

IBM Security Directory Integrator
Version 7.2.0.1

Installation and Administrator Guide



IBM Security Directory Integrator
Version 7.2.0.1

Installation and Administrator Guide



Note

Before using this information and the product it supports, read the general information under "Notices" on page 377.

Edition notice

Note: This edition applies to version 7.2.0.1 of *IBM Security Directory Integrator* licensed program (5724-K74) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2003, 2014.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this publication	vii
Access to publications and terminology	vii
Accessibility	ix
Technical training	ix
Support information	ix
Statement of Good Security Practices	x

Chapter 1. Introduction	1
IBM Security Directory Integrator Editions	1

Chapter 2. Installation instructions for IBM Security Directory Integrator	3
Before you install	3
Disk space requirements	3
Memory requirements	3
Platform requirements	3
Components in IBM Security Directory Integrator	3
Other requirements	5
Root or Administrator Privileges	5
Security Enhanced (SELinux)	6
Authentication of AMC on Unix/Linux	7
Graphics packages for UNIX systems	7
Prerequisites for CE on AIX operating system	7
Prerequisite for upgrading from V7.1.1 to V7.2 on Windows 2012 operating system	8
Installing IBM Security Directory Integrator	8
Launching the appropriate installer	9
Using the platform-specific IBM Security Directory Integrator installer	12
Installing using the graphical installer	12
Install Panel flow	12
Uninstall Panel flow	31
Add Feature Panel flow	36
Migration Panel flow	38
Installing using the command line	41
Temporary file space usage during installation	43
Performing a silent install	43
Service name limitation on UNIX systems	43
Post-installation steps	44
CE Update Site	44
Plug-ins	44
Administration and Monitoring Console (AMC)	44
Documentation	45
Migration	45
Installing local Help files	45
Deploying AMC to a custom ISC SE or IBM Dashboard Application Services Hub	47
Installing or Updating using the Eclipse Update Manager	48
Post-installation steps	50
Uninstalling	50
Launching the uninstaller	50
Performing a silent uninstallation	51
Default installation locations	51
Default solution directory	52

Chapter 3. Update Installer	53
The registry file	55
Installation of fix packs	57
Rollback	57
Troubleshooting	58

Chapter 4. Supported platforms	59
---	-----------

Chapter 5. Migrating	61
Migrate files to a different location	61
Which files do not need to be modified to be used in another location?	61
Which files need to be modified before they can be used in another location?	62
Which files should not be used in another location under normal circumstances?	63
Migrating files that contain encrypted data	63
Migrate files to a newer version	63
Installer-assisted migration	64
Tool-assisted migration	65
Manual migration	65
Backing up important data	78
Files backed up by the Installer	78
Upgrade from version 6.0 to 7.1	78
Upgrade from version 6.1.x to 7.1	78
Upgrade from version 7.0 to 7.1	79
Upgrade from version 7.1 to 7.1.1	79
Upgrade from version 7.1.1 to 7.2	79
Backup tools	79
Manual backup	80
Migrating AMC 7.x configuration settings to another AMC deployment	80
Converting from EventHandlers to corresponding AssemblyLines	81
TCP Server Connector	82
Mailbox Connector	83
JMX Connector	83
SNMP Server Connector	83
IBM Security Directory Server Changelog Connector	84
HTTP Server Connector	84
LDAP Server Connector	85
Sun Directory Change Detection Connector	85
Active Directory Change Detection Connector	86
DSMLv2SOAPServerConnector	87
Migrating BTree tables and BTree Connector to System Store	88
Migrating Cloudscape database to Derby	88
Migrating global and solution properties files using migration tool	90
Migrating Password plug-ins properties files using migration tool	91
Chapter 6. Security	93
Manage keys, certificates and keystores	93

Background	93	Creating an encrypted IBM Security Directory Integrator configuration file from scratch	135
Public/private keys and certificates	94	Editing an encrypted IBM Security Directory Integrator configuration file	135
Secret keys	94	Standard encryption of global.properties or solution.properties.	135
Keystores	94	Encryption of properties in external property files	136
Keys for SSL	94	The IBM Security Directory Integrator Encryption utility	136
Keys for encryption	95	IBM Security Directory Integrator System Store Security	138
Tools	96	Miscellaneous Config File features	140
List the contents of a keystore	96	Component Password Protection	140
Create keys	96	Saving passwords to configured Properties	141
Secure Sockets Layer (SSL) Support	99	Protecting attributes from being printed in clear text during tracing	142
Server SSL configuration of IBM Security Directory Integrator components	100	Encryption of IBM Security Directory Integrator Server Hooks	142
Client SSL configuration of IBM Security Directory Integrator components	101	Remote CE and SSL	142
SSL client authentication.	102	Using the Remote Configuration Editor	143
IBM Security Directory Integrator and Microsoft Active Directory SSL configuration	102	Summary of configuration files and properties dealing with security	144
Summary of properties for enabling SSL and PKCS#11 support	104	Web Admin Console Security	147
SSL example.	105	Miscellaneous security aspects.	147
IBM Security Directory Integrator component as a server	105	HTTP Basic Authentication	147
IBM Security Directory Integrator component as a client	106	Lotus Domino SSL specifics	147
Remote Server API	107	Certificates for the IBM Security Directory Integrator Web service Suite	148
Configuring the Server API.	108	IBM WebSphere MQ Everyplace authentication with mini-certificates	148
Remote Server API access on a Virtual Private Network	110	Chapter 7. Reconnect Rule Engine	151
Server API access options	110	Reconnect Rules	151
Server API SSL remote access	111	User-defined rules configuration	153
Using Server API specific SSL properties	111	Exception considerations	154
Using the standard SSL Java System properties	112	General reconnect configuration	155
Server API authentication	113	Chapter 8. System Queue	157
Local client session	113	System Queue Configuration	157
Remote client session.	113	Apache ActiveMQ parameters.	158
JAAS authentication	113	Configuration	158
SSL-based authentication	114	Logging	159
Username/password based authentication	115	Using SSL with ActiveMQ	159
LDAP Authentication support	116	IBM WebSphere MQ Everyplace parameters	160
LDAP Authentication Configuration.	116	IBM WebSphere MQ parameters	160
LDAP Authentication Logic	118	Microbroker parameters	161
LDAP Group Support	118	JMSScript Driver parameters	161
Host based authentication	120	The env JavaScript object	162
Summary of Server API Authentication options	121	The ret JavaScript object.	162
Server API JMX layer.	121	JavaScript example for Fiorano MQ	163
Server API authentication setup examples	122	System Queue Configuration Example	163
Server API Authorization	123	Security and Authentication	164
Authorization roles	124	IBM WebSphere MQ Everyplace Configuration Utility	164
Server API User Registry	125	Authentication of IBM WebSphere MQ Everyplace messages to provide Queue Security.	165
Server Audit Capabilities	128	Support for DNS names in the configuration of the IBM WebSphere MQ Everyplace Queue	166
Auditing scope.	129		
Suppression of notifications	129		
Sending notifications	130		
IBM Security Directory Integrator Server Instance Security	131		
Stash File.	131		
Server Security Modes	132		
Working with encrypted IBM Security Directory Integrator configuration files	133		

Configuration of High Availability for IBM WebSphere MQ Everyplace transport of password changes	167
Providing remote configuration capabilities in the IBM WebSphere MQ Everyplace Configuration Utility	167

Chapter 9. Encryption and FIPS mode 169

Configuring IBM Security Directory Integrator to run FIPS mode	169
Symmetric cipher support	169
FIPS encryption	170
Connectors, Function Components, Parsers	170
The IBM Security Directory Integrator server and FIPS	171
Configuring SSL and PKI certificates	179
Encrypting and decrypting using CryptoUtils	180
Working with certificates	180
Comparing CA-signed and Self-signed certificates	181
Configuring certificates using PKI and SSL	181
Using cryptographic keys located on hardware devices	182
Using IBMPCSKS11.	183
Enabling or disabling padding	183
Maintaining encryption artifacts - keys, certificates, keystores, encrypted files	183

Chapter 10. Configuring the IBM Security Directory Integrator Server API. 185

Server ID.	185
Exception for password protected Configs.	185
Server RMI	186
Config load time-out interval	186

Chapter 11. Properties 187

Working with properties.	187
Global properties	188
Solution properties	188
Java properties	189
System properties	190

Chapter 12. System Store 191

Property stores	194
Third-party RDBMS as System Store.	195
Oracle	196
MS SQL Server	196
IBM DB2	197
IBM solidDB	197
Using Derby to hold your System Store	198
Configuring Apache Derby Instances	199
Starting Apache Derby in networked mode	199
Enabling user authentication in System Store	199
Create statements for System Store tables	200
Backing up Apache Derby databases	201
Troubleshooting Apache Derby issues	201

Chapter 13. Command-line options 205

Configuration Editor	205
Server	206
Command Line Interface – tdisrvctl utility.	209
Command Line Reference	210
Operations	211

Chapter 14. Logging and debugging 223

Script-based logging	224
Logging using the default Log4J class	224
Log Levels and Log Level control	228
Log4J default parameters	229
Creating your own log strategies	230

Chapter 15. Tracing and FFDC 233

Tracing Enhancements	233
Understanding Tracing	233
Configuring Tracing	234
Setting trace levels dynamically	234
Useful JLOG parameters.	235

Chapter 16. Administration and Monitoring 237

Installation and Configuration.	237
Deploying AMC into the Integrated Solutions Console	237
Deploying AMC as a Windows service or UNIX process using the IBM Security Directory Integrator installer	238
Deploying AMC on existing IBM WebSphere Application Server environment	238
Starting and logging in the AMC and Action Manager	238
Enabling AMC	239
Running Action Manager remotely	240
AMC Logs	242
AMC in the Integrated Solutions Console	242
Action Manager	243
Enabling Action Manager	248
Action Manager status in real time	250
AMC force trigger for a given rule	250
AMC and Action Manager security	251
AMC and SSL	251
AMC and remote IBM Security Directory Integrator server	253
AMC and role management	253
AMC and passwords	255
AMC and encrypted configs	255
Administration and Monitoring Console User Interface	255
Log in and logout of the console	255
AMC Console Layout	256
Logging off the console	257
Using AMC tables.	257
Select action drop-down menu	258
Paging	258
Sorting	258
Finding	259
Filtering	259
Servers	260

Add a server	260
Modify a server	261
Console Properties	261
Solution Views	263
Configure ACLs	263
Local variables	264
Add a Solution View	264
Config files (allows loading/reloading of configurations)	266
Custom load	267
Monitor Status and Action Manager	267
Monitor Status	268
Solution View Details.	268
View Components	271
Show Preferred Solution Views	271
Refreshing Solution View Details in AMC	271
Action Manager	272
Add/Edit configuration rules	272
Add/Modify Action	274
Substitute variable for event data.	278
View Rules Summary.	280
Property Stores	280
Log Management	281
Preferred Solution Views	282
AMC and AM Command line utilities	283
Example walkthrough of creating a Solution View and Rules	288

