Dispatcher10 Installation and Configuration Guide

IBM

Contents

Table of Contents

Figures

Tables

Chapter 1. Overview

The Dispatcher10 is a key component for adapters that are based on Security Verify Directory Integrator. The Dispatcher10 provides the link between a Verify Governance (Identity Manager) server and the IBM Security Verify Directory Integrator.

The Dispatcher10 is not installed with the base Security Verify Directory Integrator product. It must be installed separately to enable the Security Directory Verify Integrator-based adapters to run.

The Dispatcher10 runs as an instance of the Security Directory Verify Integrator and is a prerequisite to install and run all Security Verify Directory Integrator-based adapters. Multiple adapters can be installed on the same Security Verify Directory Integrator where the Dispatcher10 is installed. The adapters consist of assembly line configurations that initialize and run Security Verify Directory Integrator connectors.

The Dispatcher10 is the user management API for the Security Verify Directory Integrator provider. The Dispatcher10 loads and runs assembly line configurations specified by the Security Verify Directory Integrator provider.

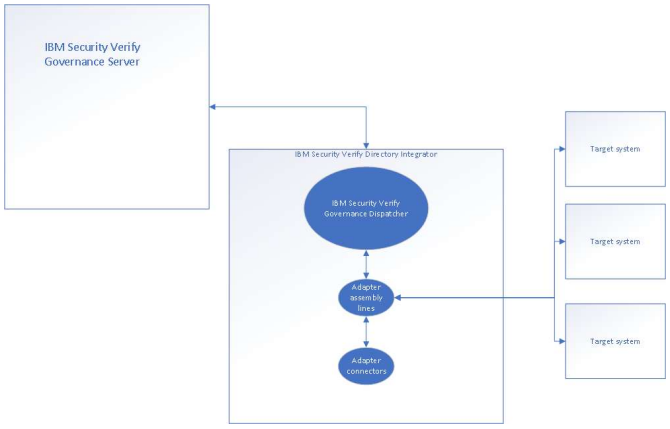

Figure 1. The architecture of the Dispatcher

For more information about Security Verify Directory Integrator, see

https://www.ibm.com/docs/en/svdi/10.0.

Chapter 2. Planning

Installing and configuring the Dispatcher10 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” 3.
Obtain the installation software.	Software downloads.
Obtain the necessary information for the installation and configuration.	“Installation worksheet” 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 Dispatcher10 in GUI mode” 7.
Verify the installation.	See “Verifying the adapter installation” 10.
Configure the dispatcher.	See “Configuring the Dispatcher” 15.

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**
Verify Directory Integrator	• See the release notes
Verify Governance server	The following servers are supported: • see the release notes
Operating system	See the release notes
System Administrator Authority	The person who performs the Dispatcher10 installation procedure must have root authority to complete the steps in this chapter.

The Dispatcher10 must be installed on the same server as the Security Verify Directory Integrator server. For information about the system requirements and supported operating systems for Security Verify Directory Integrator, see https://www.ibm.com/software/reports/compatibility/clarity-reports/report/html/osForProduct

Security Verify Directory Integrator adapters solution directory

The adapters solution directory is a Security Verify Directory Integrator work directory for adapters.

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

- The adapters solution directory
- The Security Verify Directory Integrator home directory

The Dispatcher10 installation prompts you to enter the complete path of the adapter solution directory. For example, enter *C*:\Program Files\ibm\SDI\V10.0*timsol*, where *timsol* is the adapter solution directory. The parent directory that you enter for the adapter solution directory must exist.

For every subsequent Dispatcher10 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 Dispatcher10 correctly and to avoid errors during the installation, do not use the Security Verify 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 *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
Security Verify Directory Integrator Home Directory	The *SDI_HOME* directory contains the jars/connectorssubdirectory that contains adapter JAR files. For example, the jars/connectors subdirectory contains the JAR file for the UNIX adapter.	If Security Verify Directory Integrator is automatically installed for version 10.0, the default directory path depends **UNIX and Linux operating systems** /opt/IBM/SDI/V10.0
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 "Security Verify Directory Integrator adapters directory" 4.	The default solution directory for version 10.0 depends on the operating system. **UNIX and Linux operating systems** /opt/IBM/SDI/V10.0/*timsol*

Chapter 3. Dispatcher Container

This is a containerized form of Dispatcher10 built on top of Security Verify Directory Integrator Base image.

This dispatcher container utilizes persistent volumes to store adapter files (connectors, 3rd party jars, Property Files, XSL files, Keystores, etc.). To install the adapters into the container in Docker or Kubernetes environments, their respective adapter util scripts can be used. These utility scripts are shipped with Dispatcher10 package which can be obtained from the IBM Passport Advantage website, see "Software download" of this guide. See the Dispatcher Container Utility Script (adapterUtil.sh) of this guide for more details about the script.

Configuring

Configuration format

See Configuration Format under the *Containers* chapter of the *IBM Security Verify Directory Integrator* documentation for details.

Configuration example

The following examples are configuration YAML files for the IBM® Security Verify Directory Integrator Dispatcher container.

The following YAML shows the minimum configuration to start a IBM® Security Verify Directory Integrator Dispatcher container.

```
general:
  license:
    key:    "insert-license-key-here"
    accept: true

server:
  assembly-line-file: "ITIM_RMI.xml"
```

The following example is a more typical and complete YAML configuration.

```
general:
  memory:
    minimum: 256
    maximum: 512

  logging:
    json-logging: false

  license:
    accept: true
    key: "insert-license-key-here"

server:
  assembly-line-file: "ITIM_RMI.xml"

keyfile:
  keys:
  - label: "server-key"
    key:   "@/opt/IBM/dispatcher/config/server.key"
```

Note: Details on the configuration entries that are supported by the containers can be found in the YAML

specification.

YAML specification

See Verify Directory Integrator Base under the *Containers* chapter of the *IBM Security Verify Directory Integrator* documentation for details.

Deploying

Docker

The containers can be deployed in a Docker environment.

Repository

The IBM® Security Verify Directory Integrator images are available from the IBM Cloud Repository: *icr.io/isvdi*. To load the image from the repository, the 'docker pull' command can be used. The image name can be supplied with the pull command, along with a tag that corresponds to the image version number. For example,

```
docker pull icr.io/isvdi/verify-directory-integrator-dispatcher:latest
```

Security considerations

See Security Considerations under the Containers chapter of the IBM Security Verify Directory Integrator documentation for details.

Quick start

A command to start a IBM® Security Verify Directory Integrator Dispatcher container, which would specify a bind mounted configuration volume, a data volume, and standard environment variables, is shown by the following example.

```
docker run --hostname ibm-svdi-dispatcher-test --name ibm-svdi-dispatcher-test \
   --detach \
   --volume /Users/test/ibm-svdi-dispatcher-config:/opt/IBM/dispatcher/config \
   --volume ibm-svdi-dispatcher-test-volume:/opt/IBM/svgadapters \
   --env YAML_CONFIG_FILE=/var/IBM/svdi/config/config.yaml \
   icr.io/isvdi/verify-directory-integrator-dispatcher:latest

## Examine the log file of the container.
docker logs -f ibm-svdi-dispatcher-test
```

Kubernetes

Creating the Dispatcher Container

Use the following steps to create the Dispatcher container.

1. Ensure that the kubectl context is set to the correct environment. The mechanism differs based on the Kubernetes environment in use.
2. Create a ConfigMap definition file that is named ibm-svdi-dispatcher-config.yaml. This definition file contains the configuration YAML for the Directory Integrator Dispatcher container. For example,

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ibm-svdi-dispatcher-config
  namespace: default
data:
  config.yaml: |
    general:
      license:
        key: "add-your-license-key-here"
        accept: true

    server:
      assembly-line-file: "ITIM_RMI.xml"
```

3. Create the ConfigMap.

```
kubectl create -f ibm-svdi-dispatcher-config.yaml
```

4. Create the persistent volume claim that is used by the container to store the Directory Integrator server data. The mechanism varies based on the Kubernetes environment in use. The following example shows how to create a persistent volume claim in an IBM cloud environment.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ibm-svdi-dispatcher-claim
  labels:
    billingType: "hourly"
    region: au-syd
    zone: syd04
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 45Gi
  storageClassName: ibmc-block-silver
```

5. Create a deployment file that is named ibm-svdi-dispatcher-container.yaml. This deployment file defines a Directory Integrator Dispatcher container for your environment.

```
#
# The deployment description of the Verify Directory Integrator Dispatcher container.
#

apiVersion: apps/v1
kind: Deployment

metadata:
 name: ibm-svdi-dispatcher
 labels:
   app: ibm-svdi-dispatcher

spec:
 selector:
   matchLabels:
     app: ibm-svdi-dispatcher

 template:
   metadata:
     labels:
       app: ibm-svdi-dispatcher

   spec:
     volumes:
     - name: config-volume
       configMap:
         name: ibm-svdi-dispatcher-config
     - name: data-volume
       persistentVolumeClaim:
         claimName: ibm-svdi-dispatcher-claim

     containers:
     - name: ibm-svdi-dispatcher

       # The fully qualified name of the image.
       image: icr.io/isvdi/verify-directory-integrator-dispatcher:latest

       # Environment definition.
       env:
       - name: YAML_CONFIG_FILE
         value: /opt/IBM/dispatcher/config/config.yaml

       # The configuration volume.
       volumeMounts:
       - name: config-volume
         mountPath: /opt/IBM/dispatcher/config
       - name: data-volume
         mountPath: /opt/IBM/svgadapters
```

6. Create the container.

```
kubectl create -f ibm-svdi-container.yaml
```

7. You can monitor the bootstrapping of the container by using the logs command.

```
kubectl logs -f `kubectl get pods | grep "ibm-svdi-dispatcher" | awk '{ print $1 }'`
```

Kubernetes environments

See Kubernetes Environments under Containers chapter of IBM Security Verify Directory Integrator documentation for details.

