaltered.

For example, the following PUT function calls are functionally equivalent.

```
=PUT(" email ", "-TO fred@hotrods.com -ENCODE ' SOAP (-T)'", Message)
=PUT(" SOAP ", "-T -TRANSPORT ' email (-TO fred@hotrods.com)'",
Message)
```

See "Transport Encode/Decode Examples" for more information.

Encode/decode scenarios

The following scenarios apply when using the encode/decode commands in an input card, output card, or in a GET function.

Input card or GET

In an input card or GET, where the -DECODE command is specified, data is:

- received by the transport adapter,
- decoded by the decoding adapter.

Output card or PUT

In an output card or PUT function, where the -ENCODE command is specified, data is:

- sent by the transport adapter,
- encoded by the encoding adapter.

GET function

In a GET function performing a request/reply using the optional third parameter, the following activities occur:

- Encoding by the encoding adapter
- Sending by the transport adapter
- Receiving by the transport adapter
- Decoding by the decoding adapter

Transport encode/decode examples

The following examples show how the transport adapter encodes or decodes the data when used in an input card, an output card, or in a GET function.

SOAP Input card

Adapter
: SOAP

Command
: `-ENV -TRANSPORT ’HTTP(-URL` *http://www.mysite.com*)’

Meaning
: The HTTP adapter is called for the transport, then the data is passed to the SOAP adapter to decode.

SOAP Output card

Adapter
: SOAP

Command
: `-ENV -TRANSPORT ’HTTP(-URL` *http://www.mysite.com*)’

Meaning
: The SOAP adapter is called to encode the data, then the data is passed to the HTTP adapter to transport.

SOAP request/reply in a GET function

Function Call
: Response = `GET(″SOAP″, ″-ENV -TRANSPORT ’HTTP(-URL` *http://www.mysite.com* `, Request)’`

Meaning
: The SOAP adapter is called to encode the data. The data is then passed to the HTTP adapter to send the request and pick up the response. The SOAP adapter is called again to decode the response.

Chapter 2. Using resource adapters

Resource adapters can be specified:

- As a resource in the Map Designer
- In map rules and component rules when using the functions RUN, GET, or PUT
- From the execution settings in the Integration Flow Designer
- On the command line for a Command Server or TX Programming Interface

  A wide range of execution commands are available with the Command Server or Software Development Kit. For a complete list of the available options, refer to the Execution Commands documentation.

  See the Map Designer documentation for information on source and target settings in input and output cards.

Chapter 3. Using an adapter in the Map Designer

Use the **Source** setting on an input card or the **Target** setting on an output card to specify an adapter, for example, HTTP, or database, representing any of the database adapters. The adapter-specific commands are entered in the **Command** setting on the card. The adapter-specific commands define how to connect to the data source or target and how to retrieve or route the data.

Using the Map Designer, select an adapter for a source to retrieve files or data, or a target to transfer files or data.

Maps define how to generate data objects of a certain type. A map contains input and output cards that transform data content from source formats to target formats.

- The **Command** setting on the input card of a map identifies the source of the input data and specifies the use of a resource adapter for input data.
- The **Command** setting on the output card specifies the use of a resource adapter for output data.

On an input card, a **Source** setting of **File** defines the input source as a file and the **Source FilePath** setting is used to define the **Name** and **Path** of the input file.

If the input data is coming from a database, the **GET Source → Command** setting might include commands specifying the database name, user ID, and password.

See the following example to learn how to specify the FTP adapter as a data source in the Map Designer.

Specifying the FTP adapter in the Map Designer

The FTP value for the **Source** setting in this input card identifies the FTP adapter as the source of the input data.

To specify the FTP adapter:

1. In the input card for a map, set **Source** to **FTP**.
2. For the **Command** setting, enter the required FTP URL adapter command (**-URL**), the name of the file to transfer, and other FTP adapter commands as needed.
3. Click **OK**.

The **Source Command** value for the FTP input card above is:

```
-TS -URL FTP://sales@host/c:/forms/myfile.txt
```

When this map is run, a file named **myfile.txt** is retrieved from a remote host named **sales**. The Summary Trace adapter command **-TS** specifies that the adapter will create a summary trace file to report adapter activity information during the FTP process.

Chapter 4. Using an adapter in a map or component rule

Specify adapter commands using the GET, PUT, RUN, DBLOOKUP or DBQUERY functions when defining map rules in the Map Designer or in component rules in the Type Designer. The DBLOOKUP and DBQUERY functions are used for databases only. You must specify the resource adapter with an execution command and a resource adapter alias. For information about adapter aliases, see ″Adapter Command Aliases″.

Use adapter commands in functions and in component rules in the Type Designer and map rules in the Map Designer.

Examples of database adapter commands

- The following example shows a map rule using a DBQUERY function with database adapter commands to specify particular adapter settings.

  ```
  DBQUERY ("SELECT PART_NAME from PARTS where PART_NUMBER =1",
  "-DBTYPE ORACLE -CONNECT MyDatabase -USER janes"
  ```

  This example uses the -DBTYPE ORACLE adapter command to specify the database adapter type as Oracle and -CONNECT MyDatabase to designate MyDatabase as the Oracle connect string. The -USER janes adapter command identifies janes as the user ID.

For information about using the DBLOOKUP and DBQUERY functions, refer to the Database Interface Designer documentation.

- Use adapter commands with either the Input Source Override - Database execution command (-ID) or the Output Source Override - Database execution command (-OD) in a RUN function to override particular settings. The following example shows using a RUN function to specify adapter commands without using an **.mdq** file:

  ```
  RUN ("somemap.mmc" ,
        "-od1 '-dbtype oracle -connect remotedb -user " +
        userid:profile + " -password " + password:profile +
        " -table purchordr'")
  ```

In this example, a map named **somemap.mmc** is being run. The output of this map will be inserted in the PurchOrdr table in an Oracle database defined by the RemoteDB connect string. The logon, user ID, and password settings are retrieved from an input card named Profile.

For information about using the RUN, GET and PUT functions, refer to the Functions and Expressions documentation.

Chapter 5. Using an adapter in the Integration Flow Designer

In the Integration Flow Designer, while accessing the Launcher or Command Server execution settings for a map, you can use the adapter as a source to retrieve files or as a target to transfer files.

The **Execution Mode** of the system component determines the execution settings modified to use a resource adapter.

If the **Execution Mode** setting is Launcher, edit the **Event Server Settings** for that system component.

If the **Execution Mode** setting is Command Server, edit the **Command Server Settings**.

For documentation purposes, these settings are referred to as execution settings regardless of the execution mode selected.

You can use the adapter as a source to retrieve files or as a target to transfer files.

For example, to override database-specific adapter settings in an input card or to use the adapter for a source, access the execution settings for a map. From the **GET > Source** list in the execution settings of the map, select **Database**. In the **GET → Source → Command** setting, specify the adapter commands.

An example of a database-specific adapter command is as follows:

```
-M c:\MDQfiles\Orders.mdq -DN Parts -TE+
```

where `-M` and `-DN` are used to specify usage of a database/query file named **Orders.mdq** and a database named **Parts**. The `-TE+` command causes database trace information containing only errors occurring during map execution to be produced and appended to the existing trace file.

Specifying the TIB/RV adapter in the Integration Flow Designer

The following example illustrates the use of the **TIB/RV** adapter in the Integration Flow Designer.

To specify the TIB/RV adapter:

1. Open a system in the Integration Flow Designer.
2. Open the Launcher settings for a map component.
3. Go to the **Input(s)** *#1 in1* **> GET → Source** setting.
4. For the **Source** setting, select **TIB/RV** from the drop-down list.
5. In the **Command** field, enter a TIB/RV adapter command. For example, enter:

   ```
   -SBN rvcm.test.in -CONFIRM -T -CMN input_CM -LFN rvcmcfmin.lgr
   -LSN 100
   ```

This example input card retrieves incoming messages with the subject name defined as **rvcm.test.in** using the Subject Name adapter command (`-SBN rvcm.test.in`). The -CONFIRM adapter command requests that the messaging adapter send an explicit confirmation to the sender that the message was received after successful completion of the map. The Trace adapter command (-T) dictates that a trace file be created to report adapter activity information, recording the events that occur while the adapter is retrieving this data.

The adapter returns messages with the certified session named **input_CM** using the required Certified Session Name adapter command (`-CMN input_CM`). The ledger file named **rvcmcfmin.lgr** is specified as the ledger file to log the TIB/Rendezvous PRO certified delivery for this particular certified session using the Ledger File Name adapter command (`-LFN rvcmcfmin.lgr`). The adapter waits 100 seconds for a message receipt, as specified using the Listen adapter command (`-LSN 100`).

Chapter 6. On the command line

In addition to using a command string on the command line, you can also use adapter commands in command files, batch files, shell scripts, or with the Platform API.

