Dispatcher Installation and Configuration Guide

Dispatcher Installation and Configuration Guide

Contents

Figures

Tables

Chapter 1. Overview

The Dispatcher is a key component for adapters that are based on Tivoli® Directory Integrator. The Dispatcher provides the link between IBM® Security Identity server and the IBM Tivoli Directory Integrator.

The Dispatcher is not installed with the base Tivoli Directory Integrator product. It must be installed separately to enable the Tivoli Directory Integrator-based adapters to run.

The Dispatcher runs as an instance of the Tivoli Directory Integrator and is a prerequisite to install and run all Tivoli Directory Integrator-based adapters. Multiple adapters can be installed on the same Tivoli Directory Integrator where the Dispatcher is installed. The adapters consist of assembly line configurations that initialize and run Tivoli Directory Integrator connectors.

The Dispatcher is the user management API for the Tivoli Directory Integrator provider. The Dispatcher loads and runs assembly line configurations specified by the Tivoli Directory Integrator provider.

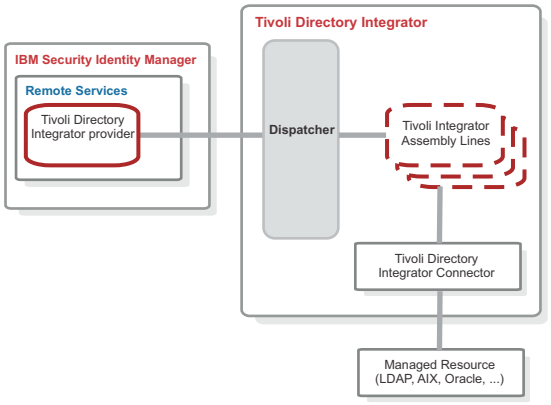

Figure 1. The architecture of the Dispatcher

For more information about Tivoli Directory Integrator, see https://www.ibm.com/support/knowledgecenter/SSCQGF_7.1.1/com.ibm.IBMDI.doc_7.1.1/welcome.htm.

Chapter 2. Planning

Installing and configuring the dispatcher involves several steps that you must complete in a specific sequence. As such, follow the roadmaps.

Use the Preinstallation roadmap to prepare the environment.

Table 1. Preinstallation roadmap

Task	**For more information, see**
Verify that your environment meets the software and hardware requirements for the adapter.	"Prerequisites."
Obtain the installation software.	Software downloads.
Obtain the necessary information for the installation and configuration.	"Installation worksheet" on page 5.

Use the Installation and configuration roadmap to complete the actual installation and configuration of the adapter.

Table 2. Installation and configuration roadmap

Task	**For more information**
Install the dispatcher.	See "Installing the Dispatcher in GUI mode" on page 7.
Verify the installation.	See "Verifying the adapter installation" on page 10.
Configure the dispatcher.	See "Configuring the Dispatcher" on page 17.

Prerequisites

Verify that your environment meets the software and hardware requirements for the Dispatcher.

The following table identifies the software and operating system requirements for the Dispatcher.

Table 3. Prerequisites to run the dispatcher

Prerequisites	Description
Directory Integrator	• IBM Tivoli Directory Integrator Version 7.1.1 + 7.1.1-TIV-TDI-FP0004 + 7.2.0-ISS-SDI-LA0008 • IBM Security Directory Integrator Version 7.2 **Note:** • Earlier versions of IBM Tivoli Directory Integrator that are still supported might function properly. However, to resolve any communication errors, you must upgrade your Directory Integrator release to the versions that the adapter officially supports. • The adapter supports IBM Security Directory Integrator 7.2, which is available only to customers who have the correct entitlement. Contact your IBM representative to find out whether you have the entitlement to download IBM Security Directory Integrator 7.2.
IBM Security Identity server	The following servers are supported: • IBM Security Identity Manager server Version 6.0 • IBM Security Identity Manager server Version 7.0 • IBM Security Privileged Identity Manager Version 2.0 • IBM Security Identity Governance and Intelligence server Version 5.2.2
Operating system	The Dispatcher can be used on any operating system that is supported by Tivoli Directory Integrator.
System Administrator Authority	The person who performs the Dispatcher installation procedure must have system administrator authority to complete the steps in this chapter. The person who performs the installation must also have execute permissions on the **ps** command on non-Windows platforms.

The Dispatcher must be installed on the same workstation as the Tivoli Directory Integrator server. For information about the system requirements and supported operating systems for Tivoli Directory Integrator, see the *Tivoli Directory Integrator 7.1: Administrator Guide*.

Tivoli Directory Integrator adapters solution directory

The adapters solution directory is a Tivoli Directory Integrator work directory for adapters.

The person who installs the Tivoli Directory Integrator must have read and write access to these directories:

- The adapters solution directory
- The Tivoli Directory Integrator home directory

The first Dispatcher installation prompts you to enter the complete path of the adapter solution directory. For example, enter `C:\Program Files\ibm\TDI\V7.1\timsol`, where *timsol* is the adapter solution directory. The parent directory that you enter for the adapter solution directory must exist.

For every subsequent Dispatcher installation, the installer uses the *timsol* directory that is already set in the `global.properties` file. It does not prompt for an adapter solution directory.

Note: To install the Dispatcher correctly and to avoid errors during the installation, do not use the Tivoli Directory Integrator home directory as the adapter solution directory.

Software download

Download the software through your account at the IBM Passport Advantage® website.

Go to IBM Passport Advantage.

See the corresponding *IBM Security Identity server Download Document* for instructions.

Note:

You can also obtain additional adapter information from IBM Support.

Installation worksheet

The installation worksheet lists the information that is required to install and configure the adapter. Complete this worksheet before you start the installation procedure for ease of reference. Make a copy of the worksheet for each adapter instance you install.

Table 4. Required information to install the Dispatcher

Required information	Description	Value
Tivoli Directory Integrator Home Directory	The *ITDI_HOME* directory contains the `jars/connectors` subdirectory that contains adapter JAR files. For example, the `jars/connectors` subdirectory contains the JAR file for the UNIX adapter.	If Tivoli Directory Integrator is automatically installed for version 7.1, the default directory path depends on the operating system. **Windows operating systems** *drive*`\Program Files\IBM\TDI\V7.1` **UNIX and Linux operating systems** `/opt/IBM/TDI/V7.1`
Solution Directory	This directory is the default directory. When you install the dispatcher, the adapter prompts you to specify a file path for the solution directory. See "Tivoli Directory Integrator adapters solution directory" on page 4.	The default solution directory for version 7.1 depends on the operating system. **Windows operating systems** *drive*`\Program Files\IBM\TDI\V7.1\`*timsol* **UNIX and Linux operating systems** `/opt/IBM/TDI/V7.1/`*timsol*

Chapter 3. Installing

You must install the Dispatcher on the same Tivoli Directory Integrator server where you want to install the adapter.

Multiple Tivoli Directory Integrator-based adapters installed on the same Tivoli Directory Integrator server can use the same Dispatcher. However, you must install a Dispatcher on each Tivoli Directory Integrator server on which you want to install an adapter.

Note:

- Before you install this version of the Dispatcher, you must uninstall earlier versions. You cannot run the Dispatcher installer on an existing installation.
- During upgrade, the Dispatcher installer does not request an instance name and port number.
- To run the Dispatcher installer on non-Windows systems, you must have **execute** permissions on the **ps** command.

Obtain the dispatcher installer from the IBM Passport Advantage website, see "Software download" on page 5.

Installing the Dispatcher in GUI mode

You must install the Dispatcher before you can use any of the adapters based on Tivoli Directory Integrator.

About this task

If you install the Dispatcher in GUI mode, then you can uninstall it in GUI, console, or silent mode.

Procedure

1. Extract the contents of the compressed file in the temporary directory.
2. Use the Java™ Virtual Machine (JVM) supplied by Tivoli Directory Integrator. The JVM is in the *ITDI_HOME*/jvm/jre/bin/ directory, where *ITDI_HOME* is the directory where Tivoli Directory Integrator is installed. Run the Java installer:

   ```
   ITDI_HOME/jvm/jre/bin/java -jar DispatcherInstall.jar
   ```

3. On the Welcome page, click **Next**.
4. In the Directory Name field, specify the location of the Tivoli Directory Integrator home directory.
5. In the Solution Directory field, specify the complete path of the adapter solution directory. For more information about adapter solution directory, see "Tivoli Directory Integrator adapters solution directory" on page 4.
6. Review the installation settings on the Install Summary page and perform one of the following steps:
   - Click **Back** to return to a previous page to modify any of the settings.
   - Click **Next** when you are ready to begin the installation.
7. Click **Finish** when the software displays the Install Completed window.

Installing the Dispatcher in console mode

You can install the Dispatcher with console mode.

About this task

If you install the Dispatcher by using console mode, then you can uninstall the Dispatcher only with console mode or silent mode.

Procedure

1. Open a command-line interface.
2. Run the following command:

   ITDI_HOME/jvm/jre/bin/java -jar DispatcherInstall.jar -i console

Installing the Dispatcher in silent mode

You can install the Dispatcher in silent mode.

About this task

You can install the Dispatcher in silent mode by using the default settings. You also can override the default settings with the commands described in Table 5.

If you use the default settings, then the Dispatcher is installed in the following location, depending on your operating system:

- On Windows, in %SYSTEM_DRIVE_ROOT%\Program Files\IBM\TDI\V7.1
- On UNIX and Linux, in /opt/IBM/TDI/V7.1

You can override the default settings with the -D parameter. The -D must be immediately followed by an option-value pair. There is no space after the -D option.

Note: If the value contains spaces, then you must use quotation marks around the value.

If you install the Dispatcher by using silent mode, then the uninstaller runs in silent mode regardless of whether you use the -i silent option.