Security considerations

See Security Considerations under Containers chapter of IBM Security Verify Directory Integrator documentation for details.

License monitoring

See License Monitoring under Containers chapter of IBM Security Verify Directory Integrator documentation for details.

Starting dispatcher container

To start dispatcher container in Docker or Kubernetes environment, refer sample commands as given below:

Docker command:

```
docker start <container_name>
```

Kubernetes command:

```
Kubectl start deployment <deployment_name>
```

Note: These commands are just samples, the commands may differ based on the Kubernetes environment being used for dispatcher container.

Stopping dispatcher container

To stop dispatcher container in Docker or Kubernetes environment, refer sample commands as given below:

Docker command:

```
docker stop <container_name>
```

Kubernetes command:

```
Kubectl stop deployment <deployment_name>
```

Note: These commands are just samples, the commands may differ based on the Kubernetes environment being used for dispatcher container.

Restarting dispatcher container

To start dispatcher container in Docker or Kubernetes environment, refer sample commands as given below:

Docker command:

```
docker restart <container_name>
```

Kubernetes command:

```
Kubectl rollout restart deployment <deployment_name>
```

Note: These commands are just samples, the commands may differ based on the Kubernetes environment being used for dispatcher container.

Enabling TLSv1.2 or TLSv1.3 protocol

To enable TLSv1.2 protocol, add below entries to dispatcher config.yaml file:

```
advanced:
 - attr: com.ibm.di.SSLProtocols
   value: 'TLSv1.2'
 - attr: com.ibm.di.SSLServerProtocols
   value: 'TLSv1.2'
```

Similarly, if TLSv1.3 protocol has to be enabled then, entries similar to above example needs to be added to dispatcher config.yaml file and replace TLSv1.2 with TLSv1.3.

See YAML Schema documentation for ISVDI container for details.

Logging and Auditing

See YAML Schema documentation for ISVDI container to add **root-level** and **json-logging** configuration entries to the dispatcher config.yaml file. To enable debug logs, set value for **root-level** to **debug** and to disable json logging, set value for **json-logging** element to **false**. Here is an example of the entries:

```
general:
  license:
    accept: true
    key: "see the contents of SVDI_10.0_Con_Lic_Key_ML.txt which can be downloaded from xtreme leverage"
  logging:
    json-logging: false
    root-level: "debug"
```

The ibmdi logs can then be captured using the logs commands from Docker or Kubernetes. Here are couple the examples for the commands for Docker and Kubernetes environments:

```
docker logs -f <isvdi-dispatcher-container-instance>

kubectl logs -f <isvdi-dispatcher-pod-instance>
```

See Logging and Auditing under *Containers* chapter of *IBM Security Verify Directory Integrator* documentation for details.

Dispatcher Container Utility Script (adapterUtil.sh)

The adapterUtil.sh script will load the necessary files for a given adapter into the ISVDI container. It can also

be used to list the currently installed adapters, and retrieve details about a given adapter. In Kubernetes environment, to support multiple clusters of ISVDI servers, where each cluster can run different adapters, an optional deployment name can be provided, and the script will connect to a pod in the specified cluster. If not supplied, the default "isvdi" deployment will be used. Similarly, namespace name can be provided, and the script will connect to a pod in the specified namespace. If no namespace is specified, "default" namespace will be used.

Note: After adding an adapter or related files or updating any configuration requires these changes to be activated. See Activating changes to the configuration of this guide for details.

Working with adapterUtil.sh script

Displaying help menu

Use the following command to display help menu:

```
./adapterUtil.sh -help
```

Displaying adapterUtil script version

Use the following command to display adapterUtil.sh script version:

```
../adapterUtil.sh -version
```

Working with adapters

Loading an adapter

Use the following command to load the adapter into the container:

```
./adapterUtil.sh -loadAdapter /path/to/Adapter.zip accept
```

The first parameter to the option is path to the adapter zip file on the machine running the adapterUtil.sh script and the second parameter is the acceptance of the adapter license. An example of the usage for this option is as below:

```
./adapterUtil.sh -loadAdapter /tmp/stagingDirectory/Adapter-IBMSecurityVerify-10.0.9.zip accept
```

Listing adapters

Use the following command to the list of adapters installed using adapterUtil.sh script:

```
./adapterUtil.sh -listAdapters
```

Displaying info of an adapters

Use the following command to display the information about an adapter installed using adapterUtil.sh:

```
./adapterUtil.sh -infoAdapter <Adapter_Name>
```

The first parameter to the option is name of the adapter. An example of the usage for this option is as below:

```
./adapterUtil.sh -infoAdapter IBMSecurityVerifySaaS
```

Removing an adapter

Use the following command to remove the adapter which was installed using adapterUtil.sh:

```
./adapterUtil.sh -removeAdapter <Adapter_Name>
```

The first parameter to the option is name of the adapter. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeAdapter IBMSecurityVerifySaaS
```

Working with container instance

Running adapterUtil options against a container/pod instance (Docker only)

Use the following command to execute any options from the adapterUtil script against any container instance:

```
./adapterUtil.sh -dispatcherPodName <Container_Name> <Other_option> <parameters for other option>
```

The first parameter to the dispatcherPodName option is the name of the target container instance and the other option can be any of the options supported by adapterUtil to perform any file copy / list / remove operation. An example of the usage for this option is as below:

```
./adapterUtil.sh -dispatcherPodName isvdi2 -removeAdapter IBMSecurityVerifySaaS
```

Running adapterUtil options against a deployment (Kubernetes only)

Use the following command to execute any options from the adapterUtil script against any dispatcher deployment:

```
./adapterUtil.sh -deploymentName <Deployment_Name> <Other_option> <parameters for other option>
```

The first parameter to the deploymentName option is the name of the target dispatcher deployment and the other option can be any of the options supported by adapterUtil to perform any file copy / list / remove operation. An example of the usage for this option is as below:

```
./adapterUtil.sh -deploymentName isvdi2 -removeAdapter IBMSecurityVerifySaaS
```

Running adapterUtil options against a namespace (Kubernetes only)

Use the following command to execute any options from the adapterUtil script against any namespace:

The first parameter to the namespace option is the name of the target Kubernetes namespace and the other

```
./adapterUtil.sh -namespace <Namespace_Name> <Other_option> <parameters for other option>
```

option can be any of the options supported by adapterUtil to perform any file copy / list / remove operation. An example of the usage for this option is as below:

```
./adapterUtil.sh -namespace isvgim2 -removeAdapter IBMSecurityVerifySaaS
```

This can also be used in combination with the deploymentName as below:

```
./adapterUtil.sh -namespace isvgim2 -deploymentName isvdi2 -removeAdapter IBMSecurityVerifySaaS
```

Working with external keystores

Copying an external keystore

Use the following command to copy external keystore file (jks, p12, etc.) to the **/opt/IBM/svgadapters/timsol/keystores** directory:

```
./adapterUtil.sh -copyToExternalKeystore /path/to/keystore_file
```

The first parameter to this option is the path to the keystore file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
./adapterUtil.sh -copyToExternalKeystore /tmp/stagingDirectory/keystore.jks
```

Listing external keystores

Use the following command to list the keystore files in the **/opt/IBM/svgadapters/timsol/scripts** directory:

```
./adapterUtil.sh -listExternalKeystoreFiles
```

Removing an external keystore

Use the following command to remove a keystore file from the **/opt/IBM/svgadapters/timsol/keystores** directory:

```
./adapterUtil.sh -removeFromExternalKeystore <Keystore_Filename>
```

The first parameter to the option is the name of the keystore file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeFromExternalKeystore keystore.jks
```

Working with external dependency jars

Copying an external dependency jar

Use either of the following command to copy an external dependency jar file to the **/opt/IBM/svgadapters/jars/3rdparty/others** or **/opt/IBM/svgadapters/jars/patches** directory:

```
./adapterUtil.sh -copyTo3rdpartyOthers /path/to/external_dependency_jar_file

OR

./adapterUtil.sh -copyToPatches /path/to/external_dependency_jar_file
```

The first parameter to this option is the path to the connector jar file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
./adapterUtil.sh -copyTo3rdpartyOthers /tmp/stagingDirectory/httpclient-4.4.16.jar

OR

./adapterUtil.sh -copyToPatches /tmp/stagingDirectory/httpclient-4.4.16.jar
```

Listing external dependency jars

Use the following command to list the external dependency jar files in the **/opt/IBM/svgadapters/jars/3rdparty/others** or **/opt/IBM/svgadapters/jars/patches** directory:

```
./adapterUtil.sh -listFrom3rdpartyOthers

OR

./adapterUtil.sh -listFromPatches
```

Removing an external dependency jar

Use the following command to remove an external dependency jar file from the **/opt/IBM/svgadapters/jars/3rdparty/others** or **/opt/IBM/svgadapters/jars/patches** directory:

```
./adapterUtil.sh -removeFrom3rdpartyOthers <External_Dependency_Jar_Filename>

OR

./adapterUtil.sh -removeFroPatches <External_Dependency_Jar_Filename>
```