Chapter 17. Touchpoint Server 295

Touchpoint concepts	295
Touchpoint Server	295
Touchpoint Provider	296
Touchpoint Type	296
Touchpoint Instance	298
Touchpoint Template	301
Resource Persistence	305
Touchpoint Schema	306
Touchpoint Configuration	310
Instance Configuration	310
Destination Configuration	311
Touchpoint Instance communication protocol	312
Provider Touchpoint	312
Initiator Touchpoint	313
Intermediary Touchpoint	313
Representation of Entry objects as HTTP content	313
Touchpoint Status Entry schema	314
Property sheet definitions	314
XML Schema locations	316
Error flows	316
Configuration	317
Authentication	319
Examples	319
Shipped example	319
Example steps for creating a Touchpoint Instance using a JDBC Connector.	320
Provider Touchpoint Instance	320
Initiator Touchpoint Instance	321
Intermediary Touchpoint Instance	322

Chapter 18. Tombstone Manager 325

Configuring Tombstones.	325
Configuration Editor Configuration screen.	325
AssemblyLine Configuration screen	327
The Tombstone Manager	328

Chapter 19. Multiple IBM Security Directory Integrator services. 331

IBM Security Directory Integrator as Windows Service	331
Installing and uninstalling the service	331
Installing the service	331
Uninstalling the service	332
Starting and stopping the service.	333
Logging	333
Configuring the service	333
IBM Security Directory Integrator as Linux/UNIX Service	335
Command line support	337

Appendix A. Example Property files 339

Log4J.properties	340
jlog.properties	341
derby.properties	342
global.properties	342

Appendix B. Monitoring with external tools 349

Monitoring IBM Security Directory Integrator with ITM	350
Short presentation of the ITM architecture.	350
Importing an existing Agent configuration in ITM Agent Builder 6.2	350
Creating an IBM SDI agent for ITM using ITM Agent Builder 6.2	351
Generating the ITM Agent	357
Configuring the ITM Agent.	358
Monitoring IBM Security Directory Integrator data	359
Defining thresholds	360
Creating links between tables	364
Purpose of links	364
Construction of links	365
Send custom notifications to ITM.	370
Limitations	370
Monitoring IBM SDI using OMNIbus	371
Configuring the EIF probe props file	371
Determine the severity for the events	371
Working with the EventPropertyFile.properties file	372
Send custom notifications to OMNIbus.	374

Appendix C. Accessibility features for IBM Security Directory Integrator 375

Notices 377

Index 381

About this publication

This publication contains the information that you require to develop solutions by using components that are part of IBM® Security Directory Integrator.

IBM Security Directory Integrator components are designed for network administrators who are responsible for maintaining user directories and other resources. It is assumed that you have practical experience with installation and usage of both IBM Security Directory Integrator and IBM Security Directory Server.

The information is also intended for users who are responsible for the development, installation, and administration of solutions by using IBM Security Directory Integrator. The reader must be familiar with the concepts and the administration of the systems that the developed solution would connect to. Depending on the solution, these systems might include, but are not limited to, one or more of the following products, systems, and concepts:

- IBM Security Directory Server
- IBM Security Identity Manager
- IBM Java™ runtime environment (JRE) or Oracle Java runtime environment
- Microsoft Active Directory
- Windows and UNIX operating systems
- Security management
- Internet protocols, including HyperText Transfer Protocol (HTTP), HyperText Transfer Protocol Secure (HTTPS) and Transmission Control Protocol/Internet Protocol (TCP/IP)
- Lightweight Directory Access Protocol (LDAP) and directory services
- A supported user registry
- Authentication and authorization concepts
- SAP ABAP Application Server

Access to publications and terminology

Read the descriptions of the IBM Security Directory Integrator Version 7.2.0.1 library and the related publications that you can access online.

This section provides:

- A list of publications in the “IBM Security Directory Integrator library.”
- Links to “Online publications” on page viii.
- A link to the “IBM Terminology website” on page ix.

IBM Security Directory Integrator library

The following documents are available in the IBM Security Directory Integrator library:

- *IBM Security Directory Integrator Version 7.2.0.1 Federated Directory Server Administration Guide*

Contains information about using the Federated Directory Server console to design, implement, and administer data integration solutions. Also contains

information about using the System for Cross-Domain Identity Management (SCIM) protocol and interface for identity management.

- *IBM Security Directory Integrator Version 7.2.0.1 Getting Started Guide*
Contains a brief tutorial and introduction to IBM Security Directory Integrator. Includes examples to create interaction and hands-on learning of IBM Security Directory Integrator.
- *IBM Security Directory Integrator Version 7.2.0.1 Users Guide*
Contains information about using IBM Security Directory Integrator. Contains instructions for designing solutions using the Security Directory Integrator designer tool (the Configuration Editor) or running the ready-made solutions from the command line. Also provides information about interfaces, concepts and AssemblyLine creation.
- *IBM Security Directory Integrator Version 7.2.0.1 Installation and Administrator Guide*
Includes complete information about installing, migrating from a previous version, configuring the logging functionality, and the security model underlying the Remote Server API of IBM Security Directory Integrator. Contains information on how to deploy and manage solutions.
- *IBM Security Directory Integrator Version 7.2.0.1 Reference Guide*
Contains detailed information about the individual components of IBM Security Directory Integrator: Connectors, Function Components, Parsers, Objects and so forth – the building blocks of the AssemblyLine.
- *IBM Security Directory Integrator Version 7.2.0.1 Problem Determination Guide*
Provides information about IBM Security Directory Integrator tools, resources, and techniques that can aid in the identification and resolution of problems.
- *IBM Security Directory Integrator Version 7.2.0.1 Message Guide*
Provides a list of all informational, warning and error messages associated with the IBM Security Directory Integrator.
- *IBM Security Directory Integrator Version 7.2.0.1 Password Synchronization Plug-ins Guide*
Includes complete information for installing and configuring each of the five IBM Password Synchronization Plug-ins: Windows Password Synchronizer, Sun Directory Server Password Synchronizer, IBM Security Directory Server Password Synchronizer, Domino[®] Password Synchronizer and Password Synchronizer for UNIX and Linux. Also provides configuration instructions for the LDAP Password Store and JMS Password Store.
- *IBM Security Directory Integrator Version 7.2.0.1 Release Notes*
Describes new features and late-breaking information about IBM Security Directory Integrator that did not get included in the documentation.

Online publications

IBM posts product publications when the product is released and when the publications are updated at the following locations:

IBM Security Directory Integrator Library

The product documentation site (<http://www-01.ibm.com/support/knowledgecenter/SSCQGF/welcome>) displays the welcome page and navigation for this library.

IBM Security Systems Documentation Central

IBM Security Systems Documentation Central provides an alphabetical list of all IBM Security Systems product libraries and links to the online documentation for specific versions of each product.

IBM Publications Center

The IBM Publications Center site (<http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss>) offers customized search functions to help you find all the IBM publications you need.

Related information

Information related to IBM Security Directory Integrator is available at the following locations:

- IBM Security Directory Integrator uses the JNDI client from Oracle. For information about the JNDI client, see the *Java Naming and Directory Interface™ Specification* at <http://download.oracle.com/javase/7/docs/technotes/guides/jndi/index.html> .
- Information that might help to answer your questions related to IBM Security Directory Integrator can be found at https://www-947.ibm.com/support/entry/myportal/over-accesspubsview/software/security_systems/tivoli_directory_integrator.

IBM Terminology website

The IBM Terminology website consolidates terminology for product libraries in one location. You can access the Terminology website at <http://www.ibm.com/software/globalization/terminology>.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see the Accessibility Appendix in *Configuring Directory Integrator*.

Technical training

For technical training information, see the following IBM Education website at <http://www.ibm.com/software/tivoli/education>.

Support information

IBM Support provides assistance with code-related problems and routine, short duration installation or usage questions. You can directly access the IBM Software Support site at <http://www.ibm.com/software/support/probsub.html>.

Troubleshooting provides details about:

- What information to collect before contacting IBM Support.
- The various methods for contacting IBM Support.
- How to use IBM Support Assistant.
- Instructions and problem-determination resources to isolate and fix the problem yourself.

Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service or security measure can be completely effective in preventing improper use or access. IBM systems, products and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.

Chapter 1. Introduction

Read about the general concepts of IBM Security Directory Integrator before you start the installation and administration tasks.

For an overview of the general concepts of IBM Security Directory Integrator, refer to "IBM Security Directory Integrator concepts," in *Configuring Directory Integrator*.

For more detailed information about IBM Security Directory Integrator concepts, see *Reference*.

IBM Security Directory Integrator Editions

Use the information provided here to know about the different editions of the product.

The IBM Security Directory Integrator Version 7.2 exists in two different editions (for which different licensing agreements apply):

- General Purpose Edition: Licensing for this edition is done on a per-processor basis.
- Identity Edition: Licensing is done on a per-user basis.

Unlike earlier versions of IBM Security Directory Integrator, in Version 7.2, both the General Purpose Edition and the Identity Edition are identical in their content, functionality, and capabilities. They only differ in their licensing agreements.

Chapter 2. Installation instructions for IBM Security Directory Integrator

Installing IBM Security Directory Integrator comprises of checking the requirements in prior, installing the software and performing some tasks to finally get the software working.

Before you install

You must ensure that your system meets the minimum requirements before you begin installing IBM Security Directory Integrator.

The IBM Security Directory Integrator installer uses the InstallAnywhere 2012 SP1 technology.

Disk space requirements

Check the disk space requirements at the link provided here.

See the Software requirements in the IBM Security Directory Integrator documentation.

Memory requirements

Check the memory requirements at the link provided here.

See the Software requirements in the IBM Security Directory Integrator documentation.

Platform requirements

Check the platform requirements at the link provided here.

See the Software requirements in the IBM Security Directory Integrator documentation.

Components in IBM Security Directory Integrator

Go through the information provided here about the available components.

With some exceptions, the following components are available and selectable for installation as part of IBM Security Directory Integrator:

Runtime Server

A rules engine used to deploy and run IBM Security Directory Integrator integration solutions.

Configuration Editor

A development environment for creating, debugging and enhancing IBM Security Directory Integrator integration solutions.

Note: IBM Security Directory Integrator does not support the Configuration Editor (CE) on the following operating systems:

- Linux PPC
- Linux 390

- AIX® PPC 64

For information on how to develop solutions without a local Configuration Editor, see “Using the Remote Configuration Editor” on page 143.

Configuration Editor Update Site (Eclipse update site for CE)

Use the CE Update Site folder to install the IBM Security Directory Integrator Configuration Editor into an existing Eclipse installation. Use the Eclipse software update tool and use this folder as a local update site. The CE update site is only supported for deployment on Eclipse 3.5.1 or later.