Table 5. Parameters for installing the Dispatcher in silent mode

Parameter	Description
-DUSER_INSTALL_DIR	This parameter overrides the default installation path. For example, -DUSER_INSTALL_DIR="D:\security\MyFolder"
-DUSER_SELECTED_SOLDIR	This parameter overrides the default adapters solutions directory. For example, -DUSER_SELECTED_SOLDIR="/opt/IBM/TDI/V7.1/mysol"
-DUSER_INPUT_RMI_PORTNUMBER	This parameter overrides the default RMI port number on which the dispatcher listens. For example, -DUSER_INPUT_RMI_PORTNUMBER=1234

Table 5. Parameters for installing the Dispatcher in silent mode (continued)

Parameter	Description
-DUSER_DISPATCHER_SERVICE_NAME	This parameter specifies the name of the Dispatcher service on Windows. For example, -DUSER_DISPATCHER_SERVICE_NAME="ISIM Adapters"

Procedure

1. Open a command-line interface.
2. Run one of the following commands.
   - To install the Dispatcher in silent mode with the default settings, run the command:

     ```
     ITDI_HOME/jvm/jre/bin/java
     -jar DispatcherInstall.jar -i silent
     ```

   - To install the adapter in silent mode and with one or more custom settings, use the -D parameter. For example:

     ```
     ITDI_HOME/jvm/jre/bin/java
     -jar DispatcherInstall.jar -i silent
     -DUSER_INSTALL_DIR="/opt/IBM/TDI/V7.1"
     -DUSER_SELECTED_SOLDIR="/opt/IBM/TDI/V7.1/timsol"
     -DUSER_INPUT_RMI_PORTNUMBER=1099 -DUSER_INPUT_WS_PORTNUMBER=8081
     ```

Installing the Dispatcher on a z/OS operating system

You must install both a binary UNIX tar file and a shell script to install the Dispatcher on a z/OS operating system.

About this task

After the installation of the Dispatcher is complete, verify the startup and shutdown of the Dispatcher. See "Start, stop, and restart the Dispatcher service" on page 11.

Procedure

1. Locate the delivered Dispatcher or adapter compressed file.
2. Extract the contents of the compressed file into a temporary directory and navigate to that directory.
3. From the temporary directory, locate and navigate to the zSystem directory.
4. Under the zSystem directory, locate the following two files:
   - Dispatcher.tar
   - instDispatcher_zOS.sh

   Note: Dispatcher.tar is a binary UNIX tar file and instDispatcher_zOS.sh is a UNIX shell script.
5. Transfer the two files to the zOS workstation where the adapter is to be installed. Both files must be copied to the same directory.
6. Set the execution flag on instDispatcher_zOS.sh:

   ```
   chmod +x instDispatcher_zOS.sh
   ```

7. Run the installer by issuing the command:

   ```
   ./ instDispatcher_zOS.sh
   ```

The following dialog is displayed.

Note: The path given in the following example might be different on your system.

```
*********************************************
   ITIM RMI Dispatcher Installation Program
*********************************************

You will prompted to enter the following information:

        TDI home directory.
        Your TDI solution directory.

Make sure you have the above information available and
the Dispatcher.jar is located in the current directory
before you continue

1. Install
2. Quit

Please enter choice: 1

Extracting content of Dispatcher...

Enter TDI home directory,
Hit [Enter] to accept [/usr/lpp/itdi]
or type new value (full path):

Enter the solution directory name (full path): /u/user2/rmi/soldir

extracting content of Dispatcher.jar...
setting up solution directory tree /u/user2/rmi/soldir...
getting files from TDI home directory /usr/lpp/itdi...
updating /u/user2/rmi/soldir/solution.properties file...
getting dispatcher files from /u/user2/rmi/Dispatcher...
updating /u/user2/rmi/soldir/ITIMAd file...

Installation complete, press any key to continue...
```

Verifying the adapter installation

You must verify that the Dispatcher installation placed components in the correct directories on the Tivoli Directory Integrator server.

Table 6. Dispatcher components

Directory	Dispatcher component
ITDI_HOME\jars\3rdparty\IBM	• rmi-dispatcher.jar • itim-dispatcher-ws-transport.jar • itim-dispatcher-ws-config.jar
ITDI_HOME\jars\3rdparty\others	• antlr-2.7.2.jar • jakarta-regexp-1.4.jar

Table 6. Dispatcher components (continued)

Directory	Dispatcher component
adapter_solution_directory	• ITIM_RMI.xml • log4j.properties This component is available on Windows operating system. • ibmdiservice.props This component is available on Windows operating system. • ibmdiservice.exe This component is available on Windows operating system • ITIMAd Tivoli Directory Integrator on operating systems other than Windows. • itimadpid This component is available on a Solaris operating system.
ITDI_HOME	• itim_listener.properties
ITDI_HOME\SOL_DIR\idm_respository\modules	• itim-dispatcher-authn.mar
ITDI_HOME\SOL_DIR\idm_respository\services	• itim-dispatcher-ws.aar
ITDI_HOME\SOL_DIR\	• axis2.xml • svcConfigDB This component is the database instance.

Review the installer log files Dispatcher_Installer.log and Dispatcher_Installer_opt.log in the installer directory for any errors.

If this installation is to upgrade a Dispatcher, send a request from IBM Security Identity server. Verify that the version number in the ibmdi.log matches the version of the Dispatcher. Navigate to the ADAPTER_SOLDIR/logs directory and search for *RMIDispatcherImpl: Starting*. Verify that the version number of the Dispatcher is correct.

Start, stop, and restart the Dispatcher service

When you edit an adapter or Tivoli Directory Integrator properties file, you must stop and restart the Dispatcher service for the changes to take effect.

Select the appropriate method based on your operating system.

Starting, stopping, and restarting the Dispatcher service on AIX, HP-UX, Linux, and Solaris operating systems

When you edit an adapter or Tivoli Directory Integrator properties file, you must stop and restart the dispatcher service for the changes to take effect.

About this task

The ITIMAd script file starts and stops the service. The adapter installation copies the file to a specific directory, depending on the operating system.

Table 7. UNIX based and Linux directories

Operating system	Directory
AIX	`timsol`
HP-UX	`timsol`
Linux and Solaris	`timsol`

On Solaris operating system, the `ITIMAd` script file creates the `itimadpid` file in the adapter solution directory. The file contains the process ID of the dispatcher service. **Do not modify or delete this file.** When you start the dispatcher service, the `ITIMAd` script file creates the `itimadpid` file. When you stop the dispatcher service, the `ITIMAd` script file deletes the `itimadpid` file. This file is not created on other platforms.

Procedure

1. From the command line, navigate to the directory that contains the `ITIMAd` script file.
2. Run the following commands to start, stop, and restart the dispatcher service:

Table 8. UNIX based and Linux commands

AIX	HP-UX	Linux and Solaris
`ITIMAd startsrc`	`ITIMAd start`	`ITIMAd start`
`ITIMAd stopsrc`	`ITIMAd stop`	`ITIMAd stop`
`ITIMAd restartsrc`	`ITIMAd restart`	`ITIMAd restart`

Starting, stopping, and restarting the Dispatcher service on the Windows operating system

When you edit an adapter or Tivoli Directory Integrator properties file, you must stop and restart the Dispatcher service for the changes to take effect.

About this task

You can use the Windows graphical user interface to start or stop the Dispatcher service.

Procedure

1. In the Control Panel, click **Administrative Tools** > **Services**.
2. In the Services window, you can start and stop the Dispatcher service. The service name is `IBM Tivoli Directory Integrator (TIM Adapters)`.

Starting, stopping, and restarting the Dispatcher service on Linux for System z and z/OS operating systems

When you edit an adapter or Tivoli Directory Integrator properties file, you must stop and restart the Dispatcher service for the changes to take effect.

About this task

The `ITIMAd` script file starts and stops the service. The adapter installation copies the file to the `timsol` directory.

Procedure

1. Navigate to the `timsol` directory.
2. Run the following commands:

Table 9. Linux for System z and z/OS commands

	Linux for System z	**z/OS**
To start the dispatcher	`% ./ITIMAd start`	`% ./ITIMAd start`
To verify whether the `ibmdisrv` or the `ibmdisrv_ascii` process is running	`% ps –ef | grep ibmdisrv`	`% ps –ef | grep ibmdisrv_ascii`
To stop the adapter	`% ./ITIMAd stop`	`% ./ITIMAd stop`
To verify that the `ibmdisrv` or `ibmdisrv_ascii` process is not running	`% ps –ef | grep ibmdisrv`	`% ps –ef | grep ibmdisrv_ascii`

Chapter 4. Upgrading

The Dispatcher is upgraded by installing the new version of the Dispatcher.

Before you upgrade the Dispatcher, verify the version of the Dispatcher.

- If the Dispatcher version mentioned in the release notes is later than the existing version on your workstation, install the Dispatcher.
- If the Dispatcher version mentioned in the release notes is the same or earlier than the existing version, do not install the Dispatcher.

If the Dispatcher service is running when you upgrade the Dispatcher, then the Dispatcher installer stops the service and restarts it after completing the upgrade process.

If the Dispatcher is not running when you upgrade the Dispatcher, then the Dispatcher installer does not start the service after completing the upgrade process.

If you want to force start the Dispatcher service, use the following command-line option when you run the Dispatcher installer:

```
ITDI_HOME/jvm/jre/bin/java -jar DispatcherInstall.jar
-DFORCE_DISPATCHER_SERVICE_START_ONINSTALL=yes
```

Valid values for `FORCE_DISPATCHER_SERVICE_START_ONINSTALL` are `YES` or `NO`.

Chapter 5. Configuring

After you install the adapter, configure it to function correctly. Configuration is based on your requirements or preference.

Configuring the Dispatcher

You must do several tasks to configure the Dispatcher.

Configuration properties of the Dispatcher

The `solution.properties` and the `itim_listener.properties` files contain the configuration properties for the dispatcher. To configure the properties for the dispatcher, you must change one of these files.

Restart the dispatcher service after you change the properties for the dispatcher. Table 10 lists the properties contained in the properties files.