The first parameter to the option is the name of the external dependency jar file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeFrom3rdpartyOthers httpclient-4.4.16.jar


OR

./adapterUtil.sh -removeFromPatches httpclient-4.4.16.jar
```

Working with connectors

Copying a connector jar

Use the following command to copy connector jar file to the **/opt/IBM/svgadapters/jars/connectors** directory:

```
./adapterUtil.sh -copyToConnectors /path/to/connector_jar_file
```

The first parameter to this option is the path to the connector jar file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
./adapterUtil.sh -copyToConnectors /tmp/stagingDirectory/CustomConnector.jar
```

Listing connector jars

Use the following command to list the connector jar files in the **/opt/IBM/svgadapters/jars/connectors** directory:

```
./adapterUtil.sh -listConnectorsFiles
```

Removing a connector jar

Use the following command to remove a connector jar file from the **/opt/IBM/svgadapters/jars/connectors** directory:

```
./adapterUtil.sh -removeFromConnectors <Connector_Jar_Filename>
```

The first parameter to the option is the name of the connector jar file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeFromConnectors CustomConnector.jar
```

Working with functions

Copying a functions jar

Use the following command to copy functions jar file to the **/opt/IBM/svgadapters/jars/functions** directory:

```
./adapterUtil.sh -copyToFunctions /path/to/functions_jar_file
```

The first parameter to this option is the path to the functions jar file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
./adapterUtil.sh -copyToFunctions /tmp/stagingDirectory/CustomFunctions.jar
```

Listing functions jars

Use the following command to list the functions jar files in the **/opt/IBM/svgadapters/jars/functions** directory:

```
./adapterUtil.sh -listFunctionsFiles
```

Removing a functions jar

Use the following command to remove a functions jar file from the **/opt/IBM/svgadapters/jars/functions** directory:

```
./adapterUtil.sh -removeFromFunctions <Functions_Jar_Filename>
```

The first parameter to the option is the name of the functions jar file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeFromFunctions CustomFunctions.jar
```

Working with library files

Copying a library file

Use the following command to copy library file (so or dll) to the **/opt/IBM/svgadapters/libs** directory:

```
./adapterUtil.sh -copyToLibs /path/to/lib_file
```

The first parameter to this option is the path to the library file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
./adapterUtil.sh -copyToLibs /tmp/stagingDirectory/CustomFile.so
```

Listing library files

Use the following command to list the library files in the **/opt/IBM/svgadapters/libs** directory:

```
./adapterUtil.sh -listLibsFiles
```

Removing a library file

Use the following command to remove a library file from the **/opt/IBM/svgadapters/libs** directory:

```
./adapterUtil.sh -removeFromLibs <Library_Filename>
```

The first parameter to the option is the name of the library file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeFromLibs CustomFile.so
```

Working with properties files

Copying a properties file

Use the following command to copy properties file to the **/opt/IBM/svgadapters/timsol/properties** directory:

```
./adapterUtil.sh -copyToProperties /path/to/properties_file
```

The first parameter to this option is the path to the properties file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
./adapterUtil.sh -copyToProperties /tmp/stagingDirectory/CustomFile.properties
```

Listing Properties files

Use the following command to list the properties files in the **/opt/IBM/svgadapters/timsol/properties** directory:

```
./adapterUtil.sh -listPropertiesFiles
```

Viewing Properties files

Use the following command to list the properties files in the **/opt/IBM/svgadapters/timsol/properties** directory:

```
./adapterUtil.sh -viewPropertiesFile <Properties_Filename>
```

The first parameter to the option is the name of the properties file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -viewPropertiesFile CustomFile.properties
```

Removing a properties file

Use the following command to remove a properties file from the **/opt/IBM/svgadapters/timsol/properties** directory:

```
./adapterUtil.sh -removeFromProperties <Properties_Filename>
```

The first parameter to the option is the name of the functions jar file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeFromProperties CustomFile.properties
```

Working with xsl files

Copying a xsl file

Use the following command to copy xsl file to the **/opt/IBM/svgadapters/timsol/xsl** directory:

```
./adapterUtil.sh -copyToXsl /path/to/xsl_file
```

The first parameter to this option is the path to the xsl file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
./adapterUtil.sh -copyToXsl /tmp/stagingDirectory/CustomFile.xsl
```

Listing xsl files

Use the following command to list the xsl files in the **/opt/IBM/svgadapters/timsol/xsl** directory:

```
./adapterUtil.sh -listXslFiles
```

Removing a xsl file

Use the following command to remove a xsl file from the **/opt/IBM/svgadapters/timsol/xsl** directory:

The first parameter to the option is the name of the xsl file to be removed. An example of the usage for this option

```
./adapterUtil.sh -removeFromXsl <Xsl_Filename>
```

is as below:

```
./adapterUtil.sh -removeFromXsl CustomFile.xsl
```

Working with script files

Copying a script file

Use the following command to copy script file to the **/opt/IBM/svgadapters/timsol/scripts** directory:

```
./adapterUtil.sh -copyToScripts /path/to/script_file
```

The first parameter to this option is the path to the script file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
./adapterUtil.sh -copyToScripts /tmp/stagingDirectory/CustomFile.sh
```

Listing script files

Use the following command to list the script files in the **/opt/IBM/svgadapters/timsol/scripts** directory:

```
./adapterUtil.sh -listScriptsFiles
```

Removing a script file

Use the following command to remove a script file from the **/opt/IBM/svgadapters/timsol/scripts** directory:

```
./adapterUtil.sh -removeFromScripts <Script_Filename>
```

The first parameter to the option is the name of the script file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeFromScripts CustomFile.sh
```

Working with swidtag files

Copying a swidtag file

Use the following command to copy swidtag file to the **/opt/IBM/svgadapters/swidtag** directory:

```
./adapterUtil.sh -copyToSwidtag /path/to/swidtag_file
```

The first parameter to this option is the path to the swidtag file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
./adapterUtil.sh -copyToSwidtag /tmp/stagingDirectory/CustomFile.swidtag
```

Listing swidtag files

Use the following command to list the swidtag files in the **/opt/IBM/svgadapters/swidtag** directory:

```
./adapterUtil.sh -listSwidtagFiles
```

Removing a swidtag file

Use the following command to remove a swidtag file from the **/opt/IBM/svgadapters/swidtag** directory:

```
./adapterUtil.sh -removeFromSwidtag <Swidtag_Filename>
```

The first parameter to the option is the name of the swidtag file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeFromSwidtag CustomFile.swidtag
```

Working with custom files

Copying a custom file

Use the following command to copy custom file to the any directory in the container:

```
./adapterUtil.sh -copyFile localFile containerPath
```

The first parameter to this option is the path to the file on the machine running the adapterUtil.sh script and the second parameter is the directory path inside the container where the file needs to be copied. An example of the usage for this option is as below:

```
./ adapterUtil.sh -copyFile /tmp/stagingDirectory/customFile.txt /opt/IBM/svgadapters/customFiles
```

Listing custom files

Use the following command to list the files in the specified path inside the container:

```
./adapterUtil.sh -listFiles /opt/IBM/svgadapters/customFiles/customFile.txt
```

Removing a custom file

Use the following command to remove a file from the specified path inside the container:

```
./adapterUtil.sh -removeFile containerPath
```

The first parameter to the option is the name of the file to be removed. An example of the usage for this option is as below:

```
./adapterUtil.sh -removeFile /opt/IBM/svgadapters/customFiles/customFile.txt
```

Working with private keys (Docker only)

Loading a private key file

Use the following command to load private key:

```
./adapterUtil.sh -loadKeyToSDI /path/to/Key.pem Key_label
```

The first parameter to the option is path to the certificate file on the machine running the adapterUtil.sh script and the second parameter is the label name to be given to the key file. An example of the usage for this option is as below:

```
./adapterUtil.sh -loadKeyToSDI /tmp/stagingDirectory/ISVDI_Key.pem isvdi_key
```

Listing private key files

Use the following command to display list of private keys:

```
adapterUtil.sh -listKeysFromSDI
```

Removing a private key file

Use the following command to remove a private key:

```
adapterUtil.sh -removeKeyFromSDI <Label_Name>
```

The first parameter to the option is path to the label name to be given to the key file. An example of the usage for this option is as below:

```
./ adapterUtil.sh -removeKeyFromSDI isvdi_key
```

Working with signer certificates (Docker only)

Loading a signer certificate

Use the following command to load signer certificate:

```
adapterUtil.sh -loadTrustedCertToSDI /path/to/trustedCert.crt
```

The first parameter to the option is path to the certificate file on the machine running the adapterUtil.sh script. An 0example of the usage for this option is as below:

```
./adapterUtil.sh -loadTrustedCertToSDI /tmp/stagingDirectory/Entrust_Signer_Cert.pem
```

Listing signer certificates

Use the following command to display list of signer certificates:

```
adapterUtil.sh -listTrustedCertsFromSDI
```

Removing a signer certificate

Use the following command to remove a trusted certificate:

```
adapterUtil.sh -removeTrustedCertFromSDI <Trusted_Certificate_Filename>
```

The first parameter to the option is path to the certificate file on the machine running the adapterUtil.sh script. An example of the usage for this option is as below:

```
adapterUtil.sh -removeTrustedCertFromSDI EntrustSignerCert.pem
```

Working with logs (Docker only)

Updating log level for ibmdi.log files

Use the following command to update the root logger level:

```
adapterUtil.sh -setRootLevelLogging <logging_level>
```

The first parameter to the option is the logging level. For supported values, see Verify Directory Integrator Base under the *Containers* chapter of the *IBM Security Verify Directory Integrator* documentation for details. An example of the usage for this option is as below:

```
adapterUtil.sh -setRootLevelLogging debug
```

Reference table for all options

Sr. No.	Command line option	Supported in Docker	Supported in Kubernetes	Comments
1	-help	Yes	Yes	Displays help menu with all commands supported by the script
2	-version	Yes	Yes	Displays version of the script
3	-dispatcherPodName dispatcher	Yes	**No**	This option cannot be used alone. It should be used in combination with other op-

				tions. The default value for this option is **dispatcher**. If the container name is different than **dispatcher**, then this option is applicable.
4	-deploymentName DEPLOYMENT_NAME	**No**	Yes	This option cannot be used alone. It should be used in combination with other options. The default value for this option is **isvdi**. If the deployment name is different than **isvdi**, then this option is applicable.
5	-namespace NAMESPACE_NAME	**No**	Yes	This option cannot be used alone. It should be used in combination with other options. The default value for this option is **default**. If the namespace name is different than **default**, then this option is applicable. **Note:** This option is not present in the script which is shipped with ISIM Container Starter Kit. In ISIM Container, the namespace value is fetched internally based on the config files.
6	-loadAdapter /path/to/Adapter.zip accept	Yes	Yes	Loads the adapter in the container this option should be used. It expects 2 parameters. 1st parameter is the path to the Adapter zip file on the local filesystem of machine running this script and 2nd parameter indi-

				cates acceptance of the included license. The bundledefinition.json file should be present in the adapter zip file to use this option.
7	-infoAdapter Adapter_Name	Yes	Yes	Displays information of the adapter.
8	-listAdapters	Yes	Yes	Lists the adapters that were installed using this script.
9	-removeAdapter Adapter_Name	Yes	Yes	Removes the adapter that was loaded in the container using this script.
10	-copyToExternalKeystore /path/to/keystore_file	Yes	Yes	Copies external keystore files (JKS, P12, etc.) required by adapter as per installation and configuration guide.
11	-listExternalKeystoreFiles	Yes	Yes	Lists external keystore files which are copied into the container using this script.
12	-removeFromExternalKeystore keystore_filename	Yes	Yes	Removes the external keystore files which are copied into the container using this script.
13	-copyTo3rdpartyOthers /path/to/jar_file	Yes	Yes	Copies external dependency jar files (httpcore, httpclient, etc.) to **3rdparty/others** directory of SDI_HOME required by adapter as per installation and configuration guide.
14	-list3rdpartyOthersFiles	Yes	Yes	Lists dependency jar files which are copied into container to **3rdparty/others** directory of SDI_HOME using this script.
15	-removeFrom3rdpartyOthers jar_filename	Yes	Yes	Removes dependency jar files which are copied into container to

				3rdparty/others directory of SDI_HOME using this script.
16	-copyToPatches /path/to/jar_file	Yes	Yes	Copies external dependency jar files (httpcore, httpclient, etc.) to **patches** directory of SDI_HOME required by adapter as per installation and configuration guide.
17	-listPatchesFiles	Yes	Yes	Lists dependency jar files which are copied into container to **patches** directory of SDI_HOME using this script.
18	-removeFromPatches jar_filename	Yes	Yes	Removes dependency jar files which are copied into container to **patches** directory of SDI_HOME using this script.
19	-copyToConnectors /path/to/connector/jar_file	Yes	Yes	Copies custom connector jar files to **connectors** directory of SDI_HOME.
20	-listConnectorsFiles	Yes	Yes	Lists custom connector jar files which are copied into container to **connectors** directory of SDI_HOME using this script.
21	-removeFromConnectors connector_jar_filename	Yes	Yes	Removes custom connector jar files which are copied into container to **connectors** directory of SDI_HOME using this script.
22	-copyToFunctions /path/to/functions/jar_file	Yes	Yes	Copies custom function jar files to **functions** directory of SDI_HOME.
23	-listFunctionsFiles	Yes	Yes	Lists custom function jar files which are copied into containers to **functions** directory of SDI_HOME using this

				script.
24	-removeFromFunctions functions_jar_filename	Yes	Yes	Removes custom function jar files which are copied into container to **functions** directory of SDI_HOME using this script.
25	-copyToLibs /path/to/lib_file	Yes	Yes	Copies library files (.so, .dll, etc.) to **libs** directory of SDI_HOME.
26	-listLibsFiles	Yes	Yes	Lists library files which are copied into containers to **libs** directory of SDI_HOME using this script.
27	-removeFromLibs lib_filename	Yes	Yes	Removes library files which are copied into container to **libs** directory of SDI_HOME using this script.
28	-copyToProperties /path/to/properties_file	Yes	Yes	Copies property files to **properties** directory of SDI Solution directory.
29	-listPropertiesFiles	Yes	Yes	Lists property files which are copied into containers to **properties** directory of SDI Solution directory using this script.
30	-viewPropertiesFile properties_filename	Yes	Yes	Displays content of property file which is copied into containers to **properties** directory of SDI Solution directory using this script.
31	-removeFromProperties properties_filename	Yes	Yes	Removes property files which are copied into container to **properties** directory of SDI Solution directory using this script.
32	-copyToXsl /path/to/xsl_file	Yes	Yes	Copies xsl stylesheet files to **xsl** directory of SDI Solution directory.
33	-listXslFiles	Yes	Yes	Lists xsl stylesheet files which are copied into containers to **xsl** directory of SDI Solution

				directory using this script.
34	-removeFromXsl xsl_filename	Yes	Yes	Removes xsl stylesheet which are copied into container to **xsl** directory of SDI Solution directory using this script.
35	-copyToScripts /path/to/script_file	Yes	Yes	Copies shell script files to **scripts** directory of SDI Solution directory.
36	-listScriptsFiles	Yes	Yes	Lists shell script files which are copied into containers to **scripts** directory of SDI Solution directory using this script.
37	-removeFromScripts scripts_filename	Yes	Yes	Removes shell script files which are copied into container to **scripts** directory of SDI Solution directory using this script.
38	-copyToSwidtag /path/to/swid_tag_file	Yes	Yes	Copies swidtag files to **swidtag** directory of SDI_HOME.
39	-listSwidtagFiles	Yes	Yes	Lists swidtag files which are copied into containers to **swidtag** directory of SDI_HOME using this script.
40	-removeFromSwidtag swidtag_filename	Yes	Yes	Removes swidtag files which are copied into container to **swidtag** directory of SDI_HOME using this script.
41	-copyFile localFile containerPath	Yes	Yes	Copies any local file to any path inside the container. **Note:** This file if not placed in the persistent volume will be lost when the container restarts.
42	-listFiles containerPath	Yes	Yes	Lists files from given path inside containers.

43	-removeFile containerPath	Yes	Yes	Removes file from given path insdie container.
44	-loadKeyToSDI /path/to/Key.pem Key_label	Yes	**No**	Loads private key file into the persistent volume and also adds reference in config.yaml file used while creating container instance. **Note:** In Kubernetes certificates should be loaded using secrets and configmaps. Hence this option is not provided in Kubernetes.
45	-listKeysFromSDI	Yes	**No**	Lists private key files from config.yaml file used while creating container instance. **Note:** In Kubernetes certificates should be loaded using secrets and configmaps. Hence this option is not provided in Kubernetes.
46	-removeKeyFromSDI isvgim	Yes	**No**	Removes private key file reference from config.yaml file used while creating container instance and also removes the file from the persistent volume. **Note:** In Kubernetes certificates should be loaded using secrets and configmaps. Hence this option is not provided in Kubernetes.
47	-loadTrustedCertToSDI /path/to/trustedCert.crt	Yes	**No**	Loads trusted certificate file into the persistent volume and

				also adds reference in config.yaml file used while creating container instance. **Note:** In Kubernetes certificates should be loaded using secrets and configmaps. Hence this option is not provided in Kubernetes.
48	-listTrustedCertsFromSDI	Yes	**No**	Lists trusted certificates from config.yaml file used while creating container instance. **Note:** In Kubernetes certificates should be loaded using secrets and configmaps. Hence this option is not provided in Kubernetes.
49	-removeTrustedCertFromSDI DigiCertGlobalRootCA.crt	Yes	**No**	Removes trusted certificate file reference from config.yaml file used while creating container instance and also removes the file from the persistent volume. **Note:** In Kubernetes certificates should be loaded using secrets and configmaps. Hence this option is not provided in Kubernetes.
50	-setRootLevelLogging debug	Yes	**No**	Updates the config.yaml file used while creating container instance to enable debug logging for ibmdi logs. **Note:** In Kubernetes dispatcher config yaml

				file should be loaded using configmaps. Hence this option is not provided in Kubernetes.

Note: Due to any reason, if adapterUtil.sh script could not be executed in the Docker or Kubernetes environments, then adapter files (connector, external dependency jars, property files, etc.) needs to be manually copied to persistent volumes defined while creating the container and the certificates need to be added to the config persistent volume and config yaml file needs to be updated manually referring to YAML Schema documentation for ISVDI container.

Paths relative to SDI_HOME or Solution directory (SOL_DIR)	**Paths mapped with persistent storage**
SOL_DIR/keystores	/opt/IBM/svgadapters/timsol/keystores
SOL_DIR/properties	/opt/IBM/svgadapters/timsol/properties
SOL_DIR/xsl	/opt/IBM/svgadapters/timsol/xsl
SOL_DIR/scripts	/opt/IBM/svgadapters/timsol/scripts
SDI_HOME/jars/connectors	/opt/IBM/svgadapters/jars/connectors
SDI_HOME/jars/3rdparty/others	/opt/IBM/svgadapters/jars/3rdparty/others
SDI_HOME/jars/patches	/opt/IBM/svgadapters/jars/patches
SDI_HOME/jars/functions	/opt/IBM/svgadapters/jars/functions
SDI_HOME/libs	/opt/IBM/svgadapters/libs
SDI_HOME/swidtag	/opt/IBM/svgadapters/swidtag

To import the Trusted / Signer certificates into the SDI keystore, see YAML Schema documentation for ISVDI container to add **trusted-certificates** element to the dispatcher config.yaml file. Extract the root certificate / Signer Certificate from the secured URL of the Target and place the certificate in the **certificates** directory of config volume which contains the config.yaml file. The default location for this config volume is **/opt/IBM/dispatcher/config**.

Provide this path of the certificate in config.yaml file as shown in the example below:

```
keyfile:
  trusted-certificates:
  - '@/opt/IBM/dispatcher/config/certificates/ca_cert.pem'
```

Note: After adding an adapter or related files or updating any configuration requires these changes to be activated. See Activating changes to the configuration of this guide for details.

Activating changes to the configuration

Any changes done to the configuration files of the container or after deploying any adapter or its related files require the changes to be activated. See Restarting dispatcher container of this guide for details.

Troubleshooting

Several strategies exist to help diagnose and solve problems in a container deployment. Most of them requires debug enabled logs. Below properties can be set in config yaml file used while creating container instance.

Configuration option	Description
general.logging.json-logging	The *general.logging.json-logging* boolean configuration entry, when set to **false**, disables the JSON based logging, and changes the message format to a simple text string. It makes the logs readable and generates logs similar to the ones generated in the on-prem version of SDI.
general.logging.root-level	The *general.logging.root-level* configuration entry controls the amount of logging that is generated by the server. Set this configuration value to **debug** and restart the container. For other supported values, see Verify Directory Integrator Base under the *Containers* chapter of the *IBM Security Verify Directory Integrator* documentation for details.

Note: After adding an adapter or related files or updating any configuration requires these changes to be activated. See Activating changes to the configuration of this guide for details.

Chapter 4. Installing

You must install the Dispatcher10 on the same Security Verify Directory Integrator server where you want to install the adapter.

Multiple Security Verify Directory Integrator-based adapters installed on the same Security Verify Directory Integrator server can use the same Dispatcher. However, you must install a Dispatcher10 on each Security Verify Directory Integrator server on which you want to install an adapter.

Note:

- During upgrade, the Dispatcher10 installer does not request an instance name and port number.
- You must have **execute** permissions on the **ps** command.

Obtain the Dispatcher10 installer from the IBM Passport Advantage website, see "Software download" 5.

Installing the Dispatcher10 in GUI mode

You can install the Dispatcher10 with GUI mode.

1. Open a command-line interface.
2. Run the following command:
   SDI_HOME/jvm/jre/bin/java -jar DispatcherInstall.jar -i gui
3. On the Welcome page, click Next.
4. In the Directory Name field, specify the location of the Security Verify 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 Security Directory Integrator adapters solution directory.
6. Enter the Dispatcher10 Instance Name. Click **Next**
7. Enter the Port number. Click **Next**.
8. Provide credentials to secure access to the Java Virtual Machine. For more information refer Enabling JVM Security(Hyperlink to this section in Dispatcher10 guide). Click **Next**.
9. Select SSL level. When the check-box is checked the SSL is enabled. If you wish to disable SSL then deselect the check-box Enable SSL. If you disable SSL the communication therefore will be unencrypted i.e. in plain text. For more information refer "Configuring SSL communication".
10. 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.
    - Click Finish when the software displays the Install Completed window.

Installing the Dispatcher10 in console mode

You can install the Dispatcher10 with console mode.

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

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

Installing the Dispatcher10 in silent mode

You can install the Dispatcher10 in silent mode.

You can install the Dispatcher10 in silent mode by using the default settings. You also can override the default settings with the commands described in Table 5 8.

You can override the default settings with the -Dparameter. The -Dmust 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 Dispatcher10 by using silent mode, then the uninstaller runs in silent mode regardless of whether you use the -i silentoption.

Table 5. Parameters for installing the Dispatcher10 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/SDI/V10.0/mysol"
-DUSER_INPUT_RMI_PORTNUMBER	This parameter overrides the default RMI port number on which the Dispatcher10 listens. For example, -DUSER_INPUT_RMI_PORTNUMBER=1234

-DUSER_DISPATCHER_SERVICE_NAME	This parameter specifies the name of the Dispatcher10 service on Windows. For example, -DUSER_DISPATCHER_SERVICE_NAME="Verify Governance Adapters"
-DUSER_SYSQUEUE_USERNAME_INPUT_RESULT	This parameter provides the username for JVM security.
-DUSER_SYSQUEUE_PASSWORD_INPUT_RESULT	This parameter provides the Password for JVM security.
-DUSER_SYSQUEUE_REPASSWORD_INPUT_RESULT	This parameter provides the retype password filed for JVM security.
-DUSER_INPUT_RESULTS_FOR_SSL=Enable	Provides SSL level; default value is Enable

1. Open a command-line interface.
2. Run one of the following commands.

- To install the Dispatcher10 in silent mode with the default settings, run the command:

```
SDI_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:

SDI_HOME/jvm/jre/bin/java

-jar DispatcherInstall.jar -i silent

-DUSER_INSTALL_DIR="/opt/IBM/SDI/V10.0"

-DUSER_SELECTED_SOLDIR="/opt/IBM/SDI/V10.0/timsol"

-DUSER_INPUT_RMI_PORTNUMBER=1099 -DUSER_INPUT_WS_PORTNUMBER=8081

-DUSER_SYSQUEUE_USERNAME_INPUT_RESULT=”disp_user”

-DUSER_SYSQUEUE_PASSWORD_INPUT_RESULT= “admin@123”

-DUSER_SYSQUEUE_REPASSWORD_INPUT_RESULT= “admin@123”

-DUSER_INPUT_RESULTS_FOR_SSL=Enable

Verifying the adapter installation

You must verify that the Dispatcher10 installation placed components in the correct directories on the Security Verify Directory Integrator server.

Table 6. Dispatcher10 components

Directory	Dispatcher10 component
SDI_HOME\jars\3rdparty\IBM	• rmi-dispatcher.jar • itim-dispatcher-ws-transport.jar • itim-dispatcher-ws-config.jar
SDI_HOME\jars\3rdparty\others	• antlr-x.x.x.jar • jakarta-regexp-x.x.jar
adapter_solution_directory	• ITIM_RMI.xml • ITIMAd • itimadpid • run_db_script.sh • createDb.sql
SDI_HOME	• itim_listener.properties
SDI_HOME\SOL_DIR\idm_respository\modules	• itim-dispatcher-authn.mar
SDI_HOME\SOL_DIR\idm_respository\services	• itim-dispatcher-ws.aar
SDI_HOME\SOL_DIR\	• axis2.xml • svcConfigDBThis component is the database instance.

•

Review the installer log files Dispatcher_Installer.logand Dispatcher_Installer_opt.login the installer directory for any errors.

If this installation is to upgrade a Dispatcher, send a request from the ISVG server. Verify that the version number in the ibmdi.log matches the version of the Dispatcher. Navigate to the ADAPTER_SOLDIR/ logsdirectory and search for *RMIDispatcherImpl: Starting*. Verify that the version number of the Dispatcher10 is correct.

Start, stop, and restart the Dispatcher10 service

When you edit an adapter or Security Verify Directory Integrator properties file, you must stop and restart the Dispatcher10 service for the changes to take effect.

Select the appropriate method based on your operating system.

Starting, stopping, and restarting the Dispatcher10 service on AIX and Linux operating systems

When you edit an adapter or Security Verify Directory Integrator propertiesf i l e , you must stop and restart the Dispatcher10 service for the changes to take effect.

The ITIMAdscript 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
Linux	timsol

The ITIMAdscript file creates the itimadpidfile in the adapter solution directory. The file contains the process ID of the Dispatcher10 service. **Do not modify or delete this file.** When you start the Dispatcher10 service, the ITIMAdscript creates the itimadpid file. When you stop the Dispatcher10 service, the ITIMAdscript deletes the itimadpid file. This file is not created on all platforms.

1. From the command line, navigate to the directory that contains the ITIMAdscript file.
2. Run the following commands to start, stop, and restart the Dispatcher10 service:

Table 8. UNIX based and Linux commands

AIX	**Linux**	
ITIMAd startsrc	ITIMAd start	
ITIMAd stopsrc	ITIMAd stop	
ITIMAd restartsrc	ITIMAd restart	

When you edit an adapter or Security Verify Directory Integrator properties file, you must stop and restart the Dispatcher10 service for the changes to take effect.

The ITIMAdscript file starts and stops the service. The adapter installation copies the file to the timsol directory.

Chapter 5. Upgrading

The Dispatcher10 is upgraded by installing the new version of the

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

Dispatcher.

- If the Dispatcher10 version mentioned in the release notes is later than the existing version on your workstation, install the Dispatcher.
- If the Dispatcher10 version mentioned in the release notes is the same or earlier than the existing version, do not install the Dispatcher.

If the Dispatcher10 service is running when you upgrade the Dispatcher, then the Dispatcher10 installer stops the service and restarts it after completing the upgrade process.