Use adapter-specific execution commands to:

- Override particular settings for an existing source, such as a user ID and password for database adapters.
- Override an existing source or target, using file, application, message or another entity.

Specify the preferred adapter commands using an execution command string on the command line, or, create a command file that contains adapter commands dictating the execution settings.

For information on adapter-specific input and output execution command overrides, see the Execution Commands documentation.

Database-specific adapter commands on the command line

For a database-specific adapter, an existing data source which is a file, can be overridden and specified to be a database using the Input Source Override - Database execution command (`-ID`). Or, you can override an existing data target that is a table in one database with a table in a different database using the Output Source Override - Database execution command (`-OD`). For information about all of the options to use within these database execution commands, refer to the Execution Commands documentation.

The values specified using the Input Source Override - Database execution command (`-ID`) and the Output Source Override - Database execution command (`-OD`) can be used to designate a query or table to be used as the data source or target, or to override one or more of the adapter commands of an existing database definition as follows:

- using an **.mdq** file

  Specify the name of an .mdq file and the name of a particular database that will override the appropriate adapter settings. In this usage, the .mdq file must be available at map execution time. The values in the .mdq file will override all values defined in the map.
- not using an **.mdq** file

  Use a command string that includes all of the database adapter commands necessary to specify the desired settings for a map. An **.mdq** file is not used.

Command line examples

The following examples show various ways to use the database-specific adapter commands on the command line.

The following examples for data sources and targets are for Oracle database adapters using UNIX-specific syntax. For the Windows platform, adapter commands are enclosed in single quotation marks. Specify adapter commands using an **.mdq** file as follows:

```
dtx parts -ID1 "'-MDQ crib1.mdq -DBNAME Inventory
-QUERY 003A'"
```

This example shows a data source override. The data for input card 1 for the parts map will come from the Inventory database using the query named 003A, all of which are defined in the **crib1.mdq** database/query file.

- Override individual adapter commands as follows:

  ```
  dtx XlatLgcy -ID3 "'-STMT SELECT CustID, Name,
  Addr FROM Cust_Table'"
  ```

This example shows how a SELECT statement may be specified on the command line. Note that the SQL statement consists of all of the input following `-STMT` until the next adapter command or until the single quotation mark ending the adapter command is reached. In this example, the remaining database information from the values compiled into the map file XlatLgcy is used (for example, the **.mdq** file, database, user ID, and other non-overridden values).

- Specify and override the settings in an **.mdq** file:

  ```
  dtx CvtApps -OD6 "'-MDQ /test/new.mdq -DBNAME Purchases -USER
  John -PASSWORD sds34u -TABLE NewerTable -UPDATE'"
  ```

This example shows that the target for output card 6 in a map named CvtApps. This map is first overridden by the database definition Purchases in **/test/new.mdq** by using the Database/Query File adapter command (`-MDQ`) and the Database Name adapter command (`-DBNAME`). The **new.mdq** database/query file is first read to override the data target that is compiled into the CvtApps map. Subsequently, the User ID adapter command (`-USER`), the Password adapter command (`-PASSWORD`), the Table Name adapter command (`-TABLE`), and the Update adapter command (`-UPDATE`) override those settings obtained from the **.mdq** file.

- Override both a data source and a target adapter command:

  ```
  dtx XferMsgs -ID1 "'-MDQ /share/prod.mdq -DBNAME LgcyProd
  -QUERY xToday'"    -OD1 "'-DBTYPE ORACLE -CONNECT ProdDB
  -USER shaz -PASSWORD shaz01 -TABLE ActiveMsgs -UPDATE'"
  ```

This example shows both an input and an output being overridden. For the data source (`-ID1`), the **.mdq** file (**/share/prod.mdq**), database name (**LgcyProd**), and query (**xToday**) are specified. For the data target (`-OD1`), an Oracle database (**ProdDB**) is designated by the connect string. Subsequent adapter commands specify the user ID (**shaz**), password (**shaz01**), and table (**ActiveMsgs**) to be updated.

For Windows sources and targets on the command line

When using execution commands on Windows platforms, the specified adapter commands must begin with a single quote and end with a single quote. Separate each command with a space within the single quotes.

For example:

`-IMTUX1` `'`*`command1 command2`*`'`

When reference is made to a single quotation mark in this documentation, it is referring to the ASCII character 039 that is created by selecting the key to the left of the **Enter** key on a PC keyboard. The double quotation mark character is the ASCII character 034, created on the PC keyboard by pressing **Shift** plus the double quotation mark-single quotation mark key to the left of the **Enter** key.

For UNIX sources and targets on the command line

When using execution commands, while running in a shell environment on UNIX platforms, the specified adapter commands must begin with a double quote, followed by a single quote and end with a single quote, followed by a double quote. Separate each command with a space within the single quotes.

For example:

```
-IAMQS1 "'command1 command2'"
```

Chapter 7. Functions

For detailed information about using the GET, PUT and all other functions, refer to the Functions and Expressions documentation.

Using the GET function

The GET function returns messages from the adapter as a single text item.

The following example shows a map rule using a **GET** function with messaging adapter commands to specify particular adapter settings:

```
GET ("MQS","-QMN globalque -QN top1")
```

This **GET** example retrieves messages from the MQSeries server messaging adapter as a single text item. The Queue Manager Name adapter command (**-QMN globalque**) specifies a queue manager named **globalque**. The Queue Name adapter command (**-QN top1**) specifies the **top1** queue.

For more adapter-specific usage information, refer to adapter-specific documentation.

Using the PUT function

The PUT function sends messages to the adapter as output and returns a single text item that is always ″none″.

For all adapters, the scope of the PUT function is burst. The executable map controls the transaction behavior, commit or roll back, regardless of nested RUN functions.

The following example shows a map rule using a **PUT** function with messaging adapter commands to specify particular adapter settings:

```
PUT ("MQSC","-QMN myque -QN best",TEXT(messageobject))
```

This PUT example places `TEXT(messageobject)` on the MQSeries client messaging adapter and returns a single text item of NONE. The Queue Manager Name adapter command (`-QMN myque`) specifies the myque queue manager. The Queue Name adapter command (`-QN best`) specifies the best queue.

For more adapter-specific usage information, refer to each adapter-specific documentation.

Using the RUN function

In a RUN function, use adapter commands with the appropriate Input Source Override execution command or the Output Source Override execution command to override particular settings.

The following example shows using a RUN function to specify adapter commands without using a database/query file (**.mdq**):

```
RUN ("somemap.mmc",
    "-od1 '-dbtype oracle -connect remotedb -user " +
    userid:profile + "-password" + password:profile +
    "-table purchordr'")
```

In this example, a map named **somemap.mmc** is being executed. The output of this map is to be inserted in the **PurchOrdr** table in an Oracle database defined by the **RemoteDB** connect string. The logon, user ID, and password settings are retrieved from an input card named **Profile**.

For more information about using the RUN function, refer to the Functions and Expressions documentation.

For more adapter-specific usage information, refer to adapter-specific documentation.

Chapter 8. Using wildcards

In most cases, the use of wildcards is supported when using resource adapter commands. Refer to the specific adapter documentation to see if wildcards are supported.

If resource adapter commands with wildcards are used with event triggers on input, events are coordinated to ensure that wildcards are matched. A wildcard value returned from an event trigger is used in other non-input event input cards or output cards, substituting any wildcard value that may exist in any of their properties.

For example:

Input card 1 (event):	MQSeries	MID = *
Input card 2 (not-event):	MQSeries	MID = 123*456
Output card 1:	File	Filename = OUT*.TXT

If a message is received from input card 1 with a message ID of **XYZ**, then:
- input card 2 is called with the MID property set to **123XYZ456**.
- the file created from output card 1 is named **OUTXYZ.TXT**.

Chapter 9. Scope settings and FetchUnit interpretation

The following table shows the **Scope** settings and **FetchUnit** interpretation available by adapter type:

	Transactional Data Units				
Source Adapter	**Source > Scope Settings**	**Target > Scope Settings**	**Aggregates**	**Divides**	**FetchUnit Interpretation**
Application	M [1], B [2], C [3]	M [1], B [2], C [3]	No	No	Logical
Archive (Tar, Zip)	M, B, C	M, B, C	Yes	Yes	File in archive
Batch File	M, B, C	M, B, C	No	No	Logical
COM Automation	M, B, C	M, B, C	No	Yes	Logical
Database	M, C	M, B, C	No	Yes	Row
E-mail	M, B, C	M, B, C	Yes	Yes	Message
File	M	M, C	No	No	Logical
FS Manager	M, B, C	M, B, C	Yes	Yes	Message
FTP	M, B, C	M, B, C	Yes	Yes	File
GZIP	M, B, C	M, B, C	Yes	Yes	File in archive
HTTP	M, C	M, C	No	No	Logical
MessageQ (client & server)	M, B, C	M, B, C	Yes	Yes	Message
MQSeries (client & server)	M, B, C	M, B, C	Yes	Yes	Message
MSMQ	M, B, C	M, B, C	Yes	Yes	Message
Oracle AQ	M, B, C	M, B, C	Yes	Yes	Message
PeopleSoft Message Agent	M, B, C	M, B, C	No	No	Logical
R/3 (ALE, BAPI, BDC)	M, C	M, B, C	No	No	Logical
Shell Script	M, B, C	M, B, C	No	No	Logical
Socket	M, B, C	M, B, C	Yes	Yes	Message
TIB/RV (7.x, TX)	M, B, C	M, B, C	Yes	Yes	Message
Tuxedo (7.1)	M, B, C	M, B, C	Yes	Yes	Message
VAN	M, B, C	M, B, C	No	No	Logical

[1] M = Map

[2] B = Burst

[3] C = Card

Chapter 10. Using global transaction management

Certain database and messaging adapters support triggering and global transaction management. Triggering and global transaction management allows monitoring of transactions that can include operations on two or more different data sources. For more information, refer to the Global Transaction Management documentation.