Table 10. Configuration properties for the Dispatcher

Property	Properties File	Description
`ALShutdownTimeout`	`itim_listener.properties`	Specifies the number of seconds before the RMI Dispatcher shuts down when a shutdown request is sent to the dispatcher. When the dispatcher shuts down, it terminates all the maintained assembly lines. The default value is `300` seconds.
`com.ibm.di.dispatcher.bindName`	`solution.properties`	Specifies the RMI bind name. The default value is `ITDIDispatcher`.
`com.ibm.di.dispatcher.objectPort`	`solution.properties`	Specifies the port on which the dispatcher remote object listens for RMI requests. The default value is `0`, which means a random port is selected at run time.
`com.ibm.di.dispatcher.registryPort`	`solution.properties`	Specifies the port on which the RMI Dispatcher listens for provisioning requests from IBM Security Identity server.
`FailTimeoutRequest`	`itim_listener.properties`	Specifies the Boolean value 1 or 0, indicating true or false respectively. Use this property when the timeout feature is enabled. By setting this value as 1, IBM Security Identity server will fail the time-out requests when the timeout occurs. Default value is 0, which executes the default behavior, IBM Security Identity server retries the time-out requests.

Table 10. Configuration properties for the Dispatcher (continued)

Property	Properties File	Description
SearchALUnusedTimeout	itim_listener.properties	Specifies the number of seconds the dispatcher waits before it deletes the search assembly lines that are unused. The default value is 600 seconds.
SearchReaperThreadTimeOut	itim_listener.properties	Specifies the number of seconds after which the dispatcher releases data from memory. The reconciliation process uses this property. The default value is 300 seconds.
SearchResultSetSize	itim_listener.properties	Specifies the number of records, per response, the dispatcher returns during a reconciliation between IBM Security Identity server and the adapter. The default value is 100.
ALCacheSize	itim_listener.properties	Specifies the number of assembly lines (add, modify, delete) that the dispatcher caches. The default assembly line cache size is 100. Setting the assembly line cache size to 0 disables the caching in the dispatcher.
AssemblylineCacheTimeout	itim_listener.properties	Specifies the number of seconds after which the reaper thread clears the non-executed assembly lines from the assembly line cache. The default timeout period is 600 seconds. **Note:** This property is applicable only for the add, modify, and delete operations. The search operation assembly lines are not cached.
GlobalRunALCount	itim_listener.properties	Specifies the maximum number of assembly lines that the dispatcher can run simultaneously. The default value is 100. **Note:** Setting the GlobalRunALCount to 0 does not limit the number of assembly lines that the dispatcher can run simultaneously. All the assembly lines are started immediately.
MaxWaitingALcount	itim_listener.properties	Specifies the maximum number of assembly lines that you can keep in the queue. When requests exceed the maximum number, subsequent requests fail. The default value of the property is 0, which means there is no limit on the number of assembly lines in the queue.

Table 10. Configuration properties for the Dispatcher (continued)

Property	Properties File	Description
SleepAfterInterrupt		Specifies the time in seconds that the Dispatcher sleeps after a timeout interrupt, to allow cleanup operations to complete. Use this property when the timeout feature is enabled. The default value of the property is 20 seconds.

Changing the port number for the IBM Tivoli Directory Integrator Dispatcher

If you run the Dispatcher as a service, the default port number is 1099. The installer automatically sets this parameter in the global.properties and solution.properties files.

About this task

In IBM Tivoli Directory Integrator version 7.0 or higher, the default setting for the api.remote.on property is true. This setting causes the IBM Tivoli Directory Integrator to listen on port 1099, as defined by the api.remote.naming.port property.

If the api.romote.on property is set to false, IBM Tivoli Directory Integrator listens on the port defined by the com.ibm.di.dispatcher.registryPort property. The default value for this setting is 16231.

To modify the port number for the Dispatcher, you must change the property value in the *ITDI_HOME*/*timsol*/solution.properties directory.

Procedure

1. Stop the service that runs the adapter. See "Start, stop, and restart the Dispatcher service" on page 11.
2. Perform one of the following actions to change the port number:
   - Edit the api.remote.naming.port property in the solution.properties file. You can change the port number to any unused port. For example:

     api.remote.naming.port=12345
   - Change the property to false and edit the file:
     a. Set the api.remote.on property to false.
     b. Edit the com.ibm.di.dispatcher.registryPort property in the solution.properties file. You can change the port number to any unused port. For example:

        com.ibm.di.dispatcher.registryPort=12345
3. Save your changes.
4. Start the service.

Configuring filtering for the Dispatcher

If you do not want the Dispatcher to do case-sensitive filtering, add the CaseInSensitiveFilter property to the search operation in the service.def file.

About this task

The `service.def` file is available in the adapter profile.

The `CaseInSensitiveFilter` property specifies whether the filtering by the Dispatcher must be case-sensitive or not case-sensitive. If the property is set to `false`, you must specify the IBM Security Identity server filter in the same case as the data on the endpoint, otherwise the correct data is not filtered.

The Dispatcher filtering is case-sensitive for adapters that do not support this property.

To add the **`CaseInSensitiveFilter`** property to the adapter, take the following steps:

Procedure

1. Extract the adapter profile jar file.
2. Open the `service.def` file from the extracted adapter profile jar file.
3. Add the following dispatcher parameters in the search operation and save the `service.def` file:

```
<dispatcherParameter name="CaseInSensitiveFilter">
   <default>true</default>
</dispatcherParameter>
```

4. Create the adapter profile jar file with updated `service.def` file.
5. Import the updated adapter profile on the IBM Security Identity server.

Extracting the current Request ID from the assembly line

When a cached assembly line is used for non-reconciliation operations, the dispatcher logs display the cached Request ID. You can display the current Request ID by extracting it from the assembly line.

About this task

A new attribute, `CurrentTCBReqId`, is added in TCB. This attribute preserves the current Request ID of a request.

Procedure

Add the following code in your assembly line. In this code, the `transactionId` holds the Request ID of a request.

```
var tcbfield = task.getClass().getDeclaredField("tcb");

tcbfield.setAccessible(true);

var tcb = tcbfield.get(task);

var transactionId=tcb.getProperty("CurrentTCBReqId");
```

Multiple instances of the Dispatcher on one system

The Dispatcher, version 6.0, 7.0, can support multiple instances of the Dispatcher on the same system. However, there can be only one Dispatcher per IBM Tivoli Directory Integrator instance.

To run multiple dispatchers on the same system, you must specify a unique Service name on Windows systems or subsystem name on AIX® systems. All platforms require a unique port number on which the Dispatcher service can listen.

Configuring the Dispatcher JVM properties for Windows operating systems

The Tivoli Directory Integrator is a Java application that runs its own JVM. You can supply standard JVM properties to the Dispatcher.

About this task

Standard JVM properties are:

- encoding
- memory allocation initial size
- memory allocation maximum size

The Dispatcher process is a running instance of the Tivoli Directory Integrator server.

As an example, this procedure sets the dispatcher encoding to UTF-8.

Procedure

1. Stop the `IBM Tivoli Directory Integrator (ISIM Adapters)` service. See "Starting, stopping, and restarting the Dispatcher service on the Windows operating system" on page 12.
2. Navigate to the adapter *timsol* directory.
3. Open the `ibmdiservice.props` file in the notepad.
4. Set the value of the `jvmcmdoptions` property to the Java property that you want to change. For example, if you want the Dispatcher JVM to run with UTF-8 encoding, then set `jvmcmdoptions=- Dfile.encoding=UTF-8`.

   Note: Separate more than one property with a space.
5. Save and close the `ibmdiservice.props` file.
6. Start the `IBM Tivoli Directory Integrator (ISIM Adapters)` service.

Configuring the Dispatcher JVM properties for UNIX operating systems

The Tivoli Directory Integrator is a Java application that runs its own JVM. You can supply standard JVM properties to the Dispatcher.

About this task

Standard JVM properties are:

- encoding
- memory allocation initial size
- memory allocation maximum size

The Dispatcher process is a running instance of the Tivoli Directory Integrator server.

As an example, this procedure sets the Dispatcherr encoding to UTF-8.

Procedure

1. Navigate to the *TDI_HOME* installed directory.
2. Run the following command:

   ```
   vi ibmdisrv
   ```

3. Modify the string value in the following format:

   ```
   "$JRE_PATH/java" -cp "/opt/IBM/TDI/V7.1/jars/3rdparty/IBM/
   db2jcc_license_c.jar" "-Dlog4j.configuration=file:etc/
   log4j.properties" -jar "/opt/IBM/TDI/V7.1/IDILoader.jar"
   com.ibm.di.server.RS "$@"
   ```

   For example, if you want the JVM to use UTF-8 encoding, then modify the command to:

   ```
   "$JRE_PATH/java" -cp "/opt/IBM/TDI/V7.1/jars/3rdparty/IBM/
   db2jcc_license_c.jar" "-Dfile.encoding=UTF-8" "
   -Dlog4j.configuration=file:etc/log4j.properties" -jar
   "/opt/IBM/TDI/V7.1/IDILoader.jar" com.ibm.di.server.RS "$@"
   ```

4. Restart the service. See "Start, stop, and restart the Dispatcher service" on page 11.

Configuring logging for the adapter

Log files provide information that you can use to diagnose or troubleshoot adapter errors. Logging for the adapters is configured with default settings. Optionally, you can configure the name, the size, and the logging levels for the file. You can also configure the log to append information.

About this task

When multiple adapters run on the server where the IBM Tivoli Directory Integrator is installed, logging information for the adapters is stored in the same log file. The Dispatcher log entries are also stored in this log file. You cannot configure logging to store information about the different components in different log files.

The settings in the `log4j.properties` file determine the type of information that is stored in your log file. To configure logging for the adapter, you must update this file.

The location of the `log4j.properties` file depends on the operating system.

Windows operating systems
*ITDI_HOME**timsol*

UNIX or Linux operating systems
ITDI_HOME/*timsol*/etc

Where *timsol* is the adapters solution directory that is defined by the `ADAPTER_SOLDIR` entry in the *ITDI_HOME*`/etc/global.properties` file.

By default, log file information is deleted and recreated each time the Dispatcher starts. You can append information to an existing log file before or after the dispatcher starts.