If the Dispatcher10 is not running when you upgrade the Dispatcher, then the Dispatcher10 installer does not start the service after completing the upgrade process.

If you want to force start the Dispatcher10 service, use the following command-line option when you run the Dispatcher10 installer:

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

Valid values for FORCE_DISPATCHER_SERVICE_START_ONINSTALLare YESor NO.

Chapter 6. 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 Dispatcher10 service after you change the properties for the dispatcher. Table 10 15 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 Dispatcher10 shuts down when a shutdown request is sent to the dispatcher. When the Dispatcher10 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 SDIDispatcher.
com.ibm.di.dispatcher.objectPort	solution.properties	Specifies the port on which the Dispatcher10 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 Dispatcher10 listens for provisioning requests from Verify Governance server.

Table 10. Configuration properties for the Dispatcher10 (continued)

Property	Properties File	Description
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, Verify Governance server will fail the time-out requests when the timeout occurs. Default value is 0, which executes the default behavior, Verify Governance server retries the time-out requests.
SearchALUnusedTimeout	itim_listener.properties	Specifies the number of seconds the Dispatcher10 waits before it deletes the search assembly lines that are unused. The default value is 600seconds.
SearchReaperThreadTimeOut	itim_listener.properties	Specifies the number of seconds after which the Dispatcher10 releases data from memory. The reconciliation process uses this property. The default value is 300seconds.
SearchResultSetSize	itim_listener.properties	Specifies the number of records, per response, the Dispatcher10 returns during a reconciliation between Verify Governance server and the adapter. The default value is 100.
ALCacheSize	itim_listener.properties	Specifies the number of assembly lines (add, modify, delete) that the Dispatcher10 caches. The default assembly line cache size is 100. Setting the assembly line cache size to 0disables the caching in the dispatcher.

Table 10. Configuration properties for the Dispatcher10 (continued)

Property	Properties File	Description
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 Dispatcher10 can run simultaneously. The default value is 100. **Note:** Setting the GlobalRunALCount to 0does not limit the number of assembly lines that the Dispatcher10 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.
SleepAfterInterrupt		Specifies the time in seconds that the Dispatcher10 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 20seconds.

Changing the port number for the IBM Security Verify Directory Integrator Dispatcher

If you run the Dispatcher10 as a service, the default port number is 1099. The installer automatically sets this parameter in the global.properties and solution.propertiesfiles.

In IBM Security Verify Directory Integrator , the default setting for the api.remote.on property is true. This setting causes the IBM Security Verify Directory Integrator to listen on port 1099, as de ined by the api.remote.naming.portproperty.

If the api.remote.on propertyis set to false, IBM Security Verify Directory Integrator listens on the port de ined by the com.ibm.di.dispatcher.registryPortproperty. The default value for this setting is 16231.

To modify the port number for the Dispatcher, you must change the property value in the *SDI_HOME*/*timsol*/solution.propertiesdirectory.

1. Stop the service that runs the adapter.
   See "Start, stop, and restart the Dispatcher10 service" 10.
2. Perform one of the following actions to change the port number:
   - Edit the api.remote.naming.portproperty in the solution.propertiesfile. 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.onproperty to false.
     b. Edit the com.ibm.di.dispatcher.registryPortproperty 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 Dispatcher10 to do case-sensitive filtering, add the CaseInSensitiveFilter property to the search operation in the service.deffile.

The service.deffile is available in the adapter profile.

The CaseInSensitiveFilter property specifies whether the filtering by the Dispatcher10 must be case-sensitive or not case-sensitive. If the property is set to false, you must specify the Verify Governance server filter in the same case as the data on the endpoint, otherwise the correct data is not filtered.

The Dispatcher10 filtering is case-sensitive for adapters that do not support this property. To add the **CaseInSensitiveFilter** property to the adapter, take the following steps:

1. Extract the adapter profile jar file.
2. Open the service.deffile from the extracted adapter profile jar file.
3. Add the following Dispatcher10 parameters in the search operation and save the service.deffile:

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

4. Create the adapter profile jar file with updated service.deffile.
5. Import the updated adapter profile on the Verify Governance server.

Extracting the current Request ID from the assembly line

When a cached assembly line is used for non-reconciliation operations, the Dispatcher10 logs display the cached Request ID. You can display the current Request ID by extracting it from the assembly line.

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

Add the following code in your assembly line. In this code, the transactionIdholds 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 Dispatcher10 on one system

The Dispatcher, can support multiple instances of the Dispatcher10 on the same system. However, there can be only one Dispatcher10 per IBM Security Verify Directory Integrator instance.

To run multiple dispatchers on the same system, you must specify a unique subsystem name on AIX® systems. All platforms require a unique port number on which the Dispatcher10 service can listen.

Configuring the Dispatcher10 JVM properties for UNIX operating systems

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

Standard JVM properties are:

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

The Dispatcher10 process is a running instance of the Security Verify Directory Integrator server. As an example, this procedure sets the Dispatcherr encoding to UTF-8.

1. Navigate to the *SDI_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 Dispatcher10 service" 10.

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.

When multiple adapters run on the server where the IBM Security Verify Directory Integrator is installed, logging information for the adapters is stored in the same log file. The Dispatcher10 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 etc/log4j2.xml file determine the type of information that is stored in your log file. Update etc/log4j2.xmlto configure logging for the adapter.

The location of the etc/log4j2.xml file depends on the operating system.

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

a) Specify the name of the log file.

```
  append="true"
  fileName="logs/ibmdi.log"
  filePattern="logs/ibmdi-%i.log">
  PatternLayout pattern="%d{DEFAULT} %-5p [%c] - %m%n"/>
```

b) Specify the number of log files you want to generate.

```
<DefaultRolloverStrategy max="3" fileIndex="min"/>
```

c) Specify the policies:

```
<Policies>
<OnStartupTriggeringPolicy minSize="1" />
<SizeBasedTriggeringPolicy size="10MB" />
```

d) Specify theRolloverStrategy:

```
<DefaultRolloverStrategy max="3" fileIndex="min" />
```

There are messages that are logged to the console (standard out):

```
<Console name="CONSOLE" target="SYSTEM_OUT">
<PatternLayout pattern="[%t] %-5p - %m%n"/>
```

e) Specify the name of the log file:

```
<RollingFile name="SCIM"
 append="true"
 fileName="SCIM/logs/audit.log"
 filePattern="SCIM/logs/audit.%d{yyyy-MM-dd}.log">
 <PatternLayout pattern="%m%n" />
```

f) Specify the loggers:

```
<Loggers>
<Root level="info">
<AppenderRef ref="Default"/>

<Logger name="Console" level="warn" >
<AppenderRef ref="CONSOLE"/>

<Logger name="com.ibm.di.config" level="warn" >
```

```
<AppenderRef ref="CONSOLE"/>

<Logger name="com.ibm.di.TDIProperties" level="warn" >
<AppenderRef ref="CONSOLE"/>

<Logger name="com.ibm.di.loader" level="warn" >

<Logger name="org.apache.wink" level="error" >

<Logger name="com.ibm.di.ScimService" level="info" additivity="false">
<AppenderRef ref="SCIM"/>
```

3. Save the file.
4. Stop and restart the Dispatcher10 service.
   See "Start, stop, and restart the Dispatcher10 service" 10
   For more information about logging, see *IBM Security Verify Directory Integrator Installation and Administrator Guide*.

Enabling FIPS mode

The keystore file created should be copied to TIMSOL folder and/or the encryption/decryption should be with the newly generated FIPS compliant keystore file.

Note: If FIPS mode is enabled, changes done to any authentication attributes in solution.properties file may not get affected directly and this can result in an error related decryption in ibmdi.log file. To resolve this error re-encrypt the solution.properties file with the key created for FIPS mode.

Sample command for encryption:

cryptoutils -input ../timsol/solution.properties -output ../timsol/solution.properties
-mode encrypt_props -keystore ../server.jck -storepass mypass -alias server

-transformation AES/CBC/PKCS5Padding -storetype jceks -keypass mykeypass

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

Service scaling and tuning

On the adapter service form, you can use attributes to scale and tune the Dispatcher10 instance that runs within the Security Verify Directory Integrator.

Disable AL Caching

The Dispatcher10 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 "Dispatcher10 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 Dispatcher10 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 Dispatcher10 has a global cache setting. Use the ALCacheSize property in the *SDI_HOME*/ itim_listener.properties file to specify the maximum number of assembly lines that the Dispatcher10 caches for all services. See Table 10 15 for more information.

Max Connection Count

The Dispatcher10 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 Dispatcher10 can run simultaneously for the service, enter a positive integer value for "Max Connection Count" on the service form under the "Dispatcher10 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 Dispatcher10 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 *SDI_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 “Con iguration properties of the Dispatcher” 15 for more information.

AL FileSystem Path

Optionally, you can store the assembly lines on the file system where the Dispatcher10 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 UNIX or Linux operating systems

/opt/IBM/SDI/*SDI_VERSION*/profiles

Transaction timeout

You can configure a transaction timeout for the Dispatcher10 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 Dispatcher10 ibmdi.log file contains an error message such as:

```
Time Out ….Dispatcher10 Interrupts Initialization Thread due to AL TimeOut….
```

For example:

```
executeALRequest ():2226 Time Out: 60 request id: 7226427570134735752
Dispatcher10 Interrupts Initialization Thread due to AL TimeOut.
Service Name :OracleTestService Assembly Line Name is :OracleManageUserAL
```

Using the itim_listener.properties file in the *SDI_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 Dispatcher10 level setting. Use these properties:

- AddRequestTimeOut
- ModifyRequestTimeOut
- DeleteRequestTimeOut
- SearchRequestTimeOut

Service instance
Affects one service instance only. This setting takes precedence over the Dispatcher10 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.deffiles of the adapter profile JAR file.

