IBM WebSphere Transformation Extender

Launcher

Version 8.1

Note

Before using this information, be sure to read the general information in "Notices" on page 135.

October 2006

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

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

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. Introduction

The Launcher automates the execution of systems of maps and can control multiple systems.

On Windows platforms, the Launcher runs as a multi-threaded service. On UNIX platforms, the Launcher runs as a multi-threaded daemon. For information about the platform and version-specific information, see the Release Notes on the product Web site (www.ibm.com/software/integration/wtx).

Note: When running the Launcher on z/OS through UNIX Systems Services (USS), follow the instructions for UNIX. The z/OS environment is treated as another UNIX system. Any differences between the standard UNIX and z/OS usages are noted where appropriate.

Installation considerations

For information about the installation details, see the Release Notes on the product Web site (www.ibm.com/software/integration/wtx).

Note: Ensure that the hostname 'localhost' resolves to 127.0.0.1 or Launcher execution (**launcher.sh**) will produce unexpected results.

Note: *<install_dir>* refers to the directory where your product is installed.

Requirements for HP-UX

For the Launcher running on an HP-UX platform, the recommended settings for **max_thread_proc** and **masdsiz** are as follows:

For the running on an HP-UX platform, the recommended settings for **max_thread_proc** and **masdsiz** are as follows:

Kernel Parameters	Default	Recommended
max_thread_proc	64	512
maxdsiz	64 MBytes	<user dependent>

The **max_thread_proc** default of 64 processes is typically not enough and should be increased to 512. For **maxdsiz**, the value is dependent on the amount of available memory and the size of the data used in the maps being run. The default of 64 MBytes is typically not enough. With large data sizes and if memory permits, 512 MBytes is recommended for **maxdsiz**.

Optional local z/OS administration interfaces

There are optional local z/OS administration interfaces available for configuring and managing the Launcher on z/OS operating systems.

The two optional local z/OS administration interfaces ("Local z/OS Administration Interfaces") are:

- ISPF Panel Interface (″ISPF Panel Interface″)
- z/OS Operator Console Interface (″z/OS Operator Console Interface″)

  Note: See ″z/OS Launcher″ for more information.

How the Launcher works

The Launcher runs systems of maps that are created and generated using the Integration Flow Designer (IFD).

These systems of maps that are generated specifically to run in the Launcher are called system files (**.msl**), sometimes referred to as Launcher control files.

When the Launcher starts running, it is initialized with **.msl** files in the deployment directory.

You can specify a deployment directory from the Launcher Administration interface. (For more information about this, see ″Launcher Administration″.)

Note: When using the z/OS local administration interfaces, the deployment directory defaults to ***install_dir*/systems**.

Using the Launcher

To use the Launcher, you must complete a series of steps.

The list below provides a high-level overview of the steps to follow after installing WebSphere Transformation Extender.

Note: When running the Java Launcher under UNIX Systems Services (USS), by default, the Java Launcher uses the value calculated by subtracting 1 from the starting listening port to stop, resume or pause the Java Launcher. The starting listening port is configured under the **General** tab in the Launcher Administration interface.

Note: When using the z/OS local administration interfaces, see ″z/OS Launcher″ for more information.

Basic steps

The following steps summarize the process flow of using the Launcher with the Management Tools.

1. From the LauncherAdministration interface, configure the Launcher by setting up users, port numbers, and deployment directories.
2. Use the Integration Flow Designer to create a system and then generate the system (.msl) file to the deployment directory. For help on creating or generating a system file, see the *Integration Flow Designer*documentation.
3. Start the Launcher service or daemon.
4. Open a connection to the Launcher from the Management Console and begin viewing statistical data from the process that is running. From there you can also control the compound system that is running.
5. Open a connection to the Launcher from the Launcher Monitor to view watches that are running dynamically and to take snapshots.
6. Open the Snapshot Viewer to view snapshots taken by the Monitor.

Chapter 2. Launcher architecture

The Launcher is an event-driven software model for linking and executing maps. Systems contain maps with the system workflow you specify by using the Integration Flow Designer.

A Launcher runs on a server platform and manages one or more Launcher processes (comprised of one or more systems) concurrently. Systems are asynchronous multi-threaded processes, whose atomic units are transactional maps. Systems can execute in a distributed environment where a Launcher resides on each server. When events occur, the Launcher is notified and starts maps based on that notification. Distributed data is received or routed during map execution.

Launcher functionality is based on the following premises:

- An event manager controls the initiation of maps based on sources to which a map subscribes.
- A resource manager synchronizes shared data and timing interfaces among heterogeneous sources and targets.
- Maps publish data that result from the content transformation of source data.
- Maps are transactional processes with each map having its own error detection and recovery procedures.
- The Launcher runs as a single process with multiple threads.

Triggers: Asynchronous communication

The Launcher starts maps based on *triggers* that you identify using the IFD. Triggers are asynchronous events and time or source state changes. In some cases, there might be multiple triggers that must collectively exist before a map is initiated. The Launcher manages the coordination of these events to ensure that the correct set of circumstances have occurred before a map is initiated.

Although each instance of a map is considered a separate *watch*, the Launcher must also manage the resource interfaces used by other maps and other instances of the same map to properly synchronize these asynchronous events.

Synchronous coordination

Situations exist where synchronous interaction between maps is required. For example, you might not want to complete a transaction until you receive an acknowledgment of a particular action. Or you might want to communicate with a program or function to receive input data or send output data. Synchronous interaction can be performed in different ways:

- Communicate with other maps from a map rule during the data transformation process. Using a map rule in this manner with the Launcher manages coordination among maps to assure deadlock situations are avoided and shared resources are reliably accessed or deleted
- Synchronize coordination to interact with other applications using a resource adapter for a data source or target. For example, if you send a message by using the e-mail adapter, receipt notification might occur before the mapping process completes. When using resource adapters, you can configure recovery procedures for errors that occur during communication. For example, you use

the Value Added Network (VAN) source adapter to get electronic data interchange (EDI) trading partner data from your VAN. If the line goes down during transmission, you can specify retry sessions during the mapping process to recover from such an error.

System execution consistency

System execution consistency implies data resources get to the right place at the right time with the right priority and that resources are manageable so that data or timing interfaces do not interfere with each other.

The smallest unit in a system configuration is a map. The definition of a map requires you to identify logical source and target data objects and rules for transforming source content to target content. For example, a logical data object can be an order, a ship notice, a news report, or a point-of-sale transaction. In this case, content transformation rules determine how an order and inventory schedule produces a ship notice.

If the transformation is consistent, the result is predictable. If the transformation is consistent, the same rules can execute on a variety of computing platforms; different sources and targets can be applied to the same logical data. For example, orders might be in a distributed database, inventory schedules might appear as messages, and ship notices might need to be contained in files.

Maps communicate when data can flow among them. To visualize that transaction data flow, the architecture of map interaction is defined as a system using the IFD. From a graphical diagram of system components, you can generate consistent control information to distribute maps and subsystems across different servers and execute all *atomic* maps in a system as an integrated whole with an Launcher.

Map function and requirements

The primary function of a map is to transform and route data from source formats to target formats. Sources and targets are independent objects, with respect to the metadata that specifies the content, the applications to which they might link, and the transport mechanisms used to obtain or deliver the content.

Defining a system

Using the IFD, you can define a system as a set of distributed subsystems, a set of maps, or a combination of maps and subsystems. You can create these definitions because a system is an isolated set of components whose interfaces are as explicit as the interfaces for an atomic map.

For example, this isolation enables the Launcher to manage a system as easily as it manages a map.

To complete successfully, a map needs the proper resources to be in place. For example:

- Every data source must be accessible.
- Every output must be routable to its assigned target.
- The server-required resources to perform the transformation (work files and audit logs, for example) must not be in use by any other process during the transformation process.

To accomplish this reliably, the transformation must be able to succeed or fail as a whole. That is, when a map does not complete successfully, the original data must persist in a form that is recoverable and partial results can be rolled back and made inaccessible to other maps or applications.

You can use a single map to simplify logical connections between multiple heterogeneous applications; direct links using application adapters; and indirect links using file, messaging, and database adapters.

Chapter 3. Getting started

In this documentation, you will learn how to set up and start using the Launcher.

Creating systems

To create Launcher processes (**.msl** system files) that run on your Launcher platform, you must use the Integration Flow Designer (IFD).

Additionally, you can use the IFD to:

- Define systems and Launcher settings.
- Build maps for a target platform, if required.
- Move all files that are required for system execution (that is, ported map files, Launcher system files, and so on) to a target platform in a designated deployment directory. For example, generate the system file (**.msl**) to the deployment directory.

See *Integration Flow Designer* documentation for more information about systems.

Note: When using the z/OS native administration interfaces, see ″z/OS Launcher″ for more information. The remainder of this topic does not apply.

Launcher Administration

The Launcher Administration is the interface to the Launcher from which you can do the following configuration tasks:

- Choose the deployment directories from which systems run.
- Set up users and user access rights.
- Select a listening port and port range.
- Select a resource configuration file (**.mrc**).
- Specify system properties for the Java Remote Method Invocation (RMI)

Note: For more information about the Launcher Administration, see the help for the Management Console application. From the toolbar, navigate to **Help** → **Help Topics** and click Launcher **Administration** from the navigation pane.

Accessing Launcher Administration

The Launcher Administration is included with the WebSphere Transformation Extender package and runs on both Windows and UNIX platforms.

To start the Launcher Administration (Windows)

Select Launcher **Administration** from the **Start** → **Programs** menu under **IBM** WebSphere Transformation Extender *n.n* (where *n.n* represents the version number) → **Launcher** .

To start the Launcher Administration application (UNIX)

1. From the installation directory, type:

```
. setup
```

2. At the command prompt type:

```
launcheradmin.sh.
```

Using Launcher Administration

Before starting the Launcher, it is recommended that you confirm the default settings. If you do not change settings in the Launcher Administration before starting the Launcher, the Launcher Administration uses the following default values:

- Automatic startup option is enabled.
- Listening ports are 5015 and 5016.
- Port range is 7000 - 8000.
- There are no user profiles.
- The deployment directory is:
  - ***install_dir*\Systems** on Windows platforms
  - ***install_dir*/Systems** on UNIX platforms

At the minimum, you must set up at least one user profile before using the Launcher. The documentation describes all of the options.

Note: When you configure the Launcher to run with the multiple-processes feature enabled, you can edit some of the settings while it is running as described in ″Multiple Processes.″
When you configure the Launcher to run as a single Launcher process (Compound System), you cannot edit settings.

Automatic startup option

In the Launcher Administration, the **Automatic startup** option is enabled by default. The **Automatic startup** option triggers the **.msl** system files located in the deployment directory to start running automatically upon starting the Launcher service or daemon.

Note: One or more system files appear in the Management Console as one ″CompoundSystem″. See ″Multiple Processes″ for additional information when using the multiple-processes feature.

If you disable the **Automatic Startup** option and then start the Launcher service, you must manually start the system from the Management Console. See ″Management Console″ for more information about the Management Console.

Connection ports

A **Listening Port** is a network port that listens for incoming requests. The default listening ports are **5015** and **5016**. The second listening port is reserved for communicating with the Launcher Monitor and is displayed in a read-only field. The Launcher Administration application uses an increment of 1 to determine the second listening port number. The system-generated number for this port is based on the **Listening Port** value plus **1**. For example, if you change the listening port to **3050**, the second listening port number automatically changes to **3051**.

Note: When running the Java Launcher on UNIX platforms and using the stop, pause or restart command with the **launcher.sh** shell script command, the

Java Launcher internally uses a port setting that is 1 less than the first **Listening Port** value. The default internal **Port** is **5014**, which is **1** less than the first **Listening Port**, **5015**.

The **Port Range** values determine the port range that is used for systems running in the Launcher. There are *three* ports used for each Launcher running. The default range is **7000-8000**. You can change the port range to any value except the **Listening Port** number plus 1 (for example, **5016**), which is reserved for communication with the Launcher monitor.

Note: When using WebSphere Transformation Extender for z/OS, the specified **Listening Port** and **Port Range** ports should be added to the **TCPIP.ETC.SERVICES** file. These ports cannot be currently in use by another application. The purpose of specifying them in the file is to document that these ports are reserved and should not be used by another application. It does not prevent other applications from using them.
This is true for all platforms, although the services file name is different. For more details, see the IBM *z/OS Communications Server: IP Configuration Reference* documentation.

Note: See "Multiple Processes" for additional information about connection ports when using the multiple-processes feature.

Note: When running the Java Launcher on UNIX and LINUX platforms, the specified **Listening Port** and **Port Range** ports should be added to the **etc\services** file.

User profiles

Before starting the Launcher, you must create at least one user profile for Launcher authentication.

To add a user profile:

1. From the **Access** tab, click **Add**.

   The User Properties window opens.
2. Complete each field:
   - **User Name** - the user's name (up to 32 alphanumeric characters).
   - **Login Name** - the user's login ID for the network (up to 9 alphanumeric characters).
   - **Password** - the password for the **Login Name** (up to 12 alphanumeric characters).

     By default, each action category (**Start/Stop**; **Pause/Resume**; **Monitor**) is set to **Revoke**, which means that this user does not have any access rights.

     Note: The access rights that you grant to a user apply to all systems running in the Launcher.
3. Under each action category (**Start/Stop**; **Pause/Resume**; **Monitor**), click **Revoke** to view a drop-down menu that contains the two options: **Grant** or **Revoke**.
4. After you select the security level for each category, click **OK** to save changes.

   The new user appears in the profiles list on the **Access** tab.

   Note: You must set up users with the appropriate access rights in the Launcher Administration interface to view systems running in the Management Console.

About Access Permission

By default, the Launcher process has Local System account privileges on Windows platforms because it is run as a service. However, Local System account privileges are insufficient for performing tasks associated with mapped network drives.

If you have an issue with access permission on mapped network drives, the solution is to configure the Launcher service to log on using a specific account.

To configure user account information:

1. Open the Windows **Computer Management** console.
2. Locate **Services** and double-click **WebSphere Transformation Extender Launcher**.

   The Properties window is displayed, from which you can make account changes.

Deployment directories

The deployment directories are where the **.msl** files are placed to run in the Launcher. The default deployment directory (**Systems**) is located in the **WebSphere Transformation Extender** installation directory; however, you can modify the default path or add new deployment directories as needed.

Note: You can create several deployment directories, however all systems will run simultaneously. The statistics that are displayed in the Management Console ("Management Console") will include all systems running.

To add a deployment directory:

1. From the **Deployment Directories** tab, click **Add**.

   The Select window opens.
2. Navigate to the deployment directory.
3. Click once on the directory and click **Select**.

   The new directory is displayed in the directory list.

To remove a deployment directory:
From the **Deployment Directories** tab, select a path from the directory list and click **Remove**.

The path is removed from the directory list.

Resource resolution

In the Launcher Administration, you can choose a resource configuration file (**.mrc**) that contains specifications for the Launcher. For example, the file can contain information such as an active virtual server and its associated resource name file (**.mrn**).

For more information about resource configuration files, see the *Resource Registry* documentation.

Cluster support

The Launcher service uses the Java Remote Method Invocation (RMI) to communicate with clients. When the Launcher service is running in a cluster environment or if there are machines with multiple network interface cards (NIC), you must manually specify the RMI system properties. You can do so from the Launcher Administration application **Advanced** tab.

The Launcher service sets these properties before running the RMI server. In a cluster environment, you must enter `java.rmi.server.hostname` in the **Property** column and the Launcher service host name in the **Value** column.

For example, **java.rmi.server.hostname** can be the **Property**. The host name **mycluster.fl.company.com** (under **Value**) can be the ″value″ of the property, which indicates where the Launcher service is running.

Firewall configuration

To use a firewall, settings in both the Launcher Administration and Management Console must be configured. Configuration for assigning port numbers for the firewall must be enabled for both.

By default, the Launcher opens anonymous ports for 2-way communication with the Management Console. If there is a firewall in place, fixed port numbers are required. To assign specific port numbers for the Launcher, you must enable the **Enable Configuration** option and either accept the default port numbers or enter specific ones.

To assign fixed port numbers in the Launcher Administration for firewall use:

1. From the Launcher Administration **Options** tab, select the **Firewall** option.

   Firewall configuration options appear.
2. Enable the **Enable Configuration** option.

   The **Server Port** and **Client Port** fields become active.
3. Enter specific port numbers in the **Server Port** and **Client Port** fields, or accept the default values (**4634** and **4635**).

   Firewall Option

   Value

   Enable Configuration

   This option is turned off by default.

   Select the check box to configure firewall ports.

   When this option is enabled, the **Server Port** and **Client Port** fields become active.

   Server Port

   Enter a server port number or accept the default value of **4634**.

   Client Port

   Enter the client port number or accept the default value of **4635**.
4. Click **OK** to save the changes.

   Note: If you are using a firewall, ensure that the Enable Configuration firewall option is also enabled in the Management Console.

To assign fixed port numbers in the Management Console for firewall use:

1. From the Management Console, select **Tools** → **Options**.

   The Options window opens.
2. From the **Options** list, select **Firewall**.

   The override configuration options appear.

3. Enable the **Enable Configuration** option.

   The **Client Port** field becomes active.
4. Enter the client port number in the **Client Port** field, or accept the default value (**7777**).

   Firewall Option
   Value

   Enable Configuration
   This option is turned off by default.

   Select the check box to configure firewall ports.

   When this option is enabled, the **Client Port** field becomes active.

   Client Port
   Enter the client port number or accept the default value of **7777**.
5. Click **OK** to save the changes.

Launcher Options

In the **Options** list of the Options window, when you select **Launchers**, the configuration options described below are displayed. These options allow you to manually configure and override connection information for each Launcher connection in the Management Console.

Override configuration:

You can manually configure and override connection information for each Launcher connection in the Management Console. Typically, the Management Console relays control commands (start, stop, pause, resume, disconnect) to a Java-based Launcher service, which in turn executes the commands to the process running. However, you can manually override the typical process so that the Management Console bypasses the Java-based service and communicates directly with the process running. In this case, the Management Console can relay only limited control commands (stop, pause, resume) to the process running. The Launcher itself must be manually started from the CPU where it resides.

The following list describes the communication process that occurs when you use the override configuration option. The port numbers that are referenced are the default numbers.

- Control information including stop, pause, and resume commands, are communicated between the Launcher process that is running and the Management Console over port 7002.
- Status information is communicated between the Launcher process that is running and the Management Console over port 7000.
- Monitoring information is communicated between the Launcher process that is running and the Launcher Monitor over port 7001.

To access override configuration options:

1. In the Management Console, select **Tools** → **Options**.

   The Options window opens.
2. From the **Options** list, select **Launchers**.

   The override configuration options appear.

About multiple Launchers

For all supported Windows or UNIX platforms, if there are multiple Launchers running on the same machine, they must run on different listening ports and have different port ranges. If this condition is violated, the Launcher might not work as intended.

Preparing to execute defined systems

After you generate **.msl** files to the deployment directory, prepare the Launcher to execute the maps defined in those systems by performing one of the following actions:

- If the Launcher service or daemon is not currently started, start the Launcher.
- If the Launcher service or daemon is running, stop and then restart the Launcher.

To run a Launcher process (comprised of one or more systems) using the Launcher:

- The Launcher system files must be located in the designated deployment directory.
- All maps and source event gateways specified for system components defined within the system must be available.

Starting and stopping the Launcher

The following procedure describes the control functionality for running services on Windows platforms.

For UNIX Platforms, commands are listed in ″UNIX Commands″ .

To start the Launcher service (Windows)

1. From your desktop, right-click **My Computer** and select **Manage**.
2. From the Computer Management window, expand the **Services and Applications** list and select **Services**.
3. In the right pane of the **Computer Management** window, right-click **WebSphere Transformation Extender Launcher** and select **Properties**.

   The Properties window opens.
4. Click **Start**.

To stop or restart the Launcher service (Windows)

From the **Computer Management** window, right-click **WebSphere Transformation Extender Launcher** and select **Stop** or **Restart** from the context menu.

UNIX Commands

From the UNIX command line, you can use shell script commands to control the Launcher.

To start, stop, pause, or resume the Launcher (UNIX)

1. From the command prompt, enter the following command:

   ```
   . setup
   ```
2. From the same command prompt, enter one of the following commands:

Option	Command
Start	launcher.sh -start <*listening_port*>
Stop	launcher.sh -stop <*listening_port*>
Pause	launcher.sh -pause <*listening_port*> <*timeout*>
Resume	launcher.sh -resume <*listening_port*> <*timeout*>

Based on the following command example, if the Launcher pauses before 60 seconds, **TRUE** or **SUCCESS** is returned. Otherwise **launcher.sh** exits, returning **FALSE**.

```
launcher.sh -pause 3000 60
```

For the codes that are returned from using these UNIX commands, see "Return Codes for the Status Option".

Environmental debug:

To enable the environmental debug option for the UNIX Launcher, you must set the DTX_DEBUG variable to **TRUE** before starting the Launcher. Use the following command to turn on environmental debug:

```
sh: DTX_DEBUG=TRUE
    export DTX_DEBUG
```

Listening port:

Pertaining to the Start command, the <*listening_port*> number is the port on which the Launcher will start. For the remaining commands (Stop, Pause, and Resume), the <*listening_port*> is the port on which the Launcher is running. The default port number is 5015.

Timeout:

The <*timeout*> value is the amount of time in seconds that the command line is blocked if the Launcher has not paused or resumed.

A timeout value of zero indicates that the command line is to be blocked for an infinite amount of time until the Launcher pauses or resumes. The default timeout value is zero. The default timeout value for the Stop command is 2 minutes.

If the pause or resume command is invoked and the Launcher is not running, an error message is displayed at the command prompt and the launcher.sh command returns a false or non-zero value. For the successful completion of the Pause or Resume commands, the value returned by **launcher.sh** is **TRUE** or zero (**0**). If a failure occurs, a **FALSE** or a non-zero value is returned.

Modifying systems while the Launcher is running

You can make changes to a Launcher system file (**.msl**) while the Launcher is running, however, you can only see the changes if the system is restarted from the Management Console. To add a new system, both the Launcher service or daemon and the Management Console need to be restarted.

The procedures in each of the following scenarios assume that a system is currently running in the Management Console.

To modify an .msl file

If the changes are only to the contents of one particular **.msl** file that is already part of the **CompoundSystem** for a Launcher process (see ″Multiple Processes″ for additional information when using the multiple-processes feature), and the name of the system file is the same, then you must do the following actions:

1. Stop the **CompoundSystem** from the Management Console.
2. Overwrite the existing file in the deployment directory with the updated **.msl** file.
3. Restart the system from the Management Console.

To add or delete files

1. Stop the **CompoundSystem** from the Management Console.
2. Disconnect the Management Console from the relevant Launcher.
3. Stop the Launcher service.
4. After the service has stopped, make additions or deletions as necessary.
5. Restart the Launcher service.
6. In the Management Console, reconnect to the Launcher and restart the **CompoundSystem**.

Chapter 4. Configuration parameters

The **dtx.ini** file is the configuration file for **WebSphere Transformation Extender** and contains specific settings for the Launcher. The **dtx.ini** file is created in the root directory during the installation of **WebSphere Transformation Extender**. You can use a text editor to edit option values as needed.

Launcher settings in dtx.ini

The following sections of the **dtx.ini** file contain default startup settings for the Launcher:

- M4File (″M4File″)
- Launcher (″Launcher″)
- Connections Manager (″Connections Manager″)

Enabling or disabling settings

If you have an unmodified **dtx.ini** file, the default value of the settings is listed on the commented-out line. To change the setting, first uncomment the line and then change the value.

A commented-out line is indicated by a semicolon (**;**) in front of the line. Uncommenting a line without changing the value does not change anything. Therefore, the following two lines are equal:

```
KeepDB=0
;KeepDB=0
```

M4File

In the **dtx.ini** file, the following settings pertain to the file listener in the Launcher - listed under the heading **M4File**:

- Journaling (″Journaling″)
- Logging (″Logging″)
- FileListenCount (″FileListenCount″)
- ListenSleepTime (″ListenSleepTime″)

Journaling

See the *Cluster Support on Sun Solaris* or the *Cluster Support for Windows* documentation for information about this setting.

Logging

Use the **Logging** setting to enable trace (logging) functionality for the file listener. See ″Troubleshooting Tip″ for details.

FileListenCount

This setting is the number of files to process at a time for each directory (0 meaning all). The default value for this setting is 25 (**FileListenCount=25**). For trigger uniformity, decrease this value as more directory watches are added.

ListenSleepTime

This setting is the time in milliseconds between directory processing. The default value is 50 (**ListenSleepTime=50**). If the CPU level is high when Launcher is idle, you should increase this value.

Launcher

In the **dtx.ini** file, the following Launcher settings are listed under **Launcher**:

- **Logging Categories** (″Logging Categories″)
- **ResourceCfgFile** (″ResourceCfgFile″)
- **LoadPDH** (″LoadPDH″)
- **MaxThreads** (″MaxThreads″)
- **TriggerTime** (″TriggerTime″)
- **LauncherLog** (″LauncherLog″)
- **InitPendingHigh** (″InitPendingHigh″)
- **InitPendingLow** (″InitPendingLow″)
- **WatchMaxThreads** (″WatchMaxThreads″)
- **DisableMaxThreads** (″DisableMaxThreads″)
- **InitPendingIdleMS** (″InitPendingIdleMS″)
- **RunMapCacheMaxNum** (″RunMapCacheMaxNum″)

Each setting is described here.