Note: IBM Security Directory Integrator does not support the Configuration Editor Update Site on the following operating systems:

- Linux PPC
- Linux 390
- AIX PPC 64

For information on how to develop solutions without a local Configuration Editor, see “Using the Remote Configuration Editor” on page 143.

Java API documentation

Full HTML documentation of IBM Security Directory Integrator internals. Essential reference material for scripting in solutions, as well as for developing custom components.

Examples

A series of short, illustrative example Configs that highlight specific IBM Security Directory Integrator features or components.

Help system (Host IBM Security Directory Integrator help locally. The default is online.)

Note: This feature is deprecated and will be removed in a future version of IBM Security Directory Integrator.

You can install an IBM User Interface Help System built on Eclipse (formerly known as IBM Eclipse Help System, or IEHS) locally as an alternative to using the global online help service. This option requires manual download and deployment of the IBM Security Directory Integrator help files after installation.

If your platform meets the system requirements, you can proceed with the download and installation instructions documented in “Installing local Help files” on page 45.

embedded Web platform (includes Integrated Solutions Console SE) v8.1.0.3

IBM Security Directory Integrator includes an embedded lightweight Web server platform, sometimes referred to as LWI. This server platform is based on the Eclipse and Open Services Gateway Initiative (OSGI) architecture and supports running web applications and Web services. The runtime provides a secure infrastructure with a small footprint and minimal configuration. The embedded Web platform includes Integrated Solution Console SE, which is used as the default alternative for deploying AMC on an existing ISC installation. The embedded Web platform provides an OSGI based lightweight infrastructure for hosting Web applications and Web services with the following characteristics:

- Minimal footprint
- Minimal configuration

- Compatibility with OSGI based ISC

Note: The AMC feature and the embedded Web platform are deprecated and will be removed in a future version of IBM Security Directory Integrator.

AMC: Administration and Monitoring Console

A browser-based application for monitoring and managing running IBM Security Directory Integrator Servers. AMC runs in the Integrated Solutions Console (ISC). In previous releases, AMC was a servlet application that was deployed into an embedded or existing instance of IBM WebSphere Application Server.

Note: The AMC feature is deprecated and will be removed in a future version of IBM Security Directory Integrator. IBM WebSphere Application Server supports ISC SE 7.2.0.2 and IBM Dashboard Application Services Hub Version 3.1.

Password Synchronization Plug-ins

A solution built with the IBM Security Directory Integrator that can intercept password changes on a number of systems.

Additional components automatically installed that are not selectable:

Java Runtime Environment (JRE) 7.0.4

A subset of the Java Development Kit (JDK) that contains the core executable files and other files that constitute the standard Java platform. The JRE includes the Java Virtual Machine (JVM), core classes, and supporting files.

Note: The JRE used for any of the installed IBM Security Directory Integrator packages is independent of any system-wide JRE or JDK you may have installed on your system. The JRE is installed no matter what features are selected. The uninstaller requires the JRE, so it is always installed.

Miscellaneous

Contains the License Package, the Uninstaller, and the Update Installer.

The IBM Security Directory Integrator License Package contains the license files for IBM Security Directory Integrator.

Other requirements

You must ensure that your system meets certain other requirements as described here.

Root or Administrator Privileges

Ensure that you meet the requirements related to root or Administrator privileges.

Note the following differences when installing IBM Security Directory Integrator with administrator as opposed to non-administrator privileges:

- Anyone installing IBM Security Directory Integrator must have write privileges when installing to the specified installation location.
- Non-administrator users have different Configuration Editor shortcuts from administrative users.

- Users who do not have administrator privileges when installing IBM Security Directory Integrator do not see the "Register AMC as a Service" and "Register Server as System Service" windows.
- Once IBM Security Directory Integrator is installed using one particular non-root user ID, that same user ID must be used to carry out any further maintenance on that installation, like un-installation or migration to newer versions.

Security Enhanced (SELinux)

You can use the instructions provided here to modify the settings and run SELinux. This will help you to run SELinux error free.

RedHat Linux (RHEL) has a security feature known as Security Enhanced Linux or SELinux. SELinux provides security that protects the host from certain types of malicious attacks. RHEL version 5.0 defaults SELinux to enabled. The RHEL 5.0 SELinux default settings have been known to prevent Java from running properly. If you try to run the RHEL 5.0 IBM Security Directory Integrator installer, an error resembling the following output may display:

```
# ./install_sdiv72_linux_x86_64.bin

Initializing Wizard.....
Verifying JVM...

No Java Runtime Environment (JRE) was found on this system.
```

The reason for this error is that the Java Runtime Environment (JRE) that InstallAnywhere 2012 SP1 extracts to the /tmp directory does not have the required permissions to run. To avoid this error:

1. Disable SELinux: `setenforce 0`.
2. Run the IBM Security Directory Integrator installer.
3. Enable SELinux again: `setenforce 1`.

You can also edit the `/etc/selinux/config` configuration file to enable and disable SELinux. The default settings for the `/etc/selinux/config` file resemble the following lines:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - SELinux is fully disabled.
SELINUX=enforcing
# SELINUXTYPE= type of policy in use. Possible values are:
# targeted - Only targeted network daemons are protected.
# strict - Full SELinux protection.
SELINUXTYPE=targeted
```

Modifying SELINUX to either `SELINUX=permissive` or `SELINUX=disabled` allows the IBM Security Directory Integrator installer to run. However, both modifications of the SELINUX property, to either `SELINUX=permissive` or to `SELINUX=disabled`, affect the level of security for the host.

The IBM Security Directory Integrator installer uses a JRE located at `install_dir/jvm` that cannot run with the SELinux default settings. The installer makes a best effort to avoid the problems with the SELinux default settings by trying to change the IBM Security Directory Integrator JRE security permissions that are blocking the installer. The IBM Security Directory Integrator installer issues a command that changes the security permissions for the IBM Security Directory Integrator JRE so that it can run. The IBM Security Directory Integrator installer issues the following command:

```
chcon -R -t textrel_shlib_t install_dir/jvm/jre
```

Note: If the installer cannot issue the `chcon` command, or if there is an error when issuing the command, you must edit the permissions manually. Errors that resemble the following output indicate that the `chcon` command did not work:

```
[root@dyn9-37-225-164 V7.2]# ./ibmdisrv
Failed to find VM - aborting
```

```
[root@dyn9-37-225-164 V7.2]# ./ibmditk
Failed to find VM - aborting
```

```
[root@dyn9-37-225-164 V7.2]# bin/amc/start_tdiadc.sh
Failed to find VM - aborting
```

Authentication of AMC on Unix/Linux

There are some limitations of a non-root user while working on AMC. Use the information provided here to know more about it and the available workaround.

On some UNIX platforms the Administration and Monitoring Console (AMC) in ISE SE fails to authenticate users, even when correct credentials are specified. Such behavior is observed when AMC is run as a non-root user and the operating system uses a password database (for example, a `/etc/shadow` file). For more information on this issue, and for a workaround see "Authentication failure on UNIX when LWI runs as non-root user" in *Troubleshooting*.

Graphics packages for UNIX systems

Use the instructions provided here to resolve the error generated while running the CE without required graphics packages.

If the required graphics packages are not installed on UNIX systems, the following error might occur later when you run the `ibmditk` command to start the Configuration Editor:

```
No fonts found; this probably means that the fontconfig library is not correctly
configured. You may need to edit the fonts.conf configuration file.
More information about fontconfig can be found in the fontconfig(3) manual page
and on http://fontconfig.org
```

To avoid such errors complete the following steps:

1. Ensure that the following graphics packages are installed on UNIX systems:

- `libgtk-x11-2.0.so.0`
- `libgthread-2.0.so.0`

2. Run the following command:

```
export LD_LIBRARY_PATH=/usr/sfw/lib/:usr/lib:/lib
```

3. Install or ensure that the following file is installed:

```
/etc/fonts/fonts.conf
```

4. Run the following command:

```
export FONTCONFIG_PATH=/etc/fonts
```

Prerequisites for CE on AIX operating system

Use the instructions provided here to install the RPMs on AIX.

When the CE is installed as a plug-in in Eclipse on an AIX operating system, it does not launch and creates a log file.

To use the CE, the `gtk+` RPM and dependencies must be available on AIX. Install the following RPMs on AIX:

```
atk-1.12.3-2.aix5.2.ppc.rpm
cairo-1.8.8-1.aix5.2.ppc.rpm
expat-2.0.1-1.aix5.2.ppc.rpm
```

```
fontconfig-2.4.2-1.aix5.2.ppc.rpm
freetype2-2.3.9-1.aix5.2.ppc.rpm
gettext-0.10.40-6.aix5.1.ppc.rpm
glib2-2.12.4-2.aix5.2.ppc.rpm
gtk2-2.10.6-4.aix5.2.ppc.rpm
libjpeg-6b-6.aix5.1.ppc.rpm
libpng-1.2.32-2.aix5.2.ppc.rpm
libtiff-3.8.2-1.aix5.2.ppc.rpm
pango-1.14.5-4.aix5.2.ppc.rpm
pixman-0.12.0-3.aix5.2.ppc.rpm
xcursor-1.1.7-3.aix5.2.ppc.rpm
xft-2.1.6-5.aix5.1.ppc.rpm
xrender-0.9.1-3.aix5.2.ppc.rpm
zlib-1.2.3-3.aix5.1.ppc.rpm
```

Note: The installed RPMs must be the versions listed here because earlier or later versions may not be compatible.

To install these RPM versions, complete the following steps:

1. Download the RPMs to a new directory. You can find the RPMs at <ftp://ftp.software.ibm.com/aix/freeSoftware/aixtoolbox/RPMS/ppc/>.
2. Install the downloaded RPMs by using the following command. If an existing version of an RPM is already installed, the command upgrades or downgrades it to the downloaded version.