Chapter 11. File Adapter

The File adapter is a listener for the Launcher that supports file triggering. The `GET` and `PUT` commands are internal for this adapter.

The only parameter available for the file adapter is the filename specified in the **FilePath** card setting. The settings for this adapter are:

- **GET Source** → **FilePath** for the input card
- **PUT Target** → **FilePath** for the output card

  The File adapter does not require any adapter commands for its operation. It uses only the full path and filename for a parameter.

  The File adapter is internal to the Command Server, Launcher, and Platform API. However, listener functionality for the Launcher is an external adapter.

Chapter 12. Text File Importer

The Text File Importer is accessible through the Importer Wizard. The Importer Wizard is an applet that is run from the Type Designer by selecting **Import** from the **Tree** menu.

The Importer Wizard uses a series of maps to convert metadata into a type tree script file (**.mts**). The Type Tree Maker then processes this type tree script and generates a type tree that contains all the supported types that are defined in the imported metadata.

Many of the type trees that are generated by the Importer Wizard can be immediately used for map development. However, depending on the contents of the interface-specific metadata file, it may be necessary for the generated type tree to be modified using the Type Designer.

You cannot simultaneously have a type tree open in the Type Designer while importing a type tree with the same name.

Supported character sets

The Text File Importer supports the listed character sets.

Western character sets, international and non-international versions)

- ASCII
- EBCDIC
- UTF - 8
- UNICODE Little Endian
- UNICODE Big Endian

Japanese character sets (international version)

- SJIS
- EUC
- UNICODE Big Endian
- UNICODE Little Endian
- UTF - 8

Running the Text File Importer

The Text File Importer automatically generates type trees that describe the format of the data in text files.

To run the Text File Importer:

1. Start the Type Designer and select **Import a type tree** from the Startup window.

   The Importer Wizard window opens.

2. Select **Text File**.

   The Importer Wizard-Text File window opens.
3. Select the file to import and click **Next**.
4. If applicable, specify the language and the character set of the input file. Choices for language are Western for non-international versions and Western and Japanese for International versions. See ″Supported Character Sets″ for a complete list of supported character sets.
5. Click **Next**.
6. In the **File Name** field, enter the name of the type tree you are about to create.
7. Optionally, select the **Merge with existing tree** check box if you want to merge trees and then select one of the following **Merge Options**:
   - Replace existing types with types imported.
   - Do not overwrite existing types.
8. Click **Next**.

   The data stucture is displayed.
9. (Optional) To create a type, double-click a row of data.

   The Importer Wizard displays the group properties for the type.
10. Under Row Format, select **Fixed** or **Delimited**.
11. Define the group type properties and click **Next**. See ″To define group properties″ for instructions on defining group types.
12. You can continue to create types, or continue clicking **Next** until you are prompted to click **Finish**.

ASCII

If you select ASCII, the data is displayed as it is in the text file, but all non-printable characters are replaced by ASCII symbols.

When you double-click on a row in the Importer Wizard to create a type, a window opens.

From this window you can make changes to the type, as you would typically do when creating a type in the Type Designer.

UTF-8

When you select UTF-8 to describe the data character set of execution-time data, all multi-byte characters are represented by their hex values.

UNICODE Little Endian, UNICODE Big Endian, and EBCDIC

When you select UNICODE Little Endian, UNICODE Big Endian, or EBCDIC to describe the data character set of execution-time data, the data displays in their hex values.

To define group properties

Use the following procedure to define group properties from within the Text File Importer.

1. Choose the Row Format: **Fixed** or **Delimited**. The following instructions pertain to fixed format. For instructions for delimited format, see"To create delimited format."
2. Highlight the data to define.
3. Right-click on the data and select a property. For example, select **Terminator**.

   The initiators, terminators, and release characters selected are color-coded red.
4. After defining group properties, click **Next**.

To define item properties

1. To create items, highlight data in the data preview pane and drag it to the **Types** pane or double-click on the rule bar and click at the start and end of the type you want to create.
2. Continue to create types, or click **Next** and then**Finish** to complete the tree.

To create delimited format

1. Choose the **Delimited** row format option.
2. From the **Delimited Format** Importer Wizard window, select either **Custom** or **Use Delimiter** for type generation.
3. For a **Custom** delimiter, highlight the text in the data preview pane and drag it under **Types**. For the **Use Delimiter** option, click **Browse** to select a delimiter to parse the text.

   The delimiter value will appear in the **Delimiter** field.
4. Click **Finish**.
5. Continue to create types, or click **Next** and then **Finish** to create the type tree.

Chapter 13. Sink adapter

The Sink adapter is used as a temporary data destination for an output map card which then discards the mapped data.

This capability is useful when a temporary destination is needed to accept output data as part of the map execution without writing the output to a stationary destination.

Using the Sink adapter

The Sink adapter does not require any adapter commands for its operation. In addition, the Sink adapter does not access a persistent source or destination, therefore, a rollback and retry capability is not provided.

For example, use a Sink adapter instead of a file as the destination for temporary output. The advantages of using the Sink adapter include:

- faster map performance due to less file I/O
- reduced file resource conflicts when running multiple instances of the map
- reduced disk space usage

This adapter can be used with the Command Server, Launcher, Software Development Kit, or in a map rule.

For example, using the Map Designer, select **Sink** for the **Target** setting in an output card dialog box. The Sink adapter does not require any additional card settings or adapter commands.

You can use the Output Target Override execution command (`-OA`) with the Sink adapter alias. For example, to override the output card with the Sink adapter so that the data is discarded, the command string would be:

```
-OASINK1 .
```

The Sink adapter command line, for example, `-OASINK1 .`, must be followed by any dummy argument, for example, a period (`.`) or an asterisk (`*`) in order for the adapter to parse the information correctly.

Chapter 14. Application adapter

The application adapter and the APP alias is provided only for backwards compatibility for pre-6.0 application adapters.

Chapter 15. Source and target card settings

Each input of data and each output of data require content definition settings. These settings are specific to input and output cards, although some settings are common to both cards. For detailed information about source and target settings, see the Map Designer documentation.

Source settings

The Source settings identify the source of the input data. The data may be in a file, from a specific application, in a database, from a message queue, or from other possible sources. Defining the adapter source specifies where to go to get the data and specifics for the data to get.

The significance of the various source settings is specific to the adapter being used to get the input data.

Target settings

The Target settings identify the target for the data produced by the map. The data may be written to a file, inserted into or updated in a table in a database or copied into an e-mail message. The adapter target is the destination of the output data after it is built by the map.

The significance of the various target settings is specific to the adapter being used to send the output data.

Chapter 16. Communication adapters overview

Communication adapters provide connectivity to other data sources or targets using industry standards or by using protocols for moving data, for example, the FTP, HTTP, or VAN adapters.

This group of adapters also provides access to customer applications, either data objects, for example, the Java Class or CORBA adapter, or transactions such as the CICS adapter. Each adapter includes a set of commands to customize its operation.

Chapter 17. VAN access through the FTP adapter

VANs provide services beyond normal transmission, such as protocol conversion or message storing and forwarding, and are often used to transport Electronic Data Interchange (EDI) data.

In addition to asynchronous access, which is supported by the VAN adapter, many VANs now support access through FTPs.

For example, you can access the GXS VAN with the FTP adapter and send and receive secure data using a customized map. This is convenient for the following reasons:

- No special hardware is required
- Internet usability
- Secure transfer of data using FTPs

For an example of how to gain VAN access through the FTP adapter, open the **Examples** interface and navigate to **Resource Adapters** → **VAN.**

Specifying the FTP adapter for GXS VAN access

The following steps explain how to specify the **FTP** adapter while using the GXS VAN and sending example data to an EDI mailbox.