Logging categories descriptions

Logging Category	Value	Description
LogTrace	0	Disabled (default)
	1	Enables the logging of sequence of events Examples: Before Resource Pending <*map*> After Resource Pending <*map*>
LogDebug	0	Disabled (default)
	1	Enables the logging of information to diagnose Launcher behavior Examples: New audit file name <*name*> Adapter wildcard.<*wildcard*>
LogInfo	0	Disabled (default)

Logging Category	Value	Description
	1	Enables the logging of general information about the state of the Launcher and maps Examples: :Map started InitPendingHigh reached
`LogWarning`	`0`	Disabled (default)
	1	Enables the logging of issues that are not severe, but might be important to know Examples: Resource Manager table too small - using default Map warning <*map*>
`LogError`	`0`	Disabled (default)
	1	Enables the logging of critical errors that won't cause the Launcher to stop Examples: Map failure <*map*> Socket error <*socket_type*>
`LogFatal`	`0`	Disabled
	1	Enables the logging of a fatal error that caused the Launcher to stop Examples: Corrupted MMC file during initialization <*mmc_name*> Adapter listener Initialization failure <*adapter*>

If you turn on logging ("Logging Option") from the Management Console or the **dtx.ini** file, the resulting files are created in the **logs** subdirectory of the **WebSphere Transformation Extender** installation directory.

ResourceCfgFile

A default resource configuration file for the Launcher is specified in the **Launcher** section of the **dtx.ini** file:

```
ResourceCfgFile=resource.mrc
```

You can specify a resource configuration file for the Launcher by modifying the default file name (**resource.mrc**) to the resource configuration file that you want to use.

For more information about the resource configuration file, see the *Resource Registry* documentation.

LoadPDH

This setting disables the central processing unit (CPU) and memory counters (Windows only).

If the setting is commented out (indicated by a semicolon at the beginning of the line), or uncommented and set to **1**, the CPU and memory counters are loaded.

If you uncomment the line and set the value to **0**, the function is disabled. Setting LoadPDH to **0** is generally not necessary; however, some adapters might require the disabling of counters.

```
LoadPDH={0|1}
```

Option	Description
0	Disables CPU and memory counter loading.
1	Enables CPU and memory counter loading.

MaxThreads

The **MaxThreads** value is used to limit the number of map threads for all watches in an environment, provided that no other settings have conflicting values. If the following settings are not set to `0`, then the **MaxThreads** value will be impacted:

- **WatchMaxThreads**
- **DisableMaxThreads**
- **Max Concurrent Map Instances** (a setting in the IFD).

  ```
  MaxThreads=n
  ```

  Option Description

  n Maximum number of maps (threads) that the Launcher can execute concurrently. Range is `0` to `32,000`.

  Default value is `20`.

  Note: For performance reasons, on the z/OS operating system, *n* should be less than or equal to `400`.

TriggerTime

Use the **TriggerTime** setting to specify file time granularity in seconds.

```
TriggerTime=n
```

Option Description

n Minimum trigger time in seconds for files. This is the minimum time that must elapse in a file's timestamp before it can cause a new event.

Default value is `1`.

Note: The **TriggerTime** setting can be used for files that reside in the Hierarchical File System (HFS) on the z/OS operating system; it is not available for use with native data sets.

LauncherLog

In the **[Launcher]** section of the **dtx.ini** file, you can specify the settings for a Launcher log file. See ″Trace and Log Files Naming Conventions″ for the specific naming conventions.

The **LauncherLog** file is not created by default. To ensure that a log file is created, you must remove the semicolon from the beginning of the designated line of text in the **dtx.ini** file. For example, the following line entry has had the semicolon removed and will create a file that contains error, warning, startup, and configuration information, and will be created in the product installation directory by default.

```
LauncherLog=ewsc
```

The default syntax is:

```
LauncherLog=[E][W][S][C]
```

Each option is described in the following table.

Option	Description
E	Logs map **e**rrors
W	Logs map **w**arnings
S	Logs Launcher **s**tartup statistics
C	Logs **c**onfiguration information

Regardless of the options you choose to log, the debug log (**.txt**) file (see "Trace and Log Files Naming Conventions" for the specific naming conventions) will contain header information that is similar to the following example:

```
***         WebSphere Transformation Extender Launcher for Windows
Version:               n.n(999)
Copyright:               2005 IBM Corporation
Started:               Thu Nov 21 10:53:05 2005
Computer:               MYCOMPUTER
Logging:               Configuration, Start Up, Errors, Warnings, Exit
Note:               This file views correctly under fixed font editors.
```

Configuration option

If you specify **c** for configuration, the following example displays how each watch is configured:

```
*** CONFIGURATION:

Time:                  Thu Nov 21 10:53:05 2002
------------------------------------------------------------------
Watch:                  1
MSL File:                  C:\mapdev.nn\mercntsv\systems\single-mq-to-file.msl
System:                  single-mq-to-file
Component:                  fromqueuetofile
Map:                  C:\testing\n.n\15010\fromquetofile.mmc

Priority:                  Normal
Map Delay:                  <none>
Pending Exp.:                  <none>
Retries:                  10 every 5 seconds

Paging:                  8 X 64K

Trace:                  Summary

Validation:                  Ignore: <none>
                  Stop On First Error: Yes

Work Area:                  Default

Init High:                  100
Init Low:                  50

Time Trigger:                  <none>
```

```
Input 1 (T):                  IBM WebSphereMQ: -QMN QM1 -QN QA.QUEUE5
                     -MID * -QTY 1
                  Rollback: Yes, Delete: No, Reuse Work File: No
                  Retries: <none>

Output 1:                  File: C:\testing\n.n\15010\output*.txt
                  Rollback: Yes, Delete: If empty, Append: No
                  Retries: <none>

Audit File:                  Burst Data                     Never
                  Burst Execution                     Always
                  Summary Execution                     Always
                  Data Settings                     Never
                  Map Settings                     Never
```

Startup option

If you specify **s** for startup, you will see an entry on whether the startup of the Launcher was successful. If successful, something similar to the following entry is displayed:

```
*** STARTUP:
Status:               Successful
Time:               Thu Nov 21 10:53:11 2002
```

If not successful, it might pinpoint where the startup problem is:

```
*** STARTUP:
Status:               Failure
Time:               Thu Nov 21 18:49:58 2002
Errors:               No systems to run
```

Error option

If you specify **e**, the file lists all maps that have an error status after being run:

```
   *** ERROR:

   System:                  single-mq-to-file
   Component:                  fromqueuetofile
   Map:                  C:\testing\n.n\15010\fromquetofile.mmc
   Start Time:                  Thu Nov 21 10:54:06 2002
   Instance:                  1
   Run Time:                  0.1957 seconds
   Return Code:                  8 - One or more inputs was invalid

   Input 1:                  IBM WebSphereMQ: -QMN QM1 -QN QA.QUEUE5
                     -MID AMQ0 -QTY 1
   Output 1:                  File: C:\testing\n.n\15010\outputAMQ.txt
   Audit File:                  C:\testing\n.n\15010\fileAMQ.log
```

Warning option

If you specify **w**, the file lists all maps that have a warning status after being run.

```
   System:                  single-mq-to-file
   Component:                  fromqueuetofile
   Map:                  C:\testing\n.n\15010\fromquetofile.mmc
   Start Time:                  Thu Nov 21 11:23:54 2002
   Instance:                  1
   Run Time:                  0.0923 seconds
   Return Code:                  21 - Input valid but unknown data found

   Input 1:                  IBM WebSphereMQ: -QMN QM1 -QN QA.QUEUE5
                     -MID AMQ1 -QTY 1
```

```
Output 1:                File: C:\testing\n.n\15010\outputAMQ.txt

Audit File:                 C:\testing\n.n\15010\fileAMQ.log
```

InitPendingHigh

Use `InitPendingHigh` to pause the listener threads if too many initialized pendings occur.

`InitPendingHigh=`*`n`*

Option Description

n Any integer value; modification not recommended.

Default setting of `0` specifies that an unlimited number of `InitPendings` might occur. For example, if you change this value to **10**, the listener thread pauses when the limit of `10` `InitPendings` reaches **10**.

InitPendingLow

Use `InitPendingLow` to resume the listener threads if paused from `InitPendingHigh`.

`InitPendingLow=`*`n`*

Option Description

n Any integer value; modification not recommended.

Default setting is `0`. The listener thread resumes when the number of occurring `InitPendings` is equal to this setting. For example, if `InitPendingHigh` is set to **10** and `InitPendingLow` is set to **5**, the listener thread pauses processing when the number of occurring `InitPendings` reaches **10** and resumes processing when the number reaches **5**.

WatchMaxThreads

The `WatchMaxThreads` option is also known as Global Max Threads per Watch.

`WatchMaxThreads=`*`n`*

Option Description

n Global limit on the number of concurrent mappings for each (*any* individual) watch.

This number cannot exceed the `MaxThreads` value, which is the total number of concurrent mappings overall.

Range is `0` to `32,000.`

Default setting is **0**.

Note: The `Max Concurrent Map Instances` setting in the IFD overrides the `MaxThreads`, `WatchMaxThreads`, and `DisableMaxThreads` Launcher settings.

DisableMaxThreads

Use the `DisableMaxThreads` setting to disable the `MaxThreads` option.

`DisableMaxThreads={0|1}`

Option Description

0 Setting is off (default)

1 Setting is on.

If set to 1, the MaxThreads value is disabled and the WatchMaxThreads value is used to place a global limit on the number of concurrent map executions for each watch.

Note: The Max Concurrent Map Instances setting in the IFD overrides the MaxThreads, WatchMaxThreads, and DisableMaxThreads Launcher settings.

InitPendingIdleMS

This entry controls how often the Launcher checks to see if initialization-pending maps can be executed. This reduces waiting time, and improves performance on most platforms.

The default value is **1**, however, higher values might help throughput when many initialization pending maps occur and the Launcher is CPU bound.

```
InitPendingIdleMS=n
```

Option Description

n Time (in milliseconds) that initiation pending threads should remain idle between mapping attempts.

Valid values are 1-10000 (one millisecond - 10 seconds).

RunMapCacheMaxNum

Use the RunMapCacheMaxNum setting to define the maximum number of RUN maps to be cached in memory.

To disable RUN map caching, set the value to -1.

```
RunMapCacheMaxNum=n
```

Option Description

n Maximum number of RUN maps to be cached in memory.

Default value is 300; the minimum value is 10.

Resource Manager

The **dtx.ini** file contains the default Launcher startup setting values for the Resource Manager. The Resource Manager synchronizes shared data and timing interfaces among heterogeneous sources and targets. You can edit the values as needed using a text editor. The Resource Manager settings are:

- **Logging Categories** (″Logging Categories″)
- **Disable** (″Disable″)
- **MaxResourcePending** (″MaxResourcePending″)
- **StackSize** (″StackSize″)
- **TransManager** (″TransManager″)
- **StreamMaxMemLimit** (″StreamMaxMemLimit″)

Logging categories

The logging categories provide a variety of debugging information for the Resource Manager. The Launcher uses them to determine which severity levels and components to debug.

For descriptions and examples of the logging categories, see ″Logging Category Descriptions″ .

Disable

Use this option to enable or disable resource management. If the Resource Manager is disabled, the file access protection scheme is deactivated. This allows you to have concurrent maps writing to the same file at the same time. If you choose to disable the Resource Manager and your system relies on the file access protection, data can become corrupt. However, if you do not need resource management, disabling this option boosts performance.

```
Disable=n
```

Option Description

0 Enables resource management.

This is the default setting.

1 Disables resource management.

MaxResourcePending

MaxResourcePending is used to specify the maximum number of resource pendings to allow.

The minimum value is 0, and the maximum is *MaxThreads + 1*. When ResourcePending is equal to MaxResourcePending, any maps that should go into ResourcePending return an error instead. Setting this to *MaxThreads - 1* (the default) provides a minimum amount of deadlock protection. If all threads are in a ResourcePending state, nothing will run. You can set MaxResourcePending to *MaxThreads - n* to ensure that *n* threads are always available to run maps.

```
MaxResourcePending=n
```

Option Description

n Number of resource pendings.

Default setting is 80.

Map threads in a resource pending state are not available to execute maps.

StackSize

Use the **StackSize** setting to specify the stack size for each thread execution. This setting is applicable only to adapters for the HP-UX 11.*x* and AIX platforms.

When using SAP R/3 on HP-UX platforms, or when using XML maps on HP-UX or AIX platforms that call multilevel RUN maps, the recommended **StackSize** value is 400.

```
StackSize = n
```

Option Description

n Maximum stack size in kilobytes (KB) for the process running.

The limits of this setting are specific to your operating system. Consult your operating system documentation for more information.

Note: The default value for pthread stack size might not be sufficient, which could result in random core dumps caused by stack corruption. Setting this value too high significantly increases the memory usage of the Launcher process and can negatively impact performance. Because the stack size is a critical operating system setting, take extreme caution when changing this value, if required.

This setting is disabled only when commented-out. Using a numerical value, such as zero, to disable the setting does not work and will cause errors.

TransManager

See the *Global Transaction Management* documentation.

StreamMaxMemLimit

When the data that is passed to or from an adapter exceeds the size specified by Stream Maximum Memory Limit value, a file is created in a temporary directory for paged data. This limits the memory consumption of the process.

```
StreamMaxMemLimit=n
```

Option Description

n Data size in megabytes.

If the size of data is exceeded, the data is paged to a temporary file.

Default value is `500`.

```
StreamMaxMemDirectory=/tmp
```

By default, the temporary directory is designated by the operating system.

For Windows this is typically what the environment variable **TEMP** is set to (such as **C:\TEMP**); for UNIX, this is typically **/tmp**.

Note: Ensure that the path points to a directory with sufficient file space.

Connections Manager

The **dtx.ini** file contains the default Launcher startup setting values for the Connections Manager. You can edit the values as needed using a text editor. The default value for all of these settings is `0`, meaning disabled or off.

How the Connections Manager works

The Connections Manager works in conjunction with adapters to maintain connections. To reduce overall run time and increase efficiency, the Connections Manager keeps a pool of connections to reduce the total number of connect or disconnect operations that have to be done. This means that a connection will not be disconnected when a map is complete, but sometime after that.

When a connection is needed, the Connections Manager searches its pool until it finds a connection that the adapter finds suitable (using the adapter's **CompareConnections** function). A suitable connection means that it must be connected to the appropriate resource, such as a database, message queue, and so forth.

If a suitable connection is found, Connections Manager then tries to verify that the connection is still valid. This is done by calling **ValidateConnection**. If the adapter does not implement **ValidateConnection**, the test will fail.

Connections Manager settings

The Connections Manager settings include:

- **Logging Categories (**″Logging Categories″ **)**
- **IgnoreGPFs** (″IgnoreGPFs″)
- **ShareConnectionThreads** (″ShareConnectionThreads″)
- **Idle** (″Idle″)
- **Soft Limit (SLim)** (″Soft Limit (SLim)″)
- **Hard Limit (HLim)** (″Hard Limit (HLim)″)
- **Min** (″Min″)
- **Keep** (″Keep″)
- **HSleep** (″HSleep″)
- **HlimTimeout** (″HLimTimeout″)
- **PollWaitTimeMin**(″PollWaitTimeMin″)
- **PollWaitTimeMax**(″PollWaitTimeMax″)

Each setting is described here.

Logging categories

The logging categories provide a variety of debugging information for the Connections Manager. The Launcher uses them to determine which severity levels and components to debug.

For descriptions and examples of the logging categories, see ″Logging Category Descriptions″ .

IgnoreGPFs

Use this setting to ignore system exceptions during processing so that only the map fails as opposed to the system.

```
IgnoreGPFs={0|1}
```

Option Description

0 Disables the option, which means the system will fail upon encountering an exception.

1 Enables the option so that exceptions are ignored on connection threads and therefore a map failure occurs instead of a system failure.

This is the default setting.

ShareConnectionThreads

The **ShareConnectionThreads** setting allows multiple connections on the same thread. There are three levels of sharing to specify the criteria used for sharing threads. Each value is described in the following table. This setting is disabled by default.

```
ShareConnectionThreads=n
```

Option Description

0 Disables the sharing option, which means each connection gets its own thread.

1 Enables sharing for GTX only. Only connections with GTX will get combined to the same thread.

2 Enables sharing for same adapter type. It provides the same functionality as 1, plus connections that are for the same adapter type can be combined to the same thread. For example, all FTP on one thread, all e-mail on another thread, all GTX on a third, and so forth.

3 Enables sharing for all adapters. It provides the same functionality as 2, except the adapter type is ignored. All connections can be on the same thread.

Idle

Use this setting to specify global default settings for all adapters or specify settings for individual adapters.

The Idle setting creates a background map thread with the purpose of periodically searching the connection pools, locating connections with idle time greater than the value specified by Idle, and then removing them. This allows connections to be disconnected even when no map is running. It is important to note that Resource Manager will shutdown the Idle thread gracefully, which means that you might potentially need to wait additional Idle seconds until it restarts and exits. This is particularly noticeable when you are using the command engine.

- To specify default settings for all adapters, use `Idle=`*n*
- To specify settings for individual adapters, use `Idle`*xxx*`=`*n*

Option	Description	
0	Disables the use of idle settings for adapters (default).	
n	Number of seconds to allow a connection to be idle before automatically disconnecting.	
xxx	Adapter type where *xxx* represents the command line alias for the adapter. Example values for typical adapters include:	
	`DB`	Database Adapters
	`MQS`	IBM WebSphere MQ (server) Adapters
	`MQSC`	IBM WebSphere MQ (client) Adapters
	`ALE`	ALE Adapters
	FTP	FTP Adapters
	For example, `IdleMQS=30` sets the connection idle limit for IBM WebSphere MQ adapters to 30 seconds.	

See the *Resource Adapters Online Library* for adapter documentation.

Soft Limit (SLim)

Most servers place a limit on the number of clients that can be connected. WebSphere Transformation Extender generally shares a server with other clients, and therefore should not be able to use all available connections. The Resource

Manager provides two limit parameters: SLim (soft limit) and HLim (hard limit). Either or both of these parameters can be used to specify a maximum number of connections. Existing connections are only removed when a new one is required.

The SLim setting is a soft (advisory) limit. This number can be exceeded at peak load, but SLim tries to limit the number of idle connections at all other times. Limits are only checked when opening a connection, so it is possible to have a greater number of idle connections.

The global SLim value is by default set to **4** and is used if any other limit for the number of connections is not set.

- To specify global default settings for all adapters, use `Slim=`*n*
- To specify settings for individual adapters, use `SLim`*XXX*`=`*n*

Option	Description	
n	Number of idle connections. Default is `4`.	
xxx	Adapter type where *xxx* represents the command line alias for the adapter. Example values for typical adapters include:	
	DB	Database Adapters
	MQS	IBM WebSphere MQ (server) Adapters
	MQSC	IBM WebSphere MQ (client) Adapters
	ALE	ALE Adapters
	FTP	FTP Adapters
	For example, `SLimMQS=30` sets the soft limit for IBM WebSphere MQ adapters to 30 idle connections.	

See the *Resource Adapters Library* for adapter documentation.

Hard Limit (HLim)

The HLim setting is a hard (mandatory) limit. This number will not be exceeded. Connections are placed in a pending state rather than exceed this number.

- To specify global default settings for all adapters, use: `Hlim=`*n*
- To specify settings for individual adapters, use: `HLim`*XXX*`=`*n*

Option	Description	
n	Maximum number of connections. Default is `0` (unlimited).	
xxx	Adapter type where *xxx* represents the command line alias for the adapter. Example values for typical adapters include:	
	DB	Database Adapters
	MQS	IBM WebSphere MQ (server) Adapters
	MQSC	IBM WebSphere MQ (client) Adapters
	ALE	ALE Adapters

Option	Description	
	FTP	FTP Adapters
	For example, `HLimMQS=30` sets the hard limit for IBM WebSphere MQ adapters to 30 connections.	

Min

The Min setting specifies a minimum number of connections to keep. When a limit (SLim or HLim) is exceeded, idle connections will be removed until no more than this value remains (if possible).

- To specify default settings for all adapters, use `Min=`*n*
- To specify settings for individual adapters, use `Min`*XXX*=*n*

Option	Description	
n	Minimum number of connections to keep. The default setting is `0`, which means there is no limit set.	
xxx	Adapter type where *xxx* represents the command line alias for the adapter. Example values for typical adapters include:	
	DB	Database Adapters
	MQS	IBM WebSphere MQ (server) Adapters
	MQSC	IBM WebSphere MQ (client) Adapters
	ALE	ALE Adapters
	FTP	FTP Adapters

Keep

Use this setting to specify the time, in seconds, an idle connection is expected to remain valid. Connections that idle longer than this are tested prior to being used. If the test fails or is not supported, the connection is not reused.

The Keep setting is an efficiency measure and represents some amount of time to assume that the connection is valid without a test. If the idle time of the connection (amount of time since it was last used) is less than the Keep value, then the Resource Manager does not attempt to verify connection validity.

Last used refers to the timestamp when the connection was last assigned to a map. Whether it was used successfully or not is not considered because it is assumed that an unsuccessful attempt is a temporary condition, not a permanent failure.

This setting should only be used for adapters that implement Validate Connection.

- To specify global default settings for all adapters, use: `Keep=`*n*
- To specify settings for individual adapters, use: `Keep`*xxx*=*n*

Option	Description
n	Specify how many seconds an idle connection is expected to remain valid. Default is **`0`**, which disables connection testing. A value of `-1` will force a test every time, but this is not recommended.

Option	Description	
xxx	Adapter type where *xxx* represents the command line alias for the adapter. Example values for typical adapters include:	
	DB	Database Adapters
	MQS	IBM WebSphere MQ (server) Adapters
	MQSC	IBM WebSphere MQ (client) Adapters
	ALE	ALE Adapters
	FTP	FTP Adapters
	For example, `KeepMQS=30` sets the Keep period for IBM WebSphere MQ adapters to wait 30 seconds for idle connections.	

Note: As an example, if you are executing maps using an IBM WebSphere MQ (server) adapter at least once every 10 seconds, a setting of `KeepMQS=10` results in the connection not being tested because they are used before they expire.

See the *Resource Adapters Library* for adapter documentation.

HSleep

The HSleep setting is the specified polling interval used to check resource availability. The value is the cycle time (in seconds) for checking connection availability. The default value is `2` and if specified should be as follows: *0<HSleep<HLimTimeout*.

Note: No validation of this value is performed. Unexpected values for this setting can cause unpredictable results.

```
HSleep=n
```

Option Description

n Number of cycle time seconds for checking connection availability.

Default value is `2` seconds.

Note: The number of retries is the HLimTimeout value divided by the HSleep value.

HLimTimeout

The HLimTimeout setting specifies a maximum amount of time (in seconds) that a map waits for a new connection to become available before failing the map. If not specified, HLimTimeout defaults to infinite.

```
HLimTimeout=n
```

Option Description

n Number of seconds that a map waits for a new connection to become available before failing the map.

Default value is infinite.

PollWaitTimeMin

The PollWaitTimeMax and PollWaitTimeMin parameters enable you to configure a polling interval that is used by adapters to detect events. If the interval is not specified, the default value is used.

Note: If this value is too small, such as below 50 milliseconds, CPU usage can go up during the idle time when the Launcher is waiting for an event.

```
PollWaitTimeMin=n
```

Option Description

n Minimal polling interval in milliseconds.

Default setting is `2000`.

PollWaitTimeMax

The PollWaitTimeMax and PollWaitTimeMin parameters enable you to configure a polling interval that is used by adapters to detect events. If the interval is not specified, the default value is used.

```
PollWaitTimeMax=n
```

Option Description

n Maximal polling interval in milliseconds.

Default setting is `10000`.

Configuring Java-based adapters

From the **dtx.ini** file, you can add **.jar** files to the CLASSPATH variable. To do this, you must add an **External Jar Files** section to **dtx.ini** and list the applicable jar files, similar to the following example:

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

For more information, see the *Resource Adapters* documentation.

Trace and log files naming conventions

The Launcher trace and log files are located in the logs directory under the install_dir, which refers to the directory where your product is installed.

The Launcher trace and log files are located in the logs directory under the install_dir, which refers to the directory where your product is installed. The naming convention for the Launcher trace files, previously named launcher.txt, is the following:

Trace File Naming Convention	Description
CompoundSystem `<date><time>.txt`	For all systems in a single Launcher process

Trace File Naming Convention	Description
`<unique_name><date><time>.txt`	For one or more systems run in separate Launcher processes. `<unique_name>` is the name configured in the Launcher Administration.

The naming convention for the Launcher log files, previously named mercntsv.log on Windows platforms and debug.log on UNIX platforms, is shown in the following table:

Log File Naming Convention	Description
CompoundSystem `<date><time>.log`	For all systems in a single Launcher process
`<unique_name><date><time>.log`	For one or more systems run in separate Launcher processes. `<unique_name>` is the name configured in the Launcher Administration.

The name and location of these log files within the initialization file are not used. The content of the log files from the resource manager and connections manager, previously named dtxrmgr.log, are written to the Launcher log for each process.

Result codes in log files

The Launcher log files contain result codes from processes run by the Launcher. These codes are returned from the operating system and do not have the same meaning as the codes returned after issuing command line options.

Chapter 5. Launcher settings

This documentation describes the settings that are configured in the IFD for maps to be run by the Launcher. These map settings, when configured from the IFD, are called *Launcher settings*.

Map settings are originally set during map creation in the Map Designer. When you change the map settings using the IFD, you are overriding the map settings. However, the overrides from the IFD only affect the map during the generation of Launcher system files (**.msl**)-the settings in the original map source file are not affected.

See the *Integration Flow Designer* documentation for information about how to configure all settings.

Note: The UNIX operating systems are case sensitive. Systems created using the IFD to be executed by a UNIX Launcher must observe the proper case for path names, map names, and so on.

Accessing Launcher settings

Systems defined in the IFD contain source map components, compiled map components, and Launcher settings configured to control map execution. Systems can include options such as time and source event triggers to initiate maps and source and target settings for inputs and outputs.

The following procedure assumes that execution mode for the open system is **Launcher**.

To view Launcher settings:

From the IFD, do one of the following actions to view Launcher settings for a map component:

- Right-click on a map component and select **Edit** Launcher **Settings**.
- Click the execution settings button at the bottom of the map component icon.

  The Launcher Settings window appears.