Procedure

1. Access the `log4j.properites` file with a text editor.
2. 1. Set the name and size of the log file and specify its maximum size.

a. Specify the name of the log file by modifying the `log4j.appender.Default.file` entry. In the following example, the log file is generated with the name `ibmdi.log`:

```
log4j.appender.Default.file=ibmdi.log
```

b. Specify the maximum size of the log file by modifying the `log4j.appender.Default.MaxFileSize` entry. In the following example, the log file size can be up to 8 MB:

```
log4j.appender.Default.MaxFileSize=8MB
```

c. Specify the number of log files you want to generate by modifying the `log4j.appender.Default.MaxBackupIndex` entry. The following example generates 10 log files:

```
log4j.appender.Default.MaxBackupIndex=10
```

a. Specify the type of Appender you want to use as the default by modifying the `log4j.appender.Default` property. The following example rolls over log files when they reach a certain size that is specified by the MaxFileSize parameter:

```
log4j.appender.Default=log4j.apache.log4j.RollingFileAppender
```

3. Set the logging levels by modifying the `log4j.rootCategory` attribute in the `log4j.properites` file. You can choose one of the following logging levels:

ERROR
Logs error conditions and provides the least amount of logging information.

WARN
Logs information when an operation completes successfully, however, a warning message is displayed.

INFO Logs information about the workflow. It generally explains how an operation occurs. This level is the default level for logging.

DEBUG
Logs all the details that are related to a specific operation. This level is the highest level of logging. If logging is set to DEBUG, all other levels of logging information are displayed in the log file. Because this setting consumes large amounts of system resources, specify `DEBUG` only when directed to do so.

Note: Other IBM Tivoli Directory Integrator components might have their own log levels. The `log4j.rootCategory` attribute setting does not change the settings of those components. For example, `log4j.logger.com.ibm.config` and the `log4j.logger.com.ibm.loader` logging categories are set to `WARN` by default. To control the level of information, either edit the component log level settings to be identical to the `log4j.rootCategory` attribute or comment out the individual component log statement. For example, if you set `log4j.rootCategory=ERROR`, then you must also change the component log level settings to:

```
log4j.logger.com.ibm.di.config=ERROR
log4j.logger.com.ibm.di.loader=ERROR
```

or comment out the statements:

```
# log4j.logger.com.ibm.di.config=WARN
# log4j.logger.com.ibm.di.loader=WARN
```

4. To append information to an existing log file before or after the dispatcher starts, change the value in `log4j.appender.Default.append` in the `log4j.properties` file to true.

```
log4j.appender.Default.append=true
```

5. Save the file.
6. Stop and restart the dispatcher service. See "Start, stop, and restart the Dispatcher service" on page 11

   For more information about logging, see your *IBM Security Directory Integrator Installation and Administrator Guide*.

Service scaling and tuning

On the adapter service form, you can use attributes to scale and tune the Dispatcher instance that runs within the Tivoli Directory Integrator.

Disable AL Caching

The Dispatcher caches assembly lines for the "add, modify, delete" operations. Caching an assembly line retains the connection to the managed resource and might improve performance. However, caching might introduce issues such as memory allocations and timeouts by the managed resource.

To disable assembly line caching for a particular service, check the "Disable AL Caching" option on the service form under the "Dispatcher Attributes" panel.

Note: When a test operation completes successfully, the assembly lines for the service are removed from the Assembly Line cache. In addition, any assembly lines for the service, that are running when the test operation is fired, will not be cached when they complete. So, now the dispatcher will not require a restart if any attribute on the service form is changed and the test operation is completed successfully.

Additional caching options - ALCacheSize

The Dispatcher has a global cache setting. Use the ALCacheSize property in the *ITDI_HOME*/itim_listener.properties file to specify the maximum number of assembly lines that the dispatcher caches for all services. See Table 10 on page 17 for more information.

Max Connection Count

The Dispatcher controls the maximum number of simultaneous connections that all services can run to handle requests. However, you can use the Max Connection Count property to configure individual services to use fewer assembly lines.

To specify the maximum number of assembly lines that the Dispatcher can run simultaneously for the service, enter a positive integer value for "Max Connection Count" on the service form under the "Dispatcher Attributes" panel. A value of 0 implies no limit.

In order for "Max Connection Count" to take effect, the following steps must be done:

1. The GlobalRunALCount, in itim_listener.properties file, must be set to nonzero. A zero setting specifies unlimited assembly lines and ignores any Max Connection Count settings.
2. After changing the value of "Max Connection Count", you must restart the service.

   Note: The dispatcher uses the **HostNameUrl** parameter as a key for the connection pool. Any adapter that uses this feature must provide the **HostNameUrl** parameter.

Additional caching options - GlobalRunALCount
Use the GlobalRunALCount property in the *ITDI_HOME*/itim_listener.properties file to set the upper limit for the maximum number of assembly lines that can be run simultaneously for all services. See "Configuration properties of the Dispatcher" on page 17 for more information.

AL FileSystem Path
Optionally, you can store the assembly lines on the file system where the Dispatcher is running. This field is the full path to where the assembly lines files are located. The assembly file names are the same as specified in the resource.def file.

Use this feature to load customized assembly lines without rebuilding and importing the profile.

For example, if an assembly line file is saved in a directory named "profiles", you must specify the full path to the directory.

For Windows operating systems
c:\Program Files\IBM\TDI*TDI_VERSION*\profiles

For UNIX or Linux operating systems
/opt/IBM/TDI/*TDI_VERSION*/profiles

Transaction timeout

You can configure a transaction timeout for the Dispatcher when transactions fail or take too long to complete. For example, transaction failure occurs when a managed resource is not correctly configured.

You can set the timeout interval for a specific transaction time, such as ADD, Delete, or Reconciliation. The timeout feature does not determine the cause of the delay. The timeout ends the transaction and frees its resources.

After timeout, the Dispatcher ibmdi.log file contains an error message such as:

```
Time Out ....Dispatcher Interrupts Initialization Thread due to AL TimeOut....
```

For example:

```
executeALRequest ():2226 Time Out: 60 request id: 7226427570134735752
Dispatcher Interrupts Initialization Thread due to AL TimeOut.
Service Name :OracleTestService Assembly Line Name is :OracleManageUserAL
```

The IBM Security Identity server marks the service instance that is associated with the adapter. All requests for that service remain pending until IBM Security Identity server determines that the service is up and running.

You can also configure the dispatcher to send a failure for the requests that have timed out, so that IBM Security Identity server does not retry the requests. See "Fail timed out transactions" on page 28.

Transaction timeout settings

There are alternate ways to set transaction timeout on the Dispatcher.

Dispatcher level
Affects all adapters running under the Dispatcher.

Using the itim_listener.properties file in the *TDI_HOME* directory, the following properties set the transaction timeout interval:

- ExecuteSearchALTimeOut

- ExecuteAddALTimeOut
- ExecuteModifyALTimeOut
- ExecuteDeleteALTimeOut

Specify all values in seconds as a positive integer, in an amount of time that your deployment requires. A value of zero (the default) specifies that the transaction timeout interval is unlimited (disabled). To implement a change, restart the Dispatcher.

Service type
Affects all services of the same type. This setting takes precedence over the Dispatcher level setting. Use these properties:

- AddRequestTimeOut
- ModifyRequestTimeOut
- DeleteRequestTimeOut
- SearchRequestTimeOut

Service instance
Affects one service instance only. This setting takes precedence over the Dispatcher level and service type settings. You can specify these attributes:

- *my*AddRequestTimeOut
- *my*ModifyRequestTimeOut
- *my*DeleteRequestTimeOut
- *my*SearchRequestTimeOut

where *my* indicates that you can define the attribute label. For example: **JonesAddRequestTimeOut**

Configuring a service type

To configure a service type setting, you must change the `service.def` files of the adapter profile JAR file.

Procedure

1. Extract the content of the adapter profile JAR file.
2. In the `service.def` file, add the following XML text under each operation:

```
<dispatcherParameter name="AddRequestTimeOut">
   <default> 60 </default >
</dispatcherParameter>

<dispatcherParameter name="ModifyRequestTimeOut">
   <default> 60 </default >
</dispatcherParameter>

<dispatcherParameter name="DeleteRequestTimeOut">
  <default> 60 </default >
</dispatcherParameter>

<dispatcherParameter name="SearchRequestTimeOut">
  <default> 600 </default >
</dispatcherParameter>
```

   Specify all values in seconds as a positive integer, in an amount of time that your deployment requires. A value of zero specifies that the transaction timeout interval is unlimited (disabled). To implement a change, restart the Dispatcher.
3. Re-create the adapter profile JAR file.

4. Import the profile.
5. Restart the Dispatcher.

Configuring a service instance

To configure a service instance setting, you must change the `service.def`, `schema.dsml`, and `CustomLabels.properties` files of the adapter profile JAR file.

Procedure

1. Extract the content of the adapter profile JAR file.
2. In the `schema.dsml` file, create the following attributes and add them to the adapter service object class:

   myAddRequestTimeOut
   myModifyRequestTimeOut
   myDeleteRequestTimeOut
   mySearchRequestTimeOut

   Specify all values in seconds as a positive integer, in an amount of time that your deployment requires. A value of zero specifies that the transaction timeout interval is unlimited (disabled). To implement a change, restart the Dispatcher.
3. For each attribute, add the following statements in the `schema.dsml` file in the attribute definition section. Each attribute must have a unique name and object-identifier.

   <attribute-type single-value = true>
   <name>**myAddRequestTimeOut**</name>
   <description>Time out period of Add request</description>
   <object-identifier>**myAddRequestTimeOut-OID**</object-identifier>
   <syntax>1.3.6.1.4.1.1466.115.121.1.15</syntax>
   </attribute-type>
4. Update the adapter service object class in the `schema.dsml` file to include the new attributes as optional attributes.
5. Modify the `CustomLabels.properties` file to include meaningful labels for the new attributes:

   ```
   MyAddRequestTimeout=Add requests time out
   myModifyRequestTimeout=Modify requests time out
   myDeleteRequestTimeout=Delete requests time out
   mySearchRequestTimeout=Reconciliation requests time out
   ```
6. Modify the `service.def` file to map the service attributes to the dispatcher parameters:

   ```
   <dispatcherParameter name="AddRequestTimeOut" source= "myAddRequestTimeOut">
     <default>60</default>
   </dispatcherParameter>

   <dispatcherParameter name="ModifyRequestTimeOut" source= "myModifyRequestTimeOut">
    <default>60</default>
   </dispatcherParameter>

   <dispatcherParameter name="DeleteRequestTimeOut" source= "myDeleteRequestTimeOut">
    <default>60</default>
   </dispatcherParameter>

   <dispatcherParameter name="SearchRequestTimeOut" source= "mySearchRequestTimeOut">
    <default>600</default>
   </dispatcherParameter>
   ```
7. Re-create the adapter profile JAR file with the updated files.
8. Import the profile.