Note that this is only an example and the maps and type trees that are provided in the examples folder are example data developed for testing purposes. You must substitute your actual data when using this procedure.

In this example scenario, we will be sending a test file from a map to a test mailbox and then retrieving the file for viewing.

1. Using the Map Designer, open the **gxscert.mms** file.
2. Verify that you have the **gxs1to1.edi** file located in the GXS VAN folder.
3. Open the gxs_transmit #1 Output Card.
4. Edit the PUT > Target > Command by changing the `<LUSERID:LPASSWORD>` to the `<LUSERID:LPASSWORD>` needed to access your EDI mailbox.

   For example, change: <LUSERID:LPASSWORD> to AAA22406:A3KPP3HB as the respective User ID and Password.

   You must enter a colon (:) between the User ID and Password.
5. Build and run the **gxs_transmit map**.
6. Open the gxs_retrieve #1 Intput Card.
7. Edit the GET > Source > Command by changing the `<LUSERID:LPASSWORD>` to the `<LUSERID:LPASSWORD>` needed to access your EDI mailbox.

   For example, change: <LUSERID:LPASSWORD> to AAA22406:A3KPP3HB as the respective User ID and Password. As previously noted, there must be a colon (:) between the User ID and Password.

   We use the same User ID and Password that we used in the transmit (Output card) because the file has been sent to the same location.
8. Build and run the **gxs_retrieve map**.

9. View the results. Your EDI data should be in the **gxs_receive.out** file, which you can view using any text editor.

GXS VAN command syntax

The following command is used in the gxs_transmit #1 Output card example for accessing the GXS VAN through the FTP adapter. Although this is only an example, the command structure is the same for when you use the adapter in your work environment.

```
-TV -implicit -URL FTPS://
AAA22406:A3KPP3HB@sftp.beta.am.gxsics.com:6366/send/
sendfile;type=A
```

Chapter 18. Messaging adapters overview

Messaging-specific adapters are a collection of middleware drivers used to provide a way to specify data sources and targets for maps. Each adapter includes a set of commands used to customize its operation.

The use and operation of a specific messaging adapter is documented separately under its own name.

Message flow

Maps control the flow of messages between messaging applications in the following ways:

- One-to-one

  Messages can flow from a single data source message to a single data target message using a map.
- One-to-many

  Messages can flow from a single data source message to many data target messages using a map.
- Many-to-one

  Messages can flow from many data source messages to a single data target message using a map.
- Many-to-many

  Messages can flow from many data source messages to many data target messages using a map.

  Additionally, message flow can be sequenced between applications and used to integrate non-messaging environments.

Adapter transactions

Map definitions provide settings to control the scope of messaging adapter transactions. It is important to understand how connections to messaging systems are controlled and how this relates to transactional scope.

Transactional control

Use the **Scope** setting for a source or target in a map to determine transaction commitment. The **Scope** setting can be one of the following:

- Map
- Burst
- Card

Connection management

Connection management results in fewer connections being made to the messaging system which may improve performance. Existing connections are reused whenever possible. Two situations exist where connection sharing occurs:

- between cards and maps within a map
- between multiple maps running serially within the Launcher

In the situation in which connection sharing occurs between cards and maps within a map, the connection is described as ″active″ as long as at least one card or rule is accessing a database and has yet to be committed or rolled back (that is, there is an active transaction). When all cards or rules complete the database access, connection is ″inactive″, but still open.

Messages as a data source

The two data source processing modes are: integral mode and burst mode.

For each input card in the map, the mode is set to either **Burst** or **Integral** in the Map Designer Input Card dialog box using the **SourceRule > FetchAs** setting.

Within a map, some input cards may be in burst mode while others are in integral mode, indicating that these cards are executed only once.

Example of data processing modes

In addition to the burst mode on a per card basis, the map itself may be set to operate iteratively, thereby invoking multiple source card bursts in a single map invocation. For example, if the Quantity adapter command (`-QTY`) is specified with a setting of `50` on the input card **GET > Source > Command** and the **FetchUnit** value is `5`, the map processes the input and output cards until 50 data source objects (messages) are retrieved. The map executes 10 (50÷5) times, with each input card returning five messages concatenated together.

Messages as a data target

When any messaging adapter is specified as the data target, the messaging adapter formulates and sends a single message.

Chapter 19. Database adapters overview

This section discusses database-specific adapter information. The use and operation of these adapters is covered in each database-specific adapter's documentation.

Using the Database Interface Designer with the database adapters, data stored in a database under the control of a Relational Database Management System (RDBMS) can be easily identified, defined, and accessed as the source or target for a map.

The Database Interface Designer is used to:

- specify the databases to use for a source or target.
- define query statements.
- automatically generate type trees for queries or tables.

After defining the specifications for accessing each database in the Database Interface Designer, define your map in the Map Designer where, in map cards, you can specify a query or stored procedure as an input source or a table or a stored procedure as an output target.

The Database Interface Designer is installed as part of the Design Studio, along with all of the Windows-based database adapters. These Windows-based adapters provide connectivity to databases residing on your local PC and on other platforms so that you can automatically create type trees for those queries and tables. The adapters installed as part of your Design Studio provide a test environment for maps that access the database.

Non-Windows database adapters do not require the Database Interface Designer if you plan to only use **mtsmaker** without a database or query file (**.mdq**) to generate each type tree. For information about using **mtsmaker**, refer to ″Generating Type Trees Using mtsmaker″.

Restrictions and limitations

The Database Interface Designer and database adapters offer options and functions for accessing and manipulating data contained within a database. However, there are some restrictions and limitations for the following functions when using certain adapters:

- **Database triggers**: Using a data source as an input event trigger for the Launcher.
- **Bind facility**: Using bind values in database functions.
- **Stored srocedures**: Using stored procedures to access adapter commands and return values from stored functions.

The following table indicates the functions **not** supported by a specific adapter:

Database-specific Adapter	Triggers	Bind Facility	Stored Procedures
DB2 (z/OS)	NO	NO	NO
DB2 (z/OS ODBC)	NO		

Database-specific Adapter	Triggers	Bind Facility	Stored Procedures
DB2 (Windows/UNIX)	NO		
Informix	NO		
Microsoft SQL Server		NO	
ODBC	NO		
OLE DB	NO	NO	
Sybase SQL Server	NO	NO	

The Oracle adapters support all finctions.

Chapter 20. Using database-specific adapters

Database-specific adapter commands can be used in any of the following ways:

- When specifying adapter settings for sources and targets.
- With functions in map and component rules.
- On the command line using execution commands to override source or target settings.

See ″General Rules for Adapter Commands″ and ″Using Resource Adapters″ for more information.

Adapter usage in functions

The set of adapter commands specified in functions must begin with one double quotation mark and end with one double quotation mark. Each adapter command is separated by one space. For example:

```
DBLOOKUP ("SELECT * FROM PARTS","command1 command2...")
```

Adapters in Input Data and Output Data Settings

Use the Map Designer or Integration Flow Designer to specify a database adapter either as a source or as a target by selecting **Database** as the value for the **GET → Source** or **PUT → Target** setting, respectively. In addition to the common adapter settings such as **Scope**, **Retry**, and so forth, you can enter any of the database adapter commands for the **GET > Source > Command** or **PUT > Target > Command** settings.

Chapter 21. List of commands

The following table lists each database-specific adapter command and whether the command can be used for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function. Usage is indicated with ✓. For a `RUN` function, any adapter command can be used as indicated for a source or target.

Name	**Adapter Command Syntax**	**Source**	**Target**	**DBLOOKUP DBQUERY**	**GET**	**PUT**
Audit (-A or -AUDIT)	-AUDIT[G][+] [*full_path*]	✓	✓	✓	✓	✓
Bad Data (-BD or -BADDATA)	-BADDATA[+] *full_path*		✓			✓
Connect String (-C or -CONNECT) (Oracle only)	-CONNECT *connect_string*	✓	✓	✓	✓	✓
Commit by Card (-CC or -CCARD) *	-CCARD	✓	✓	✓	✓	✓
Commit by Statement (-CS or -CSTMT)	-CSTMT [*number*]	✓	✓	✓	✓	✓
Delete (-D or -DELETE)	-DELETE		✓			✓
Database Name (-DN or -DBNAME) *	-DBNAME *database_name*	✓	✓	✓	✓	✓
Database Adapter Type (-DT or -DBTYPE)	-DBTYPE *database_adapter*	✓	✓	✓	✓	✓
Data Source (-DS or -SOURCE) • ODBC, DB2 • Microsoft SQL Server, Sybase SQL Server	• -SOURCE *datasource* • -SOURCE *server\\database*	✓ ✓	✓ ✓	✓ ✓	✓ ✓	✓ ✓
File (-F or -FILE)	-FILE [*directory*]	✓			✓	
Global Transaction Management (-GTX) (Oracle, Oracle AQ, Microsoft SQL Server*only*)	-GTX	✓	✓	✓	✓	✓
Database/Query File (-M or -MDQ) *	-MDQ *mdq_file_name*	✓	✓	✓	✓	✓
Password (-PW or -PASSWORD)	-PASSWORD *password*	✓	✓	✓	✓	✓
Stored Procedure Name (-PR or -PROC)	-PROC *procedure_name*		✓			✓
Query (-Q or -QUERY)	-QUERY *query_name*	✓			✓	