```
rpm -U *.rpm --force
```
3. Verify that the environment variable LIBPATH contains a path to the closure of the libraries. For example: LIBPATH=/opt/freeware/64/lib/.

Prerequisite for upgrading from V7.1.1 to V7.2 on Windows 2012 operating system

Use the information provided here to know more about the prerequisite for upgrading the versions.

If you are planning to upgrade IBM Security Directory Integrator Version 7.1.1 to Version 7.2 on Windows 2012 operating system, then ensure that **Windows 7 compatibility mode** is enabled on the Version 7.1.1 uninstaller.exe before you start the Version 7.2 installer. For more information, see the technical note at <http://www-01.ibm.com/support/docview.wss?uid=swg21634336>.

Installing IBM Security Directory Integrator

Use the installer to install IBM Security Directory Integrator in its entirety, only those IBM Security Directory Integrator components that you need, upgrade a previous version (versions 7.0, 7.1, or 7.1.1), or add features to an existing IBM Security Directory Integrator installation.

Note:

- Upgrading IBM Security Directory Integrator from versions 6.x or earlier, directly to Version 7.2 is not supported. You must first upgrade from version 6.x to version 7.1.1 and then from version 7.1.1 to Version 7.2.
- IBM Security Directory Integrator does not support the Configuration Editor (CE) on the following operating systems:
 - Linux PPC
 - Linux 390
 - AIX 64-bit

See “Using the Remote Configuration Editor” on page 143 and for more information on using the product without a locally-installed Configuration Editor.

When you choose to upgrade from a previous version, IBM Security Directory Integrator uninstalls the previous version; the uninstallation does not remove any files that the user has created. User created files are still available after the new installation completes. Configuration files such as `global.properties` and `am_config.properties` are migrated to IBM Security Directory Integrator Version 7.2, keeping any custom configuration changes that have been made.

Note: Though the IBM Security Directory Integrator installer backs up and restores some of the pre-defined configuration and property files, it is a good practice to also manually back up your files and databases that have critical data before you start the installation.

The IBM Security Directory Integrator Version 7.2 installation continues to include the features available in previous versions of IBM Security Directory Integrator:

- Administration and Monitoring Console (AMC).
- Configuration Editor (CE)
- Examples
- IBM User Interface Help System built on Eclipse
- Java API Documentation
- Runtime Server

Note: For the remainder of this *Installing and Administering*, the variable `TDI_install_dir` refers to the installation directory location chosen by the user on the **Destination Panel** during installation. See “Default installation locations” on page 51 for information on where IBM Security Directory Integrator is usually installed.

Launching the appropriate installer

You can launch the Installer either by using the Launchpad or installing it directly. Use the information provided here to know the detailed steps for launching.

You can launch the IBM Security Directory Integrator Installer by using one of the following methods:

Launch the installer from the Launchpad

The IBM Security Directory Integrator Launchpad provides essential getting started installation information and links to more detailed information on various installation, migration, and post installation topics. In addition, Launchpad allows you to launch the IBM Security Directory Integrator installer.

Note: Using the Launchpad requires that you have a supported Web browser installed and configured; if this is not the case, you cannot use the Launchpad. However, you can still use the platform-specific installer directly; see “Using the platform-specific IBM Security Directory Integrator installer” on page 12 for instructions on how to use the IBM Security Directory Integrator Installer.

1. Open the IBM Security Directory Integrator Launchpad by typing the following command at the command prompt:

- For Windows platforms, type:

```
Launchpad.bat
```

- For all other platforms, type:

```
Launchpad.sh
```

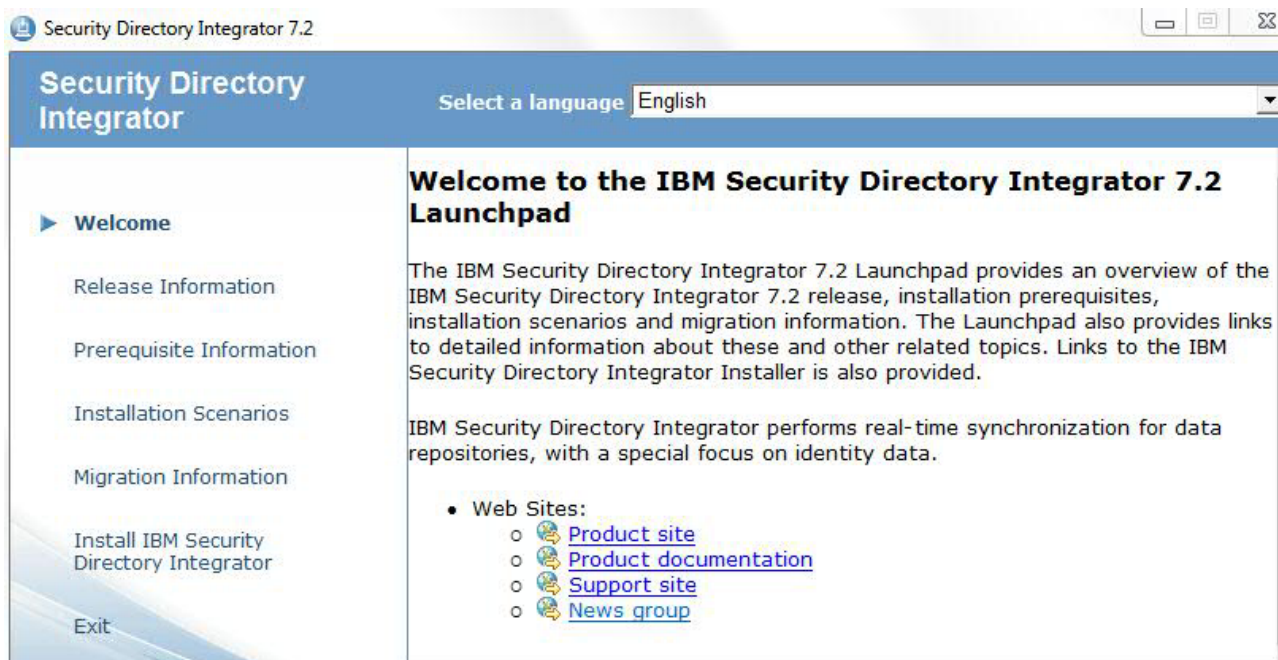
The menu on the left of the Launchpad allows you to navigate the Launchpad windows. Click a menu item to view information about it. The following menu items are available:

Welcome

The installation Welcome window contains links to:

- IBM Security Directory Integrator Web site
- IBM Security Directory Integrator Documentation
- Support Web site
- IBM Security Directory Integrator news group

Screen capture name: Adminst-1



The following options on the left are IBM Security Directory Integrator Launchpad windows:

Release Information

This window contains a list of some of the new and improved features available this release, as well as links to documentation about the release.

Prerequisite Information

This window contains links to information about platform support and hardware requirements.

Installation scenarios

This window contains a description of the IBM Security Directory Integrator components available for installation. You can install some or all of these components during installation. This window also contains a description of the Password Synchronization Plug-ins components available for installation.

Migration Information

This window contains a link to information about migrating

from IBM Security Directory Integrator 7.0, 7.1, or 7.1.1 to Version 7.2. It also contains information about migrating the Derby System Store.

Note: Upgrading IBM Security Directory Integrator from versions 6.x or earlier, directly to version 7.2, is not supported. You must first upgrade from version 6.x to version 7.1.1 and then from version 7.1.1 to version 7.2.

Install IBM Security Directory Integrator

This window contains links to the IBM Security Directory Integrator Installer, as well as links to installation, migration and supported platforms documentation. See “Using the platform-specific IBM Security Directory Integrator installer” on page 12 for instructions on how to use the IBM Security Directory Integrator Installer.

Install IBM Security Directory Integrator Password Synchronization Plug-ins

This window contains links to the IBM Security Directory Integrator Password Synchronizer Plug-ins Installer, as well as links to installation and supported platforms documentation.

Note: This window is not available on Linux PPC and Linux 390 platforms.

Exit Exits the Launchpad, without installing anything.

2. On the installation window, click **IBM Security Directory Integrator Installer**. This launches the installer. See “Using the platform-specific IBM Security Directory Integrator installer” on page 12 for instructions on how to use the installer.

Launch the installer directly

You can launch the installer directly using the installation executable file:

1. Locate the installation executable file for your platform in the `tdi_installer` directory on the product CD.

Windows Intel

`install_sdiv72_win_x86.exe`

Windows 64-bit

`install_sdiv72_win_x86_64.exe`