9. Use the form designer to add the new attributes to the adapter service form.

   Note: You can use one attribute for all timeout values on the service object class by mapping the same attribute to each Dispatcher parameter. You can also use two attributes: one for reconciliation and the other for all of the other operations.
10. Restart the Dispatcher.

Fail timed out transactions

You can configure the dispatcher to send a failure for the requests that have timed out, so that IBM Security Identity server does not retry the requests. You can configure this feature at the dispatcher level, service type level or service instance level.

Dispatcher level

Dispatcher level affects all adapters running under the Dispatcher. Using the `itim_listener.properties` file in the `TDI_HOME` directory, set the following property:

`FailTimeoutRequest`
By setting this value as 1, the IBM Security Identity server will fail the time-out requests when the timeout occurs. Default value is 0, which executes the default behavior, for example, IBM Security Identity server retries the time-out requests.

To implement a change, restart the Dispatcher.

Service type level

Service type level affects all the services of the same type. This setting takes precedence over the Dispatcher level setting.

To configure a service type setting, you must change the service.def files of the adapter profile JAR file.

1. Extract the content of the adapter profile JAR file.
2. In the `service.def` file, add the following XML text under each operation:

   ```
   <dispatcherParameter name="FailTimeoutRequest">
   <default>true</default >
   </dispatcherParameter>
   ```

   Specify **`FailTimeoutRequest`** value in Boolean. A true value implies, the feature is enabled and IBM Security Identity server fails the request when timeout occurs.

   False value implies that the request goes to a pending state and IBM Security Identity server retries the request.
3. Restart the Dispatcher.

Service instance level

Service instance level affects one service instance only. This setting takes precedence over the Dispatcher level and service type level settings.

To configure a service instance setting, you must change the `service.def`, `schema.dsml`, and `CustomLabels.properties` files of the adapter profile JAR file.

1. Extract the content of the adapter profile JAR file.
2. In the schema.dsml file, create an attribute **myFailTimeoutRequest**:

```
<attribute-type single-value = true>
 <name>myFailTimeoutRequest </name>
 <description>Optionally Fail the timed out request</description>
 <object-identifier>myFailTimeoutRequest -OID</object-identifier>
 <syntax> 1.3.6.1.4.1.1466.115.121.1.7</syntax>
</attribute-type>
```

3. Update the adapter service object class in the schema.dsml file to include the new attribute as optional attribute.
4. Modify the CustomLabels.properties file to include a meaningful label for the new attribute:

```
myFailTimeoutRequest = Fail the Timeout Request
```

5. Modify the service.def file to map the service attributes to the dispatcher parameters:

```
<dispatcherParameter name="FailTimeoutRequest " source= " myFailTimeoutRequest">
<default>true</default>
</dispatcherParameter>
```

6. Recreate the adapter profile JAR file with the updated files.
7. Import the profile.
8. Use the form designer to add the new attributes to the adapter service form.
9. Restart the Dispatcher.

Locking feature for assembly line synchronization

As an option, you can synchronize assembly lines at the dispatcher level by using a locking mechanism.

The dispatcher provides a lock to the assembly lines, which must acquire the lock before running code that requires synchronization. The lock must be released after the code is run. Using the lock, assembly lines can achieve synchronization between assembly lines by acquiring and releasing the lock.

For example, an LDAP adapter can use assembly line synchronization after the following changes to schema.dsml and service.def files in the adapter profile:

Note: This example applies to the LDAP adapter. Similar changes must be made to other adapters.

- schema.dsml

  You must change this file if you want to include the **LockName** attribute on the service form. For example:

  1. Attribute Definitions section

```
<!-- ******************************************************** -->
<!-- erLdapLockName                                           -->
<!-- ******************************************************** -->
<attribute-type single-value = "true" >
<name>erLdapLockName</name>
<description>Lock name for AL synchronization</description>
<object-identifier>1.3.6.1.4.1.6054.3.139.2.31</object-identifier>
<syntax>1.3.6.1.4.1.1466.115.121.1.15</syntax>
</attribute-type>
```

  2. RMI Service class section

```
<attribute ref = "erLdapLockName" required = "false" />
```

- service.def

For each operation in the `service.def` file, add a dispatcher parameter. For example:

```
<dispatcherParameter name="LockName" source= "erLdapLockName">
    <default>$(SO!erservicename)</default>
</dispatcherParameter>
```

The source attribute in the **dispatcherParameter** would be required only if the **LockName** value is taken from the service form. If the field is not on the service form, the default value is taken. The **dispatcherParameter** name must always be **LockName**.

This example sets the default value of the lock name to be same as the service name. However, you can change its value based on your requirements.

For example, you might provide it with a default name or add a field on the service form, where the lock name can be set and the default value points to that field. The dispatcher uses the value of the **LockName** dispatcher parameter to create the lock. The lock is created before the assembly line begins to run if a lock with the same name does not already exist.

To acquire and release the lock, you can add code similar to the following code snippet to any hook of your assembly line. However, do not add this in the PROLOG section when assembly line caching is enabled. The PROLOG section is not run again after the assembly line is in the cache.

```
var myALCfg = task.getConfigClone();     //Get AL config object.
var myALSettings = myALCfg.getSettings(); //Get AL settings object from AL config.
var LockName = myALSettingsgetStringParameter("LockName");
task.logmsg("Lock name is"+LockName);
var lock = java.lang.System.getProperties().get(LockName);
var timeout = 240; //The maximum time that AL should wait to acquire the lock.

if ( lock.tryLock(timeout, java.util.concurrent.TimeUnit.SECONDS) )
{
       /*
                  Critical Section
       */
 }
else
 {
      task.logmsg("Failed to acquire lock");
 }
```

The critical section is the interval from when the lock is acquired to the point when it is released. The lock can be released using the following:

```
if (lock!=null)
{
      lock.unlock(); //Releases the lock
}
```

You can add this specification in the same hook, or in any hook. However, you must release the lock at appropriate places, even in error paths if required. Not doing so can cause an **IllegalMonitorStateException** event.

Configuring SSL communication

You must configure Secure Sockets Layer (SSL) communication between the adapters that are based on Tivoli Directory Integrator and the WebSphere® Application Server.

You can configure the Tivoli Directory Integrator to use SSL and also configure WebSphere with the default keystore and default truststore. For more information

about WebSphere SSL configuration, see the WebSphere online help from the WebSphere Application Server Administrative Console.

SSL terminology for adapters

There are several SSL terms that apply to adapters.

SSL server
The workstation on which the Tivoli Directory Integrator is installed is the SSL server. It listens for connection requests.

SSL client
The workstation on which the IBM Security Identity server and WebSphere Application Server are installed. The client submits connection requests to the Tivoli Directory Integrator.

Signed certificates
An industry-standard method of verifying the authenticity of an entity, such as a server, a client, or an application. Signed certificates are issued by a third-party certificate authority for a fee. Some utilities, such as the `iKeyman` utility can also issue signed certificates. Use a certificate authority (CA) certificate to verify the origin of a signed digital certificate.

Signer certificates (CA certificates)
When an application receives the signed certificate of another application, the application uses a CA certificate to verify the originator of the certificate. You can configure many applications. For example, you can configure web browsers with the CA certificates of well-known certificate authorities. This type of configuration can eliminate or reduce the task of distributing CA certificates across the security zones in a network.

Self-signed certificates
A self-signed certificate contains information about the owner of the certificate and the signature of the owner. You can also use a signed certificate as a CA certificate. To use self-signed certificates, you must extract the CA certificate to configure SSL.

SSL keystore
A key database file that is designated as a keystore. The file contains the SSL certificate.

Note: You can use a keystore and truststore as the same physical file.

SSL truststore
A key database file that is designated as a truststore. The SSL truststore contains the list of signer certificates (CA certificates) that define, which certificates the SSL protocol trusts. Only a certificate that is issued by one of the listed trusted signers is accepted.

Note: You can use a keystore and truststore as the same physical file.

One-way SSL communication
For one-way SSL communication, you must have a:

- Keystore and a certificate on the SSL server (the Tivoli Directory Integrator server)
- Truststore on the SSL client-side (the IBM Security Identity server)

Two-way SSL communication
For two-way SSL (client-side) communication, you must have a:

- Keystore with a certificate

- Truststore that contains the signer certificate that issued the certificate from the other side.

You require the keystore and the truststore on the SSL server and the SSL client-side.

One-way and two-way SSL authentication

Configuring communication between an SSL server and client can use one-way or two-way SSL authentication.

For the following tasks, the SSL client is the computer on which the IBM Security Identity server is installed, and the SSL server is theTivoli Directory Integrator.

Configuring SSL for one-way SSL communication

Use one-way SSL communication when the client must authenticate the server.

Before you begin

This procedure requires you to use the following tasks:

- "Creating a keystore for the Tivoli Directory Integrator server" on page 35
- "Creating a truststore for the Tivoli Directory Integrator server" on page 35
- "Creating a self-signed certificate for the Tivoli Directory Integrator server" on page 36
- "Extracting a CA certificate for the Tivoli Directory Integrator" on page 37
- "Importing the Security Directory Integrator CA certificate in the WebSphere Application Server truststore" on page 41
- "Configuring the Tivoli Directory Integrator to use the keystores" on page 38
- "Configuring Tivoli Directory Integrator to use truststores" on page 38
- "Enabling the adapter service to use SSL" on page 39
- "Start, stop, and restart the Dispatcher service" on page 11

About this task

One-way authentication requires a truststore on the client and a keystore on the server. In this example, CA certificate "A" exists in the truststore on the SSL client and also in the keystore on the SSL server. The client sends a request to the SSL server. The SSL server sends Certificate A from the keystore to the client. The client validates Certificate A against the certificates that are contained in the truststore. If the certificate is found in the truststore, the client accepts communication from the SSL server.