Name	Adapter Command Syntax	Source	Target	DBLOOKUP DBQUERY	GET	PUT
Row Count (-RC or -ROWCNT) (ODBC, Oracle, DB2)	-ROWCNT *row_count*	✓		✓	✓	
SQL Statement (-S or -STMT)	-STMT *SQL_statement*	✓			✓	
Table Name (-TB or -TABLE) *	-TABLE *table_name*		✓			✓
Trace (-T or -TRACE)	-TRACE[+] [*full_path*]	✓	✓	✓	✓	✓
Trace Error (-TE or -TRACEERR)	-TRACEERR[+] [*full_path*]	✓	✓	✓	✓	✓
Trigger (-TR or -TRIG) (Oracle, Microsoft SQL Server only)	-TRIG *trigger_string*	✓				
Update (-UP or -UPDATE)	-UPDATE [OFF|ONLY]		✓			✓
User ID (-US or -USER)	-USER *user_ID*	✓	✓	✓	✓	✓
Variable (-V or -VAR)	-VAR name=*value...*	✓		✓	✓	

*A card setting exists. Using the adapter command will override any value set in the corresponding card setting. For examples, refer to Command Line Examples.

Audit (-A or -AUDIT)

Use the Audit adapter command (`-AUDIT`) to create a file that records the adapter activity for each specified input and output card. This adapter command can be specified for individual input and output cards on a card-by-card basis, or, optionally, as a global audit that encompasses all database activity for the entire map. This command can be used for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function. The default usage is to produce a file named **audit.dbl** in the directory in which the map is located. Optionally, you can append the audit information to an existing file or specify a name or full pathname for the file. For more information, refer to the Database Interface Designer documentation.

```
-AUDIT[G][+] [full_pathname]
```

Value	Description
G	Activate a global audit for the entire map by specifying this command for the first database card that will have database activity. This records all database activity for all sources, targets, and functions for all defined cards.
+	Append audit information to an existing file.
full_path	Specify the name of the audit file that can include the directory pathname.

Bad Data (-BD or -BADDATA)

Use the Bad Data adapter command (`-BADDATA`) for a target or in a PUT function so that if any inserts, updates, or procedure calls fail to execute, the data that could not be processed is written to a file you specify, and processing continues. The map completes successfully and the following adapter warning code and message are recorded in the execution section of the audit log:

```
1  One or more records could not be processed
-BADDATA[+] full_path
```

Value	Description
+	Append the information to the specified bad data file. This allows multiple cards or multiple iterations of the same map to be able to write to the same file.
full_path	Specify the name for the bad data file or the full pathname to specify the directory. By default, the directory is where the map is located. However, you must specify the file name.

If the **PUT > Target > Transaction > Warnings** setting is set to Fail in an output card, the data that could not be processed is still written to the specified file. However, the map fails.

Connect String (-C or -CONNECT)

Use the Connect String adapter command (`-CONNECT`) to specify the Oracle host connect string. This command can be used for a source or target, or in a `DBLOOKUP`, `DBQUERY`, `GET`, or `PUT` function. If an SQL*Net host connection string is specified, the connection is established through SQL*Net. If none is specified, a direct connection is established to the database identified by the ORACLE_SID environment setting.

```
-CONNECT connect_string
```

Commit by Card (-CC or -CCARD)

Use the Commit by Card adapter command (`-CCARD`) to commit or roll back the transaction when the card has been processed. This command can be used for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function. When used for a source or within DBLOOKUP, DBQUERY, or GET functions, the behavior is identical to the Commit by Statement adapter command (`-CSTMT`) because there is only a single statement or action to be executed.

Note that even if there are multiple DBLOOKUP, DBQUERY, GET and PUT map functions in a single output card, each of them will be committed immediately upon execution and not after the card completes, even if they had the `-CCARD` adapter command specified.

If specified for a database output card, a commit or rollback, based upon the **OnFailure** setting of a card, is made after the completion of the execution of the output card in which multiple INSERT, UPDATE, or stored procedure statements may have been made.

For a source, this adapter command can be used when the input is a query that executes a stored procedure to perform inserts, updates, or deletes.

```
-CCARD
```

Using this command is equivalent to selecting Card as the **PUT > Target > Transaction > Scope** value. If you specify the Commit by Card adapter command (-CCARD), it overrides the value in the **Scope** card setting.

Commit by Statement (-CS or -CSTMT)

Use the Commit by Statement adapter command (-CSTMT) to commit or roll back the transaction after the execution of every statement, or, optionally, after a specified number of statements. A statement is one of the following:

- An INSERT or UPDATE statement as generated in an output card or PUT function.
- The execution of a stored procedure.
- The statement specified in a DBLOOKUP, DBQUERY, or GET function.

In an example for a target, if an output card is a database table for which the Update adapter command (-UPDATE) has also been specified, the Commit by Statement adapter command (-CSTMT) would cause each row to be committed as it is inserted or updated into the database.

For a source, the Commit by Statement adapter command (-CSTMT) can be used when the input is a query that executes a stored procedure performing inserts, updates, or deletes.

```
-CSTMT [number]
```

Value	Description
number	Specify the number of statements to be executed before committing the data.

The Commit by Statement adapter command (-CS) is an extra call to the database and may degrade performance. However, this command may be beneficial in situations in which the number of statements is large because the database server writes all such updates to its rollback segment, which increases continually with each call.

Commit by Card (-CCARD) and Commit by Statement (-CSTMT) are mutually exclusive. Also, if -CSTMT is specified, it overrides the value in the **PUT > Target > Transaction > Scope** card setting that was compiled into the map.

Delete (-D or -DELETE)

Use the Delete adapter command (-DELETE) to delete all rows in the output table before processing the output data. Use this command for a target or in a PUT function.

```
-DELETE
```

Database Name (-DN or -DBNAME)

Use the Database Name adapter command (`-DBNAME`) to specify the name of a database defined in the **.mdq** file that is specified using the Database/Query File adapter command (`-MDQ`). The Database Name adapter command can be used for a source or target, or in a `DBLOOKUP`, `DBQUERY`, `GET`, or `PUT` function.

```
-DBNAME database_name
```

The `database_name` you supply is case-sensitive. Also, if `-DBNAME` is specified, it overrides the value in the **GET > Source > DatabaseQueryFile > Database** card setting.

Database Adapter Type (-DT or -DBTYPE)

Use the Database Adapter Type adapter command (`-DBTYPE`) to specify the database adapter type you are using. This command can be used for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function.

```
-DBTYPE {ODBC|ORACLE|SQLSVR7|SYBASE|DB2|DB2MVS|OLEDB|IFMX}
```

The adapter type must be specified if the original card is not a database and no database is specified using the Database/Query File adapter command (`-MDQ`) and the Database Name adapter command (`-DBNAME`).

Value	Description
ODBC	ODBC database adapter
ORACLE	Oracle database adapter
SQLSVR7	Microsoft SQL Server database Version 7.0 or later
SYBASE	Sybase SQL Server database
DB2	DB2 database on a Windows or UNIX platform
DB2MVS	DB2 database on an z/OS (formerly MVS) platform
OLEDB	OLE DB database adapter
IFMX	Informix database adapter

Data Source (-DS or -SOURCE)

Use the Data Source adapter command (-SOURCE) to specify the name of the ODBC, Microsoft SQL Server, Sybase SQL Server, DB2, or Informix data source. This command can be used for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function.

```
-SOURCE datasource|server\\database
```

Value	Description
datasource	Specify the ODBC, DB2 (Windows/UNIX), or Informix data source.

server\\database
Specify the Microsoft SQL Server or Sybase SQL Server data source.

File (-F or -FILE)

Use the File adapter command (`-FILE`) for database sources, or in a `GET` function to write the retrieved data to a temporary file that is read by either the Command Server or Launcher. This is useful when the amount of data retrieved by the adapter is large, possibly resulting in high memory consumption. By default, that is, without this command, data is passed between the adapter and the database server in memory. The default specification for this adapter command is for the temporary file to be created in the same directory as the map. The temporary file is then deleted after the map completes.

```
-FILE [directory]
```

Value Description

directory
Specify the directory in which the temporary file should be created.

It is recommended that you specify a value for the StreamMaxMemLimit Resource Manager setting in the **dtx.ini** file to limit the memory consumption of the process you are running, instead of using the File database adapter command (-F or -FILE).