1. Extract the content of the adapter profile JAR file.
2. In the service.deffile, 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.propertiesfiles of the adapter profile JAR file.

1. Extract the content of the adapter profile JAR file.
2. In the schema.dsmlfile, 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.dsmlfile 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.dsmlfile to include the new attributes as optional attributes.
5. Modify the CustomLabels.propertiesfile 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.deffile to map the service attributes to the Dispatcher10 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 Dispatcher10 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 Dispatcher10 to send a failure for the requests that have timed out, so that Verify Governance server does not retry the requests. You can configure this feature at the Dispatcher10 level, service type level or service instance level.

Dispatcher10 level

Dispatcher10 level affects all adapters running under the Dispatcher. Using the itim_listener.propertiesfile in the SDI_HOME directory, set the following property:

FailTimeoutRequest
By setting this value as 1, the Verify Governance server will fail the time-out requests when the timeout occurs.
Default value is 0, which executes the default behavior, for example, Verify Governance 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 Dispatcher10 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.deffile, 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 Verify Governance server fails the request when timeout occurs.

   False value implies that the request goes to a pending state and Verify Governance 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 Dispatcher10 level and service type level settings.

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

1. Extract the content of the adapter profile JAR file.
2. In the schema.dsmlfile, 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.dsmlfile to include the new attribute as optional attribute.

4. Modify the CustomLabels.propertiesfile to include a meaningful label for the new attribute:

   myFailTimeoutRequest = Fail the Timeout Request
5. Modify the service.deffile to map the service attributes to the Dispatcher10 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 Dispatcher10 level by using a locking mechanism.

The Dispatcher10 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.dsmland service.deffiles 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 Dispatcher10 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 Dispatcher10 uses the value of the **LockName**Dispatcher10 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 JVM Security

Since WAS and Dispatcher10 server communicate using RMI. It's mandatory to secure the JVM.
In order to do so, default Dispatcher10 installation is prompted with providing strong credentials for JVM security. These credentials will be required by an outside RMI process to access the RMI stub. You can always modify the existing credentials by changing following properties in the solutions.properties file:
{protect}-systemqueue.auth.username
{protect}-systemqueue.auth.password
systemqueue.on=false

Configuring SSL communication

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

You can configure the Security Verify Directory Integrator to use SSL and also configure WebSphere with the default keystore and default truststore. For more information about WebSphere SSL con iguration, see the WebSphere online help from the WebSphere Application Server Administrative Console.

As an adapter manages sensitive data of the users it is essential that communication should be encrypted.
SSL facilitates the encrypted communications between an adapter and end resource. SSL requires certificates to be installed.

During installation you may see the panel Enable SSL. The check-box is present on a panel. It is by default checked. When the check-box is checked the SSL is enabled. If you wish to disable SSL then deselect the check-box Enable SSL. If you disable SSL the communication therefore will be unencrypted i.e. in plain text. Whether SSL is enabled or disabled can be verified after installation.
The property "*com.ibm.di.dispatcher.ssl*" in **solution.properties** is set to true if SSL is enabled otherwise it is set to false.

SSL terminology for adapters

There are several SSL terms that apply to adapters.

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

SSL client
The workstation on which the Verify Governance server and WebSphere Application Server are installed. The client submits connection requests to the Security Verify 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 iKeymanutility 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 Security Verify Directory Integrator server)

Truststore on the SSL client-side (the Verify Governance 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 Verify Governance server is installed, and the SSL server is theSecurity Directory Integrator.

Configuring SSL for one-way SSL communication

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

server. This procedure requires you to use the following tasks:

- "Creating a keystore for the Security Verify Directory Integrator server"
- "Creating a truststore for the Security Verify Directory Integrator server"
- "Creating a self-signed certificate for the Security Verify Directory Integrator server"
- "Extracting a CA certificate for the Security Verify Directory Integrator"
- "Importing the Security Verify Directory Integrator CA certificate in the WebSphere Application Server truststore" 45
- "Configuring the Security Verify Directory Integrator to use the keystores"
- "Configuring Security Verify Directory Integrator to use truststores"
- "Enabling the adapter service to use SSL"
- "Start, stop, and restart the Dispatcher10 service"

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.

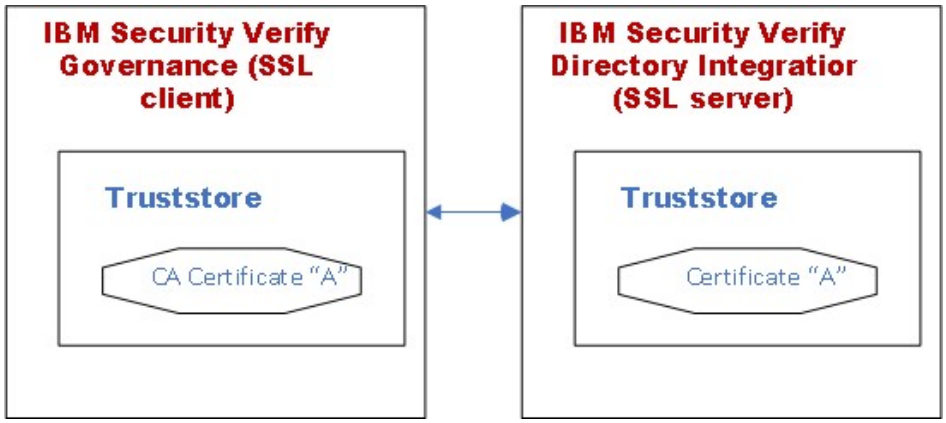

Figure 2. One-way SSL communication (server communication)

Note: The Verify Governance server uses the existing truststore of theWebSphere Application Server.

1. Create a keystore for the Security Verify Directory Integrator server.
2. Create a truststore for the Security Verify Directory Integrator server. One-way SSL communication on the Security Verify 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 Security Verify Directory Integrator server.
4. Create a CA certificate for the Security Verify Directory Integrator server.
5. Import the Security Verify 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 Security Verify Directory Integrator to use keystores.
7. configure the Security Verify 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.

Specifying the TLS level in Security Verify Directory Integrator

To specify the TLS level to be used, add the following lines to the solution.properties file

```
######################
# # Protocols to enforce SSL protocols in a SDI Server
# # Optional values for com.ibm.di.SSL* property . # # This can be a multi-valued comma separated property
######################
com.ibm.di.SSLProtocols=TLSv1.2,TLSv1.3
com.ibm.di.SSLServerProtocols=TLSv1.2
######################
```

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.

This procedure requires you to use the following tasks:

- "Creating a keystore for the Security Verify Directory Integrator server" 40
- "Creating a truststore for the Security Verify Directory Integrator server" 41
- "Creating a self-signed certificate for the Security Verify Directory Integrator server" 41
- "Extracting a CA certificate for the Security Verify Directory Integrator" 42
- "Importing the Security Verify Directory Integrator CA certificate in the WebSphere Application Server truststore" 45
- "Configuring the Security Verify Directory Integrator to use the keystores" 43
- "Configuring Security Verify Directory Integrator to use truststores" 43
- "Enabling the adapter service to use SSL" 44
- "Creating a self-signed certificate for the Security Verify Directory Integrator server" 41
- "Extracting a WebSphere Application Server CA certificate" 45
- "Importing the WebSphere CA certificate in the Security Verify Directory Integrator truststore" 42
- "Start, stop, and restart the Dispatcher10 service"

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.

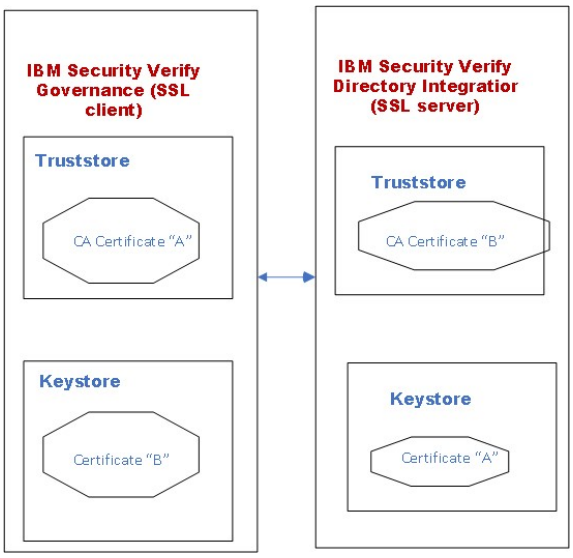

Figure 3. Two-way SSL communication (client communication)

Note: the Verify Governance server uses the existing truststore and keystore of the WebSphere Application Server. To configure two-way SSL, do the following tasks:

1. Create a keystore for the Security Verify Directory Integrator server.
2. Create a truststore for the Security Verify 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 Security Verify Directory Integrator server.
4. Create a CA certificate for the Security Verify Directory Integrator server.
5. Import the Security Verify 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 Security Verify Directory Integrator to use keystores.
7. configure the Security Verify Directory Integrator to use truststores.
8. Enable the adapter service to use SSL.
9. Create a certificate for the Verify Governance server.
10. Create a CA certificate for Verify Governance server.
11. Import the WebSphere Application Server CA certificate in Security Verify 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 Security Verify Directory Integrator as the SSL server. Complete all tasks on the Security Verify Directory Integrator server workstation.