The following figure describes SSL configuration for one-way SSL communication.

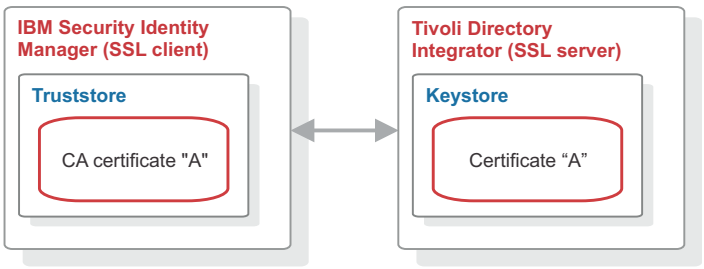

Figure 2. One-way SSL communication (server communication)

Note: IBM Security Identity server uses the existing truststore of theWebSphere Application Server.

Procedure

1. Create a keystore for the Tivoli Directory Integrator server.
2. Create a truststore for the Tivoli Directory Integrator server. One-way SSL communication on the Tivoli Directory Integrator server does not require the truststore. However, you must configure the truststore for the Remote Method Invocation (RMI) SSL initialization.
3. Create a server-signed certificate for the Tivoli Directory Integrator server.
4. Create a CA certificate for the Tivoli Directory Integrator server.
5. Import the Tivoli Directory Integrator CA certificate in the WebSphere Application Server truststore.

   Note: You can modify the `solution.properties` file for steps 6, 7, and 8 in a single operation. When you do so, do not stop and restart the adapter service at the end of steps 6 and 7.
6. Configure the Tivoli Directory Integrator to use keystores.
7. Configure the Tivoli Directory Integrator to use truststores.
8. Enable the adapter service to use SSL.
9. Stop and restart the adapter service.
10. Stop and restart WebSphere Application Server.

Configuring SSL for two-way SSL communication

Use two-way SSL communication when the client must authenticate the server and the server must authenticate the client.

Before you begin

This procedure requires you to use the following tasks:

- “Creating a keystore for the Tivoli Directory Integrator server” on page 35
- “Creating a truststore for the Tivoli Directory Integrator server” on page 35
- “Creating a self-signed certificate for the Tivoli Directory Integrator server” on page 36
- “Extracting a CA certificate for the Tivoli Directory Integrator” on page 37
- “Importing the Security Directory Integrator CA certificate in the WebSphere Application Server truststore” on page 41
- “Configuring the Tivoli Directory Integrator to use the keystores” on page 38
- “Configuring Tivoli Directory Integrator to use truststores” on page 38
- “Enabling the adapter service to use SSL” on page 39

- “Creating a self-signed certificate for the Tivoli Directory Integrator server” on page 36
- “Extracting a WebSphere Application Server CA certificate” on page 40
- “Importing the WebSphere CA certificate in the Tivoli Directory Integrator truststore” on page 37
- “Start, stop, and restart the Dispatcher service” on page 11

About this task

Two-way authentication requires a truststore and a keystore on both the client and the server. In this example, CA certificate "A" exists in the truststore and a CA certificate "B" in the keystore of the client. CA certificate "B" exists in the truststore and a CA certificate "A" in the keystore of the server. The client sends a request to the SSL server. The SSL server sends Certificate A from the keystore to the client. The client validates Certificate A against the certificates that are contained in the truststore.

If the certificate is found in the truststore, the client accepts communication from the SSL server. The server sends an authentication request to the client. The client sends Certificate B from the keystore to the server. The server validates Certificate B against the certificates that are contained in the truststore. If the certificate is found in the truststore, the server accepts communication from the client.

The following figure describes SSL configuration for two-way SSL communication.

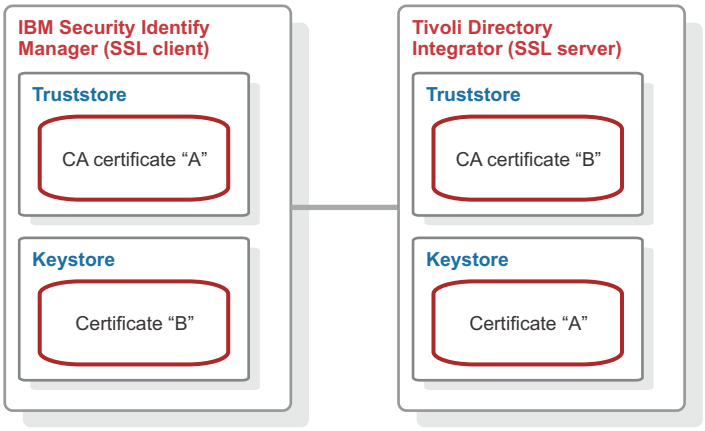

Figure 3. Two-way SSL communication (client communication)

Note: IBM Security Identity server uses the existing truststore and keystore of theWebSphere Application Server.

Procedure

To configure two-way SSL, do the following tasks:

1. Create a keystore for the Tivoli Directory Integrator server.
2. Create a truststore for the Tivoli Directory Integrator server. Do not do this task if you use the same file for keystore and truststore.
3. Create a server-signed certificate for the Tivoli Directory Integrator server.
4. Create a CA certificate for the Tivoli Directory Integrator server.

5. Import the Tivoli Directory Integrator CA certificate in the WebSphere Application Server truststore.

   Note: You can modify the `solution.properties` file for steps 6, 7, and 8 in a single operation. When you do so, do not stop and restart the adapter service at the end of steps 6 and 7.
6. Configure the Tivoli Directory Integrator to use keystores.
7. Configure the Tivoli Directory Integrator to use truststores.
8. Enable the adapter service to use SSL.
9. Create a certificate for the IBM Security Identity server.
10. Create a CA certificate for IBM Security Identity server.
11. Import the WebSphere Application Server CA Certificate in Tivoli Directory Integrator truststore.
12. Stop and restart the adapter service.
13. Stop and restart WebSphere Application Server.

Tasks done on the SSL server

You can configure the Tivoli Directory Integrator as the SSL server.

Complete all tasks on the Tivoli Directory Integrator server workstation.

Note: File names such as `tdikeys.jks` and locations such as *ITDI_HOME*`\keys` are examples. Actual file names and locations might differ.

Creating a keystore for the Tivoli Directory Integrator server

You must create a keystore to hold the certificates that the SSL server uses to authenticate itself to clients.

About this task

A keystore is a database of private keys and the associated certificates that authenticate the corresponding public keys. Digital certificates are stored in a keystore file. A keystore also manages certificates from trusted entities.

Procedure

1. Navigate to the *ITDI_HOME*`/jvm/jre/bin` directory.
2. Start the `ikeyman.exe` file (for Windows operating systems) or `ikeyman` (for UNIX and Linux operating systems).
3. From the **Key Database File** menu, select **New**.
4. Select the key database type of **JKS**.
5. Type the keystore file name. For example, type `tdikeys.jks`.
6. Type the location. For example, type *ITDI_HOME*`/keys`.

   Note: Ensure that location that you specify exists.
7. Click **OK**.
8. Type a password for the keystore. The default password is `secret`.
9. Click **OK**.

Creating a truststore for the Tivoli Directory Integrator server

You must create a truststore on the SSL server to hold trusted certificates, so that clients can authenticate to the server.

About this task

A truststore is a database of public keys for target servers. The SSL truststore contains the list of signer certificates (CA certificates) that define which certificates the SSL protocol trusts. Only a certificate that is issued by one of these listed trusted signers can be accepted. Do not do the following task if you use the same file for keystore and truststore.

Procedure

1. Navigate to the *ITDI_HOME*/jvm/jre/bin directory.
2. Start the ikeyman.exe file (for Windows operating systems) or ikeyman (for UNIX and Linux operating systems).
3. From the **Key Database File** menu, select **New**.
4. Select **JKS**.
5. Type the keystore file name. For example, type tdikeys.jks.
6. Type the location. For example, type *ITDI_HOME*/keys.

   Note: Ensure that location that you specify exists.
7. Click **OK**.
8. Type a password for the keystore. The default password is secret.
9. Click **OK**.

Creating a self-signed certificate for the Tivoli Directory Integrator server

A self-signed certificate contains information about the owner of the certificate and the signature of the owner. This type of certificate is typically used in a testing environment.

Before you begin

To use self-signed certificates, you must extract the CA certificate from the self-signed certificate to configure SSL. See "Extracting a CA certificate for the Tivoli Directory Integrator" on page 37

About this task

A self-signed certificate is a signed certificate and also a CA certificate. To use self-signed certificates, you must extract the CA certificate from the self-signed certificate to configure SSL. You can purchase a certificate from a well-known authority, such as VeriSign. You can also use a certificate server, such as the one included with the MicrosoftWindows 2003 Advanced Server, to generate your own certificates.

Procedure

1. Navigate to the *ITDI_HOME*/jvm/jre/bin directory.
2. Start the ikeyman.exe file (for Windows operating system) or ikeyman (for UNIX and Linux operating systems).
3. From the **Key Database File** menu, select **Open**.
4. Navigate to the keystore file that was created previously: *ITDI_HOME*/keys/tdikeys.jks.
5. Enter the keystore password. The default password is secret.
6. Select **Create > New Self Signed certificate**.

7. Set the Key Label to **tdiserver**.
8. Use your system name (DNS name) as the Common Name (workstation name).
9. Enter the name of your organization. For example, enter `IBM`.
10. Click **OK**.

Extracting a CA certificate for the Tivoli Directory Integrator

Use a CA certificate to verify the origin of a signed digital certificate.