AIX `install_sdiv72_aix_ppc.bin`

AIX 64-bit

`install_sdiv72_aix_ppc_64.bin`

Linux 64-bit

`install_sdiv72_linux_x86_64.bin`

Power® PC Linux

`install_sdiv72_ppclinux.bin`

Solaris Sparc

`install_sdiv72_solaris_sparc.bin`

Solaris (Intel)

`install_sdiv72_solaris_x86_64.bin`

2. Double-click the executable file, or type the executable file name at the command prompt. This launches the installer. See “Using the

platform-specific IBM Security Directory Integrator installer” for information on how to use the installer.

Once you have launched the installer, you are ready to begin the process of “Using the platform-specific IBM Security Directory Integrator installer.”

Using the platform-specific IBM Security Directory Integrator installer

Use the instructions provided here to install the platform specific IBM Security Directory Integrator.

The platform-specific IBM Security Directory Integrator installer is launched either from the Launchpad or from the command line. The IBM Security Directory Integrator installer can be used to install a new copy of IBM Security Directory Integrator, add a feature to an existing instance of IBM Security Directory Integrator, or upgrade a previous version of IBM Security Directory Integrator. The default install location on your computer for IBM Security Directory Integrator varies with the platform.

During installation, the Installer will log its actions in files (sdiv72install.log and sdiv72debug.log) residing in the system's temporary files directory, typically /tmp or /var/tmp on UNIX platforms.

Installing using the graphical installer

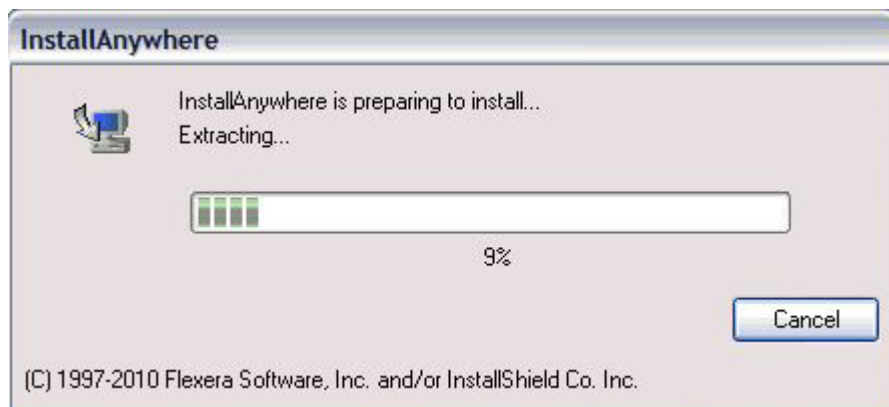
Use the instructions provided here to install the IBM Security Directory Integrator using graphical installer.

Install Panel flow

Use the instructions provided here to install the panel flow.

Pre-Initialization Panel

You invoke the installer executable either from the command line, or by double clicking the executable (Windows only). This panel will initially appear followed by a splash screen:



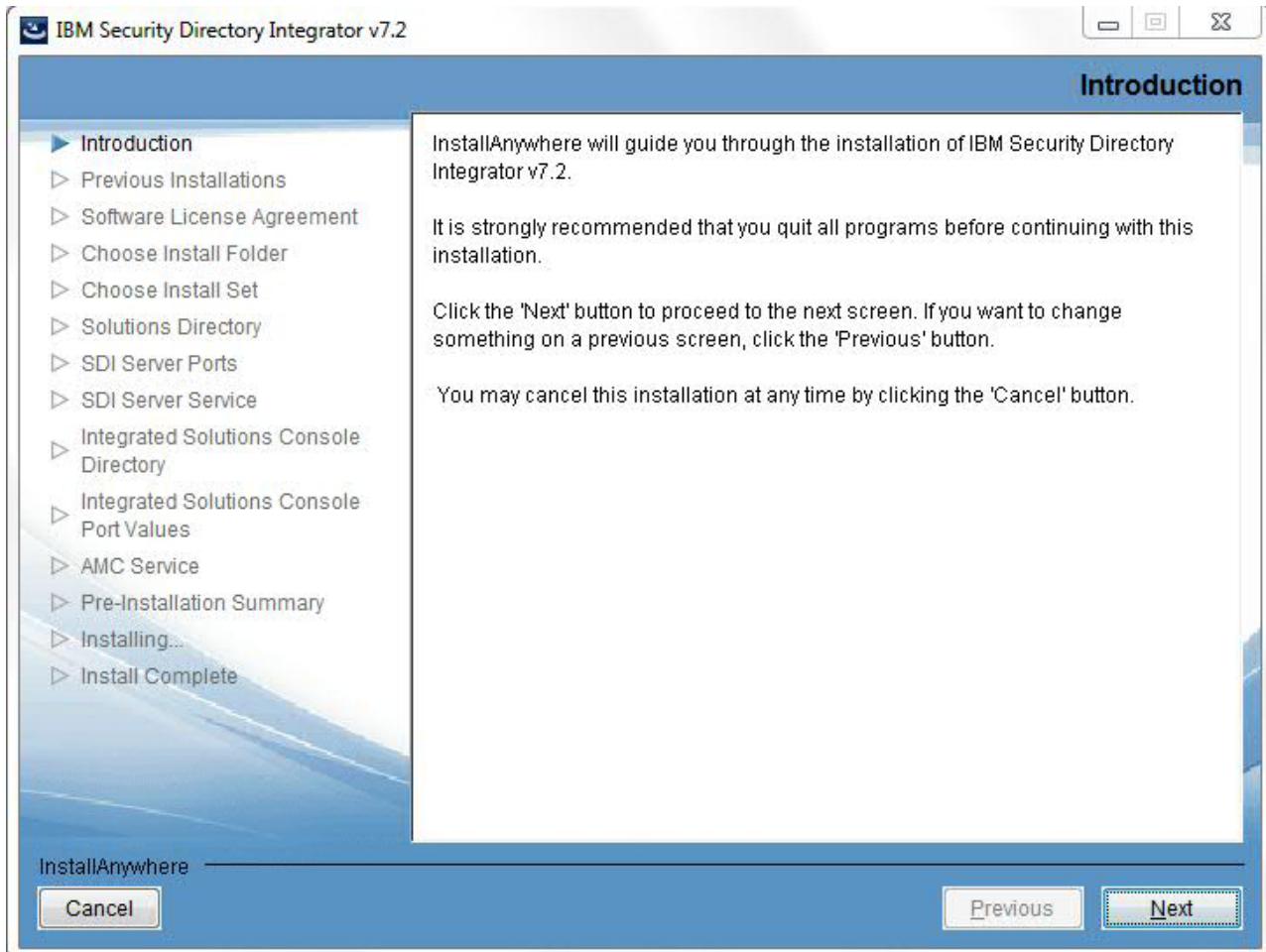
Note: The splash screen may also show a drop down list of language choices if the underlying system supports more than one. (The default is English.)





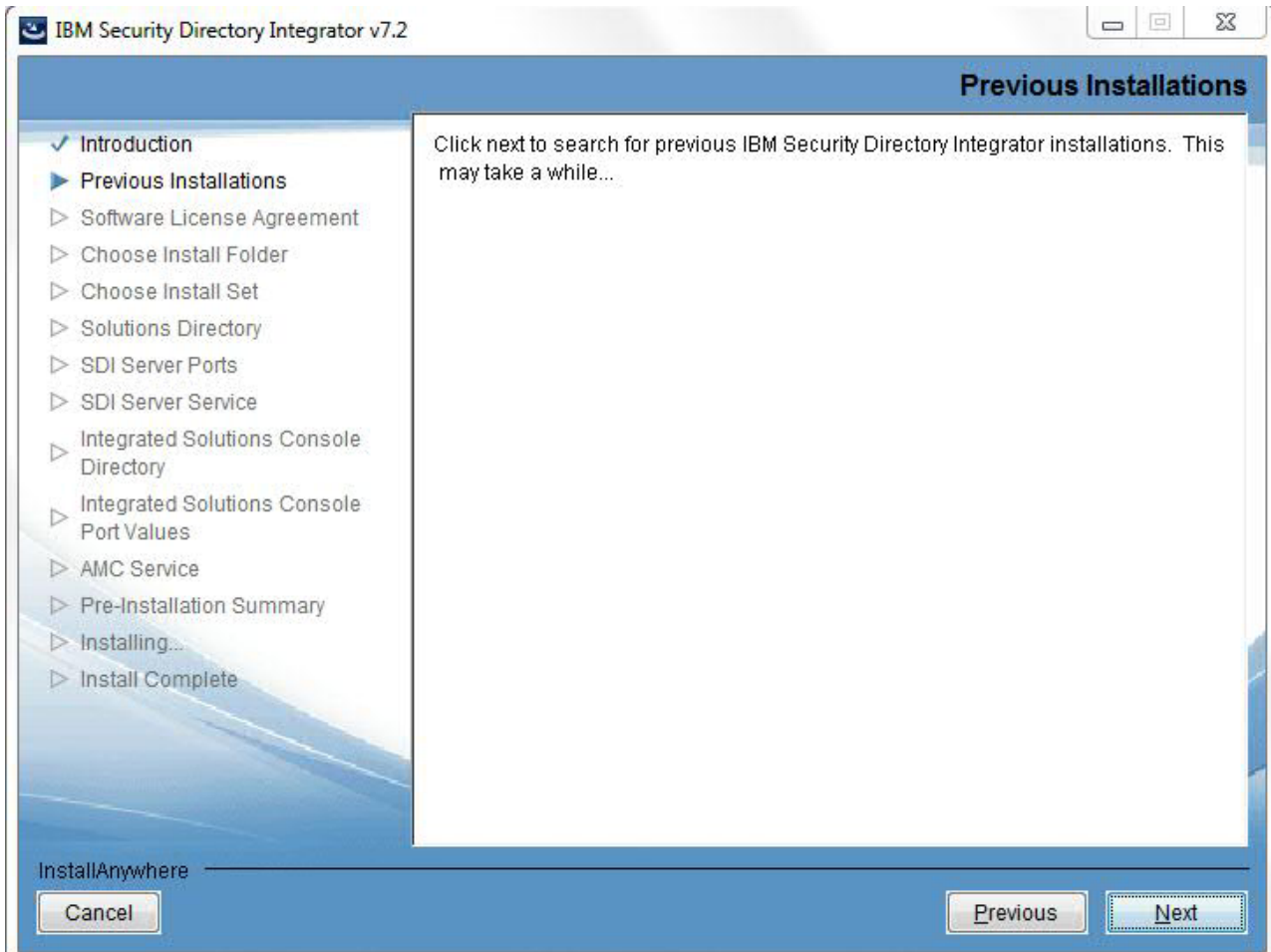
Introduction Panel

This is the welcome panel for the installer. This is the default panel provided by the InstallAnywhere installer. You have the option to continue by hitting the **Next** button or canceling out of the installer by pressing **Cancel**.



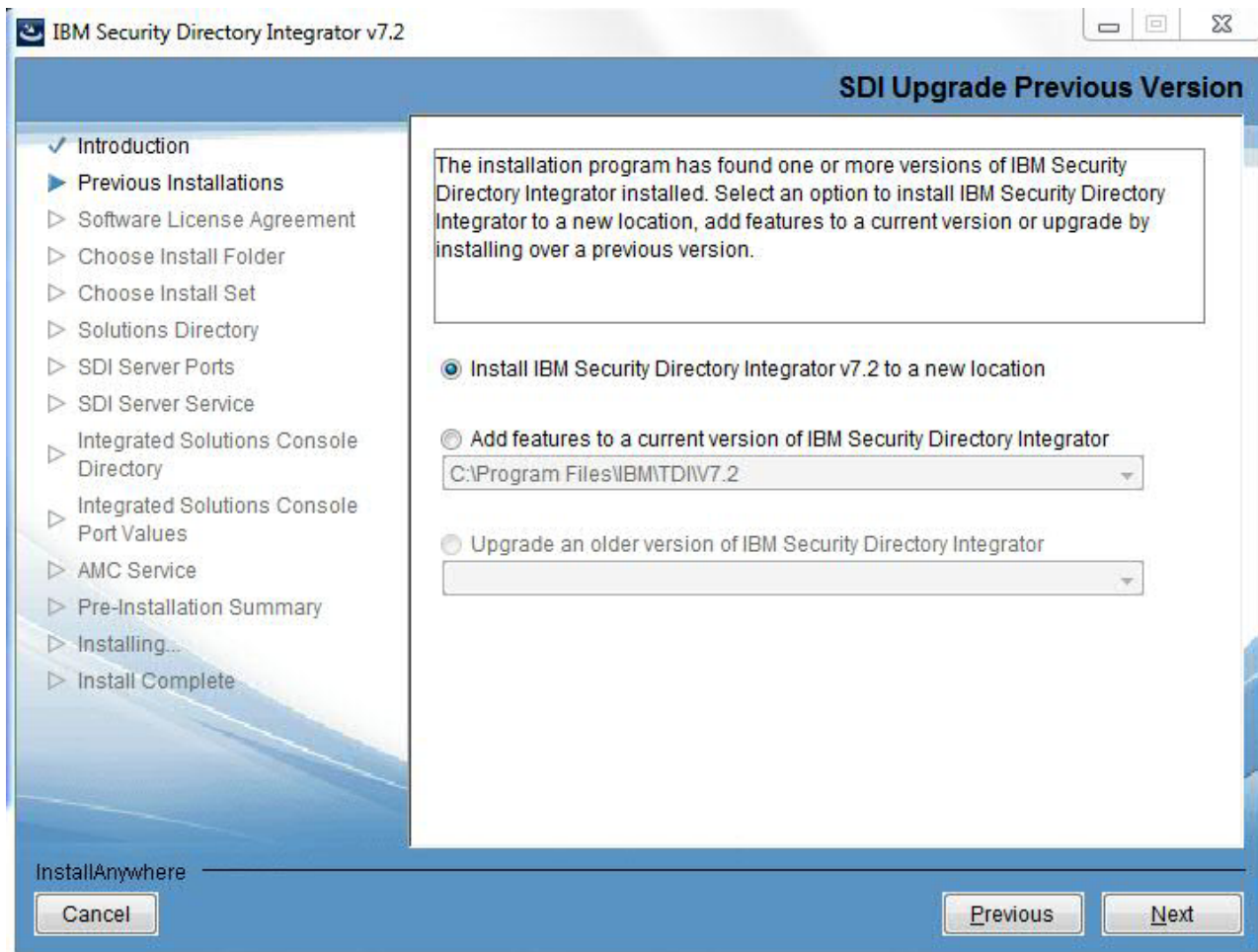
Previous Installations Panel

This panel informs your that detecting previous versions of IBM Security Directory Integrator may take some time.



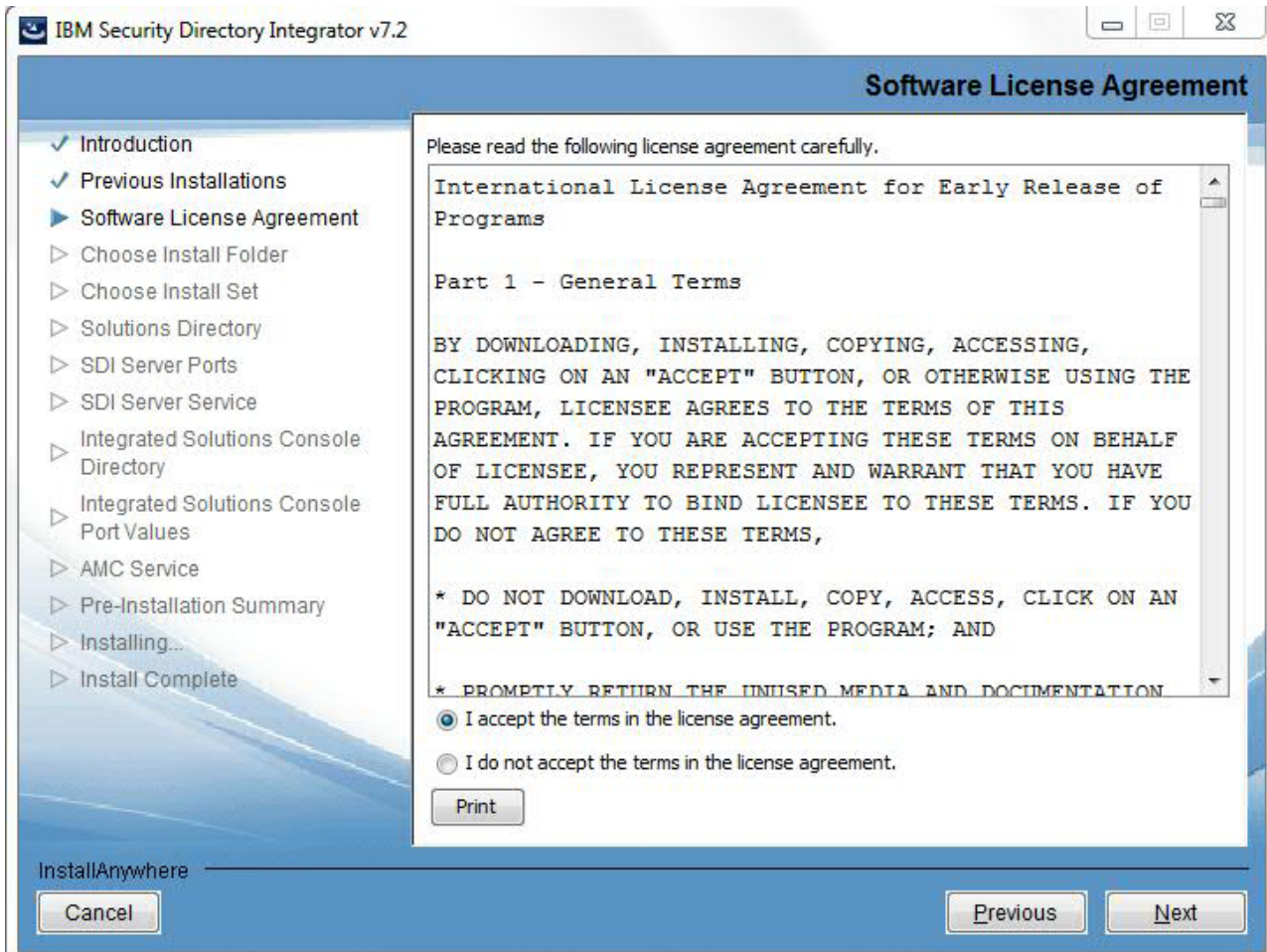
If a previous version is detected, you are presented with a number of upgrade options.