Launcher settings window

The LauncherSettings window shows a collapsed tree view of the settings that can be specified for a map component in a system that has the Launcher mode. To see an example of the LauncherSettings window, go to httpdst.msd system in the Launcher Agent example installed in the examples folder.

The following settings provide specific information needed by the Launcher:

- **MapServerLocation** (″MapServerLocation″)
- **MapSettings** → **MapAudit** (″MapSettings> MapAudit″)
- **MapSettings** → **Workspace** (″MapSettings> WorkSpace″)
- **MapDelay** (″MapDelay″)

- **PendingExpiration** (″PendingExpiration″)
- **Priority** (″Priority″)
- **Max Concurrent Map Instances** (″Max Concurrent Map Instances″)
- **TimeEvent** (″TimeEvent″)
- **Pending Instance Thresholds** (″Pending Instance Thresholds″)

MapServerLocation

The MapServerLocation setting is used to identify the path of the executable map on the server. The default value is the path of the selected map.

Note: When using the z/OS Launcher, see ″z/OS Launcher″ for more information about specifying the location of the compiled map file as a native file system object.

To modify the MapServerLocation setting

To change the default MapServerLocation setting, do one of the following actions in the MapServerLocation column next to MapServerLocation:

- Enter the path and filename location of the compiled map file associated with the map component in the system window.
- Click the browse button and navigate to the file.

MapSettings > MapAudit

You can use the **MapAudit** settings to create and store map audit logs. The audit logs contain information about data audit, execution audit, map settings, and data settings.

MapAudit settings are compiled (as overrides) into the **.msl** file, which is run by the Launcher.

To enable the audit log

From the Launcher Settings window, under **MapSettings** → **MapAudit** → **Switch**, choose **ON**. (the httpdst.msd system installed in the Launcher Agent example folder.)

The audit log is created in a file in the map directory or to a location that you specify using the **AuditLocation** settings.

When you choose **Custom**for **Directory** under the **AuditLocation** setting, enter a file path in **Value**. The value you enter in **FileName**determines the audit log file name.

There are three options from which to choose for the audit log filename (**AuditLocation** → **FileName**): **Default**, **Unique**, and **Custom**. The following table displays the filename outcome of each option.

Value **Description**

Default
The file is automatically assigned an audit log file name (based on the compiled map name) with a **.log** file name extension.

Unique

Generates a unique audit log name in the following format:

Mer_<*mapname*>_<*process_key*>_<*map_counter*>.log

Run_<*mapname*>_<*process_key*>_<*map_counter*>.log

The prefix ″Mer″ is used with top-level maps while ″Run″ is used for run maps.

<*mapname*> is the executable map name.

<*process_key*> is a unique, for each process value.

<*map_counter*> is a unique, for each map instance value (within the same process).

Custom

A custom filename. Enter the name in the **FileName Value** field.

Note: Wildcards can be used as long as an input trigger resolves them.

Note: For detailed information about map settings, see the *Map Designer* documentation.

Note: If you are using the native file system to store a map on a z/OS environment, see the documentation about the three data file location options DEFAULT (″DEFAULT″), CUSTOM (″CUSTOM″), and UNIQUE (″UNIQUE″), for the format of the audit log filename generated in the HFS.

Unique names for audit log files

If the log files produced for one map trigger another map, use **AuditLocation** → **FileName** → **Unique** to select unique names for the log files. Each log file will then trigger another instance of the other map.

For the **AuditLocation** → **FileName** setting, the following option results apply:

- The **Unique** option produces a file in the following format:

  Mer_<*mapname*>_<*process_key*>_<*map_counter*>.log
- The **MapName** option produces a log file in the following format:

  ***map_name*.log**

If you use this **MapName** option, the general **Retry** option is used to determine when one map instance will not interfere with another. If you use multiple maps that generate unique log file names and you want to use log files to trigger different maps, be sure to specify different directories for those log files.

MapSettings > WorkSpace

When a map runs, the software creates or reuses a workspace that it uses during the mapping process to find data that it needs to transform.

The **Unique** value of the **WorkFilePrefix** setting enables you to select globally-unique work file names for a map. If you want multiple instances of the same map to run at the same time, use this option. If you do not use the unique work files option, multiple instances of the same map will run sequentially.

If you do not use unique work files, the general retry setting is used (**MapSettings** → **MapRetry** → **Switch = ON**) when one map instance interferes with another.

Note: Selecting the **Memory** option for the **AuditLocation** setting also allows multiple instances of the same map executing simultaneously.

Unique filename format for work files:

Mer_<*mapname*>_<*process_key*>_<*map_counter*>.[I|O]*card_number*

MapDelay

The **MapDelay** setting defines the amount of time in milliseconds that the Launcher waits, after all triggers are satisfied, before running the map. The default value for the **MapDelay** setting is **0**.

Value	Description
n	Time, in milliseconds, to wait before starting the map. Valid entries are integers `0` to `9999`.

PendingExpiration

The **PendingExpiration** setting specifies the amount of time that a map can stay in an initiation pending state before expiring (running without all of its input events). The default value for the **PendingExpiration** setting is **Hour**.

Setting Value	Description
Hour	Measures the value of the **PendingExpiration** period in hours
Minute	Measures the value of the **PendingExpiration** period in minutes
Second	Measures the value of the **PendingExpiration** period in seconds

The **PendingExpiration** → **Value** setting specifies how many units (in hours, minutes, or seconds) to use for measuring the waiting period. Enter the number of units in the **Value** field. The default value for the **Value** setting is **0**.

If the map associated with a given watch has multiple triggers and at least one trigger occurs, the map is placed in an *initiation pending* state until all triggers occur. Maps in an initiation pending state are reevaluated each time another trigger occurs.

The initiation pending state occurs when any of the following conditions are true:

- The map is defined as single-threaded and one occurrence of the map is already running.
- The map requires an event that has not yet occurred. (This is the only initiation pending condition in which **PendingExpiration** can occur.)
- The maximum number of concurrent maps are already running.

If all preceding conditions are eliminated, the initiation pending state can be removed from the map; if the **PendingExpiration** period has not elapsed, the map can run.

Priority

Do not use. This setting is reserved for future development.

Max concurrent map instances

Specifies the maximum number of concurrent mappings for each watch. In the IFD, this setting overrides the `MaxThreads`, `WatchMaxThreads`, and `DisableMaxThreads` Launcher settings.

TimeEvent

The **TimeEvent** settings initiate (or trigger) map components of a system based on time. Each map component is run based on a time event, a source event, or a combination of multiple events. If a map component is not assigned an event trigger, the Launcher will not initiate that map to run.

Note: When you analyze a system on the IFD and your system is set to Launcher mode, a warning is registered for any source or compiled map component that has no associated event triggers.

TimeEvent > Switch

The **Switch** setting enables or disables **TimeEvent** options. The default value for the **Switch** setting is **OFF**.

Value	Description
OFF	Disables **TimeEvent** options
ON	Enables **TimeEvent** options

TimeEvent > BeginAfter

The **BeginAfter** settings specify the period of time in hours, minutes, or seconds to wait (from the time the Launcher starts) before enabling the time trigger.

BeginAfter > Hour

The **Hour** setting is used to specify how many hours to wait to begin the map. The default value for the **Hour** setting is **0**.

Value	Description
n	Enter a number to represent the time in hours to wait to begin the map. Valid entries are integers `0` to `23`.

BeginAfter > Minute

The **Minute** setting is used to specify how many minutes to wait to begin the map. The default value for the **Minute** setting is **0**.

Value	Description
n	Enter a number to represent the time in minutes to wait to begin the map. Valid entries are integers `0` to `59`.

BeginAfter4Second

The **Second** setting is used to specify how many seconds to wait to begin the map. The default value for the **Second** setting is **0**.

Value	Description
n	Enter a number to represent the time in seconds to wait to begin the map. Valid entries are integers `0` to `59`.

TimeEvent > Trigger

The **Trigger** setting is used to define the type of time event. You can trigger a map based on a singular time event or cycle, according to a specified time increment. The default value for the **Trigger** setting is **Once**.

Value	Description
Once	The trigger is a non-repeating, one-time-only event.
Every	The trigger occurs once for every interval defined using the **Interval** settings.

Trigger > Interval

Use the **Interval** setting to describe the recurring period to use for the trigger. The **Interval** setting is available only if the value of the **Trigger** setting is **Every**. The default value for the **Interval** setting is **1 Day(s)**.

Interval > At

Use the **At** settings to set a specific time for the trigger to start after the start of the interval defined in **Interval** settings. The **At** settings are only available if the value of the **Trigger** setting is **Every**. The time periods specified in the **Hour**, **Minute**, and **Second** settings describe a time based on a 24-hour clock set to system time.

At > Hour

The **Hour** setting specifies the hour in which to run the map. The default value for the **Hour** setting is **0**.

Value	Description
n	Enter a number representing the time in hours in which to run the map. Valid entries are integers `0` to `23`.

At > Minute

The **Minute** setting specifies the minute in which to run the map. The default value for the **Minute** setting is **0**.

Value	Description
n	Enter a number representing the time in minutes in which to run the map. Valid entries are integers `0` to `59`.

At > Second

The **Second** setting specifies the second in which to run the map. The default value for the **Second** setting is **0**.

Value	Description
n	Enter a number representing the time in seconds in which to run the map. Valid entries are integers `0` to `59`.

Interval > OmitDays

Use the **OmitDays** setting to set the days to omit.

OmitDays > Monday

The **Monday** setting is used to omit Mondays from the time trigger schedule. The default value for the **Monday** setting is **No**.

Value	Description
No	Do not omit Mondays.
Yes	Omit Mondays.

OmitDays > Tuesday through Sunday

These settings work the same way as the **Monday** setting.

Interval > OmitHours

Use the **OmitHours** setting to set the hours to omit.

OmitHours > 0

The **0** setting is used to omit the 0 hour (midnight to 1:00 am). The default value for the **0** setting is **No**.

Value	Description
No	Do not omit this hour.
Yes	Omit this hour.

OmitHours > 1 through 23

These settings work the same way as the **0** setting (above).

TimeEvent > SkipIfBusy

The **SkipIfBusy** setting is used to ignore the time trigger if it occurs when a watch instance is already in process. The default value for the **SkipIfBusy** setting is **Yes**.

Value	Description
Yes	Ignore time trigger if a watch is in process.
No	Do not ignore time trigger if a watch is in process; initiate another watch instance.

TimeEvent > EventCoordination

The **EventCoordination** setting is used to control the coordination of time and input events. The **EventCoordination** setting is available only if the map has both a time trigger and one or more input triggers: the **SourceEvent** setting for one of the input cards is **ON**. The source trigger might have wildcard values. The default value for the **EventCoordination** setting is **All or None**.

Value Description

All or None
If no source trigger exists when the time trigger occurs: this time trigger interval is skipped and nothing is run.

If one source trigger exists when the time trigger occurs: the map is run.

When the time trigger occurs, if multiple source events exist that satisfy a wildcard: multiple instances of the map are run for all event sets that exist.

FirstAvailable
If no source trigger exists when the time trigger occurs: then the map goes into pending state and waits for the first source trigger to appear, then the map runs.

If one source trigger exists when the time trigger occurs: the map is run.

When the time trigger occurs, if multiple source events exist that satisfy a wildcard: the map is run with the first known source event.

Pending instance thresholds

The **Pending Instance Thresholds** setting enables a listener to pause or resume for a specific watch. This setting allows high and low initialization-pending maps for any given watch to turn listeners on or off for that particular watch only.

The use of this setting can improve performance by enabling watches to run that could not run previously because of other watches monopolizing system resources.

If two watches share the same event (what an adapter is triggering), the two watches must have the same values for high and low pending instance thresholds. Otherwise, an error occurs, preventing the Launcher from starting.

During initialization, properties of the watch, including high and low parameters, are written to the debug trace (**.txt**) file (see ″Trace and Log Files Naming Conventions″ for the specific naming conventions).

Note: The global **InitPendingHigh** and **InitPendingLow** settings in **dtx.ini** regulate all listeners and events. The **Pending Instance Thresholds** Launcher setting regulates listeners for a specific watch (or watches) based on individual watch settings for high and low pending maps. If the **InitPendingHigh** and **InitPendingLow** settings in the Launcher section of the **dtx.ini** file are also set, the **Pending Instance Thresholds** Launcher settings will take precedence.

Setting Description

Hi Value (no limit) for the watch to pause the event or events for that watch. The default value is **0**.

Low Value (no limit) for the watch to resume the event or events for that watch. The default value is **0**.

Trigger event settings

Event settings initiate map components of a system. Each map component is run based on a time event, source event, or combination of multiple events. A source event can be a file, message, or database event. If a map component is not assigned an event trigger, the Launcher does not initiate that map to run.

Note: When you analyze a system on the IFD and your system is configured for Launcher mode, a warning for a source or compiled map component that has no associated event triggers results.

A *watch* consists of the map and the set of events that initiate that map. Each event that contributes to the initiation of a map is called a *trigger*. Whenever the Launcher detects a new event trigger, the watches associated with that trigger are either initiated, placed in an initiation pending state, or maintained in an initiation pending state until all specified event triggers have occurred.

The same map can appear in different watches. For more information on this situation, see ″Same Map in Different Watches″.

Time events

A time event is the occurrence of a specific time that you designate to initiate a watch. You can select to trigger a map based on a singular time event or cycle according to a specified time increment.

To specify an event based on time, select the **TimeEvent** setting in the **Launcher** Settings window in the IFD. See ″TimeEvent″ for information about how to configure these settings.

Source events

A source event is the change in state of an input you designate to be a trigger. A source event can be associated with a file, database, or message source. When the **SourceEvent** setting for a map input card is **ON** and the map has a time trigger, the **TimeEvent** → **EventCoordination** setting is available. This setting controls the coordination of time and input events.

The following table describes the file, message, and database source events.

Event **Description**

File Event
A change in state of that file, its creation, or a change in the time stamp for an existing file.

If you use file triggers, you can enter the full path name for that file as the source name. You can also enter a path name relative to where the compiled map resides (that is, the **.mmc** file). The file name can also include wildcards, as described in ″Wildcards in Adapter Names″ .

Message Event
The occurrence of a new message in a messaging system

Database Event
Database modification that triggers a map. Database sources for certain database types can be used as source events for the Launcher, which allow insertions, deletions, or updates to database tables as the trigger mechanism that launches a map. The specific database trigger activities (that is, insertions, updates or deletions) are defined for the query in the Database Interface Designer. The event or events might be the result of a map or another application. A typical use is to configure a trigger that launch a map to run when another map has completed and modified a table.

Note: For more information about database triggering, and the specific databases that support it, see *Database Triggers* in the *Database Interface Designer* documentation. For platform and version-specific information, see the release notes (**readme.txt**) file.

Troubleshooting tip

To enable trace (logging) functionality for the file listener, you must enable the option in the **dtx.ini** file.

To enable logging for the file listener

1. Open **dtx.ini**, which is located in the product installation directory.
2. Scroll down to the **[M4File]** heading and locate the following text:
   ```
   ; used to enable logging for the file listener -- it should
   ; be a valid complete path (directory and filename)
   ;Logging=
   ```
3. Uncomment the `Logging` line by removing the semicolon in front of it.
4. Enter a complete path and filename for the log file. For example:

   `Logging=` *install_dir* `\m4trace.log`
5. Save and close the file.

Time and source event coordination

When a map has a time trigger and one or more source triggers, use the Launcher settings to coordinate these different events. You specify how these events are coordinated using the **EventCoordination** setting under **TimeEvent** in the **Launcher Settings** window. This setting specifies how the Launcher executes when a map has both a time trigger and at least one source trigger (which may include wildcard values). The two options for **EventCoordination** are **FirstAvailable** or **All or NONE**.

The following table describes the conditions that might exist at execution time and the result for each Event Coordination option.

Condition	**FirstAvailable**	**All or NONE**
No source trigger exists when the time trigger occurs.	Map goes into pending state and waits for the first source trigger to appear, then the map runs.	This time trigger interval is skipped and nothing is run.
One source trigger exists when the time trigger occurs.	Map is run.	Map is run.
When the time trigger occurs, multiple source events exist that satisfy a wildcard.	Map is run with the first set that appeared, instances of the same map are run sequentially with each set at successive time trigger intervals, based on the order that the event set appeared.	Multiple instances of the map are run for all event sets that exist.

Launcher input trigger files

These are recommendations for creating Launcher input trigger files if you exhibit the following issues:

- Multiple triggering of the same file.
- Unexpected EOF (end of file) during map processing, which is reported with the following error message: `Input type contains errors`.

If you have experienced the above issues, the following command usage is recommended:

- Use the `rename` system call or `mv` shell command to create input files on UNIX platforms.
- Use the `rename` or `ren` command on Windows platforms to create input files.
- Do not use the copy command or similar non-atomic commands or functions (such as FTP) because the Launcher file adapter can detect a file before it is fully copied.

This is applicable for both Windows and UNIX platforms.

Note: To avoid the potential problems listed above, you should use a **.tmp** filename extension during file creation and rename the file afterward.

Source and target settings

When you compile a map from the Map Designer, you supply a default set of sources and targets. From the IFD, you can override those sources and targets to adapt a map to the system in which that map is used.

These source and target setting options enable you to treat a map as a transaction for mission critical applications. The option settings are important when you use multiple target adapters. For example, if one output inserts data to a database correctly, but the next output fails to produce a related message, you can rollback the database insert.

Source adapters are used to read input data from a file, query a database, get a message from a queue, or receive the results of a call to an application. Target adapters are used to write output data to a file, insert or update data into a database, put a message on a queue, or call an application to send data to it.

From the **Launcher Settings** window, you can select source adapters for inputs and target adapters for outputs. Select the adapter type from the **Source** or **Target** drop-down list and then enter specific settings for the selected adapter type.

Adapter classes

The source and target adapters are organized into four adapter *classes*. Each adapter class has an icon that appears in the system diagrams in the IFD that indicates the Launcher properties of that adapter. The following table shows the Launcher properties for each adapter class. It also shows the properties available in all Server modes (Command Server, Launcher, and SDK) and those properties that are available only on the Launcher.

Adapter Class / Type	**Application**	**Database**	**File**	**Message**
I/O Type	Library Interface	Database Type	File Type	Message Type
All Server Types				
OnSuccess Action	✓	✓	✓	✓
OnFailure Option	✓	✓	✓	✓
Retry Option	✓	✓	✓	✓
Override option for compiled adapters	✓	✓	✓	✓
Launcher ONLY				
Event Management		✓	✓	✓
Resource Management		✓	✓	✓
Wildcard instantiation based on source events			✓	✓
Wildcard coordination with source events	✓		✓	✓

Wildcards in adapter names

Use wildcard notation to identify a source or target to coordinate data threads in the same map or among many maps. The **Launcher Settings** window allows only valid source and target names. Inconsistencies are analyzed from the IFD and only valid names are accepted. In the following list, wildcard notation refers to a name with asterisks or question marks in it.

General rules for using wildcards in adapter names:

- Directories *cannot* have wildcard notation in their names.
- A database source name *cannot* contain wildcard notation.
- If wildcards are used as sources or targets, at least one source trigger must have a wildcard.
- Question marks *cannot* be used in target file names.
- Multiple wildcards can be used in a source trigger name of a map, as long as there are no other source triggers.
- If more than one source file name contains a wildcard, that wildcard must be the same wildcard.
- Sources that are not event triggers and target names can contain at most one asterisk.

File adapters

When you select a file adapter, you supply the name of a specific file or a file name that contains a wildcard. If a source adapter is a file, you can specify that the file be a source trigger. If you use an input file with a wildcard in its name, the Launcher initiates a watch for any file that matches that name. If more than one source for the same watch has a wildcard, the first one to change state is used as the value for the other input sources and output sources that have that wildcard in its name.

Files used as sources or targets have the following characteristics:

Event triggers

- A file can be used as a source event trigger for a map
- Files used as triggers cannot have a **.tmp** extension. The Launcher uses this extension for temporary files. For example, temporary files are created when rollback options are selected.

Wildcards

- Can be used within a file name
- Cannot be used for directory names
- Can be used in Audit Log file names providing the input that is a trigger has the necessary wildcards to resolve the name for resource coordination

See the *Resource Adapters* documentation for more information about wildcards.

File names, path names, directories, and overrides

- File names can be specified with a full path name or a relative path name. Relative path names are relative to the path of the compiled map.
- File names can be coordinated with other sources and destinations.
- Files can be used as an override for any source or destination type.
- If the directory for a file source that is an input trigger for a map becomes disconnected while the Launcher is running, the Launcher will attempt to reconnect to that watch directory every 10 seconds.

OnFailure and Retry settings

- An **OnFailure** setting can be specified for a file that a map writes to in case the map does not run successfully. If an **OnFailure** setting of **Commit** is specified, the changes to the file will remain intact, even if the map does not complete successfully. If an **OnFailure** setting of **Rollback** is specified and a map does not complete successfully, the file will remain in its original state. That is, if a file originally exists, it will not be updated. If a file does not originally exist, it will not be created.
- The **Retry** option applies to files that exist. **Retry** is used for files that exist, but might be in use by applications unknown to the Launcher and cause a resource conflict. For example, a file to be opened is already open for write or a file to be opened for write is already opened for read or write.

Database adapters

Database source adapters enable you to query a database and use the results of that query as input data. Target adapters enable you to insert or update data to a database. Connections to a particular database are generally made once for each map instance. However, if one database resource uses a rollback option and another does not, the database adapter will use a separate connection for the rollback option.

Before you specify the database to be used by the Launcher, you must first define and configure connections to the selected database. For example, before you specify the Oracle **Connect String** (in the Database Interface Designer) to be used by the Launcher on a computer installed with Oracle client, you must ensure that the value you provide for the **Connect String** matches the Oracle service name definition on that client.

Note: For more information about how to define or configure connections to a database, see the documentation for your specific operating system and database.

Characteristics of database adapters

Event triggers

Database sources can be used as source event triggers.

Wildcards

Wildcards cannot be used for database sources and targets.

Overrides

Database adapters can be used as an override for any source or target type.

Database connections

- Database connections are coordinated across sources, targets, and database functions used in a single map instance. For example, if one database target is specified with a rollback option and that database is accessed by a `DBLOOKUP` function within any output, the connection will apply for data accessed in that function.
- A connection to a database is specified for each unique combination of data-source name, user ID, password, rollback option and map instance. You can force a new connection, for example, by using different user IDs or data source names. You can take advantage of single connections for a particular map instance by keeping all the database specific parameters the same.

OnSuccess, OnFailure, and Retry settings

- The **OnFailure** setting applies to modifications made to a database source or target within a particular map, burst or card instance (depending on the setting for **Scope**). For example, an **OnFailure** setting of **Rollback** specified for a source query applies only when that query references a stored procedure, which modifies the database
- A database transaction is synonymous with a connection for the purposes of **OnFailure** and **OnSuccess** settings. The database transaction begins when the connection is established and ends when the map completes. If the map does not complete successfully and a rollback is associated with that transaction, the database transaction is rolled back. This means, for example, that if multiple sources and targets of the same map share the same connection and all have an adapter rollback specified, any modification made to any one of these sources or targets are part of the same database transaction
- If a map does not complete successfully, all `DBQUERY` and `DBLOOKUP` functions used in that map always rollback. If a `DBQUERY` or `DBLOOKUP` function fails and the map succeeds, the function returns `NONE`, but any modifications to a database specified in the function will rollback.
- The **Retry** option applies when attempting to connect to a database.
- The **OnSuccess** option for a source or target is not supported. However, SQL delete statements and stored procedures used as sources can be used for deletion purposes.

Note: If more than one map instance deletes the same data, the first map instance that completes will delete that data. Deleted database data is not coordinated among map instances.

Application adapters

Application adapters are programs or functions called by the Launcher after a map instance has been initiated. The use and availability of both program and library function application adapters is platform dependent. If an application adapter is a function, that function must conform to a specific prototype. Its prototype, or interface syntax and semantics are defined in the *TX Programming Interface* documentation.

Characteristics of application adapters

Event triggers

Application adapters cannot be used as source event triggers, they are client applications within a map instance.

Wildcards

Wildcards can be used in an application adapter override command to coordinate trigger events with adapter arguments.

OnSuccess, OnFailure, and Retry settings

- Custom adapters can make use of the **OnFailure** and **Retry** options. The Launcher passes the information to the adapter. The adapter is responsible for responding to the rollback and retry.
- **OnSuccess** options are passed to application adapters.
- **Retry** options apply only if an adapter can be found and if applicable to that adapter. If not available on the server where the Launcher is installed, you will get a ″source not available″ or a ″target not available″ message.

Note: If you use your own application adapter, you are responsible for including functions for **OnFailure** and **Retry** options. For more information about using your own application adapter, see the *TX Programming Interface* documentation.

Message adapters

Message adapters allow you to send or receive discrete units of information, typically identified by a unique name. The naming conventions are adapter specific. For example, IBM WebSphere MQ names or *message identifiers* are 24-byte text items. SAP R/3 message names are called *transaction identifiers*. The storage mechanism for a message is also adapter-specific. For example, IBM WebSphere MQ uses queues to store messages.

When a source message adapter is used with an Launcher, the adapter notifies the Launcher when specific source event messages are available.

To use message adapters, you must have message adapters for the server where the Launcher is installed. For information about command options, see the *Resource Adapters Online Library*.

Characteristics of message adapters

Event triggers

Messages can be used as source event triggers for a map.

Wildcards

Wildcards can be used for message identifiers. If the message identifier is specified as a wildcard, the value of that wildcard is derived from the value of the message identifier of a message event trigger or from a file source with a wildcard in its name.

OnFailure, and Retry settings

- The **OnFailure** option for a source or target depends on the specific messaging adapters. For example, for IBM WebSphere MQ the source rollback option includes an option to place the source message on an error queue.
- The **Retry** option for a source or target depends on the specific messaging adapters. In general, the command option for the adapter must be valid on the computer where the Launcher is installed. If a connection to a queue cannot be made because the specified queue is not available, you will get a source not available message or a target not available message. If connection parameters are valid, but the message is unavailable due to, for example, a connection limit that has been reached or the message is locked, the retry will be attempted.