If you are currently running processes that are using the File database adapter command (-F or -FILE) and are retrieving data without resulting in high memory consumption, you can continue to use this command.

Do not use the File database adapter command (-F or -FILE) when you are running maps in burst mode. Instead, use the StreamMaxMemLimit Resource Manager setting in the **dtx.ini** file to limit memory usage.

Global Transaction Management (-GTX)

Use the Global Transaction adapter command (`-GTX`) to indicate that the operations on a card, whether input or output, should be processed within a global transaction.

This command is available only for Oracle, Oracle AQ, and Microsoft SQL Server database adapters.

```
-GTX
```

For more information about global transaction management, refer to the Global Transaction Management documentation.

Database/Query File (-M or -MDQ)

Use the Database/Query File adapter command (`-MDQ`) to specify the name of the **.mdq** file that contains the database definition for the database to be used, as specified with the `-DBNAME` adapter command. This Database/Query File adapter command can be used for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function.

```
-MDQ mdq_file_name
```

When this adapter command is used, the **.mdq** file must be available at map execution time. Also, if the **.mdq** file is not in the same directory as the compiled map file, the full pathname is required.

If the Database/Query File adapter command (`-MDQ`) is specified, it overrides the value in the **GET > Source > DatabaseQueryFile > File** setting.

Password (-PW or -PASSWORD)

Use the Password adapter command (`-PASSWORD`) to specify the password that authenticates the user logon. This command can be used for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function.

```
-PASSWORD password
```

If you specify the parameter as `$(env_var)`where `env_var` represents the name of an environment variable, the actual value of the password is retrieved from this environment variable.

Stored Procedure Name (-PR or -PROC)

Use the Stored Procedure Name adapter command (`-PROC`) to specify the name of a stored procedure to be executed in an output of a map. Use this command either for a target or in a PUT function. If the Stored Procedure Name adapter command (`-PROC`) is specified, it overrides the value in the **DatabaseQueryFile > Table** output card setting.

```
-PROC procedure_name
```

Query (-Q or -QUERY)

Use the Query adapter command setting (`-QUERY`) for a source to specify the name of a query in an **.mdq** file as specified using the Database/Query File adapter command (`-MDQ`) and the database defined by the Database Name adapter command (`-DBNAME`).

```
-QUERY query_name
```

The name of the query (`query_name`) you supply is case-sensitive. Also, if the Query adapter command (`-QUERY`) is specified, it overrides the value in the **DatabaseQueryFile > Query** input card setting.

Row Count (-RC or -ROWCNT)

For ODBC, Oracle, DB2 for Windows, and DB2 for UNIX adapters only.

Use the Row Count adapter command (`-ROWCNT`) to specify the number of rows to be retrieved in one fetch from the database by the database adapter. This command can be used for a source or in a DBLOOKUP, DBQUERY, or GET function.

This adapter command specifies the number of rows that the adapter should retrieve from the database. It should not be confused with the **FetchAs > FetchUnit** card setting that specifies the number of rows the adapter will pass to the Command Server. For more information about the **FetchUnit** card setting, refer to the Database Interface Designer documentation.

The time to fetch is optimized in accordance with the increase in the number of rows to be retrieved. Therefore, a significant performance improvement may be gained by retrieving as many rows as possible in a single fetch. The number of rows to be retrieved is limited by the memory requirement, because the return buffer must be large enough to hold the rows that are retrieved.

```
-ROWCNT row_count
```

Without using this command, the default number of rows to be retrieved per fetch is:

- for ODBC, one row
- for DB2, one row
- For Oracle, a 64K buffer is allocated. Then, a calculation is performed to determine the number of rows that can be held in the buffer.

SQL statement (-S or -STMT)

Use the SQL Statement adapter command (`-STMT`) either for a source or in a `GET` function to specify an SQL statement or a call to a stored procedure using either the database-independent or native call syntax to be executed. This execution will provide the input data to a map.

```
-STMT SQL_statement
```

Table Name (-TB or -TABLE)

Use the Table Name adapter command (`-TABLE`) to specify the table to which or in which the map will insert or update data, respectively. Use this command either for a target or in a `PUT` function. If the Table Name adapter command is specified, it overrides the value in the **DatabaseQueryFile > Table** output card setting.

```
-TABLE table_name
```

Trace (-T or -TRACE)

Use the Trace adapter command (`-TRACE`) to produce a database trace file. The recorded Trace information details the database adapter activity. This command can be used either for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function.

To produce a trace, you must specify the Trace adapter command (`-TRACE`) or Trace Error adapter command (`-TRACEERR`). A database trace is not produced if only map trace is enabled, as was true in previous versions of the database adapters.

The default is for a database trace file to be produced in the directory in which the map is located, using the full name of the map file and a **.dbl** extension (**map_name.dbl**). Optionally, you can append the trace information to an existing database trace file, specify a file name, or specify the full pathname for the file.

```
-TRACE[+] [full_path]
```

Value	**Description**
+	Append trace information to an existing trace file.

full_path
: Create a trace file with the specified name in the specified directory. By default, the directory is where the map is located and the file name is **map_name.dbl**.

Trace Error (-TE or -TRACEERR)

Use the Trace Error adapter command (`-TRACEERR`) to produce a database trace file containing only the database errors that occurred during map execution. This command can be used either for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function.

When this adapter command is specified, a database trace file is produced in the directory in which the map is located, using the full name of the map file and a **.dbl** extension. Optionally, you can append the trace information to an existing trace file, specify a file name, or specify the full pathname for the database trace file.

The Trace Error adapter command (`-TRACEERR`) and the Trace adapter command (`-TRACE`) are mutually exclusive.

```
-TRACEERR[+] [full_path]
```

Value	Description
+	Append trace information to an existing trace file. Do not put a space between -TRACEERR and the plus [+] sign.
full_path	Create a trace file with the specified name in the specified directory. By default, the directory is where the map is located and the file name is **map_name.dbl**.

Trigger (-TR or -TRIG)

This command is available only for the Oracle and Microsoft SQL Server database adapters.

Use the Trigger adapter command (`-TRIG`) for a source to specify the trigger string for a trigger specification. For more information about specifying database triggers, refer to the Database Interface Designer documentation.

The Trigger adapter command (`-TRIG`) can be used as a trigger specification for a query that has no defined trigger in the Database Interface Designer. It can also be used to override conditions that are already defined in a trigger specification associated with a query.

The pipe character (`|`) is used to represent OR and the ampersand character (**&**) is used to represent AND. The combination of characters **&**`|` represents an AND/OR statement. The value inserted as the `when_clause` must be a valid SQL syntax with `table.column_name` enclosed in square brackets.

```
-TRIG {[R] [I tablename1 [{&||} tablename2]...]
   [D tablename1 [{&||} tablename2]...
   [U tablename1 [{&||} tablename2]...]}
   [W when_clause [[table.column_name1]]...]
```

Value Description

R This indicates that row-based triggering is being used as opposed to table-based.

I *tablename1* **[{&|}***tablename2***]...**
The insertion of rows into the specified table(s) serves as an input event trigger to the map using this query as a source.

D *tablename1* **[{&|}***tablename2***]...**
The deletion of rows from the specified table(s) serves as an input event trigger to the map using this query as a source.

U tablename1
[{&|}tablename2]...
Updating values in the specified column(s) serves as an input event trigger to the map using this query as a source.

W *when_clause* **[[***table.column_name1***]...]**
This is an expression that will be evaluated when the other events have occurred. If this expression evaluates to TRUE, the conditions of the trigger specification are met and the map is launched. If the expression is not TRUE, the state is restored as it was prior to any events occurring.

This expression may contain any SQL expressions that are valid for the target database. If database columns are referenced, the column name must be qualified with the table name and enclosed in square brackets.

The following is an example of specifying trigger events and conditions using the adapter:

```
-TR I ACCT_INFO | TRANSACTIONS D ACCT_INFO U ACCT_INFO
W [TRANSACTION.TRANS_TYPE]='PY'
```

This example causes the following to occur:

- Launch the map that uses this query as a data source when one of the following conditions is met:
  - there is insertion into the `ACCT_INFO` `TRANSACTIONS` tables
  - there is a deletion from the `ACCT_INFO` table
  - `ACCT_INFO` is updated,

but only when a row exists in `TRANSACTION` with the column `TRANS_TYPE` set to 'PY'.

You cannot combine both | and **&** in an event class (*Insert into*, *Delete from*, or *Update of*) if more than two tables or columns are being combined. For example, the following command is invalid:

```
-TR I TABLE1 & TABLE2 | TABLE3
```

Update (-UP or -UPDATE)

Use the Update adapter command (`-UPDATE`) to enable update mode. Update mode causes update operations to be performed for tables or views based upon the defined update keys and key columns to update. For information about defining updating keys and key columns to be updated, refer the Database Interface Designer documentation.