Note: Upgrading IBM Security Directory Integrator from versions 6.x or earlier, directly to version 7.2, is not supported. You must first upgrade from version 6.x to version 7.1.1 and then from version 7.1.1 to version 7.2.



Software License Agreement Panel

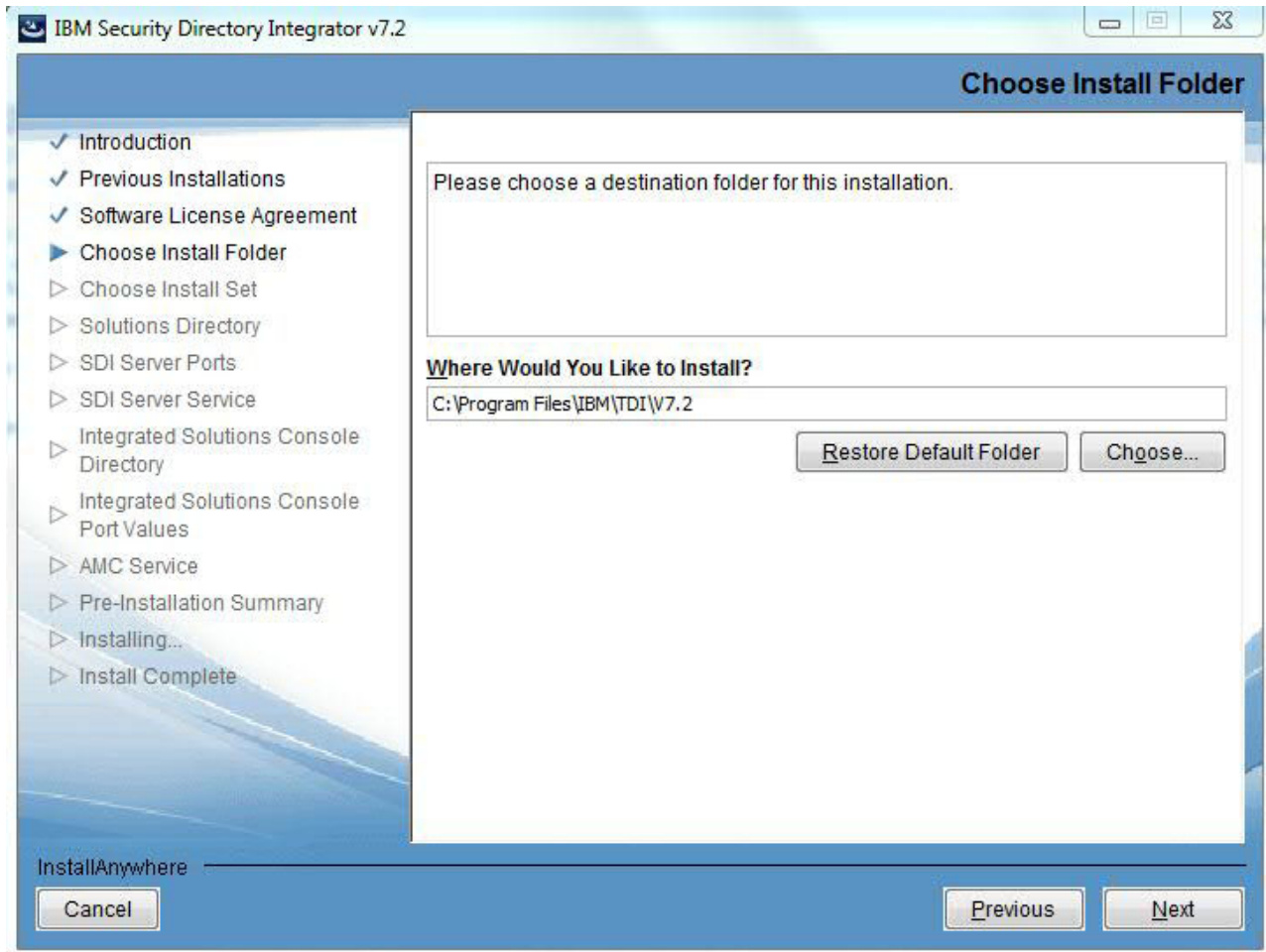
The license panel is provided by the IBM license tool. This panel will be shown in a **New Security Directory Integrator v7.2** install and **Upgrading an older Security Directory Integrator** version.



Choose Install Folder Panel

Note:

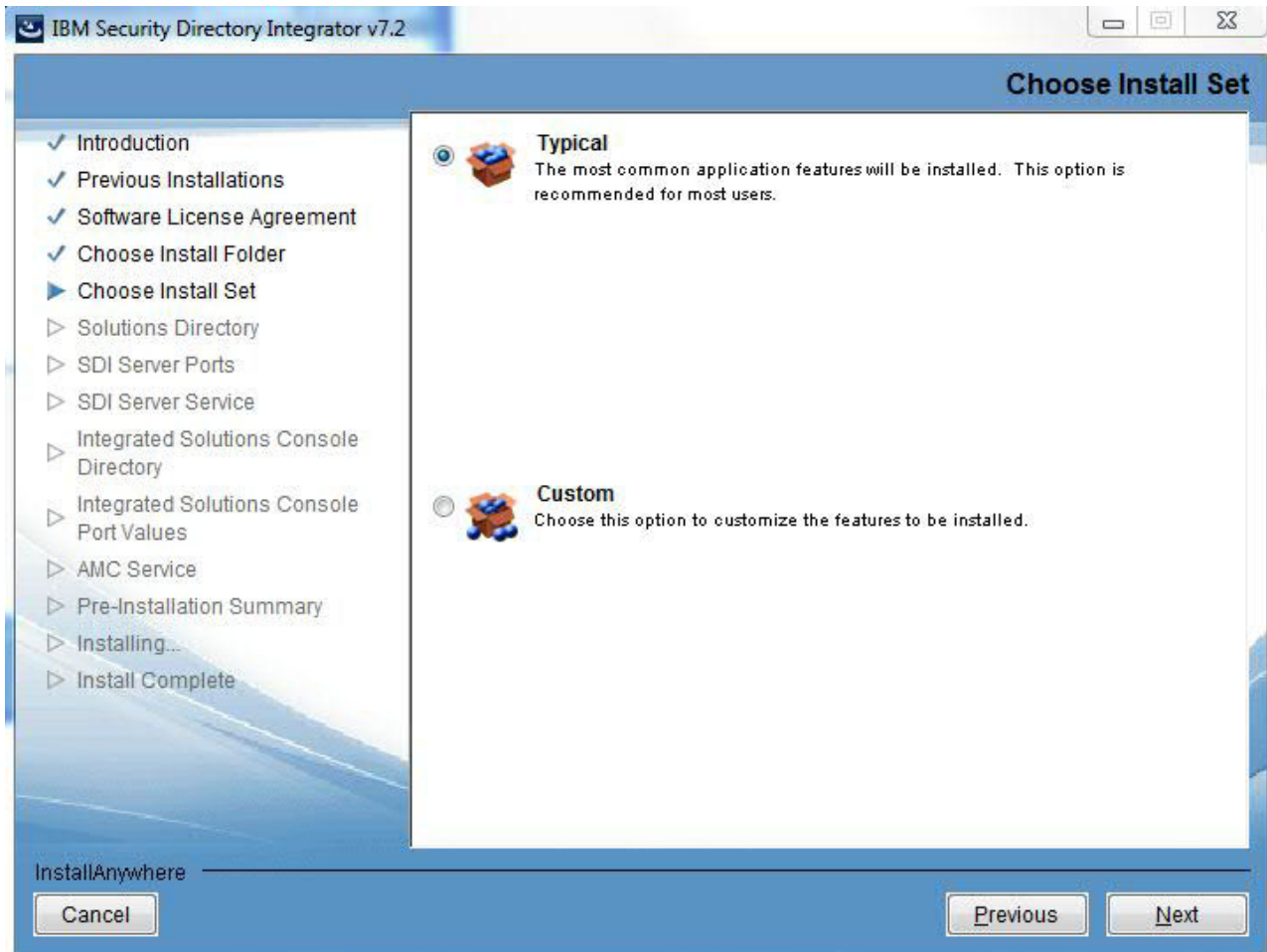
1. This panel will not be shown if an upgrade from IBM Security Directory Integrator Version 7.0, 7.1, or 7.1.1 was selected nor will it be shown if you are adding features to an existing IBM Security Directory Integrator Version 7.2 instance.
2. The destination panel will have the last value entered if you go forward in the wizard to other panels and then come back.
3. Non-ASCII characters and the following list of characters are not supported in the install path: ";|*?!#&\$',=^@%+



Choose Install Set Panel

The typical install includes the Runtime Server, the Configuration Editor (CE), Javadocs, Examples and AMC. It does not include the Configuration Editor Update Site, IBM User Interface Help System built on Eclipse, or the Password Synchronization Plug-ins.

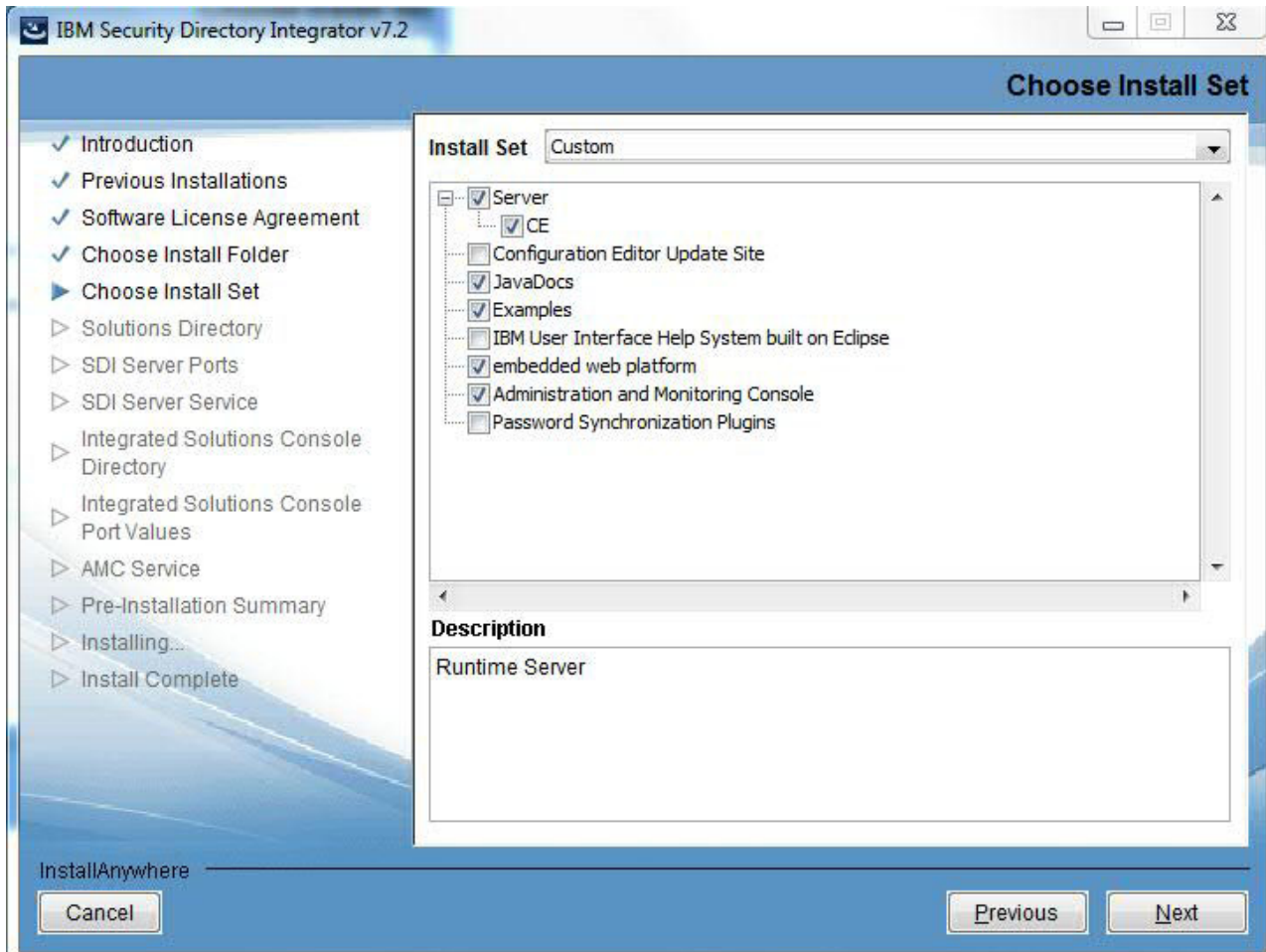
If you select **Typical**, the feature selection panel is skipped. Also, you will automatically get the bundled embedded Web platform/ISC package. The ISC Directory panel will be skipped.



Feature Selection Panel

This panel allows you to specify which features will be installed. Any feature can be individually installed if needed. The only exception to this is that if the Configuration Editor is selected, the server will be selected because the Configuration Editor is a subfeature of the server.

If any feature is not supported on the platform it will not be shown on the feature selection panel.



The following list summarizes each feature:

Runtime Server

A rules engine used to deploy and run IBM Security Directory Integrator integration solutions.

Configuration Editor

A development environment for creating, debugging and enhancing IBM Security Directory Integrator integration solutions. This feature can not be installed without installing the Runtime Server.

Configuration Editor Update Site

Patterned after the Eclipse Update Site. Contains the necessary files to install the Config Editor to an existing Eclipse. It will also be used for maintenance. (Not available on zLinux or Linux PPC.)

Javadocs

Full HTML documentation of IBM Security Directory Integrator internals. Essential reference material for scripting in solutions, as well as for developing custom components.

Examples

A series of short, illustrative example Configs that highlight specific IBM Security Directory Integrator features or components.

IBM User Interface Help System built on Eclipse (local help)