Adapters that support event triggers

The following list displays the adapters that can be used as event triggers by the Launcher.

Note: For platform and version-specific information, see the release notes (**readme.txt**) file.

• BW Staging BAPI	• IBM WebSphere MQ (server)	• Socket
• Database (See note below.)	• MSMQ	• TIB/RVTX
• File	• Oracle AQ	• Tuxedo
• FS Manager	• R/3 ALE	• Tuxedo (MT)
• HTTP	• R/3 BAPI	
• IBM WebSphere MQ (client)	• R/3 BDC	

Note: For information about the specific databases that support event triggers, see the *Database Triggers* in the *Database Interface Designer* documentation.

Connection testing adapters

The following list of adapters supports connection testing.

Note: See the release notes (**readme.txt**) file for platform and version-specific information.

• BW Staging BAPI	• MSMQ	• R/3 ALE
• DB2	• ODBC	• Siebel Business Objects
• HTTP	• OLE DB/MS SQL Server	• Socket
• IBM WebSphere MQ (client)	• Oracle	• SQL/MP
• IBM WebSphere MQ (server)	• Oracle AQ	• SQL/MX
• Informix	• PeopleSoft CI	• Sybase
• JMS	• R/3 BAPI	• TIB/RVTX
• MS SQL Server	• R/3 BDC	• Tuxedo

Subsystem component execution settings

A subsystem component references another system that you have defined. As such, a subsystem is an executable unit. You use the IFD to specify the execution settings of subsystem components. Because the IFD treats each subsystem component as a unique instance, these execution settings are maintained separately with each component.

A subsystem component has a defined set of interfaces based on the sources and targets of the various maps that it contains. The inputs and outputs of a subsystem are derived according to the following rules:

- An input of a subsystem is a source that is not internally produced by one of its components.
- An output of a subsystem is a target that is internally produced by one of its components, and is not deleted.

To access Launcher settings,

you define execution settings for each subsystem component in your system by accessing the **Launcher Settings** window. To display the **Launcher Settings** window in the IFD, click the **Execution Settings** button at the bottom of the subsystem component icon with the assigned server displayed.

You can also access this window by selecting the **Edit Execution Settings** command from the context menu for a subsystem component.

Launcher settings dialog

Use the **Launcher Settings** window to configure the settings for the subsystem component.

Server

The **Server** setting is used to specify the server on which the system will be deployed. The default value for the **Server** setting is **None**.

Value	Description
Local Server	A system is going to run on the local workstation, usually where the IFD is installed. You might want to assign **Local Server** to systems to use your workstation as a test environment to verify execution. Or, you can plan to use **Local Server** as a part of your production environment for system execution.
Distributed	Distributed server; a system is composed of subsystems that are assigned to run on different servers in a coordinated manner to distribute the processing. Components in a distributed system must be subsystems, not maps.
None	No server selected; no server is assigned. The system is not executable. Systems that have an assigned server of **None** do not appear in the Component view in the Navigator. They can be used as subsystems within a system that has a server assigned. A subsystem with an assigned server of **None** derives its server from its parent system that has a server assigned. Subsystems always display in the Navigator hierarchical view unless the parent system also has a server of **None**.
Custom	User-defined server

ExecutionType

The **ExecutionType** setting is used to specify the execution mode for the system. The default value for the **ExecutionType** setting is **Launcher**.

Value Description

Launcher
System will be deployed for execution by the Launcher.

Command
System will be deployed for execution by the Command Server.

Inputs and outputs

Expanding the **Input(s)** and **Output(s)** settings shows the map cards associated with the maps in the subsystem. The values shown are based on the settings specified in the map but the **Launcher Settings** window enables you to override these values. The values you can override are:

- **FetchAs** (for input cards)
- **WorkArea**
- **Backup**
- **Source** (for input cards)
- **Target** (for output cards)

See the *Map Designer* documentation for more information about card settings.

Chapter 6. Map initiation scenarios

This documentation describes map initiation scenarios and coordination.

Coordinating and using wildcards

You can use wildcard notation to specify sources or targets that have varying names. To determine the value of a wildcard, the Launcher requires that at least one wildcard-using source is specified as a trigger event.

Single source name

If you want a map to run each time a file appears in a particular directory, use a wildcard as part of the source name. An example is *install_dir*\receive*.any.

In this example, each file in the *install_dir*\receive directory with an extension of .**any** starts an instance of the **Run All** map. Depending on other Launcher settings, instances of the **Run All** map can be concurrent or sequential.

For some adapters, an implicit wildcard can be used. For example, if you use the IBM WebSphere MQ source adapter, the message identifier is an option. In this case, if no message identifier is specified, any message on a particular queue will initiate a map. An example is **-QMN MyManager -QN MyQueue**.

If you require all message identifiers that end in **XYZ** to run the map, use a specification that is such as **-QMN MyManager -QN MyQueue -MID *.XYZ**.

Depending on other Launcher settings, instances of the**Run All** map can be concurrent or sequential.

Matching multiple source names

If more than one source name for a map has a wildcard, you can use the first source with that wildcard to set the wildcard value for the other sources required by the same map instance.

For example, consider a business using an order system that requires a credit check for all orders. Orders that pass the credit check need to be confirmed; the business does not want to mismatch an order with the wrong credit check.

In this example, the order message is a source trigger. When a new order arrives, its message identifier is used as the value of the wildcard and **CreditCheck** will not run until a credit report message with the same message identifier arrives.

Matching source to target names

If a source has a wildcard, any target that contains an asterisk is assigned the source wildcard value. Modify the credit check example used previously to include an asterisk in the target message's identifier:

The value of the Order message identifier wildcard is assigned to the **CreditStatus** message identifier wildcard. For example, if the value of Order's message identifier is `AAA1212TW`, the **CreditStatus** message identifier is assigned the value, `STATAAA1212TW`.

Multiple wildcards in a source name

If a source name has multiple wildcards and the target name has a single asterisk wildcard, the target name asterisk is replaced with the first wildcard value, next wildcard value, and any characters that exist between the wildcards.

For example, with a source name of `*a???b.c` and a target name of `*.d` , if the source file name instance is:

```
xyazzzb.c
```

the target name assigned by the Launcher is:

```
xyazzz.d
```

Threading wildcards across multiple maps

You can track a data object and other objects produced as a result of its existence. These related objects can be produced asynchronously and feed yet other transformation steps that produce more objects.

You can tag each related object to distinguish the thread of actions taken as a result of the event of the first object that started.

For example, suppose there is a travel reservation system. One component of the travel reservation system accepts travel requests with a proposed itinerary. That component sends out request messages to airlines and hotels for availability. After there is a response from an airline and hotel for that itinerary, a response is produced so that a reservation can be made.

To track a travel request to match it to a reservation as described previously, you can design an example system to have an **Availability** map component with an input event trigger that accepts the travel requests represented by the **-QMN MyManager -QN Itinerary -MID.*** input card. The **Availability** map can send out request messages represented by the **1: AirlineRequest** and **2: HotelRequest** output cards. The responses can be represented by the **-QMN MyManager -QN Airline -MID Air*** output card from the airline and the **-QMN MyManager -QN Hotel -Htl*** output card from the hotel, which can be used as input by the **Reservation** map component with an input event trigger to produce the**-QMN MyManager -QN Response -MID** *output card.

Using wildcards, each Itinerary message identifier can be tracked by the same identifier on the Hotel queue, the Airline queue, and the Response queue.

Multiple map Instances in the same watch

The same map might be initiated more than once by the same watch. For example, a map configured to run once every minute will run multiple times. You might, or might not, want multiple instances of the same map running at the same time. How Launcher settings are specified determine specific Launcher actions.

Using time events

In this scenario, a map is configured to run every minute. The map starts at 3:00 P.M. and takes over one minute to run. Another instance of that same map could be initiated at 3:01 P.M., and you could have two instances of the same map running at the same time. You can design an example **ApptoApp** map with a time event trigger, **1: OrderAppln_In** input card and **1: OrderAppl'n_Out** output card.The map takes a minute-and-a-half to run on average. Each time it runs, it uses an application adapter to get data expected from some communication session, transforms it to its output form, and puts it in a file. Assume that this map is the only one controlled by the Launcher. Here are the different ways the map could run in real-time.

Case 1: Different map instances can run at the same time

If you select unique work files, different map instances can run at the same time.

Unique work files on their own do not guarantee map instances will run at the same time. For example, if the output file is appended, the resource manager will not let multiple map instances run at the same time because the result of one map instance is dependent on another.

Case 2: Different map instances can run sequentially

If unique work file names are not selected, different map instances will *always* run sequentially. A second map instance will begin running after the first map instance has completed running.

There are other situations for which maps run sequentially. For examples, see ″Using Source Events″ .

Case 3: Shared map system resources can make map instances take longer to complete

If the same map system resources (other than work files) are used by multiple map instances, map instances will run at the same time. However, one map instance can be placed in an I/O pending state until shared system resources are available. (Shared map system resources can be log files that are not unique, trace files, and compiled map files.)

In this case, one map instance initiates while another instance is in process, but the second seems to run longer than it should. For example:

Case 4: Skip if Busy can be used to omit map instances from starting

The Skip if Busy Option can be used to omit map instances from starting when another instance is in process.

In this case, it might not matter what options you select for work files and log files because map instances always run sequentially.

Note: If you want multiple instances of the same map to be able to run at the same time, be sure to select unique work files and unique log file run options. These options create different work files and log files for each map instance so that one map instance will not interfere with another. If you do not select these options, the Launcher will keep the second instance in a pending state until the first instance has completed.

Using source events

In this scenario, a map is configured to run each time a new file appears in a specified directory. A map instance starts when the first file appears and takes over a second to run. Another file appears a half a second after the first file. There could be two instances of the same map running at the same time. You can create an example **AppToApp** map with an event trigger, *install_dir*\MQS\APPLN*.TXT input card and **1: OrderAppl'n_Out** output card.

Assume that this map is the only one controlled by the Launcher. The following case describes different ways the map could run in real-time.

Case 1: Different map instances run at the same time, if all sources and targets are different.

How run options are selected might affect the length of time a map instance is in process. Assume run options are set as one of the following combinations:

- Select unique work file names or work area in memory and select no log file.
- Select unique work file names or work area in memory and select unique log file names.

In this case, each map instance is independent and the length of time each map instance runs depends only on the amount of processing time associated with the size of the data and the mapping rules used.

If sources and targets are independent, and unique work files are used, but map instances share log files or other map system resources, you might see map instances in process run longer than expected. This occurs because the Launcher will initiate an instance based on available event triggers and then waits for map system resources to become available.

Note: Sources can be the same if they do not have the potential of a map writing data to it. See ″Different Maps That Use the Same Source″ .

Note: If you use work files with the **WorkFilePrefix** setting of **MapName** (available under the **Workspace** setting; see ″MapSettings > WorkSpace″), be sure to use the retry option if multiple map instances can run at the same time. If a retry is not set, the map will fail to run if these resources are in use.

You might also see this same effect if a source or target resource is required, it cannot be shared, and it is in use by an application unknown to the Launcher. In this case, be sure to use the source or target retry option, or the map instance will fail to run.

Case 2: Different map instances run at same time or sequentially based on source and target settings

Unique log file names and unique work file names do *not* guarantee map instances will run at the same time. For example, if the same input file is used each time a map runs, and it is updated by a map rule, the resource manager will not let multiple map instances run at the same time because the result of one map instance is dependent on another.

Map instances will run sequentially under the following conditions:

- If a map uses the same file or message as both a source and target
- If a source is the same file each time a map runs and a map rule is used to update it
- If a target is a file and that file is appended

If any of the previous conditions apply, you will not see two instances of the same map running at the same time.

Processing HTTP input events

The Launcher Agent enables the Launcher to directly process Hypertext Transfer Protocol (HTTP) input events, including XML messages, in a safe and secure manner.

See the *Launcher Agent* for more information about the Launcher Agent.

Same map in different watches

You can have the same map in different watches. The initiations of map instances of one watch can be configured to be independent from the initiation of map instances of other watches.

For example, you want to process orders based on priority. All orders look alike before processing and are converted to the same output form. Priority orders are placed on one message queue. Normal orders are placed on another queue. This is the same map with different component names and different sources.

You want **Priority Orders** to run as fast as possible and (if more than one priority order appears at the same time) you want to process them all at the same time. Normal orders can be processed at the same time, as long as there are enough computer resources available to do so. In this case, you configure **Priority Orders** to have a higher priority than **Normal Orders**.

You also set unique work files and unique log files, so that **Priority Orders** will not need to wait for map system resources. If you just keep work files for each map in a different directory, **Priority Orders** and **Normal Orders** could run at the same time. However, this would limit **Priority Orders** instances to run sequentially. It would also limit **Normal Orders** instances to run sequentially.

Note: If you are using the log file options and are doing this for more than one map, you might want to keep these files for each watch in a separate directory. Although unique log file names are globally unique, they do not contain the name of the map. For example, if you use log files to trigger other maps, keeping these in a separate directory will make sure the logs trigger the correct map instance.

Different maps that use the same source

Different maps that use the same source can run at the same time, as long as that source is not updated by one of those maps.

For example, a patient is admitted to a hospital. The admitting department produces an admitting message. Various other departments in the hospital use the admitting message. The medical records department needs to update the patient record. The accounting department needs to track the patient's charges. The nurses' station on the floor where the patient will be staying needs to be notified. Other departments might have a need to know.

Each department uses a separate system for its internal processing. And none of these understands how to process the admitting message. The same message must be converted to many different forms, one for each department.

You could create one map and do all the conversions in one step, but some departments need the information faster than others. You therefore decide to configure a different map for each department.

For example, create an **Admit** system in the IFD with a different map for each department, **Pharmacy**, **Medical Records**,**Accounting** and **Nurses**.

In this example, each map uses the same trigger source as input. An example input trigger source is **-QMN MyManager -QN ADMIT**. Each map has different targets. Example targets are **1:HL7_Style** for the **Admit To Pharmacy** map,, **1:MedicalRecord** for the **Admit To Medical Records** map, **2:InsuranceBilling** and **1:PatientCharges** for the **Admit To Accounting** map and **1:FAX** for the **Admit To Nurses** map. Each map is independent from the other; each can run at the same time. Also, more than one instance of each map could run at the same time.

Impact of Delete settings

All maps that share the same source and have no Delete action specified are run first. When these maps complete, a map that specifies the Delete action for that shared resource can be initiated. Only one map should use the delete action for this type of resource to avoid a `Source not available` message.

Different maps use the same target

If different maps use the same target, the type of target influences how maps are run:

- If the target is a file or a message, the resource manager will prevent different maps that use the *same* file or message from running at the same time.
- If the target is an application, the Launcher relies on the application adapter to be multi-threaded and assumes resources for different instances of that adapter are independent.
- If the target is a database, the database adapter communicating with the specific database handles shared resources.

One map's target is another map's source

When the target of one map is used as a trigger source for another, the Launcher does not start any map until that trigger occurs. However, if a file or message source is not a trigger, the Launcher initiates a map based on other event criteria.

Map initiation tip

- If a single multi-threaded map is triggered 100 times, the map is read up front during initialization, and kept in memory for all runs. Thus, the map is only read once.

Chapter 7. Management Tools

This documentation describes how to configure and use the Management Tools for the Launcher.

The Management Tools are a component of WebSphere Transformation Extender.

Note: When using the z/OS local administration interfaces, see ″z/OS Launcher″ for more information. The following information does not apply.

Overview

The Management Tools provide a means to manage and view Launcher processes running in the Launcher. The Management Tools are automatically installed with WebSphere Transformation Extender and can also be installed separately on multiple machines to allow multiple users to monitor Launcher activity remotely. You can view both Windows and UNIX Launcher activities from one station.

Note: A large number of instances of the Management Tools running will adversely affect the performance of the Launcher.

Management Tools for Windows platforms

You can install the Management Tools separate from WebSphere Transformation Extender on your client machines. The Management Tools installation for Windows includes the following applications:

- Management Console (″Management Console″)
- Launcher Monitor (″Launcher Monitor″)
- Snapshot Viewer (″Snapshot Viewer″)

Management Tools for UNIX platforms

You can install the Management Tools separate from WebSphere Transformation Extender on your client machines. The Management Tools installation for UNIX includes only the Management Console application.

Note: If you are using Native Window Manager and experience symptoms such as no title bar, no insertion point, or no pop-ups, you can work around this by reconfiguring the X Window Manager (Exceed settings).

Unique keys in the Launcher Management Tools

There are some unique key combinations available in the Launcher Management Tools in addition to the standard Java navigation keys.

For enhanced accessibility, the following key combinations are available in the Launcher Management Tools applications.

Application name	Navigation description	Key combination
Launcher Administration	To navigate the tabs in the Deployment Directories panel.	Shift+F2

Application name	Navigation description	Key combination
Management Console	To navigate between panes.	Shift+F2
	To navigate the status panel tabs.	Tab

Default port settings

The Launcher Administration, Launcher, and Management Tools applications work collectively and must communicate by means of ports. Each application has default port settings. However, you can change them as needed. From the Launcher Administration interface you can configure startup, connection user security, firewall, and deployment options for the Launcher. The following list displays the default port settings for each application, and the interrelated structure of the applications.

- The Launcher relays monitoring information over port 7001 to the Monitor.
- The Launcher relays status information over port 7000 to the Management Console.
- The Launcher service sends control information (start, stop, pause, resume) to the Launcher process that is running over port 7002.
- The Management Console commands (start, stop, pause, resume, disconnect) are relayed to the Launcher service over port 5015 (also known as the listening port).
- Communications between the Monitor and the Launcher service are relayed over port 5016 (also known as the listening port).

If you run multiple Launchers on one machine, they must run on different listening ports and must have different port ranges. If this condition is violated, the Launcher might not work as intended. Also, export the display to a server that supports X Window systems for that same environment.

When you run the Launcher through Telnet or on WebSphere Transformation Extender for z/OS, ensure that the DISPLAY variable points to the machine that you are using. The value of the DISPLAY variable can be checked by typing `echo $DISPLAY` at the Telnet prompt and can be set using the export command. Otherwise, if the listening port is already in use, an exception is raised instead of the regular message that appears indicating that the listening port is in use. However, in all cases there is an entry in the log file that indicates which listening port <*number*> is in use.

Management Console

The Management Console allows you to control and view the status of Launcher processes (**.msl** system files) running in the Launcher. From the Management Console you can start, stop, and pause a compound system, which is the combination of all **.msl** files in the deployment directory. See ″Multiple Processes″ for additional information when using the multiple-processes feature.

The Management Console provides a variety of figures about the maps being run by the Launcher, such as how long the Launcher has been running, how many maps have been processed, and so on.

Features of the Management Console

- Internal information is provided for statistical analysis and troubleshooting.
- Runs on all platforms that WebSphere Transformation Extender supports.
- Data is displayed in an easy-to-read format.
- Debugging information is available.
- Ability to monitor the current state of the compound system process.
- Ability to start, stop, and pause a compound system without stopping the service.
- Ability to easily obtain both real-time and historical statistical information from the compound system running in the Launcher.

Using the Management Console

The Management Console is used to display run-time statistics of the Launcher. You must complete the following steps before using the Management Console.

Management Console quick start list

After you complete required steps outside of the Management Console, there are three main tasks to complete in the Management Console before you can begin viewing run-time statistics of the Launcher:

To prepare to view Launcher run-time statistics

1. Set up an instance of a Launcher. (See ″To set up an instance of a Launcher″ .)
2. Connect to a Launcher. (See ″To connect to the Launcher″.)
3. Start the system or systems. (See ″To start the system(s)″ .)

The procedures for each task are described in the following topics.

Management Console icon descriptions

The following icons appear in the Management Console navigation pane:

Icon	**Description**
	Management Console in connection mode
	Management Console in disconnection mode
	Management Console is in retry mode. (The icons periodically fluctuate)
	system stopped
	system running
	statistics
	system paused
	system stopping

Required steps to perform outside of the Management Console

- Generate a system file (**.msl**) and add it to either the ***install_dir*\systems** directory (which is the default) or to a deployment directory of your choice.
- Using the Launcher Administration interface, do the following activities:
  - Create a user profile with the appropriate access rights.
  - Set up the deployment directory path from where the Launcher will access the system file (**.msl**).
- Start the Launcher.

After you complete the required steps, you are ready to begin using the Management Console. Follow the steps in the following topics to quickly get started using the Management Console.

Getting started

After you open the Management Console, the first thing you need to do is set up an instance of the Launcher that you're going to monitor. After the Launcher is set up, the next step is to connect to the Launcher and begin monitoring.

To open the Management Console (Windows):
Select **Management Console** from the **Start** → **Programs** menu under IBM **WebSphere Transformation Extender** *n.n* (where ***n.n*** represents the version number).

To open the Management Console (UNIX):

1. From the command prompt, enter the following command:
   ```
   . setup
   ```
2. From the same command prompt, enter the following command:
   ```
   mgmtconsole.sh
   ```

To set up an instance of a Launcher:

Note: You must set up users with the appropriate access rights in the Launcher Administration interface to view systems running in the Management Console. If you add a new user after the Launcher service is running, you must close the Management Console and restart the Launcher before the user is recognized.

1. From the **Launcher** menu, select **New** to add a Launcher.

   The New Launcher window appears.
2. In the **Name** field, type a unique name of up to 20 alphanumeric characters.

   Note: Although the name must be unique, it can be arbitrary because it is only a label that appears on its own tab in the Management Console navigation pane.
3. In the **Host Name** field, enter the host name or IP address. For example, `192.168.1.120`.
4. In the **Port** field, enter the port number for the Launcher. Port `5015` is the default setting.

   Note: Each host name must have a separate port number.
5. In the **Refresh Interval (sec)** field, type an interval time in seconds. The minimum value is 1 second, meaning that the Launcher will check for new data every second. The default setting is 5 seconds.

6. Under **Security**, type a **Login Name** and **Password** that permits access to the host. This information must correspond to a user profile set up in the Launcher Administration interface (″Launcher Administration″).
7. Click **OK**.

   The New Launcher window closes and the new Launcher name is listed in the navigation pane of the Management Console window. The name is also displayed on a tab at the bottom of the navigation pane.

To connect to the Launcher:

Before you create the connection to the Launcher, the Launcher must be running. (See ″Starting and Stopping the Launcher″ for instructions on starting the Launcher service). See ″Multiple Processes″ for additional information when using the multiple-processes feature.

From the navigation pane, right-click on the Launcher and select **Connect**.

The connection is established. The **CompoundSystem** icon is displayed in the navigation pane for a single process.

The system is labeled **CompoundSystem**, which can represent one or more system files (**.msl**).

To start the system or systems:
From the navigation pane, right-click **CompoundSystem** and select **Start** from the context menu.

While the request is being processed, you can see at the bottom of the window which **.msl** files are starting.

In the navigation pane, if you expand the **CompoundSystem**, you will see the **Status**, **History**, and **Configuration** icons:

Note: If you see this information upon opening the Management Console, it indicates that the **Automatic startup** option in the Launcher Administration is enabled.

The statistics display in the main window.

Note: See ″Management Console Statistics″ for a description of each statistic.

Controlling systems

The Management Console enables you to control the systems running. If you right-click on **CompoundSystem** in the navigation pane (see ″Multiple Processes″ for additional information when using the multiple-processes feature), a context menu appears that has disconnect and stopping options. The following instructions describe how to disconnect from the Launcher and stop the service.

To stop an instance of the Launcher:
1. From the Management Console navigation pane, right click **CompoundSystem** and select **Stop** from the context menu.

   The status of the system changes from **Running** to **Stopping** to **Stopped**.
2. Go to the **Launcher** menu and select **Disconnect**.

The connection is terminated and the **CompoundSystem** disappears from the navigation pane.

To stop the Launcher service (Windows):

1. From your desktop, right-click **My Computer** and select **Manage**.
2. From the Computer Management window, expand the **Services and Applications** list and select **Services**.
3. Right-click **WebSphere Transformation Extender Launcher** and select **Stop** from the context menu.

To stop the Launcher (UNIX):

1. From the command prompt, enter the following command:
   ```
   . setup
   ```
2. From the command prompt, enter the following command:
   ```
   launcher.sh -stop
   ```

See ″UNIX Commands″ for more UNIX commands.

Retry option

The Management Console has a retry mode that automatically attempts to reconnect to the Launcher service in case it becomes unavailable or shuts down. This eliminates the need to manually monitor the Management Console for possible problems.

When the Launcher service fails, the Management Console attempts to reconnect to the server every 20 seconds. When the Management Console attempts to reconnect, the status bar at the bottom of the navigation pane displays the time in seconds.

To enable retry mode

From the Management Console window, select **LauncherRetry if Disconnected**.

A check mark appears next to **Retry if Disconnected**.

Because retry mode is an infinite process, you must stop the operation manually if needed.

To manually stop retry mode

From the Management Console window, select **LauncherStop Retry**.

The Management Console stops trying to connect to the Launcher.

To connect to the Launcher again, you must restart the service or daemon.

Snapshot option

The **Snapshot** option in the Management Console provides a means for saving a ″snapshot″ of Launcher information to a file in XML format. A ″snapshot″ of Launcher information refers to capturing a moment in time of status, historical, and configuration information that the Management Console provides about a **CompoundSystem** (refer to ″Multiple Processes″ for additional information when using the multiple-processes feature) running under a Launcher.

To create a snapshot file

Note: This option is not available when the **CompoundSystem** is stopped.

1. From the Management Console, select **System** → **Snapshot**.

   The Save As window opens.
2. Specify the name of the XML file to which the Launcher **CompoundSystem** information will be stored and click **Save**.

   The file is saved in XML format.