Note: File names such as sdikeys.jks and locations such as *SDI_HOME*\keys are examples. Actual file names and locations might differ.

Creating a keystore for the Security Verify Directory Integrator server

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

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.

1. Navigate to the *SDI_HOME*/jvm/jre/bin directory.
2. Start 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 sdikeys.jks.
6. Type the location.
   For example, type *SDI_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 Security Verify Directory Integrator server

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

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.

1. Navigate to the *SDI_HOME*/jvm/jre/bin directory.
2. Start 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 sdikeys.jks.
6. Type the location.
   For example, type *SDI_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 Security Verify 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.

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 to generate your own certificates.

Procedure

Navigate to the SDI_HOME/jvm/jre/bin directory.

Start the ikeyman.exe file (for Windows operating system) or ikeyman (for UNIX and Linux operating systems).

From the Key Database File menu, select Open.

Navigate to the keystore file that was created previously: SDI_HOME/keys/sdikeys.jks.

Enter the keystore password. The default password is secret.

Select Create > New Self Signed certificate.

Set the Key Label to sdiserver.

Use your system name (DNS name) as the Common Name (workstation name).

Enter the name of your organization.

For example, enter IBM.

Click OK.

Extracting a CA certificate for the Security Verify Directory Integrator

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

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.

1. Navigate to the *SDI_HOME*\jvm\jre\bindirectory.
2. Launch ikeyman(for UNIX and Linux operating system).
3. From the **Key Database File** menu, select **Open**.
4. Navigate to the keystoref i l e that was created previously: *SDI_HOME*\keys\sdikeys.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 certificatef i l e name: sdiserver.der.
9. Enter the location as *SDI_HOME*\keys.
10. Click **OK**.
11. Copy the s diserver.dercertificate f i l e to the workstation on which Verify Governance server is installed.

Importing the WebSphere CA certificate in the Security Verify Directory Integrator truststore

Verify Governance server uses the WebSphere CA certificate, to authenticate to the Security Directory Integrator.

Before you begin

Copy the timclient.der SSL Client CA certificate file created in Extracting a WebSphere Application Server CA certificate to the SDI_HOME\keys directory on the workstation on which the Security Directory Integrator is installed.

About this task

After you extract the WebSphere CA certificate, you must import it into the Security Verify 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

Navigate to the SDI_HOME\jvm\jre\bin directory.

Start ikeyman (UNIX and Linux® operating system).

3. From the **Key Database File** menu, select **Open**.
4. Select **JKS**.
5. Type the keystore file name: sditrust.jks.
6. Type the location: *SDI_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 svgclient.der file that is stored in *SDI_HOME*\keys directory. Use **svgclient** as the label.
10. Click **OK** to continue.

Configuring the Security Verify Directory Integrator to use the keystores

You can configure the Security Verify Directory Integrator to use the keystores.

You must know the location, password, and type of keystore that you created in "Creating a keystore for the Security Verify Directory Integrator server"

1. Navigate to the *SDI_HOME*\timsoldirectory.
2. Open the Security Verify Directory Integrator solution.properties file in an editor.
3. Edit the following lines under client authentication:

   javax.net.ssl.keyStore=ITDI_HOME\keys\sdikeys.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 Security Verify Directory Integrator to use the keystores and truststores. You can stop and restart the adapter after you enable the adapter service to use

Configuring Security Verify Directory Integrator to use truststores

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

1. Navigate to the *SDI_HOME*\timsoldirectory.
2. Open the Security Verify Directory Integrator solution.propertiesfile 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.propertiesfile in a single operation. Do not stop and restart the adapter service after you configure the Security Verify Directory Integrator to use the keystores and truststores. You can stop and restart the adapter after you enable the adapter service to use

When you edit an adapter or Security Verify Directory Integrator propertiesfile, you must stop and restart the Dispatcher10 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.

1. Navigate to the *SDI_HOME*/timsoldirectory.
2. Open the Security Verify Directory Integrator solution.propertiesfile 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 Verify Governance server and

Security Verify Directory Integrator.

Complete all tasks on the server workstation on which Verify Governance server and WebSphere Application Server are installed.

Creating a self-signed certificate for the Security Verify 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.

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 Security Verify Directory Integrator"

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, or create your own certificates.

1. Start ikeyman (for UNIX and Linux operating systems).
2. From the **Key Database File** menu, select **Open**.
3. Navigate to the keystore file that was created previously: *SDI_HOME*/keys/sdikeys.jks.
4. Enter the keystore password. The default password is secret.
5. Select Create > New Self Signed certificate.
6. Set the Key Label to **sdiserver**.
    1. Use your system name (DNS name) as the Common Name (workstation name).
    2. Enter the name of your organization. For example, enter IBM.

Click **OK**.

Extracting a WebSphere Application Server CA certificate

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

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\svgclient.der.
7. Select **Binary DER data** as the data type.
8. Click **OK**.

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

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

1. Copy the SSL server CA certificate file, sdiserver.der, to the c:\keys directory on the workstation on which Verify Governance server is installed.
2. Connect to the WebSphere Application Server Administrative Console.
3. Browse to **Security** > **SSL certificate and key management** > **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 Verify Governance 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 sdiserver.

b) Specify the file name of the exported IBM Security Verify Directory Integrator server certificate: C:\keys\sdiserver.der.

c) Select **Binary DER data** as the data type.

7. Click **OK** to continue and save.

Chapter 7 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 Dispatcher10 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" 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
```

Security Verify Directory Integrator Application Monitoring console

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

The port is specified in the api.remote.naming.portproperty in the *SDI_HOME*/*timsol*/solutions.propertiesfile.

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

- Change the port number that is specified in the Security Verify Directory Integrator location ield on the service form to the number specified in api.remote.naming.portproperty of the solution.propertiesfile.
- Change the port number that is specified in the api.remote.naming.portproperty of the solution.propertiesfile to the number specified in the Security Verify Directory Integrator location ield on the service form.

Troubleshooting the Dispatcher10 while using SSL Configuration

After some amount of usage, maybe an hour or so, SSL connections from Verify Governance server to the Dispatcher10 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.propertiesfile:

1. Go to SDI_HOME\timsol
   \solution.properties.
2. Set systemqueue.on=falseand save the file
3. Restart the Dispatcher10 service.

Verifying that the correct level of Security Verify Directory Integrator is installed

You must check the version level date in ibmdi.log to determine the level of the installed Security Verify 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 Security Verify Directory Integrator version and fix pack level.

To verify the level of Security Verify Directory Integrator, check the ibmdi.logfile. The log shows version levels up to three levels x.x.x. The date is the only way to verify the Security Verify Directory Integrator ix pack level.

Installer problems on UNIX and Linux operating systems

Interruptions during the Dispatcher10 installation or running an unsupported JVM can cause installation problems.

The Dispatcher10 installer creates temporary files during installation. On the UNIX and Linux platforms these files are in the /tmpdirectory. 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 /tmpdirectory:

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

2. Run the uninstaller.

3. Edit the *SDI_HOME*/etc/global.propertiesfile 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

SDI_HOME
```

4. Remove the following JARfiles from the *SDI_HOME*/jars/3rdparty/IBMdirectory:
   com.ibm.agents.installAnywhere.ITDIAgents.jar
   itim-dispatcher-ws-config.jar
   rmi-dispatcher.jar
   itim-dispatcher-ws-transport.jar

5. Remove the following JARf i l e s from the *SDI_HOME*/jars/3rdparty/othersdirectory:

```
Jakarta-regexp-x.x.jar
antlr-x.x.x.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 ITIMAdscript to start, stop, and restart the Dispatcher10 service.

The ITIMAdscript logs its output to a separate ITIMAd_stdout.logfile in the /opt/IBM/SDI/ *SDI_Version*/timsoldirectory.

If a problem occurs, examine the output in the log file, which describes the Dispatcher10 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 15.

Chapter 8. Uninstalling

The Dispatcher10 is required for all adapters that are based on Security Verify Directory Integrator. If you uninstall the Dispatcher, none of the other installed adapters function.

The mode used to uninstall the Dispatcher10 depends on which mode was used to install the Dispatcher.

If you install the Dispatcher10 by using console mode, then you can uninstall the Dispatcher10 only with console mode or silent mode.

If you install the Dispatcher10 by using silent mode, then the uninstaller runs in silent mode regardless of whether you use the -i silentoption.

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” 55.

1. Navigate to the Dispatcher10 uninstaller folder.
2. Run one of the following commands:

   - To run the uninstaller in GUI mode, use the following command:

     ```
     SDI_HOME/jvm/jre/bin/java  -jar  uninstaller.jar -i gui
     ```

   - To run the uninstaller in console mode, use the following command:

     ```
     SDI_HOME/jvm/jre/bin/java -jar uninstaller.jar -i console
     ```

   - To run the uninstaller in silent mode, use the following command:

     ```
     SDI_HOME/jvm/jre/bin/java -jar uninstaller.jar -i silent
     ```

The Dispatcher10 is uninstalled and the uninstaller creates a backup of the itim_listener.properties.

Chapter 9. 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.propertiesfile

The itim_listener.propertiesfile is a Dispatcher10 configurationfile in the SDI_HOME directory. When you upgrade the Dispatcher10 component, the Dispatcher10 replaces the itim_listener.propertiesfile with a new version while the installer creates a backup of the original file.

Similarly, when you uninstall the Dispatcher10 component, the uninstaller creates a backup of the itim_listener.propertiesfile.

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.

IBM®