About this task

When an application receives signed certificate of another application, it uses a CA certificate to verify the originator of the certificate. You can configure many applications. For example, you can configure web browsers with the CA certificates of well-known certificate authorities. This type of configuration can eliminate or reduce the task of distributing CA certificates across the security zones in a network.

Procedure

1. Navigate to the *ITDI_HOME*`\jvm\jre\bin` directory.
2. Launch the `ikeyman.exe` file (for Windows operating system) or `ikeyman` (for UNIX and Linux operating system).
3. From the **Key Database File** menu, select **Open**.
4. Navigate to the keystore file that was created previously: *ITDI_HOME*`\keys\tdikeys.jks`
5. Enter the keystore password. The default password is `secret`.
6. Extract the Server certificate for client use by selecting **Extract Certificate**.
7. Select **Binary DER data** as the data type.
8. Enter the certificate file name: `idiserver.der`.
9. Enter the location as *ITDI_HOME*`\keys`.
10. Click **OK**.
11. Copy the `idiserver.der` certificate file to the workstation on which IBM Security Identity server is installed.

Importing the WebSphere CA certificate in the Tivoli Directory Integrator truststore

IBM Security Identity server uses the WebSphere CA certificate, to authenticate to the Tivoli Directory Integrator.

Before you begin

Copy the `timclient.der` SSL Client CA certificate file created in "Extracting a WebSphere Application Server CA certificate" on page 40 to the *ITDI_HOME*`\keys` directory on the workstation on which the Tivoli Directory Integrator is installed.

About this task

After you extract the WebSphere CA certificate, you must import it into the Tivoli Directory Integrator truststore. After it is stored in the truststore, the SSL server can recognize the credentials of the client and authenticate the client.

Procedure

1. Navigate to the *ITDI_HOME*\jvm\jre\bin directory.
2. Start the ikeyman.exe file (Windows operating system) or ikeyman (UNIX and Linux operating system).
3. From the **Key Database File** menu, select **Open**.
4. Select **JKS**.
5. Type the keystore file name: tditrust.jks.
6. Type the location: *ITDI_HOME*\keys and click **OK**.
7. Click **Signer Certificates** in the dropdown menu and click **Add**.
8. Select **Binary DER data** as the data type.
9. Use **Browse** to select the timclient.der file that is stored in *ITDI_HOME*\keys directory.
10. Use **timclient** as the label.
11. Click **OK** to continue.

Configuring the Tivoli Directory Integrator to use the keystores

You can configure the Tivoli Directory Integrator to use the keystores.

Before you begin

You must know the location, password, and type of keystore that you created in "Creating a keystore for the Tivoli Directory Integrator server" on page 35

Procedure

1. Navigate to the *ITDI_HOME*\timsol directory.
2. Open the Tivoli Directory Integrator solution.properties file in an editor.
3. Edit the following lines under **client authentication**:

   ```
   javax.net.ssl.keyStore=ITDI_HOME\keys\tdikeys.jks
   {protect}-javax.net.ssl.keyStorePassword=secret
   javax.net.ssl.keyStoreType=JKS
   ```

   a. Uncomment them, if necessary.
   b. Set the location, password, and type of keystore to match the keystore you created.
4. Save your changes.
5. Stop and restart the adapter service.

   Note: You can modify the solution.properties file in a single operation. Do not stop and restart the adapter service after you configure the Tivoli Directory Integrator to use the keystores and truststores. You can stop and restart the adapter after you enable the adapter service to use SSL.

Related concepts:

"Start, stop, and restart the Dispatcher service" on page 11
When you edit an adapter or Tivoli Directory Integrator properties file, you must stop and restart the Dispatcher service for the changes to take effect.

Related tasks:

"Creating a keystore for the Tivoli Directory Integrator server" on page 35
You must create a keystore to hold the certificates that the SSL server uses to authenticate itself to clients.

Configuring Tivoli Directory Integrator to use truststores

To configure Tivoli Directory Integrator to use the truststores, take these steps:

Procedure

1. Navigate to the *ITDI_HOME*\timsol directory.
2. Open the Tivoli Directory Integrator solution.properties file in an editor.
3. Edit the following lines under **client authentication**:

   ```
   javax.net.ssl.trustStore=ITDI_HOME\keys\tditrust.jks
   {protect}-javax.net.ssl.trustStorePassword=secret
   javax.net.ssl.trustStoreType=JKS
   ```

   a. Uncomment them, if necessary.
   b. Set the location, password, and type of keystore to match the keystore you created.
4. Save your changes.
5. Stop and restart the adapter service.

Note: You can modify the solution.properties file in a single operation. Do not stop and restart the adapter service after you configure the Tivoli Directory Integrator to use the keystores and truststores. You can stop and restart the adapter after you enable the adapter service to use SSL.

"Start, stop, and restart the Dispatcher service" on page 11
When you edit an adapter or Tivoli Directory Integrator properties file, you must stop and restart the Dispatcher service for the changes to take effect.

"Enabling the adapter service to use SSL"
You can enable the adapter service to use SSL.

Enabling the adapter service to use SSL

You can enable the adapter service to use SSL.

Procedure

1. Navigate to the *ITDI_HOME*/timsol directory.
2. Open the Tivoli Directory Integrator solution.properties file in an editor.
3. Edit the following two lines, which depend on the type of secure communications you want to use.

   For no SSL

   ```
   com.ibm.di.dispatcher.ssl=false
   com.ibm.di.dispatcher.ssl.clientAuth=false
   ```

   For one-way SSL

   ```
   com.ibm.di.dispatcher.ssl=true
   com.ibm.di.dispatcher.ssl.clientAuth=false
   ```

   For two-way SSL

   ```
   com.ibm.di.dispatcher.ssl=true
   com.ibm.di.dispatcher.ssl.clientAuth=true
   ```
4. Save your changes.
5. Stop and restart the adapter service.

Tasks done on the SSL client

You must do certain tasks on the SSL client to establish SSL communication between IBM Security Identity server and Tivoli Directory Integrator.

Complete all tasks on the server workstation on which IBM Security Identity server and WebSphere Application Server are installed.

Creating a self-signed certificate for the Tivoli Directory Integrator server

A self-signed certificate contains information about the owner of the certificate and the signature of the owner. This type of certificate is typically used in a testing environment.

Before you begin

To use self-signed certificates, you must extract the CA certificate from the self-signed certificate to configure SSL. See "Extracting a CA certificate for the Tivoli Directory Integrator" on page 37

About this task

A self-signed certificate is a signed certificate and also a CA certificate. To use self-signed certificates, you must extract the CA certificate from the self-signed certificate to configure SSL. You can purchase a certificate from a well-known authority, such as VeriSign. You can also use a certificate server, such as the one included with the MicrosoftWindows 2003 Advanced Server, to generate your own certificates.

Procedure

1. Navigate to the *ITDI_HOME*/jvm/jre/bin directory.
2. Start the ikeyman.exe file (for Windows operating system) or ikeyman (for UNIX and Linux operating systems).
3. From the **Key Database File** menu, select **Open**.
4. Navigate to the keystore file that was created previously: *ITDI_HOME*/keys/tdikeys.jks.
5. Enter the keystore password. The default password is secret.
6. Select **Create > New Self Signed certificate**.
7. Set the Key Label to **tdiserver**.
8. Use your system name (DNS name) as the Common Name (workstation name).
9. Enter the name of your organization. For example, enter IBM.
10. Click **OK**.

Extracting a WebSphere Application Server CA certificate

To establish a secure communication between IBM Security Identity server and the adapter you must extract a WebSphere Application Server CA certificate for IBM Security Identity server.

Procedure

1. Connect to the WebSphere Application Server Administrative Console.
2. Navigate to **Security > SSL certificate and key management > Keystores and certificates**.
3. Select **NodeDefaultKeyStore**.
4. Select **Personal certificates**.
5. Select the check box against the certificate that you created and select **Extract**.
6. Enter a file name: *C*:\keys\timclient.der.
7. Select **Binary DER data** as the data type.
8. Click **OK**.

Importing the Security Directory Integrator CA certificate in the WebSphere Application Server truststore

After you extract a CA certificate from the IBM Security Directory Integrator, you must import the Security Directory Integrator CA certificate in the WebSphere Application Server truststore.

Procedure

1. Copy the SSL server CA certificate file, `idiserver.der`, to the `c:\keys` directory on the workstation on which IBM Security Identity server is installed.
2. Connect to the WebSphere Application Server Administrative Console.
3. Browse to **Security** > **SSL certificate and key managemnet** > **Keystores and certificates**.
4. For a single-server environment, click **NodeDefaultTrustStore** or for a cluster environment, click **CellDefaultTrustStore**.

   Note: For SSL communication between IBM Security Identity Server and LDAP, see https://www.ibm.com/support/knowledgecenter/SSRMWJ_6.0.0.13/com.ibm.isim.doc/installing/tsk/tsk_ic_ins_first_security_ldapcert.htm.
5. Select **Signer certificates**.
6. Click **Add**.
   a. Set the **Alias** to `idiserver`.
   b. Specify the file name of the exported IBM Security Directory Integrator server certificate: `C:\keys\idiserver.der`.
   c. Select **Binary DER data** as the data type.
7. Click **OK** to continue and save.

Chapter 6. Troubleshooting

Troubleshooting is a systematic approach to solving a problem. The goal of troubleshooting is to determine why something does not work as expected and how to resolve the problem. This topic provides information and techniques for identifying and resolving problems that are related to the adapter, including troubleshooting errors that might occur during the adapter installation.

Techniques for troubleshooting problems

Certain common techniques can help with the task of troubleshooting. The first step in the troubleshooting process is to describe the problem completely.

Problem descriptions help you and the IBM technical-support representative find the cause of the problem. This step includes asking yourself basic questions:

- What are the symptoms of the problem?
- Where does the problem occur?
- When does the problem occur?
- Under which conditions does the problem occur?
- Can the problem be reproduced?

The answers to these questions typically lead to a good description of the problem, which can then lead you to a problem resolution.

What are the symptoms of the problem?