Dynamic adapter tracing

Dynamic adapter tracing enables you to dynamically turn on and off adapter tracing for a map running under the Launcher. Through the configuration pane of the Management Console, you can configure adapter tracing dynamically. When you turn adapter tracing on, the result is that the map will write all the trace information to a new trace file or append the trace information to an existing file.

You do not need to enable adapter tracing in the adapter command line of your map to optimize it. If you encounter an issue with the adapter, you can dynamically turn adapter tracing on in the Management Console.

Using the dynamic adapter trace feature

This feature is helpful for resolving adapter issues without having to:

- bring the Launcher down to add tracing to the adapter command lines
- having to recompile and redeploy the maps
- having to restart the Launcher

The following procedure assumes you have connected to a Launcher from the Management Console and have selected the Compound System that appears in the navigation pane.

Refer to ″Multiple Processes″ for additional information when using the multiple-processes feature.

To use the dynamic adapter trace feature:

1. Start the Launcher.

   When the Launcher starts, it will initialize all trace settings to default values as configured on the adapter command line. Tracing will be done for all watches according to the adapter command line settings.
2. Enable the adapter tracing feature in the Management Console.
3. Configure adapter tracing dynamically through the configuration pane.
4. Modify adapter tracing configurations.

   Note: When you modify the adapter tracing configurations after a Launcher has started, the changes will take affect in the next map that is run for that component.

Enabling dynamic adapter tracing

The configuration pane in the Management Console is used to configure the settings for enabling dynamic adapter tracing.This procedure assumes that the Launcher is running and there is a Launcher system (**.msl**) file in the deployment directory.

To enable dynamic adapter tracing

1. After highlighting a Launcher system (**.msl**) file, from the menu bar in the Management Console, select **System**.
2. Select **Trace & Log** from the pull-down menu.

   A pane in the right portion of the window opens. Through that pane, you can configure the dynamic adapter tracing specifications for each map component in each system separately.

Configuration pane

When you have enabled dynamic adapter tracing through the **System menu** option in the Management Console, you use the configuration pane to configure settings for a watch. For each watch, you can:

- turn off adapter tracing, which overrides the adapter command line that was defined in a map through the Map Designer
- turn on adapter tracing for all adapters including child RUN maps
- use the default tracing configurations in the adapter command line that was defined in a map through the Map Designer

Selecting **Trace** from the **Config:** pull-down menu: in the configuration pane will present the configuration pane elements specific for dynamic adapter tracing.

Configuration pane elements

The configuration pane in the Management Console used for dynamic adapter tracing consists of the elements required to enable tracing and configure the trace specifications. The following table lists the components and provides a description of their use.

Configuration Pane Elements
  Description

Config:
  The **Config:** pane contains a pull-down menu where you can select either of the following two configuration options:

- **Trace**

  The **Trace** configuration option indicates that you want to configure dynamic adapter tracing.
  When you select **Trace**, the trace configuration for the selected Launcher will appear.
- **Logging**

  The **Logging** configuration option indicates that you want to configure dynamic logging.

Systems:
  The **Systems:** pane contains a pull-down menu where you can select the path and system file for the Launcher process.

Components:
  The **Components:** pane contains all the map components in the Launcher system, which you can configure individually. It is the list of all top-level map components running in that Launcher process for the selected system.

  When you select a map, only the trace settings for that selected map will appear. When you select a different map, a different set of trace settings will appear.

All RUN maps associated with that component will use the same settings.

Only information for a running system will be available for configuration, not for a stopped system.

Settings:
The **Settings:** pane comprises the **Trace** and **Action** panes, which are used to configure trace settings for each of the components in the **Components**: pane.

Trace The **Trace** pane contains radio buttons for the following three adapter trace setting options:

- **Default**

  The **Default** adapter trace setting option indicates that the Launcher will use the settings on the adapter command line for a map configured in the Map Designer or the Integration Flow Designer.
  The **Default** option is the default adapter trace setting.

- **ON**

  The **ON** adapter trace setting option enables adapter tracing for all adapters for a map.
  This setting overrides any setting on the adapter command line for a map configured in the Map Designer or the Integration Flow Designer.

- **OFF**

  The **OFF** adapter trace setting option disables adapter tracing for any adapter.
  This setting overrides any setting on the adapter command line for a map configured in the Map Designer or the Integration Flow Designer.

Action
The **Action** pane contains two sub-elements related to the output adapter trace file:

Adapter trace file option

Allows you to configure how you want the trace content to be written.

There are two radio buttons for the adapter trace file options:

- **Create**

  When the **Create** option is selected, the map will write all the trace information to a new trace file.

- **Append**

  When the **Append** option is selected, the map will write all the trace information to a new trace file or append the trace information to an existing file.

File Name:

Allows you to configure the trace file name and location.
See ″Using Wildcards″ for information about using wildcards in your trace file name.

Apply The **Apply** button is used to refresh the Launcher with the settings associated with the component.
The adapter trace configuration changes for the component will be passed to the Launcher process, and will take effect when the next map is initiated for the configured component.

Note: These dynamic adapter trace configurations are not saved anywhere.

Using wildcards

Wildcards in the trace file name can be used to reference map source events. They are specified in the configuration pane of the Management Console when using the dynamic adapter tracing feature.

How wildcards are used

- Used in the trace file name based on source events that also use wildcards.
- Can not be used if the source event does not use wildcards.
- Allow for unique instances of adapter trace files.

Output files

The output of using the dynamic adapter trace feature is a file. It contains the following sections:

- Header
- Trace results for a map instance
- Trailer

The header and trailer sections are written to the trace file before and after each map instance including RUN maps. They both contain the following information:

- map name including full path
- date and time of executed action type
- map instance count, which represents the cumulative number of times the map has been executed

Note: RUN maps will be nested within top map entries to enhance your ability to see the sequence of events.

See the ″File Name:″ within the configuration pane element for more detailed information about naming the adapter trace output file.

Dynamic logging

Dynamic logging allows you to configure a Launcher process for logging while the Launcher is running. You can configure the following logging entries in the Management Console:

Logging Entries

- Launcher (ES)
- Resource Manager (RM)
- Connections Manager (CM)

Each logging entry is categorized by the following severity classes and debug components listed from lowest impact to highest:

Logging Categories

- Trace
- Debug
- Info
- Warn
- Error

- Fatal

Without having to stop the Launcher, you can turn each of these logging components (**Trace**, **Debug**, **Info**, **Warn**, **Error** and **Fatal**) for each of the logging entries (**Launcher**, **Resource Manager**, and **Connections Manager**) on and off individually for an Launcher process that you select to configure.

For example, you can run the Launcher with **INFO** settings to track the status of maps. If one of the maps connecting to a database starts to fail, you can dynamically enable the setting (turn on) for the **DEBUG** and **ERROR** components under the **Connections Manager** logging entry for a Launcher process that you have selected. Now you can track what the database connection is reporting without having to stop the Launcher, configure the Launcher with different debug settings, and then restart it.

Using the dynamic logging feature

This feature is helpful for resolving Launcher issues *without* having to:

- bring the Launcher down to enable the logging feature
- reconfigure the Launcher with different logging settings
- restart the Launcher

The following procedure assumes you have connected to a Launcher from the Management Console and have selected the Compound System that appears in the navigation pane.

See ″Multiple Processes″ for additional information when using the multiple-processes feature.

To use the dynamic logging feature

1. Start the Launcher.
   - When the Launcher starts, it will use the logging category settings configured in the **INI** (**dtx.ini**) file under each logging entry section (**Launcher**, **Resource Manager**, and **Connections Manager**) to determine which severity levels and components to debug.
   - The following logging categories in the **INI** file will be configured to `0` to turn that logging category off or `1` to turn that logging category on:
     - LogTrace
     - LogDebug
     - LogInfo
     - LogWarning
     - LogError
     - LogFatal
2. Enable the logging feature in the Management Console.

Enabling dynamic logging

The configuration pane in the Management Console is used to configure the settings for enabling dynamic logging. This procedure assumes that the Launcher is running and there is a Launcher system file (**.msl**) in the deployment directory.

To enable dynamic logging

1. After highlighting a Launcher system (**.msl** file), from the menu bar in the Management Console, select **System**.
2. Select **Trace & Log** from the pull-down menu.

   A pane in the right portion of the window opens. Through that pane, you can configure the dynamic logging specifications for each logging category component for each logging entry.

Configuration pane

When you have enabled dynamic logging through the **System** menu option in the Management Console, you use the configuration pane to configure logging settings for a Launcher process.

Selecting **Logging** from the **Config:** pull-down menu in the configuration pane will present the configuration pane elements specific for dynamic logging.

Configuration pane elements

The configuration pane in the Management Console used for dynamic logging consists of the elements required for you to specify logging and configure the logging specifications.

The following table lists the components and provides a description of their use.

Configuration Pane Elements
    Description

Config:
    The **Config:** pane contains a pull-down menu where you can select either of the following two configuration options:

- **Trace**

  The **Trace** configuration option indicates that you want to configure dynamic adapter tracing.
- **Logging**

  The **Logging** configuration option indicates that you want to configure dynamic logging.
  When you select **Logging**, the Logging configuration for the selected Launcher will appear.

Launcher:
    The **Launcher:** pane contains the list of severity levels and components (logging categories) for the Launcher logging entry that you can enable or disable for the logging of the Launcher process you selected.

Resource Manager:
    The **Resource Manager:** pane contains the list of severity levels and components (logging categories) for the Resource Manager logging entry that you can enable or disable for the logging of the Launcher process you selected.

Connections Manager:
    The **Connection Manager:** pane contains the list of severity levels and components (logging categories) for the Connection Manager logging entry that you can enable or disable for the logging of the Launcher process you selected.

Apply The **Apply** button is used to:

- trigger the Management Console to pass these logging configuration changes to the Launcher process
- refresh the Launcher with the settings associated with each logging category for each logging entry

As a result, the Launcher will notify the Management Console to synchronize the logging settings.

Clear All
The **Clear All** button is used to:

- disable all the check boxes under the three logging entries: **Launcher:** , **Resource Manager:** and **Connection Manager:**

The **Clear All** button is grayed out when there are no check boxes enabled.

Output files

The output of using the dynamic logging feature is a file. It contains the log entries. A log entry is a single line entry for each event that has been logged and is comprised of components in the following format:

```
<msgid><date/time><threadid><instance><msgtext><extra>
```

Log Entry Component
Description

msgid unique identifier of the log entry

date/time
date and time of the executed action type

threadid
identifier that distinguishes each map that is running concurrently

instance
instance number of a map that is running

msgtext
log entry

extra area for additional information (optional)

Logging messages for Trace

This documentation presents the logging messages for the Trace logging category for each logging entry (Launcher, Resource Manager, Conection Manager) listed by each message category (01*xx*, 03*xx*, 05*xx*).

Launcher trace 01xx

Message ID
Message

DTXLN-T-0101
Launcher trace log enabled

DTXLN-T-0102
Launcher trace log diabled

DTXLN-T-0103
Starting Launcher <*pid*>

DTXLN-T-0104
Loading MSL <*msl_name*>

DTXLN-T-0105
Before Init Resource Manager

DTXLN-T-0106
After Init Resource Manager

DTXLN-T-0107
Before Init Connection Manager

DTXLN-T-0108
After Init Connection Manager

DTXLN-T-0109
Init socket <socket_type>

DTXLN-T-0110
Map Start <*map_instance_number*>

DTXLN-T-0111
Map End <*map_instance_number*>

DTXLN-T-0112
Before resource pending

DTXLN-T-0113
After resource pending

DTXLN-T-0114
Before adapter call <*adapter*>

DTXLN-T-0115
Before adapter call <*adapter*>

DTXLN-T-0116
RUN map call <*run_map*>

DTXLN-T-0117
Before destroy table

DTXLN-T-0118
After destroy table

DTXLN-T-0119
Before RMDestroy

DTXLN-T-0120
After RMDestroy

DTXLN-T-0121
Pending thread added

DTXLN-T-0122
Terminating --> last message to log

Resource Manager trace 03xx

Message ID
Message

DTX-RM-T-0301
Resource Manager trace log enabled

DTX-RM-T-0302
Resource Manager trace log disabled

DTX-RM-T-0303
Init global resource table

DTX-RM-T-0304
Init local resource table

DTX-RM-T-0305
rm-realloc called start

DTX-RM-T-0306
rm-realloc called end

DTX-RM-T-0307
addrmentry blocking <*ID*>

DTX-RM-T-0308
maddrunentry suspending <*ID*>

DTX-RM-T-0309
maddrunentry resuming <*ID*>

DTX-RM-T-0310
removermentry - can't resume <*ID*>

DTX-RM-T-0311
Entering rmdestroy

DTX-RM-T-0312
Leaving rmdestroy

Connection Manager trace 05xx

Message ID
Message

DTX-CM-T-0501
Connection Manager trace log enabled

DTX-CM-T-0502
Connection Manager trace log disabled

DTX-CM-T-0503
Entering destroy table

DTX-CM-T-0504
Destroy table - stopping listeners

DTX-CM-T-0505
Destroy table - closing connections

DTX-CM-T-0506
Destroy table - uploading adapters

DTX-CM-T-0507
Leaving destroy table

DTX-CM-T-0508
Entering SAM entry <*parameters*>

DTX-CM-T-0509
Leaving SAM entry <*params*>

DTX-CM-T-0510
Begin rm worker thread

DTX-CM-T-0511
End rm worker thread

DTX-CM-T-0512
RMParseAdapterCommand <*parameters*>

DTX-CM-T-0513
Begin middleboot thread

DTX-CM-T-0514
End middleboot thread

DTX-CM-T-0515
RmiddleBoot active

DTX-CM-T-0516
FireEvent - calling launchadapter <*x*>

DTX-CM-T-0517
FireEvent returned <*x*>

DTX-CM-T-0518
Begin rmlistenerthread worker

DTX-CM-T-0519
End rmlistenerthread worker

DTX-CM-T-0520
RMInittriggers - maps = <*amount*>

Logging messages for Debug

This documentation presents the logging messages for the Debug logging category for each logging entry (Launcher, Resource Manager, Conection Manager) listed by each message category (11*xx*, 13*xx*, 15*xx*).

Launcher debug 11xx

Message ID
Message

DTXLN-D-1101
Launcher debug log enabled

DTXLN-D-1102
Launcher debug log disabled

DTXLN-D-1103
Starting Launcher <*pid*>

DTXLN-D-1104
Loading MSL <*msl_name*>

DTXLN-D-1105
Before init Resource Manager

DTXLN-D-1106
After init Resource Manager

DTXLN-D-1107
Before init Connection Manager

DTXLN-D-1108
After init Connection Manager

DTXLN-D-1109
Init socket <*socket_type*>

DTXLN-D-1110
Map Start <map_instance_number>

DTXLN-D-1111
Map Start <map_instance_number>

DTXLN-D-1112
Before resource pending

DTXLN-D-1113
After resource pending

DTXLN-D-1114
Before adapter call <*adapter*>

Resource Manager debug 13xx

Message ID
Message

DTX-RM-D-1301
Resource Manager debug log enabled

DTX-RM-D-1302
Resource Manager debug log disabled

DTX-RM-D-1303
PID list exists

DTX-RM-D-1304
PID list doesn't exist

DTX-RM-D-1305
SMBASE <*address + length*>

DTX-RM-D-1306
PRMH <*address + length*>

DTX-RM-D-1307
SIG <*address + length*>

DTX-RM-D-1308
PRMT <*address + length*>

DTX-RM-D-1309
PCT <*address + length*>

DTX-RM-D-1310
PFTG <*address + length*>

DTX-RM-D-1311
PNBB <*address + length*>

DTX-RM-D-1312
RMAddrunentry in shutdown

DTX-RM-D-1313
Thread enabled due to shutdown

DTX-RM-D-1314
firstRM is null when count >1

DTX-RM-D-1315
RMremoverunentry in shutdown

Connection Manager debug 15xx

Message ID
Message

DTX-CM-D-1501
Connection Manager debug log enabled

DTX-CM-D-1502
Connection Manager debug log disabled

DTX-CM-D-1503
Queue manager: <*message*>

DTX-CM-D-1504
Session manager: <*message*>

DTX-CM-D-1505
Rmcheckadapter object <*adapter*>

DTX-CM-D-1506
OpenConnection <*initial_number*>

DTX-CM-D-1507
OpenConnection - new conn opened

DTX-CM-D-1508
OpenConnection - existing conn refused

DTX-CM-D-1509
OpenConnection - <*final_number*>

DTX-CM-D-1510
OpenConnection - GTXOPEN

DTX-CM-D-1511
CloseConnection <*number*>

DTX-CM-D-1512
CloseConnection - GTXCLOSE

DTX-CM-D-1513
FireEvent - elapsed <*value*>

DTX-CM-D-1514
FindEvent stats <*value*>

DTX-CM-D-1515
RMListenerThread - waiting <*value*>

DTX-CM-D-1516
RMListenerThread - waited <*value*>

DTX-CM-D-1517
RMListenerThread - listen/msgs <*values*>

DTX-CM-D-1518
RMListenerThread - found <*value*>

DTX-CM-D-1519
RMListenerThread - elapsed stats <*value*>

DTX-CM-D-1520
RMListenerThread - stats2 <*values*>

Logging messages for Info

This documentation presents the logging messages for the Info logging category for each logging entry (Launcher, Resource Manager, Conection Manager) listed by each message category (21*xx*, 23*xx*, 25*xx*).

Launcher info 21xx

Message ID
Message

DTXLN-I-2101
Launcher info log enabled

DTXLN-I-2102
Launcher info log disabled

DTXLN-I-2103
Starting Launcher <*version*>

DTXLN-I-2104
Command Line enterted <*value*>

DTXLN-I-2105
INI settings <*value*>

DTXLN-I-2106
Ports <*value*>

DTXLN-I-2107
Gethostname <*value*>

DTXLN-I-2108
Resource config file <*value*>

DTXLN-I-2109
MSL Name - version <*value*>

DTXLN-I-2110
MMC Name <*mapname*>

DTXLN-I-2111
MMC Sources / targets <*value*>

DTXLN-I-2112
MMC settings <*value*>

DTXLN-I-2113
Watch settings <*value*>

DTXLN-I-2114
Adatper trigger found (init) -- <*value*>

DTXLN-I-2115
M4file journaling enabled

DTXLN-I-2116
Socket opened <*type*>

DTXLN-I-2117
Missing input for multi-source event <*value*>

DTXLN-I-2118
Got all triggers -- missing time event

DTXLN-I-2119
Got all triggers -- single threaded

DTXLN-I-2120
Mapping thread created

DTXLN-I-2121
Map Start <*value*>

DTXLN-I-2122
Map End / outcome <*values*>

DTXLN-I-2123
Hi-Lo pause

DTXLN-I-2124
Hi-Lo continue

DTXLN-I-2125
Shutdown Launcher

DTXLN-I-2126
Pause Launcher

DTXLN-I-2127
Continue Launcher

DTXLN-I-2128
Got STOP

DTXLN-I-2129
Hit maximum mapping thread

DTXLN-I-2130
Status socket accepted

DTXLN-I-2131
Status socket closed

DTXLN-I-2132
Monitor socket accepted

DTXLN-I-2133
Monitor socket closed

DTXLN-I-2134
Control socket accepted

DTXLN-I-2135
Control socket closed

DTXLN-I-2136
Control socket command <*values*>

DTXLN-I-2137
InitPendingHigh met <*value*>

DTXLN-I-2138
InitPendingLow met <*value*>

DTXLN-I-2139
Terminating - last message to log

Resource Manager info 23xx

Message ID
Message

DTX-RM-I-2301
Resource Manager info log enabled

DTX-RM-I-2302
: Resource Manager info log disabled

DTX-RM-I-2303
: INI settings <*value*>

DTX-RM-I-2304
: Table size <*value*>

DTX-RM-I-2305
: Central logging enabled

DTX-RM-I-2306
: Local logging enabled

DTX-RM-I-2307
: SHM use count

DTX-RM-I-2308
: Launcher use count

DTX-RM-I-2309
: Pointer size <*value*>

DTX-RM-I-2310
: Deadlock avoided - addrmentry

DTX-RM-I-2311
: RMTABLE entries/size <*value*>

DTX-RM-I-2312
: COMMANDTABLE entries/size <*value*>

DTX-RM-I-2313
: FREETOGO entries/size <*value*>

DTX-RM-I-2314
: NBB size <*value*>

DTX-RM-I-2315
: Average command line size

DTX-RM-I-2316
: FTG resume in rmworker thread

DTX-RM-I-2317
: RMrealloc called - start <*value*>

DTX-RM-I-2318
: RMrealloc called - end <*value*>

DTX-RM-I-2319
: Entering RMIDestroy

DTX-RM-I-2320
: Leaving RMIDestroy

Connection Manager info 25xx

Message ID
: **Message**

DTX-CM-I-2501
: Connection Manager info log enabled

DTX-CM-I-2502
: Connection Manager info log disabled

DTX-CM-I-2503
INI settings

DTX-CM-I-2504
Destroy - waiting for threads

DTX-CM-I-2505
OpenConnections - shutdown time

DTX-CM-I-2506
RMIListenerThread - pause

DTX-CM-I-2507
RMIListenerThread - continue

DTX-CM-I-2508
RMIListenerThread - stop

DTX-CM-I-2509
RMIListenerThread - start

Logging messages for Warning

This documentation presents the logging messages for the Warning logging category for each logging entry (Launcher, Resource Manager, Conection Manager) listed by each message category (31*xx*, 33*xx*, 35*xx*).

Launcher warning 31xx

Message ID
Message

DTXLN-W-3101
Launcher warning log enabled

DTXLN-W-3102
Launcher warning log disabled

DTXLN-W-3103
Map Warning <*mapname* / *warning*>

DTXLN-W-3104
Socket disconnect <*socket*>

DTXLN-W-3105
Unable to load resource config file

Resource Manager warning 33xx

Message ID
Message

DTX-RM-W-3301
Resource Manager warning log enabled

DTX-RM-W-3302
Resource Manager warning log disabled

DTX-RM-W-3303
Failed to get SHM on create

DTX-RM-W-3304
Failed to get SEM on create

DTX-RM-W-3305
Create mismatch

DTX-RM-W-3306
Table size requested vs. allocated <*values*>

DTX-RM-W-3307
Reached ten second lock

DTX-RM-W-3308
RUDumpTable memory failure

Connection Manager warning 35xx

Message ID
Message

DTX-CM-W-3501
Connection Manager warning log enabled

DTX-CM-W-3502
Connection Manager warning log disabled

DTX-CM-W-3503
DestroyTable - table already destroyed

DTX-CM-W-3504
DestroyTable - table in use

DTX-CM-W-3505
OpenConnection - smcomparerm fail

DTX-CM-W-3506
OpenConnection - all connections busy

DTX-CM-W-3507
OpenConnection - couldn't open conn

DTX-CM-W-3508
CloseConnection - invalid conn type

DTX-CM-W-3509
CloseConnection - uninitialized conn

DTX-CM-W-3510
Adapter GET returns bad conn

DTX-CM-W-3511
Failed to drop bad conn (mgetputres)

Logging messages for Error

This documentation presents the logging messages for the Error logging category for each logging entry (Launcher, Resource Manager, Conection Manager) listed by each message category (41*xx*, 43*xx*, 45*xx*).

Launcher error 41xx

Message ID
Message

DTXLN-E-4101
Launcher error log enabled

DTXLN-E-4102
Launcher error log disabled

DTXLN-E-4103
Map error <*map /erro*r>

DTXLN-E-4104
Connection error <*value*>

DTXLN-E-4105
Socket error <*value*>

DTXLN-E-4106
Alloc thread param failure

DTXLN-E-4107
Rmaddpendingentry failure

DTXLN-E-4108
Linkpendingthread error

DTXLN-E-4109
Resolvewhildcards memory failure

DTXLN-E-4110
SimulateMapRun <*values*>

DTXLN-E-4111
Linkpendingthread memory failure

DTXLN-E-4112
Copythreadparam memory failure

DTXLN-E-4113
Initsocketstuff memory failure

DTXLN-E-4114
Socket send failure

DTXLN-E-4115
Socket recv failure

DTXLN-E-4116
PDH error (WIN32 only)

DTXLN-E-4117
Can’t get exit function

Resource Manager error 43xx

Message ID
Message

DTX-RM-E-4301
Resource Manager error log enabled

DTX-RM-E-4302
Resource Manager error log disabled

DTX-RM-E-4303
Lock failed

DTX-RM-E-4304
Unlock failed

DTX-RM-E-4305
Enumprocess failed (WIN32 only)

DTX-RM-E-4306
Reached max pids

DTX-RM-E-4307
Rmaccessfile failure

DTX-RM-E-4308
Rmrealloctable failed

DTX-RM-E-4309
GetAdptMaskedCmd failure

DTX-RM-E-4310
Linkrmt returns NULL

DTX-RM-E-4311
Linkct returns NULL

DTX-RM-E-4312
Linknbb returns NULL

DTX-RM-E-4313
Rmaddrunentry failure

DTX-RM-E-4314
Suspend thread failure

DTX-RM-E-4315
FTG link error

DTX-RM-E-4316
NULL CT in rmgetdetails

DTX-RM-E-4317
NULL NBB in rmgetdetails

Connection Manager error 45xx

Message ID
Message

DTX-CM-E-4501
Connection Manager error log enabled

DTX-CM-E-4502
Connection Manager error log disabled

DTX-CM-E-4503
Invalid thread - can’t be suspended <*value*>

DTX-CM-E-4504
Suspend thread failed <*value*>

DTX-CM-E-4505
Invalid thread - can’t be resumed <*value*>

DTX-CM-E-4506
Thread - not suspended <*value*>

DTX-CM-E-4507
SAM error <*values*>

DTX-CM-E-4508
InvokeAdpaterMethod System exception

DTX-CM-E-4509
Invalid rmthread signature

DTX-CM-E-4510
Rmthread --> post call restore failed

DTX-CM-E-4511
Attempt to use destroyed adpt object

DTX-CM-E-4512
Can't initialize conn. Object

DTX-CM-E-4513
Illegal GTX TLS conn. cnt # <*value*>

DTX-CM-E-4514
Attempt to destroy adpt obj in use

DTX-CM-E-4515
Rmidleboot init failure

DTX-CM-E-4516
Illegal cleanup call - context missing <*value*>

DTX-CM-E-4517
Illegal GTX cleanup call - context missing <*value*>

DTX-CM-E-4518
Failed to allocate memory - rmgetputres

DTX-CM-E-4519
Invalid source event

DTX-CM-E-4520
Malloc failed GTX TLS init

Logging messages for Fatal

This documentation presents the logging messages for the Fatal logging category for each logging entry (Launcher, Resource Manager, Conection Manager) listed by each message category (51*xx*, 53*xx*, 55*xx*).