If enabled, the default is to update each row, if possible, and upon failure, to insert the data. Optionally, you can specify to insert a row or update each row, if possible, and upon failure, to not insert the data. This command can be used either for a target or in a PUT function.

If used in a PUT function, you must define your update keys and columns to update in an **.mdq** file. Then specify this **.mdq** file in the PUT function using the Database/Query File adapter command (`-MDQ`), together with the Database Name adapter command (`-DBNAME`), the Table Name adapter command (`-TABLE`), and the Update adapter command (`-UPDATE`). The **.mdq** file must be available at runtime.

```
-UPDATE [OFF|ONLY]
```

Value	**Description**
OFF	Insert a row. Insertions are allowed regardless of success or failure.
ONLY	Update each row, if possible. Upon failure, do not insert the data.

User ID (-US or -USER)

Use the User ID adapter command (`-USER`) to specify the user ID for connection to the database. This command can be used either for a source or target, or in a DBLOOKUP, DBQUERY, GET, or PUT function.

If you specify the parameter as `$(env_var)` where `env_var` represents the name of an environment variable, the actual value of the password is retrieved from this environment variable.

```
-USER username
```

Variable (-V or -VAR)

Use the Variable adapter command (`-VAR`) for a source at runtime to pass a value for a variable defined in the SQL SELECT statement. It can also be used in a DBLOOKUP, DBQUERY, or GET function.

```
-VAR name=value
```

Chapter 22. Generating Type Trees using mtsmaker

This documentation discusses using the **mtsmaker** application to generate type trees.

Overview

The **mtsmaker** application helps generate type trees for a query, table, view, or stored procedure when you do not have connectivity between the Database Interface Designer and your non-Windows database.

Use **mtsmaker** to produce type tree script files (**.mts**) on your non-Windows database host system that you will subsequently transfer to your development PC in binary mode.

On your development PC, run the Type Tree Maker using the transferred tree script file as input. The Type Tree Maker generates the type tree for you.

For information about using the Type Tree Maker, refer to the Type Tree Maker documentation.

Generating a tree script

When using **mtsmaker**, tree scripts can be created in either of two modes:
- by using a database/query file (**.mdq**)
- by not using an **.mdq** file

Control how you want the tree script file to be generated by specifying various **mtsmaker** parameters.

Using mtsmaker with a database/query file

You can use a database/query file (**.mdq**) that was created using the Database Interface Designer and transfer it, in binary mode, to the system hosting your database. The **.mdq** file is then used as a source of information about a particular query, database, or stored procedure.

To generate a tree script file (**.mts**) for a database query, specify the following **mtsmaker** parameters. For specific information about each **mtsmaker** parameter, refer to the ″List of Parameters″ .

```
mtsmaker   -M mdq_file
   -Q query_name [-B database_name]
   -F mtt_file_name
   -O mts_file_name
   [-X {F|O|T|FOT}]
```

To generate a tree script file (**.mts**) for a table, view, or stored procedure, specify the following **mtsmaker** parameters:

```
mtsmaker   -M mdq_file
   -B database_name
   -T table_name|-G stored_procedure_name
   -F mtt_file_name
   -O mts_file_name
   [-X {F|O|T|FOT}]
```

Using mtsmaker with an .mdq File

The following is an example of how to use**mtsmaker** with a database/query file to generate a tree script file for a table:

```
mtsmaker -m mdqfile.mdq -b mydbase -t mytable -f tree.mtt
   -o mtsfile.mts -x f
```

Using mtsmaker without a database/query file

Type trees can be generated without using database/query files. However, you must specify all of the necessary information so that **mtsmaker** can generate the tree scripts for queries, tables, views, or stored procedures.

To generate a tree script file (**.mts**) for a query, specify the following **mtsmaker** parameters:

```
mtsmaker   -S "sql_select_statement"
   -Z database_adapter_type
   [-U user_id]
   [-P password]
   [-D {datasource|connect_string|server\\database}]
   -O mts_file_name
   -F mtt_file_name
   -N type_name
   [-X [F][O][T]]
```

To generate a tree script file for a table, view, or stored procedure, specify the following **mtsmaker** parameters:

```
mtsmaker   -T table_name|-G stored_procedure_name
   -Z database_adapter_type
   [-U user_id]
   [-P password]
   [-D {datasource|connect_string|server\\database}]
   -O mts_file_name
   -F mtt_file_name
   [-X [F][O][T]]
```

Using mtsmaker without an .mdq File

The following is an example of how to generate a tree script for a query without using a database/query file and specifying an Oracle connect string with the Data Source parameter (`-D`).

The `-D` parameter can also be used to specify a data source for ODBC, Sybase SQL Server, or DB2, as appropriate.

```
mtsmaker -s "select * from AddressList" -u yourid -p password
   -z oracle -d "oracle connect string" -o addlist.mts
   -f maillist.mtt -n adddata
```

As part of the SQL Statement parameter (`-s`) format, the SELECT statement must be enclosed in a double quotation mark.

General parameter rules

To properly enter **mtsmaker** parameters, you must observe the correct syntax, as well as follow these general rules:

- Each parameter must begin with a hyphen, for example, `-X`.
- All parameters can be either upper or lower case, but cannot be mixed case.
- A space is required between the parameter and its value. Refer to the following example:

  ```
  -Z oracle
  ```

- The order of parameters is unimportant.
- If you do not specify all required parameters, **mtsmaker** prompts for any missing information. The displayed parameter prompts depend upon the mode in which you are operating, whether using a database/query file or not.

List of parameters

The following table lists the **mtsmaker** parameters that can be used for generating type trees, the parameter syntax, and if each one can be used (✓) with or without a database or query file. To access the list of **mtsmaker** parameters and the syntax for each parameter, enter **mtsmaker** parameters `-H` or `?` on the command line.

Parameter Name	Parameter Syntax	With .mdq	Without .mdq
"Database Name (-B)"	-B *database_name*	✓	
"Data Source (-D)" • ODBC and DB2 • Oracle • Sybase SQL Server	• -D *datasource* • -D *connect_string* • -D *server\\database*		✓ ✓ ✓
"Type Tree File (-F)"	-F *mtt_file_name*	✓	✓
"Stored Procedure Name (-G)"	-G *stored_procedure_name*	✓	✓
"Database/Query File (-M)"	-M *mdq_file*	✓	
"Category Type (-N)"	-N *type_name*		✓
"Tree Script File (-O)"	-O *mts_file_name*	✓	✓
"Password (-P)"	-P *password*		✓
"Query Name (-Q)"	-Q *query_name*	✓	
"SQL Statement (-S)"	-S *"sql_statement"*		✓
"Table Name (-T)"	-T *table_name*	✓	✓
"User ID (-U)"	-U *user_id*		✓
"Variable Name (-V)"	-V *name=value*	✓	✓
"Format (-X)" *	-X [F][O][T]	✓	✓
"Database Adapter Type (-Z)"	-Z *database_adapter_type*		✓

*

The **O** option of the Format parameter (`-X`) can be used only with the Oracle adapter.

Database Name (-B)

Specify the name of the database as defined in the **.mdq** file.

```
-B database_name
```

Data Source (-D)

Specify the name of the ODBC, DB2, Oracle, or Sybase SQL Server data source.

```
-D {datasource|connect_string|server\\database}
```

Option Description

datasource
Specify the ODBC or DB2 data source.

connect_string
Specify the Oracle host connect string.

server\\database
Specify the Sybase SQL Server data source.

Type Tree File (-F)

Specify the name of the type tree file (**.mtt**) to be generated by the Type Tree Maker. Enter the file name in the Windows file system format. For example, if a full pathname is specified, use the backslash directory separator (\).

```
-F mtt_file_name
```

Stored Procedure Name (-G)

Specify the name of the stored procedure for which you want to generate the tree script (**.mts**).

```
-G stored_procedure_name
```

Database/Query File (-M)

Specify the name of the database/query file created using the Database Interface Designer. This **.mdq** file contains the definition of either the database, specified by the `-B` parameter, or the query, specified by the `-Q` parameter.

```
-M mdq_file
```

Category Type (-N)

Specify the name of the category type from which the types defining the table or query will stem.

```
-N type_name
```

Tree Script File (-O)

Specify the name of the tree script file (**.mts**) to be generated by **mtsmaker** and used by the Type Tree Maker to create the type tree.

```
-O mts_file_name
```

Password (-P)

Specify the password used to connect to the data source. This parameter is required if it is necessary for your database security environment. Otherwise, it is optional.

```
-P password
```

Query Name (-Q)

Specify the name of the query in the **.mdq** file for which you want to generate the tree script (**.mts**).

```
-Q query_name
```

SQL statement (-S)

Specify an SQL statement or a call to a stored procedure using either the database-independent or native call syntax that is passed to the database management system to determine the content and format of the database output. This statement should be the same as the statement you are using to generate input for your map. However, you may exclude SQL WHERE clauses because they do not affect the format of the result. The SQL statement must be enclosed in double quotation marks.

```
-S "sql_statement"
```

Table Name (-T)

Specify the name of the table, or the view that can be updated, for which you want to generate the tree script file (**.mts**).

```
-T table_name
```

User ID (-U)

Specify the user ID to use to connect to the data source. This parameter may be required based upon your database security.

```
-U user_id
```

Variable Name (-V)

Specify this parameter whenever needing variable name substitution in an SQL query that is contained in an MDQ file. The equal sign (=) must be included to properly associate the substituted value with the variable name.

```
-V name=value
```

whereby:

name is the name of the variable in the SQL query.

value is the value that should be substituted.