When you start to describe a problem, the most obvious question is "What is the problem?" This question might seem straightforward; however, you can break it down into several more-focused questions that create a more descriptive picture of the problem. These questions can include:

- Who, or what, is reporting the problem?
- What are the error codes and messages?
- How does the system fail? For example, is it a loop, hang, crash, performance degradation, or incorrect result?

Where does the problem occur?

Determining where the problem originates is not always easy, but it is one of the most important steps in resolving a problem. Many layers of technology can exist between the reporting and failing components. Networks, disks, and drivers are only a few of the components to consider when you are investigating problems.

The following questions help you to focus on where the problem occurs to isolate the problem layer:

- Is the problem specific to one operating system, or is it common across multiple operating systems?
- Is the current environment and configuration supported?
- Do all users have the problem?
- (For multi-site installations.) Do all sites have the problem?

If one layer reports the problem, the problem does not necessarily originate in that layer. Part of identifying where a problem originates is understanding the environment in which it exists. Take some time to completely describe the problem environment, including the operating system and version, all corresponding software and versions, and hardware information. Confirm that you are running within an environment that is a supported configuration. Many problems can be traced back to incompatible levels of software that are not intended to run together or are not fully tested together.

When does the problem occur?

Develop a detailed timeline of events that lead up to a failure, especially for those cases that are one-time occurrences. You can most easily develop a timeline by working backward: Start at the time an error was reported (as precisely as possible, even down to the millisecond), and work backward through the available logs and information. Typically, you use the first suspicious event that you find in a diagnostic log.

To develop a detailed timeline of events, answer these questions:

- Does the problem happen only at a certain time of day or night?
- How often does the problem happen?
- What sequence of events leads up to the time that the problem is reported?
- Does the problem happen after an environment change, such as upgrading or installing software or hardware?

Responding to these types of questions can give you a frame of reference in which to investigate the problem.

Under which conditions does the problem occur?

Knowing which systems and applications are running at the time that a problem occurs is an important part of troubleshooting. These questions about your environment can help you to identify the root cause of the problem:

- Does the problem always occur when the same task is being done?
- Is a certain sequence of events required for the problem to occur?
- Do any other applications fail at the same time?

Answering these types of questions can help you explain the environment in which the problem occurs and correlate any dependencies. Remember that just because multiple problems might occur around the same time, the problems are not necessarily related.

Can the problem be reproduced?

From a troubleshooting standpoint, the ideal problem is one that can be reproduced. Typically, when a problem can be reproduced you have a larger set of tools or procedures at your disposal to help you investigate. Problems that you can reproduce are often easier to debug and solve.

However, problems that you can reproduce can have a disadvantage: If the problem is of significant business impact, you do not want it to recur. If possible, re-create the problem in a test or development environment, which typically offers you more flexibility and control during your investigation.

- Can the problem be re-created on a test system?

- Do multiple users or applications have the same type of problem?
- Can the problem be re-created by running a single command, a set of commands, or a particular application?

Logs

When the adapter is initially configured, a default directory is selected to store the log files that record the adapter activities. Logs can help you determine the background or cause of an issue and to find the proper solution.

Logs added to the log file for the adapter or the Dispatcher have the following format:

```
<Log Level> [<Assembly Line_ProfileName>_<Request Id>]_
[<Connector Name>] - <message>
```

Log Level
Specifies the logging level that you configured for the adapter. The options are DEBUG, ERROR, INFO, and WARN. See "Configuring logging for the adapter" on page 22 for information about using the log4j.properties file to configure logging.

Assembly Line
Specifies the name of the assembly line that is logging the information.

ProfileName
Specifies the name of the profile. Profile names might vary based on the adapter that is running or the operating system.

Request ID
Specifies the number of the request. Request number is used to uniquely identify a specific request.

Connector Name
Specifies the connector for the adapter.

message
Specifies the actual message information.

The example below is an actual message that might be displayed in a log file:

```
INFO [AssemblyLine.AssemblyLines/DispatcherAdd_Ldapprofile_518536692232324188_
91ea4bb8-2801-11b2-91ba-00000a2c0670.1297881434 - Load Attribute Map
```

Tivoli Directory Integrator Application Monitoring console

The Tivoli Directory Integrator Application Monitoring console routes all the Remote Method Invocation (RMI) requests that are sent to the Tivoli Directory Integrator to a specified port.

The port is specified in the `api.remote.naming.port` property in the *ITDI_HOME*/*timsol*/`solutions.properties` file.

To route the RMI requests to another port, do either of the following tasks:

- Change the port number that is specified in the Tivoli Directory Integrator location field on the service form to the number specified in `api.remote.naming.port` property of the `solution.properties` file.
- Change the port number that is specified in the `api.remote.naming.port` property of the `solution.properties` file to the number specified in the Tivoli Directory Integrator location field on the service form.

Troubleshooting the dispatcher while using SSL Configuration

After some amount of usage, maybe an hour or so, SSL connections from IBM Security Identity server to the Dispatcher stop working because the RMI Registry loses it's reference to the SSL Connection Factory.

If Connection reset errors are found, set the property `systemqueue.on=false` in the `solution.properties` file:

1. Go to `TDI_HOME\timsol\solution.properties`.
2. Set `systemqueue.on=false` and save the file
3. Restart the Dispatcher service.

Verifying that the correct level of Tivoli Directory Integrator is installed

You must check the version level date in the `ibmdi.log` file to determine the level of the installed Tivoli Directory Integrator.

Depending on your adapter requirements, ensure that the correct version is installed. See the Release Notes that accompanied your adapter for information about the Tivoli Directory Integrator version and fix pack level.

To verify the level of Tivoli Directory Integrator, check the `ibmdi.log` file. The log shows version levels up to three levels `x.x.x`. The date is the only way to verify the Tivoli Directory Integrator fix pack level.

Installer problems on UNIX and Linux operating systems

Interruptions during the Dispatcher installation or running an unsupported JVM can cause installation problems.

The Dispatcher installer creates temporary files during installation. On the UNIX and Linux platforms these files are in the `/tmp` directory. These temporary files might cause subsequent installations to fail or not to work correctly, if either of the following conditions occur:

- The installation is interrupted.
- The installer ran with an unsupported JVM.

Symptoms

- The installation completes successfully, however, the solution directory is not created.
- The installation completes successfully, however, the solution directory is created as a file instead of a directory.

Corrective action

1. Remove any of the following files from the `/tmp` directory:

```
ITDIAsService.sh
rmITDIAsService.sh
deldispatcher.sh
createdir.sh
copyfiles.sh
copyagentfile.sh
delfiles.sh
copylog4j.sh
```

2. Run the uninstaller.

3. Edit the *ITDI_HOME*/etc/global.properties file to remove the following properties:

   ```
   ADAPTER_SOLDIR
   com.ibm.di.dispatcher.registryPort
   com.ibm.di.dispatcher.bindName
   com.ibm.di.dispatcher.ssl
   com.ibm.di.dispatcher.clientAuth
   com.ibm.di.dispatcher.disableConnectorCache
   ITDI_HOME
   ```

4. Remove the following JAR files from the *ITDI_HOME*/jars/3rdparty/IBM directory:

   ```
   itdiAgents.jar
   itdiAgents-common.jar
   rmi-dispatcher-client.jar
   rmi-dispatcher.jar
   ```

5. Remove the following JAR files from the *ITDI_HOME*/jars/3rdparty/others directory:

   ```
   jakarta-regexp-1.4.jar
   antlr-2.7.2.jar
   ```

6. Delete the *timsol* directory of file.
7. Run the installer again with the correct JVM.

Log output from the ITIMAd script

On UNIX and Linux systems, you can use the ITIMAd script to start, stop, and restart the Dispatcher service.

The ITIMAd script logs its output to a separate ITIMAd_stdout.log file in the /opt/IBM/TDI/*TDI_Version*/timsol directory.

If a problem occurs, examine the output in the log file, which describes the Dispatcher start, stop, or restart operation.

RMI configuration to traverse firewalls

If you have a firewall enabled, you must manually set the object port number.

To manually set the object port number, see the description of the **com.ibm.di.dispatch.objectPort** configuration property in Table 10 on page 17.

Chapter 7. Uninstalling

The Dispatcher is required for all adapters that are based on Tivoli Directory Integrator. If you uninstall the Dispatcher, none of the other installed adapters function.

About this task

The mode used to uninstall the Dispatcher depends on which mode was used to install the Dispatcher.

If you install the Dispatcher in GUI mode, you can uninstall it in GUI mode, console mode, or silent mode.

If you install the Dispatcher by using console mode, then you can uninstall the Dispatcher only with console mode or silent mode.

If you install the Dispatcher by using silent mode, then the uninstaller runs in silent mode regardless of whether you use the `-i silent` option.

When you uninstall the Dispatcher, the uninstaller creates a backup of the `itim_listener.properties` file. For more information, see "Backup of the `itim_listener.properties` file" on page 51.

Procedure

1. Navigate to the Dispatcher uninstaller folder.
2. Run one of the following commands:
   - To run the uninstaller in GUI mode, use the following command:
     TDI_HOME`/jvm/jre/bin/java –jar uninstaller.jar`
   - To run the uninstaller in console mode, use the following command:
     `ITDI_HOME/jvm/jre/bin/java -jar uninstaller.jar -i console`
   - To run the uninstaller in silent mode, use the following command:
     `ITDI_HOME/jvm/jre/bin/java -jar uninstaller.jar -i silent`

Results

The Dispatcher is uninstalled and the uninstaller creates a backup of the `itim_listener.properties`.

Chapter 8. Reference

Reference information is organized to help you locate particular facts quickly, such as adapter attributes, registry settings, and environment variables.

Backup of the `itim_listener.properties` file

The `itim_listener.properties` file is a Dispatcher configuration file in the `TDI_HOME` directory.

When you upgrade the dispatcher component, the dispatcher replaces the `itim_listener.properties` file with a new version while the installer creates a backup of the original file.

Similarly, when you uninstall the dispatcher component, the uninstaller creates a backup of the `itim_listener.properties` file.

The backup is created in the following format:

`format.itim_listener.000itim_listener.001itim_listener.002`

where *.000*, *.001*, and so on, indicates the version level.

Index

Printed in USA