Launcher fatal 51xx

Message ID
Message

DTXLN-F-5101
Launcher error log enabled

DTXLN-F-5102
Launcher error log disabled

DTXLN-F-5103
MSL Mismatch <*value*>

DTXLN-F-5104
Invalid MMC <*mmc_name*>

DTXLN-F-5105
Cardinfo failure <*mapname*>

DTXLN-F-5106
Mpicreateallobjects failed <*mapname*>

DTXLN-F-5107
Init adapters failure <*error_number*>

DTXLN-F-5108
Resource manager init failure <*value*>

DTXLN-F-5109
Connection manager init failure <*value*>

DTXLN-F-5110
Socket failure <*error*>

DTXLN-F-5111
Thread create failure <*error*>

DTXLN-F-5112
Memory allocation failure <*error*>

DTXLN-F-5113
Readlauncherfile failed <*error*>

DTXLN-F-5114
Loadtriggers failed <*error*>

DTXLN-F-5115
No MSL files

Resource Manager fatal 53xx

Message ID
Message

DTX-RM-F-5301
Resource Manager error log enabled

DTX-RM-F-5302
Resource Manager error log disabled

DTX-RM-F-5303
RMworker thread create failed <*error*>

DTX-RM-F-5304
Semaphore create failed <*error*>

DTX-RM-F-5305
Shared memory/table conn failed <*error*>

Connection Manager fatal 55xx

Message ID
Message

DTX-CM-F-5501
Connection Manager error log enabled

DTX-CM-F-5502
Connection Manager error log disabled

DTX-CM-F-5503
Rmloadadapter failure <*error*>

DTX-CM-F-5504
Rminittriggers failure <*error*>

Management Console statistics

The statistical information displayed in the Management Console is refreshed periodically based on the refresh interval that you set when creating a new Launcher. Statistical information is derived directly from the process that the Launcher service is running.

The Management Console displays summary, status, historical, and configuration information. Each type is described here.

Summary statistics

From the Management Console, statistical data is displayed on the **Summary** tab in the main window of the Management Console. The statistics are grouped into the following three areas by default: **Performance**, **Status**, and **Historical**.

Performance

Statistic
: **Description**

Memory Usage
: Current amount of memory usage.

  (Supported on Windows platforms only.)

CPU Usage
: Current percent of total CPU usage.

  (Supported on Windows platforms only.)

Status

Statistic
: **Description**

System Status
: Status of the Launcher process (**.msl** system file). Options include: Starting, Running, Stopping, Stopped, and Paused.

Active Component Maps
: Current number of component map instances running

Active Listeners Up
: Number of listeners running

Active Listeners Down
: Number of listeners that cannot connect

Active Connections
: Number of active connections

Start Time
: Time that the Launcher started

Success Time
: Cumulative amount of time for maps, which are successful

Failure Time
: Cumulative amount of time for maps that failed

Up Time
: Total time that the Launcher has been running

Pending Initialization
: Number of maps currently waiting to begin execution

Pending Initialization Maximum
: Maximum number of maps that have waited to begin execution at any one time

Pending Resource
Number of maps currently waiting for resources, such as a shared data target

Pending Connection
Number of maps waiting to connect

Pending Total Maps
Sum of Pending Initialization, Pending Resource, and Pending Connect

Historical

Statistic
Description

History Successes
Number of maps that have succeeded since the Launcher started

History Failures
Number of maps that have failed since the Launcher started

History Total Maps
Total number of all maps that have been executed

History Connection Failures
Total number of connections that failed

History Deadlocks Detected
Number of map failures resulting from deadlocks

History Function Failures
Number of failures in `RUN`, `GET`, `PUT`, `DBLOOKUP`, and `DBQUERY`

Status statistics

From the **Status** tab, you can view information about active components, pending components, and adapter connections.

Active Component tab

Column
Description

Component
Component name

Map Name of compiled map file (**.mmc**)

System
System name

State The state of the Launcher. Options include:

- getting input
- validating input
- building input
- running map
- `GET/PUT` adapter
- ending output

Card The card number and I/O indicator

Resource
Adapter command used for this card

Pending

Column
Description

Component
Component name

Map Name of compiled map file (**.mmc**)

System
System name

State Initialization, connect, or resource pending

Card For connect and resource pending state, which card is pending

Reason
For resource pending state, reason why card is pending

Resource Type
Type of resource or adapter alias for this card

Resource Name
Name of resource (adapter command) for this card

Adapter connections

Column
Description

Adapter
Adapter type, such as database or FTP. Default represents the global default settings.

Open Number of connections currently open

Active Number of open connections being used in a map

Idle Number of open connections not in use

Pending
Number of pending connection requests

Historical information

From the **Historical** tab, you can view information about:

- map failures (″Map Failures″)
- adapter connections (″Adapter Connections″)
- specific map function failures (″Map Function Failures″)

The number of rows of data varies dynamically.

Map failures

Column
Description

Component
Component name

Map Name of compiled map file (**.mmc**)

System
Name of system containing map failure

Card Card number and I/O indicator that caused the error

Card Error
Return code and error message from adapter

Map Error
Return code and error message from map

Adapter connections

Column
Description

Adapter
Adapter type, such as database or FTP. Default represents global default settings.

Requests
Number of connection requests

New Number of requests that required a new connection

Reused
The number of requests that reused an existing connection

Successes
Number of successful operations (`GET` or `PUT`) on the adapter

Failures
Number of failing operations (`GET` or `PUT`) on the adapter

Map function failures

Column
Description

Component
Name of the map component where the function is contained

Function Name
Name of function that failed

Argument
Map name for `RUN`; adapter name for `GET`, `PUT`, `DBLOOKUP`, and `DBQUERY`

Reason
Reason for function failure

Configuration information

From the **Configuration** tab, you can view global configuration parameters used by the Launcher.

System tab

Column
Description

Maximum Concurrent Maps
Maximum number of maps that can run at the same time

Maximum Concurrent Maps Per Watch
Maximum number of maps running at the same time for each watch

Pending Initialization High
A user-defined static amount that serves as the threshold for the number of adapter listeners and time events pending initialization. After this amount is reached, maps can no longer trigger until the Pending Initialization Low amount is reached.

Pending Initialization Low
After the **Pending Initialization High** amount is reached, the number of items pending must reach this amount before maps can begin triggering again.

Adapter connections tab

Column
Description

Adapter
Adapter type, such as database or FTP. Default represents global default settings.

Idle Time
The period of time after which an idle connection is expected to be disconnected

Keep Time
How long an idle connection is expected to remain valid

Keep Minimum
The minimum number of connections allowed

Advisory Limit
The maximum number of connections, but can be exceeded if necessary

Mandatory Limit
The maximum number of connections allowed

Logging option

If you select the logging option when you have configured the Launcher to run multiple systems as a compound system in a single Launcher process or as separate Launcher processes and then run the system or systems, there are two log files that are created in the **logs** subdirectory of the ***install_dir***. Both log files have the same name as the system (**.msl**) file with the current date/time stamp added on (date/time format *mm-dd-yy-hh-mm-ss-am*).

The first log file has a **.log** filename extension.

The second log file has a **.txt** filename extension.

See ″Trace and Log Files Naming Conventions″ for the specific naming conventions.

Example

This is an example of file names generated from running a system with logging enabled.

Run a system where the system file path is ***install_dir*\systems\ small.msl** and logging has been enabled. The date is May 29, 2004 and the time is 10:31:11 AM. The following two files are generated in the **logs** subdirectory of the ***install_dir:***

- **small-05-29-04-10-31-11-am.log**
- **small-05-29-04-10-31-11-am.txt**

With logging enabled, these files are generated with the current date/time stamp every time the system is started.

Log file types

Using the logging option produces files that contain information falling under the following log file types:

- ″Launcher Activity Log Files″
- ″Debug Log Files″
- ″Diagnostic Log Files″

Launcher activity log files

The text (**.txt**) Launcher log file produced from the Launcher contains information about Launcher activities including summary and detail information that is useful for analysis, debugging and troubleshooting. Based on default parameters, the log file contains information about:

- Launcher configuration
- Launcher startup
- Map warnings
- Map errors

The default location for the Launcher log file is in the **logs** subdirectory of the ***install_dir***. As described in an example (see ″Example″), the file name is based on the system name and current date.

For more information, see the following documentation:

- ″Trace and Log Files Naming Conventions″ for the specific naming conventions
- ″LauncherLog″

Debug log files

The text (**.txt**) debug log file produced from the Launcher contains information about Launcher activities that is helpful for analysis, debugging and troubleshooting. Based on the default parameters, this file contains information about:

- Watch triggers
- Time and input events
- Error messages

The text debug log file is also generated in the **logs** subdirectory of the ***install_dir*** and generates upon startup. As described in ″Logging Option″ , the file name is based on the system name and current date.

For more information, see the following documentation:

- ″Trace and Log Files Naming Conventions″ for the specific naming conventions

- ″LauncherLog″

Diagnostic log files

Additional files are created when certain events occur with the Launcher:

- If an unhandled system exception occurs in a map thread that is running, troubleshooting information for technical support purposes is written to a file named **GPF*ThreadId-TimeStamp*.txt** on Windows and **ABORT-*pid*.txt** on UNIX. You can find this diagnostic log file in the installation directory of WebSphere Transformation Extender.
- The **dtxAbort.log** file is a UNIX environment-specific log file that indicates when the Launcher, Command Server, Platform API, or Java API is attempting to acquire memory from the operating system but does not succeed.

  The **dtxAbort.log** file gets created in $DTX_TMP_DIR when the respective application is started. $DTX_TMP_DIR is normally empty, which indicates that there has not been a failed attempt to get additional memory.

  If a failure occurs while attempting to obtain additional memory, the message appended to the **dtxAbort.log** file will be similar to one of the following messages:

  – `Fatal Error: malloc() - Unable to Retrieve Additional New Memory.`

  – `Fatal Error: realloc() - Unable to Retrieve Additional New Memory.`

    The message is prefaced with text indicating the date and time that the failure occurred.

Environmental debug (UNIX)

To turn on the environmental debug option for UNIX Launchers, you must export the DTX_DEBUG variable to **TRUE** before starting the Launcher (using the start command: `launcher.sh -start`). The environmental debug option will not work unless this action is performed. (See ″Environmental Debug″ for instructions.)

Map initiation states

To be initiated, a map might require multiple events to occur and multiple resources to be available. The Launcher coordinates these events and resources for maps based on the Launcher settings established during the system creation process in the IFD.

A map is ready to run when all its triggers have occurred and all required resources known to the Launcher are available. After the map is initiated, if a source, target, work file, map file, or log file exists but is in use, the retry values set in **Source** → **Retry** and **Target** → **Retry** are used to determine whether to continue or complete the processing of that map instance. When a retry option is being used because a shared resource is not available, the Launcher places the map in a resource or I/O pending state until those resources are available or the retry count has been exceeded.

Initiation pending state

If the map associated with a watch has multiple triggers and at least one trigger occurs, the map is placed in an initiation pending state until all triggers occur. Maps in an initiation pending state are reevaluated each time another trigger occurs.

Resource pending state

The resource pending state refers to source and target resources shared by other map instances known to the Launcher. If you are using the Launcher Status, the resource pending status count is non-zero when all triggers for a map have occurred but the resource manager has not yet released that map for initiation. In this case, at least one source or target updates or replaces a resource used by a map in process.

I/O pending state

The input/output (I/O) pending state accounts for Launcher system resources and resources shared by applications unknown to the Launcher. When viewing status using the Launcher Status tool, the I/O pending status message may display when a map has been initiated but the Launcher has found at least one source or target file, one work file, map file, or log file cannot be accessed because it is being updated by an application unknown to the Launcher.

If you are monitoring the status of the Launcher, you might see some map instances running longer than others with similar size data sources. This condition might occur when the Launcher places a map in an I/O pending state. For example, whenever the Launcher attempts to create or open a file that is already open for write, the map instance attempting to open that file will be placed in an I/O pending state.

Command line options

You can pause, resume, and detect the status of a Launcher from the command line on both Windows and UNIX platforms. This allows for automation in situations such as using a batch file.

Net pause and net continue commands

On Windows, the Launcher is controlled by the Launcher service. ″Pausing″ entails pausing all Launchers. ″Resuming″ entails resuming or continuing any paused Launchers. The net pause and net continue commands interact with the Launcher service to pause and resume the Launcher.

The Windows net pause and net continue commands are used in the following ways:

```
net pause ServiceName
net continue ServiceName
```

where *ServiceName* is the case-sensitive name of the service that is being controlled, such as ″ *WebSphere Transformation Extender* Launcher″.

To use the Net command options, the service must be started, otherwise an error message is displayed at the command prompt: **The service has not been started**. The command line will be blocked until the service pauses or resumes.

Pause and **Resume** actions can also be initiated from the Windows Computer Management console. The service must be started; otherwise the **Pause** option will not be enabled. If the service is not in a paused state, the **Continue** option is not enabled.

The following table lists the Net command-line options. Note the following rules that need to be observed when specifying service names on the command line:

- The service name must be enclosed in double quotation marks if it contains spaces.
- Service names are case-sensitive.

Command
Description

net pause ″ *WebSphere Transformation Extender* **Launcher**″
Pauses the Launcher service that is running

net continue ″ *WebSphere Transformation Extender* **Launcher**″
Resumes the paused Launcher service

The return values for the Net commands are zero for success and non-zero for failure.

Status command

The `-status` command is available on both Windows and UNIX platforms. This command retrieves the status of a specific Launcher.

On UNIX platforms, the Launcher is controlled by **launcher.sh** and therefore the `-status` option is used with **launcher.sh**. On Windows platforms, the `-status` option is used with **launcher.bat**.

The Launcher name provided to the `-status` option must be the name of the Launcher as configured in the Management Console (″Management Console″).

The following table lists `-status` command usage for each platform.

Platform	Command	Description
Windows	`launcher.bat -status` *LauncherName*	Retrieves the status of a specific Launcher and displays a status message on the screen, providing a numeric status return code
UNIX	`launcher.sh -status` *LauncherName*	Retrieves the status of a specific Launcher and displays a status message on the screen, providing a numeric status return code

Return codes for the status option

The following list gives a description of the possible status codes returned from the `-status` command.

Code	**Description**
0	Stopped
1	Stopping
2	Pausing
3	Paused
4	Started
5	Starting

6 Not responding

7 Unknown Error

101 The Launcher (with the specific settings) is not started.

102 The name of the Launcher supplied on the command line is not valid.

103 The user name, password, or both user name and password are not valid.

104 The connection between this utility and the Launcher is lost when trying to retrieve the status. For example, if another user stops the Launcher.

105 The command line supplied is invalid.

106 Launcher is not in a running state.

107 The name of the system (`systemname`) supplied on the command line is invalid, or it is blank and there is no compoundsystem present under the Java Launcher.

108 The Launcher raised an exception.

109 Unable to connect to the Launcher.

Snapshot commands

Additional Snapshot options are available using the Windows or UNIX command line. You can use these commands for automation with batch files.

The Snapshot options from the command line facilitate the capturing of Launcher information by enabling you to:

- Request and view the information in XML or text format.
- Redirect and store the information in a text or XML file.

Using Snapshot commands

The same Snapshot command line options are available for both Windows and UNIX platforms. For Windows, the associated file name is **launcher.bat**; for UNIX, it is **launcher.sh**. The following table lists the Snapshot command options.

Note: For the list of return codes and their descriptions, see ″Return Codes for System-Based Command Options″ .

Command
Description

-summary
This option retrieves summary information about:

- Memory and CPU Usage
- System Status
- Active Component Maps
- Active Listeners Up and Down
- Active Connections
- Start, Success, Failure, and Up Time
- Pending Initialization, Resource, Connection, and Total Maps
- History Successes, Failures, Total Maps, Connection Failures, Deadlocks Detected, and Function Failures

-history
This option retrieves history information about:

- Map Failures
- Adapter Connections
- Function Failures

-statusinfo
This option retrieves status information about:

- Active Components
- Pending Components
- Adapter Connections

-configuration
This option retrieves configuration information about:

- System
- Adapter Connections

-all This option retrieves all Launcher information; or is the equivalent to invoking the `-summary`, `-statusinfo`, `-history`, and `-configuration` commands together.

It also can be combined with any of the `-startsystem`, `-stopsystem`, `-pausesystem`, and `-resumesystem` options so that all the systems can be acted upon at one time. This is equivalent to using the `Start All`, `Stop All`, `Pause All` and `Resume All` menu items in the Management Console.

-xml This option enables you to retrieve and display Launcher information in XML format. If you specify this option, the output is printed in XML format. If this option is not specified, the output is printed in a readable text format.

When using a Snapshot command, in addition to specifying either the **launcher.sh** (UNIX) or **launcher.bat** (Windows) file, you must specify the Launcher name that was configured in the Management Console.

File storage commands

On Windows platforms, the following DOS redirection command can be used to store the summary information of the **CompoundSystem** (see ″Multiple Processes″ for additional information when using the multiple-processes feature) running under a Launcher named ″Server1″ in XML format:

```
launcher.bat -summary -xml "Server1" > MySnapshot.xml
```

On UNIX platforms, the following command line can be used to store the summary information of the **CompoundSystem** running under a Launcher named ″Server1″ in XML format:

```
launcher.sh -summary -xml "Server1" >  "/usr/home1/
snapshot.xml"
```

The following displays a sample of XML output file:

Display commands

On Windows platforms, the following command can be used to display all information about the **CompoundSystem** (see ″Multiple Processes″ for additional information when using the multiple-processes feature) running under a Launcher named ″Server1″ in a readable text format:

```
launcher.bat -all "Server1"
```

On UNIX platforms, the following command line can be used to display all information about the **CompoundSystem** running under a Launcher named ″Server1″ in a readable text format:

```
launcher.sh -all "Server1"
```

The following sample shows readable text output:

Return codes for system-based command options

For all Snapshot command line options, including `-summary`, `-statusinfo`, `-history`, and `-configuration`, and system-controlling commands, including `-startsystem`, `-stopsystem`, `-pausesystem`, and `-resumesystem`, an error message is displayed if the operation is not successful.

If the operation is successful, a code will be returned by the batch file as follows:

- `0` for Snapshot command line options
- `1` for system-controlling commands s.

Code **Description**

101 The specific Launcher is not started.

102 The name Launcher supplied on the command line is not valid.

103 The user name and password are not valid.

104 The connection between this utility and the Launcher is lost during the attempt to retrieve summary information. For example, this might happen if trying to find the summary information and another user stops the Launcher.

105 Command is invalid.

106 Launcher is not in a running state.

107 The name of the system (`systemname`) supplied on the command line is invalid, or it is blank and there is no compoundsystem present under the Java Launcher.

108 The Launcher raised an exception.

109 Unable to connect to the Launcher.

110 The control code and the system state are not compatible.

Note: This code is valid only for system-controlling commands.

111 The user does not have the proper user rights to control the system.

Note: This code is valid only for system-controlling commands.

Launcher Monitor

The Launcher Monitor provides graphical, detailed information about maps running in the Launcher. Using the Monitor, you can create, update, delete, and start Launcher connections for monitoring. Although you can set up multiple Launchers, you can only connect to and view one at a time.

The Launcher Monitor provides a dynamic view of watches as they run. A watch is a single map running. Watches are sequentially numbered starting with 1. When you start the Launcher Monitor and maps begin to run, a window for each watch appears within the Launcher Monitor window.

How the Monitor works

The Launcher Monitor works with the Launcher by means of Transmission Control Protocol/Internet Protocol (TCP/IP). Because this set of protocols allows for communications over interconnected, dissimilar networks, the Monitor can monitor Windows or UNIX Launchers.

There are two things are required for the interprocess communication (IPC) between the server and client:

- An address (or computer name)
- A port number

Using the Launcher Monitor

To use the Monitor to view Launcher processes (systems) running in the Launcher, a series of steps must first take place:

- One or more **.msl** files are generated and placed in the deployment directory that you specified in the Launcher Administration interface.
- The Launcher service/daemon is running.
- The Launcher is configured and a process is running in the Management Console.
- From the Launcher Monitor, you must configure the server and port number where the Launcher is running.

To open the Launcher Monitor

Select Launcher **Monitor** from the **Start** → **Programs** menu under **WebSphere Transformation Extender** *n.n* (where ***n.n*** represents the version number).

The Launcher Monitor window opens.

To start the Launcher Monitor

1. From the Launcher Monitor window, select **Options** → **Select** Launcher.

   The Select Launcher window appears.
2. Click **Add**.

   The Add Launcher window appears.
3. Enter a name for the Launcher and click **OK**.
4. Under **User**, enter the user name and password that corresponds to the server you need to connect to so that you can monitor a Launcher.
5. Under **Connection**, enter the name or IP address of the server where the Launcher is running in the **Server** field.

   Note: The **Server** connection is either obtained from network configuration information or can be a TCP/IP address. A TCP/IP address is a set of four 8-byte numbers (from 0 though 255); for example, **192.1.1.23**.
6. In the **Port** field, type the port number that you want to use or accept the default port number, which is **5016**.

   The port number is always the Launcher port number (configured in the Management Console) plus **1**. The default port for the Launcher is `5015`; therefore, the default port here is **5016**.See "Default Port Settings" for more information about default port settings.

7. Under **Timeout**, enter a value (in seconds) in the **Response** field. This value can range from 30 to 3600 seconds, which is the time that can elapse without an initial response from the Launcher before a timeout occurs. The default setting is 30 seconds.
8. Do one of the following activities:
   - To save this Launcher connection and close the Select Launcher window, click **OK**.
   - To connect to the Launcher and begin monitoring, click **Connect**.

     The Select System window opens.
9. Select **CompoundSystem** from the list (see ″Multiple Processes″ for additional information when using the multiple-processes feature) and click **OK**.
10. From the Select Launcher window, click **OK**.

    Each map that is running in the process is displayed in an individual window as a sequentially numbered watch.

Configuring the Monitor display

There are many options that you can use to configure the Monitor display according to your preferences. For example, you can use the following options to configure the Monitor display:
- Arrange the watch windows.
- Change update intervals.
- Change background and grid line colors.

Each option is described here.

Note: All procedures to configure the Monitor display begin from the Launcher Monitor window.

To arrange watch windows

You can arrange watch windows to display in the **Launcher**Monitor window.

Select **View** → **Tile**.

Watches

There are two options for configuring watches:
- Changing the interval
- Choosing specific watches

To change the interval for watch updates:

You can change the interval at which the watch windows in the Monitor are updated. Options include ¼ second, ½ second, and 1 second.
1. Select **Options** → **Update Interval**.
2. Choose the number of seconds: **¼**, **½**, or **1**.

To select specific watches:

You can select specific watches to view as well as the watch instance limit for the maximum number of watch instances.

Select **Options** → **Snapshot Settings**.

The Watch Settings window opens.

Adjusting colors

You can choose from nine different colors for the Monitor’s background, grid lines, and maps. Additionally, you can select a different color for each type of map status. For example, maps in progress, successful maps, maps with warnings, and maps with errors. All options are available from the colors menu.

Vertical Grid

There are two options for configuring the vertical grid:

- Changing the spacing
- Changing the time interval

To change vertical grid spacing:

You can change the vertical grid spacing by choosing the number of pixels per second. Choices include 10, 20, and 40 pixels.

1. Select **Options** → **One Second Equals**.
2. Choose the number of pixels: **10**, **20**, or **40**.

To change the vertical grid time interval:

You can change the time interval between vertical grid lines. Options include ½ second, 1 second, and 2 seconds.

1. Select **Options** → **Vertical Grid**.
2. Choose the number of seconds: **10**, **20**, or **40**.

Snapshots

From the Launcher Monitor, you can take snapshots of a watch running as well as configure how the snapshots are taken. Snapshots capture Launcher activity. The Snapshot Viewer enables you to select the type of snapshot to capture each time an error occurs, for any error or warning, or for all map instances. See ″Snapshot Viewer″ for more information about the Snapshot Viewer.

Snapshots on demand

You can take a snapshot at any time during processing.

To capture a snapshot

1. Click the **Snapshot!** menu command.
2. Open the Snapshot Viewer (″Snapshot Viewer″) to view detailed information about the snapshot.

Snapshot settings

The snapshot settings available in the Monitor enable you to configure how snapshots are taken. For example, you can select specific watches or set a maximum limit for instances. The documentation provides a description of all the options.

To access Snapshot settings from the Launcher Monitor

Select **Options** → **Snapshot Settings**.

The Snapshot Settings window opens.

Each setting is described in the following table.

Control
Description

Instance Limit
Maximum number of total watch instances to include in a single snapshot. The maximum number of watch instances to be included in one snapshot is 1000. Click the arrow buttons to change the limit.

Snapshot
Select an option to specify the type of snapshot:

On Demand Only - Snapshots are only taken when you choose the **Snapshot!** menu command.

On Error - A snapshot is taken automatically whenever a map error occurs.

On Warning/Error - A snapshot is taken automatically whenever a map error or warning occurs.

Continuous - A continuous snapshot is taken, including all watch instances specified by **Instance Limit**.

Include these watches
Select a check box next to each watch to include in a snapshot.

Snapshot Viewer

The Snapshot Viewer enables the analysis of snapshots taken of Launcher activity from the Launcher Monitor.