For example:

If using the following query from an **.mdq** file, select * from BigTable where Identifier=#ID# then specify the following option on the mtsmaker command line:

```
-V ID=2
```

In this way, the query to be executed would be as follows:

```
select * from BigTable where Identifier=2
```

Note that variable name values are read by default from the **.mdq** file, unless overridden on the command line.

See the Database Interface Designer documentation for more information regarding the definition of variables in SQL statements.

Format (-X)

Specify a format for the type tree to be generated. Do not use the default, which is delimited.

```
-X [F][O][T]
```

Option	**Description**
F	Generate a fixed row format type tree.
O	(Oracle only) Generate a 1.4.*x* format type tree.
T	Generate a type tree with dates represented as text items. If this option is not specified, the default is to represent dates as date items.

Database Adapter Type (-Z)

Specify the database adapter type to connect to.

```
-Z {ODBC|ORACLE|SQLSVR7|OLEDB|SYBASE|DB2|DB2MVS|IFMX}
```

Option	**Description**
ODBC	ODBC database adapter
ORACLE	Oracle database adapter
SQLSVR7	Microsoft SQL Server database Version 7.0 or later
OLEDB	OLE DB database adapter
SYBASE	Sybase SQL Server database adapter
DB2	DB2 (Windows/UNIX) database adapter
DB2MVS	DB2 (z/OS) database adapter
IFMX	Informix database adapter

Chapter 23. Return codes and error messages

Adapter return codes and messages provide information on the adapter operations, adapter command syntax errors, and reasons for unsuccessful transactions. See the associated adapter documentation for specific error message information.

General adapter messages

The following is a list of all the codes and messages that are common among all resource adapters. Return codes with positive numbers are warning codes that indicate a successful operation. Return codes with negative numbers are error codes that indicate a failed operation.

See "Database-specific Adapter Messages" for information on database-specific return codes and error messages.

Return Code	Message
-1	Unknown error code
-2	Exception occurred
-3	Function failed
-4	Invalid property specified
-5	Property type mismatch
-6	Property not set
-7	Invalid index
-8	Null argument
-9	Invalid argument
-10	Failed to allocate memory
-11	No properties defined for the object
-12	Invalid type specified
-13	Connection failed
-14	File open failed
-15	File write failed
-16	File read failed
-17	File position function failed
-18	Invalid object specified
-19	Null object specified
-20	Illegal function call
-21	Unexpected end-of-file
-22	Only one top level element allowed
-23	Not a top level element

-24	3rd party function failed
-25	Attempt was made to resize non-resizable memory
-26	Running map failed
-27	Loading map failed
-28	Item requested does not exist
-29	Failed to load library
-30	Invalid seek performed
-31	Buffer too small - data truncated
-600	Internal error Message indicates nature of problem.

Database-specific adapter messages

After database adapter activity completes, messages display indicating the results. These messages may also be recorded in the appropriate files as specified, which may include audit logs, trace files, and execution summary. These codes and messages can be returned for the following database adapters:

- DB2 (z/OS ODBC)
- DB2 (z/OS)
- DB2 (Windows/UNIX)
- Informix
- Microsoft SQL Server
- ODBC
- OLE DB
- Oracle
- SQL/MP
- SQL/MX
- Sybase SQL Server

The following is a listing of all the codes and messages that can be returned as a result of using the database adapters for sources or targets.

You can also troubleshoot database-specific adapters by using the adapter Trace (-T or -TRACE) or the Trace Error (-TE or -TRACEERR) command.

Adapter return codes with positive numbers are warning codes that indicate a successful operation. Adapter return codes with negative numbers are error codes that indicate a failed operation.

Return Code	**Message**
0	Success
12	Database -b argument is required with arguments mdqfile -m and query -q
1001	One or more records could not be processed
1002	No data found
-1001	Memory allocation error

-1002	Failed to allocate ODBC environment
-1003	Failed to allocate ODBC statement
-1004	Failed to allocate ODBC connection
-1005	Failed to connect to the database
-1006	Failed to prepare the SQL statement
-1007	Failed to obtain definition of column
-1008	Failed to bind parameters to the SQL statement
-1009	Failed to execute the SQL statement
-1010	Failed to open a cursor
-1011	Failed to insert a row into the database
-1012	Failed to read file
-1013	Failed to open file
-1014	The input buffer was of incorrect form
-1015	Failed to write to file
-1016	Database type is unknown
-1017	Failed to parse SQL statement
-1018	Failed to define host variables
-1019	Failed to logoff the database
-1020	Failed to close the cursor
-1021	Failed to bind parameters to the SQL statement
-1022	Failed to fetch a long data value
-1023	Failed to perform a file seek
-1024	Failed to create file
-1025	The specified query could not be found
-1026	The specified database could not be found
-1027	Failed to load DLL
-1028	Failed to get address of function
-1029	Failed to fetch a row of data
-1030	Failed to get data from the database
-1031	Failed to get data from the database
-1032	No rows were updated
-1033	The number of rows affected was undeterminable
-1034	The supplied statement was syntactically invalid
-1035	Failed to create an execution thread
-1036	Failed to get extended error text
-1037	Table is not accessible
-1038	Failed to disconnect from the database

-1039	Failed to free the database connection
-1040	Failed to free the environment
-1041	MDQ file is invalid
-1042	Failed to close file
-1043	No database connection could be found
-1044	No database type was specified with -DBTYPE
-1045	Failed to find column attributes
-1046	Failed to get the number of result columns
-1047	Failed to get large data
-1048	Failed to cleanup the database connection
-1049	Databases are not supported
-1050	Failed to load DLL
-1051	Failed to get address of function
-1052	Failed to commit or rollback the transaction
-1053	Failed to free the statement
-1054	Insufficient data was provided
-1055	Failed to bind parameters to the SQL statement
-1056	Failed to define large data value
-1057	Failed to put large data value
-1058	Failed to commit the transaction
-1059	Failed to rollback the transaction
-1060	The update statement was invalid
-1061	No type could be found to match column name
-1062	No suitable presentation could be found for a column
-1063	No suitable presentation could be found for a column
-1064	No suitable presentation could be found for a column
-1065	Internal error
-1066	No more rows were returned
-1067	No trigger was defined for the input
-1068	Data was unexpectedly terminated
-1069	Required command line options missing
-1070	The database command line was invalid
-1071	Could not find environment variable
-1072	Failed to attach to database
-1073	Failed to convert datatype to native database type
-1074	Failed to get list of SQL servers
-1075	Failed to get list of databases

-1076	Failed to initialize stored procedure
-1077	Failed to set parameter to stored procedure
-1078	Failed to execute stored procedure
-1079	Failed to get column information
-1080	Failed to declare host variables
-1081	Command line token could not be found
-1082	Failed to create an event
-1083	The command line is too long
-1084	The function/procedure has no return value
-1085	A variable length binary column is not permitted
-1086	Error occurred writing to a LOB column
-1087	Error occurred reading from a LOB column
-1088	Connection to Open Server failed
-1089	Poll to Open Server failed
-1090	The map contains outdated DB parameters. Rebuild the map
-1091	An output date was of an incorrect format
-1092	Internal error occurred
-1093	Burst complete
-1094	Type already exists and override was not specified
-1095	No Row or ProcedureCall group can be found
-1096	Failed to format output from an object type
-1097	Failed to create an object
-1098	An object is not permitted in fixed type tree
-1099	Failed to convert database parameters from ASCII to EBCDIC
-1100	One or more columns/parameters are of an unsupported datatype
-1101	Failed to configure run-time environment.
-1102	Failed to Allocate context structure, unknown reason or not enough memory
-1103	A trigger could not be defined for a watch event
-1104	Unique table columns are required for row-based watch events
-1105	Procedure is not accessible
-1106	The configured transaction manager is not supported
-1107	System error occurred while waiting for a watch event
-1108	Triggering is not supported for this adapter.
-1109	User terminated database access
-1110	The trace file name is too long, or the running directory is too deep The database trace file name and/or the directory name in which the failed map is located, may be longer than 128 characters.

Ensure the file and directory name are shorter than 128 characters.

-1302	A trigger event could not be registered
-2002	Insufficient data passed to output card

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