How are Snapshots created?

A snapshot of Launcher activity is saved from the Launcher Monitor by clicking **Snapshot!** from the menu bar.

Use the Snapshot Viewer to view and analyze the snapshots taken. Each snapshot contains information that is used to display a chart of map instances over time.

When a snapshot is created, it is written to a file with this format:

`MMDD`<*four-digit-number*>`.mss`

where `MMDD` represents the two-digit month and two-digit day on which the snapshot was created. The four-digit number begins with one and is increments by one for each snapshot for the current day.

The Snapshot file (**.mss**) is created in the WebSphere Transformation Extender installation directory by default.

To view a snapshot

This procedure assumes that you have already captured a snapshot from the Launcher Monitor.

1. From the **Start** → **Programs** menu, open the Snapshot Viewer under **WebSphere Transformation Extender** *n.n* (where ***n.n*** represents the version number).
2. Select **File** → **Open**.
3. Navigate to the **.mss** file, which is located in the WebSphere Transformation Extender installation directory, and open it.

   The snapshot from the Launcher Monitor appears.

Map instances are color-coded, so it is easy to understand a map's status at a glance. The default colors are:

- **Green**-Maps that completed successfully.
- **Yellow**-Maps that encountered a warning condition during execution.
- **Red**-Maps that did not complete because of an error condition.

From the **Colors** menu, you can change colors for map instances, background, grid, and text associated with a map instance.

To see detailed information about the map instance

Right-click over a map instance and hold to see detailed information about the execution for that instance of the map.

Chapter 8. Multiple processes

The WebSphere Transformation Extender Launcher has a multiple-processes feature.

This feature enables the Launcher to run multiple Launcher processes, each containing one or more systems (**.msl** files), within a single installation. It will run each system or group of systems in a separate Launcher process allowing you to add, remove and modify a Launcher process without affecting other running Launcher processes. Running the Launcher in separate Launcher processes also offers a more scalable and robust run-time environment. The WebSphere Transformation Extender Launcher can be configured to run using the **CompoundSystem** approach (all systems run in a single Launcher process), or to run each system (**.msl** file) or group of systems in multiple Launcher processes.

The documentation describes the different types of configurations available for running multiple processes within a single installation and how each configuration type affects system processing.

The WebSphere Transformation Extender Launcher multiple-processes feature is also available to use with the z/OS operating system on IBM z/Architecture platforms.

Note: If you will be running systems specifically through the z/OS native administration interface (NAI), the WebSphere Transformation Extender Launcher multiple-processes feature allows you to run multiple Launcher processes, each containing one or more systems (**.msl** files), within a single install of the Launcher as described in the″z/OS Launcher″ documentation.

Activities summary

To run multiple processes within a single install:

- change resource management settings through the Resource Manager section of the initialization file depending on the way in which the Launcher will be running the Launcher processes
- change settings through the various windows within the Launcher Administration to configure the Launcher to use this feature
- manage the Launcher process by:
  - starting the Java Launcher
  - controlling the Launcher and viewing the status of the system either through the Management Console or by using command line utilities
  - viewing map execution information and taking snapshots of Launcher activity through the Launcher Monitor
  - analyzing the contents of debug trace and log files

Installation

During WebSphere Transformation Extender Launcher installation in Windows environments, the setup program installs WebSphere Transformation Extender Launcher as a service with **Manual** as the default value for **Startup Type** and **LocalSystem** as the default value for **Log On As**. An example of the service name that you might see is **WebSphere TX n.n Launcher**.

See the release notes (**readme.txt**) file for more information about the installation.

Note: The Launcher run on UNIX and z/OS does not run under a service.

Using resource management

The Resource Manager is a subsystem of the Launcher that controls those resources that do not have inherent resource management such as databases. File adapters are the most common adapters that require resource management. You can change resource management settings, which will affect how the Launcher will run.

There are two types of resource management:

- local - used within a single Launcher process.
- global - used across all of those Launcher processes that have been configured to use this type of resource management.

Global resource management

Use global resource management, also referred to as central resource management, when running a system or group of systems in separate Launcher processes and the systems can share resources. The extra overhead of central resource management is minimal. The Resource Manager table requires an operating system-level lock rather than a process-level lock while accessing the Resource Manager. You should use central resource management if a system or group of systems running in separate Launcher processes start to have map errors that produce the error messages such as: `source not available` or `failed to put data on output`.

When running systems in multiple Launcher processes, central resource management can be turned on or off for a select Launcher process. Central resource management enables these different processes from the same installation to manage shared resources.

Launcher systems run options

The following list presents four data transformation scenarios and the various ways in which the Launcher can run a system or group of systems

- in a single Launcher process, as a Compound System
- in separate Launcher processes, each with localized resource management
- in separate Launcher processes, all using centralized resource management
- in separate Launcher processes, with a combination of centralized and local resource management

Using the Launcher Administration

Through the Launcher Administration application, you can change settings in the various windows within it to configure the Launcher to use the multiple-processes feature. The Launcher Administration allows for separate Launcher processes for each **.msl** system file or group of **.msl** system files.

Benefits include:

- More robust run-time environment where problems occurring in maps or adapter connections in one Launcher process will not cause problems in other Launcher processes.
- The ability to add and reconfigure Launcher processes without stopping other Launcher processes.
- The ability to use systems within systems for a single **.msl** Launcher system file for logical processes definitions.
- The ability to group systems to run in a separate Launcher process and start and stop them without having to shut down the Java Launcher.

General tab

The **General** tab of the Launcher Administration window has controls that you can set to enable the Launcher to use the multiple- processes feature.

The following list presents the controls, used with the multiple-process feature, along with their location in the Launcher Administration window:

- **Automatic startup**

  Enabling **Automatic startup** gives you the ability to start up a Launcher process automatically. For information about using this option, see ″Configuring Launcher Processes to Start Automatically″ .
- **Separate Launcher system processes**

  Indicates that the Launcher will be run in different Launcher processes and that one or more system files can be configured for each process.
- **Initialization File:**

  Identifies the initialization file, which contains configuration settings for WebSphere Transformation Extender.

  Default is *install_dir*\dtx.ini.
- **Status:**

  Displays the status of the Java Launcher.

Selecting separate Launcher processes

When selecting **Separate** Launcher **processes**, configuration for different initialization files can be set for each Launcher process. Also, checking both **Automatic startup** and **Separate Launcher processes** adds the **Automatic Start** property in the Properties window for each process. The **Automatic Start** property has two options:

- **Yes**
  - Enables the specific Launcher process on which the **Yes** option is set to start up automatically.
- **No**

- Prevents the specific Launcher process on which the **No** option is set from starting up automatically.

With the **Separate Launcher processes** checkbox not checked, the Launcher will run all systems as a Compound System.

Connection ports

Three consecutive ports for each separate Launcher process are required. The Launcher automatically assigns them based on the first port entered in the port range.

For information about the listening ports, see ″Connection Ports″.

Access tab - User Properties dialog

Through the **Access** tab of the Launcher Administration window, you can invoke the User Properties window to add new or edit existing users and access permissions for the Management Console and the Launcher Monitor by clicking **Add** or **Edit**.

After clicking **Add** or **Edit**, the User Properties window will appear with default access permission set to **ALL Systems** under the **Systems** column heading. You can change the permissions for **Start/Stop**, **Pause/Resume** and **Monitor** to **Grant**.

To change the permissions in the User Properties window

1. Click on the permissions under one of the permission category columns (**Systems**, **Start/Stop**, **Pause/Resume**, and **Monitor**) to expand the drop-down menu.
2. Select the permission you want from the drop-down list.
3. To apply the changes, click **OK**.
4. You can repeat these steps for any of the permission categories you want to change.

Deployment Directories tab

Checking the **Separate Launcher processes** checkbox enables the **Edit** button in the window that appears under the **Deployment Directories** tab in the Launcher Administration application.

When the **Edit** button is clicked, the File List window is invoked. The File List window is used for the following purposes:

- adding, configuring and removing individual Launcher processes.
  - "To add a Launcher process" on page 107
  - "To remove a Launcher process" on page 107
- adding and deleting individual **.msl** system files for a specific Launcher process.
  - "To add .msl system files to a Launcher process" on page 107
  - "To delete .msl system files from a Launcher process" on page 107

Note: The Launcher process that is listed in the File List window under the **Properties** tab is treated as if it were the *main* **.msl** system file; it must exist before others can be added and it cannot be deleted until all additional systems are deleted.

To invoke the File List window

Under the **Deployment Directories** tab in the Launcher Administration application, click the **Edit** button.

The File List window opens.

To add a Launcher process

1. Click **Add** in the view under the **Properties** tab.

   The Select window appears.
2. Click on the **.msl** system files you want to deploy.

   To select multiple files, click **Ctrl** and click on the files you want. Your selected files will be highlighted.
3. Click **OK**.

   The Select window closes and the File List window reappears with the selected files listed in the left pane.
4. Click **OK**.

   The File List window closes and the window under the **Deployment Directories** tab in the Launcher Administration application reappears.

To remove a Launcher process

1. Click in the left section of the window under **File List** on the Launcher process you want to remove.
2. Click **Remove** in the view under the **Properties** tab.

   The Launcher process is removed from the list.

To add .msl system files to a Launcher process

1. Click in the left section of the window under **File List** on the Launcher process to which you want to add **.msl** system files.
2. Click **Add Files...** in the **Select additional files...** view under the **Files...** tab.

   The Select window opens.
3. Click on the **.msl** system files you want to deploy in the specified Launcher process.

   To select multiple files, click **Ctrl** and click on the files you want. Your selected files will be highlighted.
4. Click **Select**.

   The Select window closes and the File List window reappears with the selected files listed in the main pane of the **Select additional files...** view.

To delete .msl system files from a Launcher process

1. Click in the left section of the window under **File List** on the Launcher process from which you want to delete **.msl** system files.
2. Click in the main pane of the **Select additional files...** view on the specific **.msl** system file you want to delete.
3. Click **Delete files...** in the **Select additional files...** view under the **Files...** tab.

   The **.msl** system file is removed from the list.

Configuring Launcher processes in the File List dialog

When using separate Launcher processes, configure each of them in the File List window. There are two sections of the File List window. The top section of the File List window is the **Property (Directory)**. The bottom section is the **Property (File)**.

Top section of the File List dialog

The top section labeled **Property (Directory)** enables you to override the default values and configure settings for those specific Launcher processes (**.msl** files). It allows you to propagate the default settings to those Launcher processes (**.msl** files) within that directory.

To propagate the Property (Directory) default settings

1. Select the property under the **Property (Directory)**.
2. Right-click on the property.

   A menu appears.
3. Select **Propagate**.

   The default settings are copied to those Launcher processes (**.msl** files) within that directory.

Bottom section of the File List window

The bottom section labeled **Property (File)** enables you to configure the specific Launcher process to override the default settings.

The following table lists the Launcher process properties along with a description, and default and valid values.

Property (File)
  Description

Name Unique name that will be used for the Management Console and command line utilities to control the Launcher process individually.

Description
  Description for the Launcher process.

File Name
  Launcher system path and name.

Initialization File
  Valid WebSphere Transformation Extender initialization file.

  Default value is the global entry on the **General** tab of the Launcher Administration.

Resource Manager
  Configuration value indicating whether to use global or local resource management.

  Valid values are `Global` and `Local` selected from pull-down menu.

  Default value is `Local`.

  Global indicates that resource management will allow multiple Launcher processes to share common resources properly.

Resource Configuration
  Resource configuration file for this system.

  Default value is the entry on the **General** tab of the Launcher Administration.

Automatic Start
  This property row appears when **Automatic startup** is enabled on the

General tab of the Launcher Administration window. It allows you to start up this Launcher process automatically.

Valid values are **Yes** and **No**, which are selected from a pull-down menu.

Default value is **No**.

To apply your system configuration changes

Click **OK**.

The File List window closes and the window under the **Deployment Directories** tab in the Launcher Administration application reappears.

Configuring Launcher processes to start automatically

After you have configured the Launcher to run as multiple Launcher processes within a single installation, they are ready to be configured to start automatically. When **Separate Launcher processes** and **Automatic startup** are both selected on the **General** tab of the Launcher Administration, the **Automatic Start** property on the Properties window appears for each Launcher process under the **Properties** tab in the File List window.

To configure the Launcher processes to start automatically

1. Navigate to the File List window from the **Deployment Directories** tab in the Launcher Administration application.
2. Click in the left side of the window under **File List** on the specific configured Launcher process you want to start up automatically.
3. Select **Yes** from the pull-down menu of the **Automatic Start** property.

   Note: You can select **Automatic startup** if **Separate Launcher processes** is not selected. In this case, the **CompoundSystem** will automatically start when the Java Launcher starts.
4. Click **OK**.

The Launcher processes have been configured to start automatically. After they are started, while the Java Launcher is running, you can modify them through the Launcher Administration application by following the instructions.

Modifying systems while the Launcher is running

While the Java Launcher is running, you can make changes to Launcher system files (**.msl**) that are in separate Launcher processes in the Launcher Administration application. You can also add and remove Launcher processes.

Note: Before you can remove a Launcher process, you must first stop its execution.

After the Launcher processes are started, while the Java Launcher is running, the Launcher Administration application allows you to make the modifications but also has some functionality restrictions; some of the options are disabled and grayed out. They are presented in the following topics:

- ″Changing File Configurations″
- ″Modifying Deployment Directories″
- ″Modifying Administration User Access″

Changing file configurations

While the Java Launcher is running, you can change file configurations. The detailed procedures are described in the ″General Tab″ documentation.

- Cannot Change Startup, Execution and Connection Settings
  - Under the **General** tab, the **Startup**, **Execution** and **Connection** sections are disabled.
- Changing Initialization File Configuration Settings
  - You can make changes to the initialization file in the **Configuration** section.
  - The **Apply** button is enabled. When you click **Apply**, changes will take effect after the Java Launcher is restarted.
  - When the Java Launcher is started, you will see the **Running** status of the Launcher in the Launcher section of the window under the **General** tab.

Modifying deployment directories

While the Java Launcher is running, you can modify deployment directories. The detailed procedures are described in ″Deployment Directories Tab″ .

- Cannot Remove Deployment Directories
  - Under the **Deployment Directories** tab, the **Remove** button is disabled and grayed out so you are unable to remove deployment directories while the Java Launcher is running.
- Removing a Launcher Process
  - Under the **Deployment Directories** tab in the File List window invoked by clicking the **Edit** button, the **Remove** button is disabled and grayed out if the specific Launcher is running, and enabled if the specific Launcher has been stopped. (To remove a Launcher process, it must first be stopped.)
- Adding Deployment Directories
  - However you can still add new deployment directories under the **Deployment Directories** tab by clicking the **Add** button.
- Adding Launcher Processes
  - And you can still add new Launcher processes in the File List window by clicking the **Add** button. (The File List window is invoked by clicking the **Edit** button.)
  - Modification of a Launcher process that is currently running will take effect after the Launcher is restarted.

Modifying administration user access

While the Java Launcher is running, you can modify administration user access. The detailed procedures are described in ″Access Tab - User Properties window″ .

- Adding or Editing Administration User Access
  - Under the **Access** tab, you can add new or edit existing Administration users.
  - The **Edit** button settings will take effect after the Java Launcher is restarted.
- Cannot Remove Administration User Access
  - Under the **Access** tab, the **Remove** button is disabled.

Configuring the initialization file

Whether using the single process **CompoundSystem** or separate Launcher processes comprising one or more **.msl** system files, you can configure the Launcher initialization file. You can rename it (the default name is **dtx.ini**) as well as make modifications to its contents.

Some of the changes you can make to the initialization file include modifications to the following resource manager settings in the **[Resource Manager]** section:

Resource Management Settings
Description

CentralMgmt
Valid values are:
0, local resource management
1, global (centralized) resource management

Default is 0.

TableSize
Size in megabytes for the Resource Manager table.

Default is 10.

If the Launcher Administration is set to run systems in separate Launcher processes and the resource manager setting for a Launcher process is set to Local, the CentralMgmt entry in the initialization file has precedence over those other settings. Setting CentralMgmt to 1 will enable global (centralized) resource management. The default is 0 or local resource management.

The TableSize entry is used for both local and global resource management. It determines the initial size of the Resource Manager table before having to re-size it if required. Re-sizing requires all Launchers using the table to re-calculate the entries and might fail if memory cannot be obtained. The value is in megabytes, and the minimum value is 1. The default table size is 10 (10 megabytes).

The following table correlates the size of the Resource Manager table to the number of entries that the table can hold.

Resource Manager Table Size (in megabytes)	Total Entries	Unique Entries (138 bytes average size)
1	5461	4369
5	27306	21845
10	54613	43690
25	136533	109226

Note: There is, at most, one entry for each initialize pending map and one entry for each card for each resource pending map during Launcher processing, depending on the type of resource (file, database, ftp, and so on) and whether it needs WebSphere Transformation Extender resource management.

When using global resource management, the initial Launcher process that creates the Resource Manager table determines the size for subsequent processes that use global resource management.

Managing the Launcher process

There are several activities you can do to manage the Launcher process. You can manage it by:

- starting the Java Launcher
- controlling the Launcher process and viewing the status either through the Management Console or by using command line utilities
- viewing map execution information and taking snapshots of Launcher process activity through the Launcher Monitor
- analyzing the contents of debug trace and log files

Starting the Java Launcher

The Java Launcher must be started prior to starting the Launcher processes that will run within the Management Console or using the **launcher.bat** (Windows) or **launcher.sh** (UNIX).

Monitoring systems through the Management Console

After WebSphere Transformation Extender has been installed and configured, and the Java Launcher started, the Management Console can be used to control and view the statistics about the Launcher processes (**.msl** system files) that are running.

You must log in to the Management Console and connect to the Java Launcher.

If **Separate Launcher processes** in the Launcher Administration was not checked,

and the connection was successful, you will see the **CompoundSystem**.

The Management Console is connected to the Java Launcher, which is running all Launcher systems (**.msl** files) in a single Launcher process, or CompoundSystem, or in one or more separate Launcher process.

If **Separate Launcher processes** in the Launcher Administration was checked,

and the connection was successful, you will see the unique names given to the individual Launcher processes (**.msl** files) from the File List window in the Launcher Administration. The Management Console is connected to the Java Launcher, which is running separate system Launcher processes for each Launcher system (**.msl** file) or group of systems.

In this configuration, you can control and monitor each of the different Launcher processes individually without affecting another.

Expand one of the Launcher system files to see the categories of information that you can view in the Management Console. They are **Status**, **History** and **Configuration**.

The **Summary** tab of the Management Console is an example of the statistical information for one of multiple Launcher processes.

Shell script and batch file (launcher.sh and launcher.bat)

The command line utilities that control the Launcher (**launcher.sh** on UNIX operating systems and **launcher.bat** on Windows operating systems) can be configured specifically for running systems in multiple Launcher processes.

When running systems in separate Launcher processes, you control each process with the **[systemname]** parameter, which is the unique name of the system file (**.msl**) as defined in the File List window when **Edit** is selected from the **Deployment Directories** tab in the Launcher Administration. Select **-all** for all configured Launcher processes.

UNIX shell script

The following code shows the UNIX launcher.sh command usage:

```
Usage: launcher.sh [-start] [-stop [listenerport [hostname]]]
                      [-pause [listenerport [timeout [hostname]]]]
                      [-resume [listenerport [timeout [hostname]]]]
                      [-status LauncherName [systemname]]
                      [<[-summary] [-history] [-statusinfo] [-configuration]
                        [-all] [-xml]> Launchername [systemname]]
                      [-startsystem Launchername <[systemname] [-all]>]
                      [-stopsystem Launchername <[systemname] [-all]>]
                      [-pausesystem Launchername <[systemname] [-all]>]
                      [-resumesystem Launchername <[systemname] [-all]>]
```

Windows batch file

The following code shows the Windows launcher.bat command usage:

```
Usage: launcher.bat [-status Launchername [systemname]]
                       [<[-summary] [-history] [-statusinfo] [-configuration] [-all]
 [-xml]> Launchername [systemname]]
 [-startsystem Launchername <[systemname] [-all]>]
 [-stopsystem Launchername <[systemname] [-all]>]
 [-pausesystem Launchername <[systemname] [-all]>]
 [-resumesystem Launchername <[systemname] [-all]>]
```

On Windows platforms, use the operating system’s `NET` command to start, stop, pause and continue the Java Event Service through the command line.

On UNIX platforms, use the first four *Usage* commands in the **launcher.sh** shell script, start, stop, pause and resume.

Using the Launcher Monitor

When running the Launcher with the **Separate Launcher processes** checkbox in the **General** tab of the Launcher Administration enabled for multiple Launcher processes, each Launcher process (**.msl** file) that is configured will be displayed in the Select System window.

Because the Launcher Monitor can only observe one Launcher process at a time, multiple Launcher Monitor instances are required to monitor multiple processes.

Debug trace and log files

The Launcher debug trace and log files are located in the **logs** directory under the ***install_dir***, which refers to the directory where your product is installed.

See ″Trace and Log Files Naming Conventions″ for the specific naming conventions.

Chapter 9. z/OS Launcher

This documentation includes detailed information about how to enable the Launcher execution on z/OS environments.

Overview

WebSphere Transformation Extender for z/OS is an IBM Corporation product developed to enable you to run multiple maps or systems on a z/OS operating system on the IBM z/Architecture platforms.

Note: To use the z/OS Launcher, you will need to deploy systems. Although the focus of this documentation is to present the details of the z/OS Launcher, you will still need to reference other documentation for the details pertaining to deploying systems.

There are minimal differences between the z/OS implementation, and the Windows and UNIX implementation.

Among the differences are the methods available to administer the Launcher operations. They are:

- Native administration interfaces (ISPF Panel and Operator's Console)
  - For native administration interfaces, reference the specific documentation.
- Existing user interfaces based on UNIX operating systems (Launcher Administration and Management Console)
  - For administration user interfaces based on UNIX operating systems, you can reference processes presented in ″Getting Started″.

    Note: The **DISPLAY** environment variable must be exported and an X Server must be available to host the user interface displays.

    Note: Reference ″Starting the Launcher Monitor″ for details on how to start the Window Launcher Monitor:

Another difference between the implementations is the capability of the z/OS implementation to access the native file system.

Referenced terms

Throughout the documentation, the following references are made:

Term	References
WebSphere Transformation Extender for z/OS	Includes Launcher
Command Server	Batch
CICS Execution Option	CICS

See ″Glossary″ for additional information about terminology used in this documentation.

Architecture

WebSphere Transformation Extender for z/OS runs in and uses both the UNIX System Services (USS) layer and the Hierarchical File System (HFS). It runs as a single process, multi-threaded server from within the USS layer of z/OS. It relies on TCP/IP sockets as the communications method under the TCP/IP communications protocol.

Utilities and tools

WebSphere Transformation Extender for z/OS includes JCL utilities and tools to transfer log, trace, and debug data from the HFS into the native file system. A UNIX System Services Command Server (USS Command Server) is provided as a tool to verify expected map operation prior to deployment in the Launcher.

Note: The USS Command Server is used for testing map behavior only. It is not a tool to be used for running maps in a production environment.

Note: When you are testing maps using the USS Command Server, if those maps were set up in the IFD with overrides, you need to use those override names directly in the maps.

JCL tools

The following tools are available in the DTX.SDTXSAMP PDS to assist in transferring data from the HFS into the native file system.

Member Name
  Description

DTXDHFS
  Defines, allocates, and mounts an HFS filespace.

  - Installation process defines a user ID, an HFS mount point, and ’HOME’ directory.

DTXH2PDS
  Copies the file from the HFS to a PDS or PDS/E member

DTXH2SF
  Copies the file from the HFS to a sequential file

By default, all diagnostic information is stored in the HFS in the following locations:

Default Location
  File Types in the HFS

$DTX_HOME_DIR/config/dtx.ini
  Contains configuration information, including diagnostic definitions

$DTX_HOME_DIR/dump
  Contains diagnostic information when failures are generated by a Launcher execution

$DTX_HOME_DIR/logs
  The logs from each of the user interfaces

$DTX_TMP_DIR
  Contains data generated during execution of a map or system

USS Command Server

The UNIX Systems Services (USS) Command Server allows you to verify a map's behavior prior to deploying it in a system in the Integration Flow Designer.

It is the same as the UNIX Command Server with the additional functionality:
- Access to native files (PDS and PDS/E membersand Sequential files).
- Ability to specify output file-creation options for native files using allocation parameters (″Allocation Parameters″).
- Ability to specify record terminators using the Record Terminators (/V) command option (″Record Terminators (/V)″).

Note: For more information and a comparison of the UNIX and Batch/CICS Command Servers, refer to the appropriate chapters in the *Command Server* documentation.

Note: It is recommended that you avoid running the USS Command Server and the Launcher at the same time if both servers are attempting to access the same resources.

Native file system

The native file system is used in processing the Launcher for z/OS. The Launcher supports the various native file system file types (″File Types″) in various contexts.

The subsequent documentation describe native file system:
- ″File Types″
- ″File Access″
- ″File Triggering″
- ″Map Names″

File types

The Launcher for z/OS supports the following native file system file types:
- Sequential files
- PDS and PDS/E members
- Virtual Storage Method (VSAM) files
- Generation Dataset Group (GDG) Files

Note: For information on these file types, see the specific version of the IBM *z/OS C/C++ Programming Guide* for your operating system release.

File access

There are different ways to access files in the native file system through specifications in map input/output.
- IFD specifications
  - ALLOC syntax
- Map server location
  - Describe location of ′related′ data files

Allocation

Native file system files can be specified in the following locations:

- as input or outputs on maps in your systems.
- as maps in your map server location setting.

Map names

The Launcher for z/OS supports native file system conventions for specifying the location of compiled maps on the execution server.

Compiled map name specifications

When specifying a native file system location for the compiled map, the Launcher for z/OS supports native file system formats for the following file types:

- Sequential files
- PDS and PDS/E members
- Virtual Storage Method (VSAM) files
- Generation Dataset Group (GDG) Files

Note: For formatting information about these file types, see the specific version of the IBM *z/OS C/C++ Programming Guide* for your operating system release.

Related data file locations

The location of data files for compiled maps is dependent on the option chosen at WebSphere Transformation Extender for z/OS map or system definition time.

The location of related data files (for example, log and trace files) for maps that are stored in the native file system is dependent on the options configured for those map in the Integration Flow Designer prior to deployment. The options are:

- DEFAULT (″DEFAULT″)
- CUSTOM (″CUSTOM″)
- UNIQUE (″UNIQUE″)

DEFAULT

The following table presents the location of related data files for compiled maps located in the native file system when the **DEFAULT** option was specified in the map settings:

Note: For files in the HFS, the default location is **$DTX_TMP_DIR**.

Data File in
Native File System
Default Location

Audit log
$DTX_TMP_DIR/MYUSERNAME.ANY/MAP.log

Input work file
$DTX_TMP_DIR/MYUSERNAME.ANY/MAP.Inn

Output work file
$DTX_TMP_DIR/MYUSERNAME.ANY/MAP.Onn

Trace file
$DTX_TMP_DIR/MYUSERNAME.ANY/MAP.mtr

CUSTOM

The following table presents the location of data files related to the compiled maps when the **CUSTOM** option was specified at map or system definition time:

The following table presents the location of related data files for compiled maps located in the native file system when the **CUSTOM** option was specified in the map settings:

Note: For files in the HFS, the location that was specified at map or system definition time is used.

Data File in Native File System
Default Location

Any file
If the file is in the native file system, the map will fail.

Note: Do not use the **CUSTOM** setting for maps in the native file system.

UNIQUE

The following table presents the location of data files related to the compiled maps when the **UNIQUE** option was specified at map or system definition time.

Note: For files in the HFS, the default location is **$DTX_TMP_DIR**.

Data File in Native File System
Default Location

Audit log
$DTX_TMP_DIR/MYUSERNAME.ANY/Mer_MAP_process-key_map-counter.log

Input work file
$DTX_TMP_DIR/MYUSERNAME.ANY/Mer_MAP_process-key_map-counter.Inn

Output work file
$DTX_TMP_DIR/MYUSERNAME.ANY/Mer_MAP_process-key_map-counter.Onn

Map sources and targets

For map source (″Map Sources″) and targets (″Map Targets″), the Launcher for z/OS supports native file system formats for the following file types:

- Sequential files
- PDS and PDS/E members
- Virtual Storage Method (VSAM) files
- Generation Dataset Group (GDG) FilesGeneration Dataset Group (GDG) Files

Note: Many type definitions use terminators such as `<CR><LF>`. If data is created locally on the z/OS environment or transferred to the z/OS environment using a file transfer protocol (FTP), the terminators, which would indicate that there is no more data in the record, are not used to terminate the

data. When the map processes that data, it would not know when to complete because it would not understand where the end of the data in the record occurs. Therefore, an override must be used to allow the map to complete as if the terminators were contained in the data stream. This override would indicate to the map where the end of each record occurs. If the target of the FTP is UNIX Systems Services (USS), the FTP converts the `<CR><LF>` terminators (**ASCII** mode) in Windows files to `<NL>` (**EBCDIC**) on the z/OS environment.
If the target of the FTP is a file in the native file system, the `<CR><LF>` characters are removed; there is no record terminator at all.
Type trees are designed to indicate the record terminators. Make sure that the appropriate overrides are specified when running the maps on z/OS so that type validation works as expected. The override is used to place an end of record indicator in an otherwise unmarked record.

Record Terminators (/V)

Use the Record Terminators execution command (`/V`) in source and target map settings to enable special handling of data with embedded record terminators in files with variable-length records, specifically with a record format of variable (V), variable block (VB), or variable block spanned (VBS).

`[/V[`*`terminator_list`*`...]]`

Option Description

terminator_list

A comma-delimited string of bytes that can be expressed in hex format (for example: `/VX5B,X6C,X7C`) or character format (for example: `/V$,%,@`).

The default record terminator, if `none`, is indicated on the `/V` command as a single byte, hex15.

This string of terminator bytes should match the record terminator defined in the type tree that represents this data.

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

For inputs, this command causes a string of one or more bytes to be appended to each record to ensure that the data maps properly when using the Launcher for z/OS. For example, suppose you are mapping an input file on the PC that consists of lines of text terminated by carriage-return/line-feed (`0D 0A`) pairs. Even though you cannot see them in a text editor, these terminators are embedded in the data, they are defined in the type tree that is created to represent this data, and the map expects them to be there.

Now you need to run this same map using the Launcher for z/OS. However, when you upload the input file to a z/OS environment, the record terminators embedded in the data are removed during the transfer. Consequently, when the file is read in for mapping using the Launcher for z/OS, the record terminators must be reinstated or the data will not map correctly. The solution is to use the command `[/VX0D,X0A]` to append a hex `0D 0A` string to the end of each record.

For outputs, this command specifies that the indicated (or default) byte or bytes are used to mark the ends of each record in the output. When this command is used, the map must embed record terminators in the output data. These terminators indicate the places where record boundaries are to occur in the final output. The record terminators are removed when the records are created in the output target or targets and do not appear in the resulting output.

Allocation parameters

Use allocation parameters to specify output file-creation options when native files are specified as targets and the targets do not exist. It is comparable to specifying output data set definitions on the DD statement in the JCL used if you were executing the Command Server for z/OS Batch.

Allocation parameters are specified as an ALLOC statement in the **FilePath** setting in the Settings window as part of the file name.

When you use allocation parameters, the **BPXWDYN** service handles file allocation. It is used to define the new output and enables file allocation to be available to programs running outside of a TSO environment. Allocation parameters must be specified in the output card settings for the output file to be created successfully.

Note: See the z/OS *Using REXX and z/OS UNIX System Services* IBM manual for more information about file allocation and the **BPXWDYN** service.

Example

The following example demonstrates how to define the same output data set using two different methods. One method is to use allocation parameters specified in an ALLOC statement and the other method is to use data set definitions specified in the DD statement in the JCL.

Allocation (ALLOC) Parameters

```
ALLOC DD(LABELS) DSN('MYUSERNAME.LABELS') RECFM(F,B) LRECL(80)
UNIT(SYSDA) TRACKS SPACE(5,2) NEW CATALOG DELETE
```

JCL DD Statement

```
//LABELS DD DSN=MYUSERNAME.LABELS,
//          DCB=(RECFM=FB,LRECL=80),
//          UNIT=SYSDA,
//          SPACE=(TRK,(5,2)),
//          DISP=(NEW,CATLG,DELETE)
```

Map sources

This is a description of the specifications you might need to use when native files are specified as sources in the Integration Flow Designer.

Input or inputs

- Record Terminators
  - Place the record terminators in the **FilePath** setting when specifying the map input in the Launcher Settings window.
  - A set of brackets ([]) indicates to the z/OS environment that the data contained inside the brackets are the file characteristics.
  - Place the record terminators inside a set of [] in the map input **FilePath**.
  - **Input(s)** → **#1 FileIn** → **GET** → **Source FilePath**

    Example: //'MYUSERNAME.IN1'[/VX0D,X0A]

Map targets

This is a description of the specifications you might need to use when native files are specified as targets in the Integration Flow Designer.

Output or outputs

- Allocation Parameters and Record Terminators (if required)
  - Place allocation parameters and record terminators (if required) in the **FilePath** setting in the Settings window as part of the file name when specifying native files as targets.
  - A set of brackets ([]) is used to define allocation parameters and potential record terminators.
  - Place the allocation parameters and record terminators (if required) inside a set of [] in the map output **FilePath**.
  - **Output(s)** → **#1 FileOut** → **PUT** → **Target** → **FilePath**

Example: `[ALLOC DD(OUT1) DSN('MYUSERNAME.OUT1') TRACKS SPACE(5,1) NEW CATALOG RECFM(F,B) LRECL(80) /VX0D,X0A]`

Note: Systems (**.msl** files) will not reside in the native file system; they will reside in the HFS.

File triggering

The UNIX and Windows Launchers can be set to trigger on files being updated or created. Triggering on files in the native z/OS file system is more complex than in the HFS. This is most evident when trying to do file-update triggering because the z/OS operating system, in general, does not maintain update information for files.

Triggering on native file creation or update supports the following native file system file types, depending on the triggering type:

- Sequential files

  Example: `//'MYUSERID.DATA.INPUT1'`
- PDS and PDS/E members

  Example: `//'MYUSERID.DATA(INPUT1)'`

File-creation triggering

File-creation triggering is allowed for any native file system or file system member. In file-creation triggering, when a specified file is created and there is a successful open-file attempt, the event is triggered.

File-update triggering

File-update triggering is more complex because, as mentioned previously, in general, the z/OS operating system does not maintain information for files that could be used as a way of recognizing that the file changed. The z/OS Launcher will infer a change of a PDS or PDS/E member by recognizing the change in its location, TTR, or pointer.

File-update triggering is allowed for PDS or PDS/E members, but not sequential files. There is no straightforward way to check for sequential file updates.

The following process repeats until the specified event is triggered. The process awaits a specified file to be updated and when it recognizes that it has been updated, by a successful directory read attempt, it triggers execution of an event.

To trigger on an updated file:

1. Retrieve the current TTR information for the PDS or PDS/E member or members by reading the directory.
2. Save the retrieved information.
3. Repeat the loop until the process recognizes that the file has been updated.
   - Get the TTR information.
   - If the TTR information has changed, the event is triggered and the TTR is updated.
   - If the TTR information has not changed, the process repeats the loop to check for file updates.

Wildcards in native file system names

Wildcards are characters that are used to substitute unknown characters. Using wildcards for file creation or update triggers can be costly on z/OS operating systems.

Note: When using wildcards, place them in a location of native file system names where you are certain that the search will be performed for a well-defined portion of the catalog. The search that is invoked acts like a TSO Data Set List Utility search. It is very powerful and searches through the entire portion of the catalog defined by the location of the wildcard. The larger the portion of the catalog that is searched, the longer the search takes to complete.

Examples of wildcard-triggering uses

The following examples present different ways you can specify and use wildcard-triggering. The bullet points highlight their characteristics, similarities and differences.

```
//'MYUSERID.DATA.INPUT(*)'
```

- The partitioned data set (PDS) being searched is a single data set.
- Any of its members can be a potential trigger.
- The system catalog does not have to be searched.
- There will be repeated reads on the data set to check for new or changed files.

```
//'MYUSERID.DATA.INPUT*' or
//'MYUSERID.DATA.INPUT.*'
```

- The search is on all data sets that have the same specified pattern of nodes or qualifiers.
- Any data set that meets the criteria can be a potential trigger.
- The system catalog is searched.
- There will be repeated searches of the system catalog for new or changed files that match the specified pattern.

  Note: In the z/OS operating system, the system catalog is a single resource used by all tasks running on the system. CPU and resource consumption could be very high.

File pre-allocation

Some jobs are designed to create or pre-allocate a file in advance of being populated with data. If the Launcher is set up to trigger on file creation, file pre-allocation might cause the map execution not to complete as expected. Considerations about whether or not to design your jobs with pre-allocated files might be necessary when setting up your Launcher processes.

Incomplete file creation or update

It is possible to trigger on file creation or update if only a portion of the file was created or updated. If the Launcher is set up to trigger on file creation or update, incomplete file creation or update might cause the map execution not to complete as expected.

A method of triggering on file creation or update when it has not completely been created or updated is to create a file with the following disposition parameter setting so that the file will not appear until it is cataloged at the end of the job:

```
DISP=(NEW,CATLG)
```

Local z/OS administration interfaces

WebSphere Transformation Extender for z/OS includes a set of local administration interfaces that you will use to run the Launcher on the z/OS operating system running on z/Architecture platforms.

The two local administration interfaces are:

- ISPF Panel Interface (″ISPF Panel Interface″)
- z/OS Operator Console Interface (″z/OS Operator Console Interface″)

Overview

A benefit to using the local administration interfaces is that they provide the ability to connect to the user interfaces.

The only limitation of using the local administration interfaces is that you cannot start systems from the Management Console.

Activities summary

Local administration interfaces perform the following activities:

- Read the system deployment directory.
- Build the command to start the Launcher, including a fully qualified path to the Launcher, list of systems, and command ports.
- Start the Launcher.
- The operator console interface invokes Write-To-Operator (WTO) calls, which communicate status for the operator console interface. WTO messages display the results from the Launcher process for each operation.
- The ISPF interface processes return codes and messages for display in the ISPF panel.

Note: WebSphere Transformation Extender for z/OS can be managed through the Management Console and monitored through the Windows Launcher Monitor, even though the administration is performed locally.

After WebSphere Transformation Extender for z/OS is started using one of the local administration interfaces, the Windows/UNIX Management Console can be used to manage the Launcher. The Management Console can be used to view statistical data and also control the compound system that is running from the process running. See ″Multiple Processes″ for additional information when using the multiple-processes feature.

For more details, see the following documentation:

- ″Launcher Options″
- ″Using the Launcher Management Tools″
  - ″Using the Management Console″
  - ″Starting the Launcher Monitor″

Using the Launcher Management Tools

When using the local administration interfaces to perform Launcher administration tasks, there are Management Console and Launcher Monitor configuration considerations.

Management and monitoring of the Launcher process is dependent on the service having been started. Since the Launcher service is not started when using the local administration interfaces, a different method of accessing the Management Console and Launcher Monitor must be performed.

Using the Management Console

After the Management Console is started, you will perform the following activities to establish a connection to the Launcher process running on a z/OS environment.

The connection will enable you to use the Management Console to manage your systems running on the z/OS environment.

Note: Because the local administration interfaces are used to start systems, you cannot start systems from the Management Console. The Management Console performs all the other capabilities required to manage the Launcher process running your systems on a z/OS environment.

To configure the Launcher in the Windows and UNIX Management Console

1. Define a new Launcher.
2. In the Management Console, select **Tools** → **Options**.

   The Options window opens.
3. From the Options list, select **Launchers** to connect to the remote Launcher.

   The override configuration options appear.

Starting the Launcher Monitor

The Windows Launcher Monitor can be used to monitor the Launcher.

The following method can be used to start the Window Launcher Monitor:

- Issue the following command from the Windows command prompt:
  - `dtxmon -c:<` *computer_name* `> -p:<` *monitor_port_number* `>`
  - *computer_name* is the TCP/IP hostname of the system running the Launcher.
  - *monitor_port_number* is the second port in the ’port range’.
  - Default port is `7001`.

Note: The Launcher **Monitor** application saves the configuration information you entered in the user interface when you initially started the Launcher **Monitor** through the command line. It makes the configuration information available for subsequent starts that you can do through the Windows **Start** menu as follows: Select Launcher **Monitor** from the **Start** → **Programs** menu under **IBM** WebSphere Transformation Extender *n.n* (where *n.n* represents the version number) → **Launcher** .

Default port settings

The local administration interface applications used to run the Launcher on the z/OS operating system running on z/Architecture platforms work collectively and must communicate by means of ports. Each application has default port settings. However, you can change them as needed.

The following list displays the default port settings for each application, and the interrelated structure of the applications.

- The Launcher relays monitoring information over port 7001 to the Monitor.
- The Launcher relays status information over port 7000 to the Management Console.
- The local administration interface sends control information (start, stop, pause, resume) to the Launcher process that is running over port 7002.

Environment settings

To run WebSphere Transformation Extender for z/OS, a set of environment variables must be set.

They are usually established by running a shell script from the UNIX command line. Since there is no UNIX command line being used in this case, an environment-variable file is used to configure the settings for the local administration interfaces.

File location

The sample environment variable file location resides in DTX.SDTXCNFG(DTXENVAR). It contains pre-specified default values that are included in the WebSphere Transformation Extender for z/OS installation.

Note: The location of the PDS and member name containing the environment variable file must be specified in the DTXENV DD statement of the JCL for starting the z/OS Operator Console command. A sample PROC is included in the installation-defined data set, DTX.SDTXSAMP(DTXPROC).
The location of the DTXENVAR must also be specified in a CLIST or REXX EXEC to start the ISPF interface. A sample REXX EXEC is included in the installation-defined data set, DTX.SDTXCLST(DTXCLIST).

DTXENVAR environment variable file

The contents of **DTXENVAR** include settings for the following variables:

Environment Variable
Description

_CEE_RUNOPTS
Language Environment runtime options for the Launcher process.

This is an optional field that you can override. See the IBM z/OS Language Environment Programming Reference, SA22-7562 for a description of the valid runtime options.

CENTRAL_MGMT

Configuration value indicating whether to use centralized (global) or local resource management.

Valid values are `YES` for centralized (global) resource management and `NO` for local.

Centralized (global) indicates that resource management will allow multiple processes to share common resources properly.

Default value is `NO`.

CONNECTION_RETRY_COUNT

The maximum number of times the z/OS native interface is allowed to attempt the TCP/IP connection to the Launcher control port before an error condition, if attempts are not successful, is raised.

Default value is `10`.

Note: If the **CONNECTION_RETRY_COUNT** value is reached before the z/OS native interface can start or detect the Launcher, the `TX is not active...` message will appear.

DEBUG

Enable debug option. Valid values are `YES` to enable the debug option and `NO` to disable it.

Default value is `NO`.

INI_FILE

dtx.ini file

Override the default location and name of the **dtx.ini** file.

Uncomment the line by removing the # character in front of **INI_FILE** and replace `<configuration directory>/<ini file>` with a valid file location of the **.ini** file to be used.

LAUNCHER_NAME

Symbolic name assigned to the current Launcher. It uniquely identifies the Launcher instance and is also used to prefix the Launcher and debug logs when running the multiple-process Launcher.

The name is displayed in the ISPF DTXEVNT window.

Default value is `Launcher-Name`.

Note: Maximum length is 19 characters and the name should not contain any spaces.

DTX_CONTROL_PORT

TCP/IP port number used for control information when communicating with the Launcher process.

Default value is `7002`.

DTX_HOME_DIR

Installation location for the Launcher)

Default value is `/u/dtx/Launcher`.

DTX_MONITOR_PORT

TCP/IP port number used for monitoring information when communicating with the Launcher process.

Default value is `7001`.

DTX_STATUS_PORT

TCP/IP port number used for status information when communicating with the Launcher process.

Default value is `7000`.

SYSTEM_FILE

Location of a single system (**.MSL**) to be used in a multiple-process Launcher configuration.

Uncomment the line by removing the `#` character in front of `SYSTEM_FILE`. Replace `<deployment directory>/<system name>.msl` with the valid file location of the system to be executed.

TIME_ZONE

Time zone used.

This is required only if the locale defined to UNIX System Services (USS) does not contain a valid **LC_TOD** category.

Default value is `EST5EDT`.

ISPF panel interface

The ISFP panel interface allows you to manage the Launcher as well as view Launcher information on the z/OS operating system.

Commands

The ISPF commands are available to control the execution of the ISPF administration interface.

The following procedure uses the ISPF Command Shell screen to initiate the execution of the interface.

To start the ISPF administration interface

1. From ISPF, select `Primary Option Menu > option 6 - Command`.
2. At the command prompt in the ISPF Command Shell screen, enter the `EXEC` 'DTX.SDTXCLST(DTXCLIST)'.

   The ISPF administration interface starts. The Administration Main Menu appears, listing the command options available to control the execution.

Menu options

To manage the Launcher through the ISPF panel interface, select the function by entering the associated menu option number at the **Option ===>** prompt in the **Launcher Administration Main Menu** panel as follows:

Menu Function Use	Menu Option	Option Description
Start the Launcher	1	Start the Launcher identified in the **Launcher Information** area of the panel. **Status** field must be `Unknown`.

Menu Function Use	Menu Option	Option Description
Stop the Launcher	2	Stop theLauncher identified in the **Launcher Information** area of the panel. **Status** field must be `Running` or `Paused`.
Pause the Launcher	3	Pause the Launcher identified in the **Launcher Information** area of the panel. **Status** field must be `Running`.
Resume the Launcher	4	Resume the Launcher identified in the **Launcher Information** area of the panel. **Status** field must be `Paused`.
Refresh Launcher Status	5	Get the status of the Launcher identified in the **Launcher Information** area of the panel. This is allowed for any current state; **Status** field can contain any valid value.

Server information

The server information is contained in the **Launcher Information** area of the panel.

The following list describes the various fields that are used in the LauncherAdministration Main Menu panel to display information about the Launcher, the status of the Launcher, as well as the ports used.

Field **Description**

Server Name of the current Launcher (user-supplied name)

Status Current status of the Launcher.

Values are:

- Not Responding - not started.
- Starting - in the process of starting
- Started - has started
- Stopping - in the process of stopping
- Stopped - has stopped
- Pausing - in the process of being paused
- Paused - currently paused and not running deployed systems
- Unknown - in an unknown state
- Error - has detected an error and might or might not continue

Status Port
Port used to communicate the status of the Launcher.

Also used by the Management Console interface.

Monitor Port
Port used to communicate monitoring information.

Also used by the Windows Launcher Monitor interface.

Control Port
Port used to send commands (start, stop, pause, resume) to the Launcher.

Also used by the Management Console interface.

The ISPF panel resides in DTX.SDTXPENU(DTXEVNT). It must be invoked from a REXX EXEC or CLIST.

Managing multiple Launchers

Multiple Launchers are administered by starting multiple ISPF panels. They must be invoked through multiple CLIST or REXX EXECS and use different configuration files.

Note: You must create a separate environment variable file for each Launcher instance to be able to manage your systems.

z/OS operator console interface

The z/OS Operator Console interface provides a standard interface that allows the z/OS system operator to control a server by using a cataloged procedure (PROC).

Commands

The operator console interface commands are available to control the execution of the Operator Console administration interface.

The following procedure uses the start command to initiate the execution of the interface.

To start the Operator Console interface:

From the Operator Console, enter `S DTXPROC`.

The Operator Console administration interface has started.

The following table lists the available Operator Console administration commands.

Operator Console Command Use	**Commands**
Start	`S DTXPROC`
Stop	`P DTXPROC`
Pause	`F DTXPROC,APPL=PAUSE`
Resume	`F DTXPROC,APPL=RESUME`
Status	`F DTXPROC,APPL=Status`

DTXPROC is a PROC that resides in DTX.SDTXSAMP.

Note: The DTXPROC PROC must first be copied to a system procedure library, such as SYS1.PROCLIB, before the procedure can be started from the Operator Console.

Managing multiple Launchers

The z/OS Operator Console interface manages multiple Launchers in a similar manner as the ISPF panel interface. The only difference is that they are invoked through multiple PROCs, instead of multiple REXX EXECs or CLISTs.

Note: You must create a separate environment variable file for each Launcher instance so that you can manage your systems.

Multiple processes in the native administration interfaces

The WebSphere Transformation Extender Launcher multiple-processes feature, as described in″Multiple Processes″, is also available to use with the z/OS operating system on IBM z/Architecture platforms. If you will be running systems specifically through the z/OS native administration interfaces (NAI), the WebSphere Transformation Extender Launcher multiple-processes feature allows you to run multiple Launcher processes (systems or **.msl** files) within a single installation of the Launcher as described in this documentation.

To enable the multiple-processes feature, global management is selected through resource management settings in the Resource Manager section of the NAI **DTXENVAR** environment variables file. This file is included in the WebSphere Transformation Extender Launcher installation. See ″Using Resource Management″ for background information about global resource management.

The NAI will optionally launch a single system for each Launcher installation instance or recognize a request for global resource management and launch multiple systems in a single instance depending on the way in which you instruct the Launcher, through the settings in the **DTXENVAR** environment variables file, to run the system or systems. Also, there are additional settings in the file that you can change to enable the Launcher to create debug and Launcher log files.

See ″DTXENVAR Environment Variable File″ for the settings that you can configure to enable the WebSphere Transformation Extender Launcher multiple-processes feature.

Troubleshooting tip

If you encounter a problem when running the Launcher for your z/OS environment, run the maps through the UNIX Systems Services (USS) Command Server as a debugging test. The results from running maps through the USS Command Server should be the same as running maps through the Launcher for the z/OS environment. If the source, targets, and settings have changed, you can override them by using the appropriate command line options.

Chapter 10. Glossary

Term **Definition**

Batch Running in the background with no user assistance

CICS Customer Information Control System; online transaction processing program; running in the foreground communicating with users

CPU Central Processing Unit; contains the logic functionality and processes the instructions that drive a computer.

DCB Data Control Block *parameter* of DD statement in JCL

DD Data Definition *statement* in JCL

EXCP Execute Channel Program

HFS Hierarchical File System, similar to any UNIX file structure

JCL Job Control Language

LE Language Environment

LNKLST
Link List

LPA Link Pack Area

LRECL
Logical Record Length

LSR Local Shared Resources

MIPS Million Instructions Per Second

MSU Measured Service Units

PDS Partitioned Data Set

PDS/E Partitioned Data Set Extended. It is an extended version of a PDS; allows for long alias names and automatic reusing of deleted space without the need for manually compressing the file.

RECFM
Record Format used in DCB *parameter* of DD statement in JCL

TTR Track Track Record

- The relative offset of the member in a PDS.
- It is the address that describes a location within the file relative to the beginning of the file.

USS UNIX System Services

VIO Virtual Input/Output

z/Architecture
Line of IBM computer systems running operating systems such as z/OS

z/OS Operating system running on z/Architecture.

z/OS Native File System
In the context of large processors or servers running z/OS operating systems on IBM z/Architecture platforms, this is the file system that employs the following file types:

- Sequential data sets
- Partitioned Data Sets (PDS)
- Partitioned Data Sets Extended (PDS/E)

Notices

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

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″ WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

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

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

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

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

Trademarks and service marks

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

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

WebSphere
z/Architecture
z/OS
zSeries

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

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

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

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

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

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

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

IBM WebSphere Transformation Extender, Version 8.1

Index

Printed in USA