An IBM User Interface Help System (previously known as IEHS) built on Eclipse that you can install locally as an alternative to using the global online help service. This option requires manual download and deployment of IBM Security Directory Integrator help files after installation.

embedded web platform

The embedded Web platform package, which includes ISC SE.

Administration and Monitoring Console

A browser-based application for monitoring and managing running IBM Security Directory Integrator Servers.

Password Synchronization Plug-ins

IBM Security Directory Integrator password synchronization plug-ins.

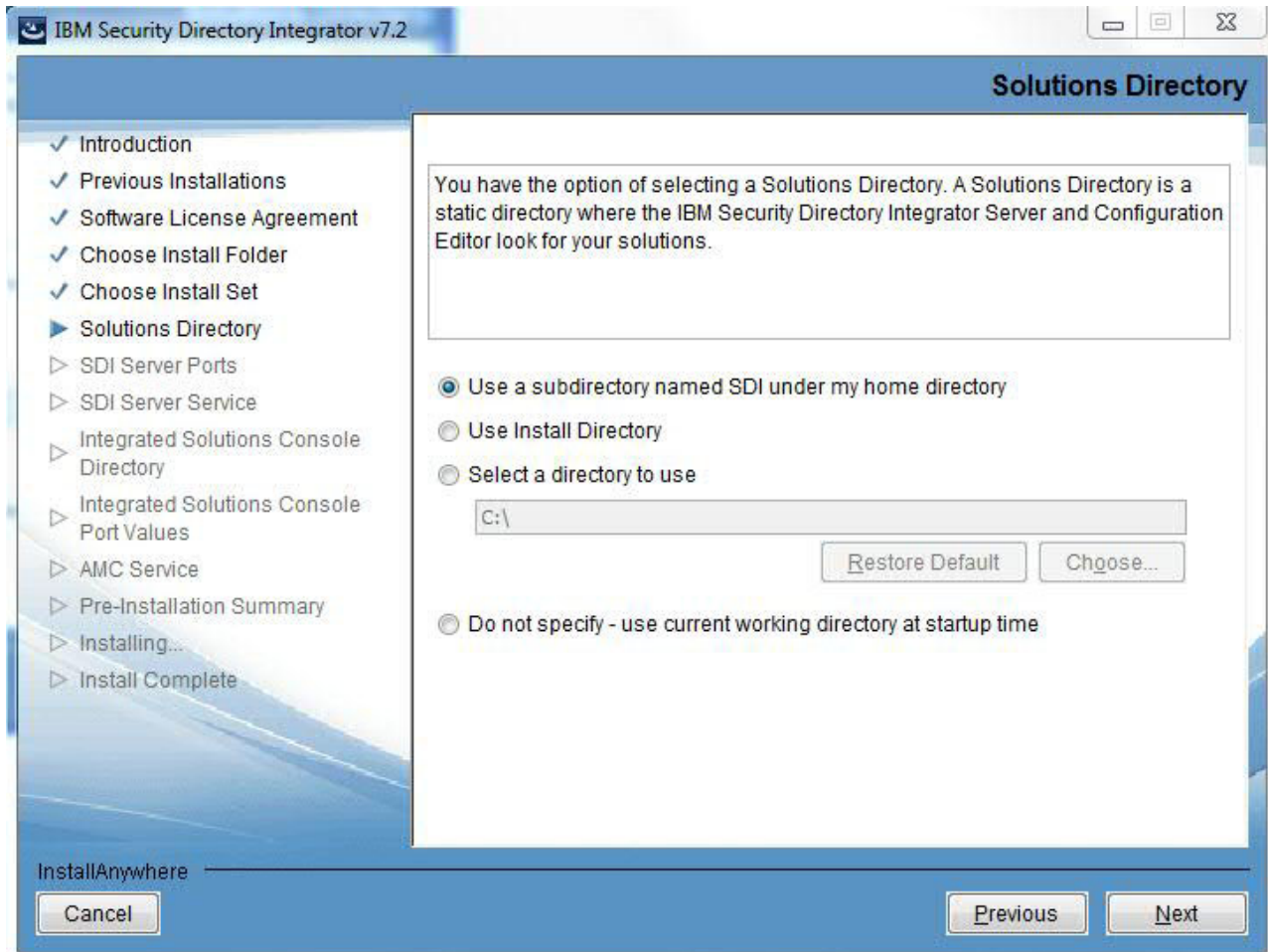
IBM Security Directory Integrator Solutions Directory Panel

This panel is only displayed if the Server feature was chosen. It lets you select the default Solution Directory for the server. The Solution Directory is a static directory containing the solutions created by the user that will be run. By default, this panel will select to have the Solution Directory set the user's home directory.

If you select the **Select a directory to use** radio button, you need to specify a valid Solution Directory. The Universal Naming Convention (UNC) path is supported for Solution Directory during installation time.

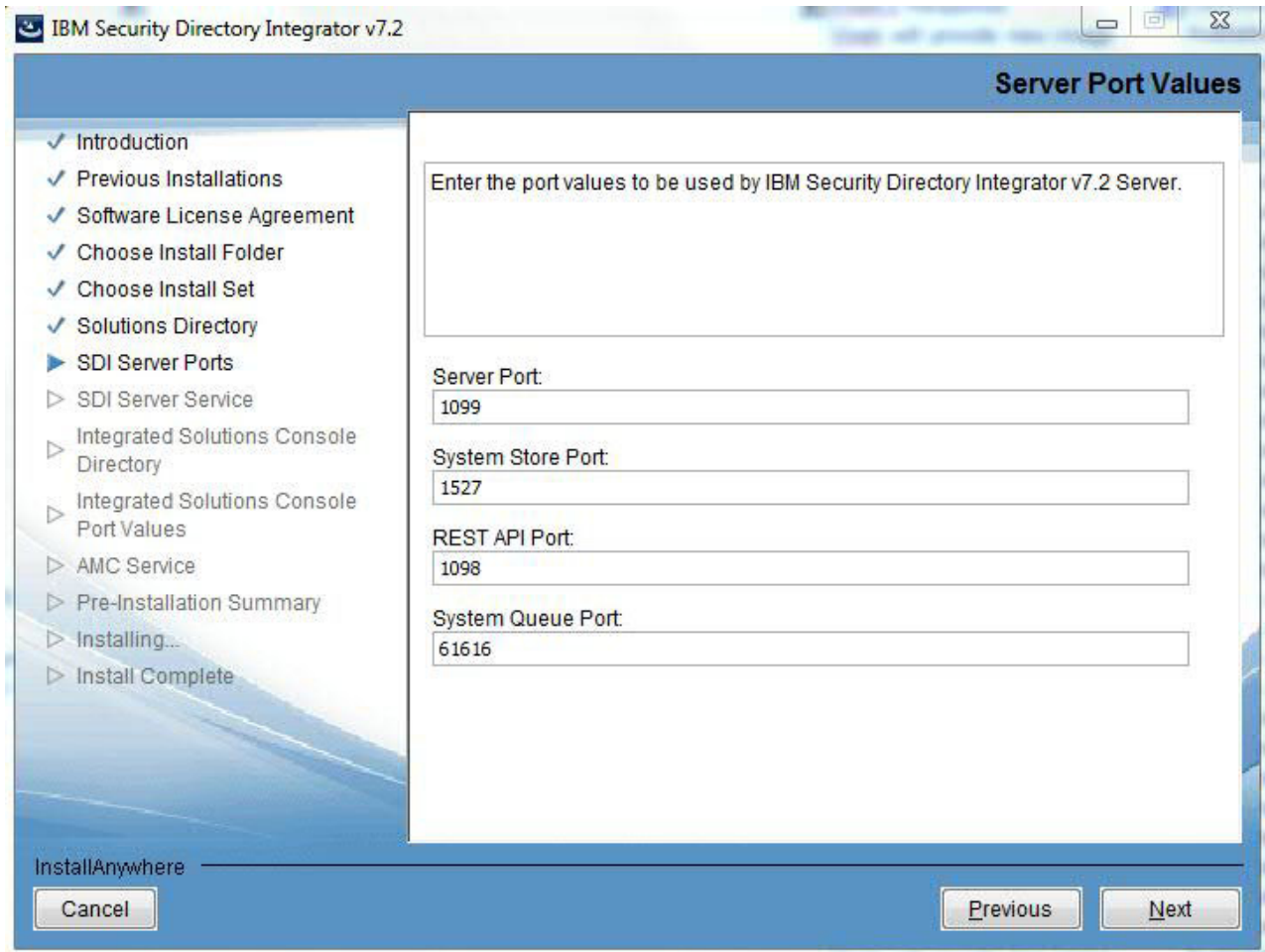
Note: This panel will not be shown in an upgrade from IBM Security Directory Integrator Version 7.0, 7.1, or 7.1.1.

If you are adding features, and the Server feature was already installed, this panel will not be shown.



Server Port Configuration Panel

You will be asked for 4 server ports numbers. There will be default values for these ports. The installer will make sure that you enter a valid and available port number (see Server Port Configuration).



Register Server as a System service panel

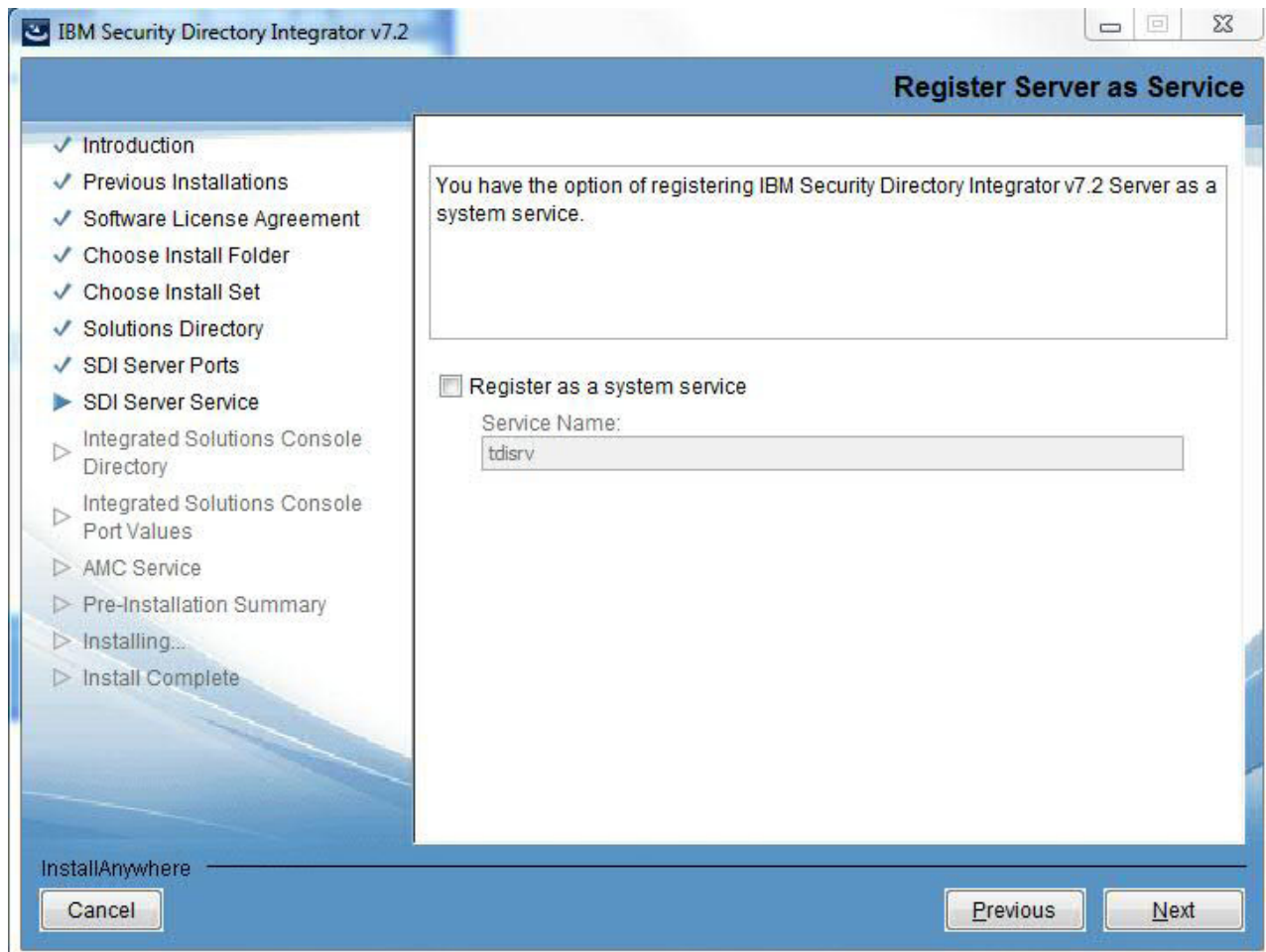
This panel will only be displayed if new instance of IBM Security Directory Integrator is getting installed and you have selected the Server to install as a feature or if it is an upgrade installation. Also this panel will only be displayed if you have Administrative privileges.

If the checkbox is checked, then only SERVER will be registered as a service for that OS.

The default is for the checkbox to be unchecked. The two text boxes will be enabled only if the checkbox is checked. The first text box is for service name and the second is for the port number that the server as a system service will use to run on.

The installer will do its best to provide a valid default value for Service Name (see Registering Server as a Windows service or Unix Process for details on this process). If the installer is unable to determine a valid Service Name, the field will be blank. You will not be able to move forward until you enter a valid service name.

Note: If you are installing on an UNIX system, ensure that the Service Name does not exceed the maximum length of 4 characters. If it exceeds 4 characters, this limitation results in an error and you cannot proceed with the installation.



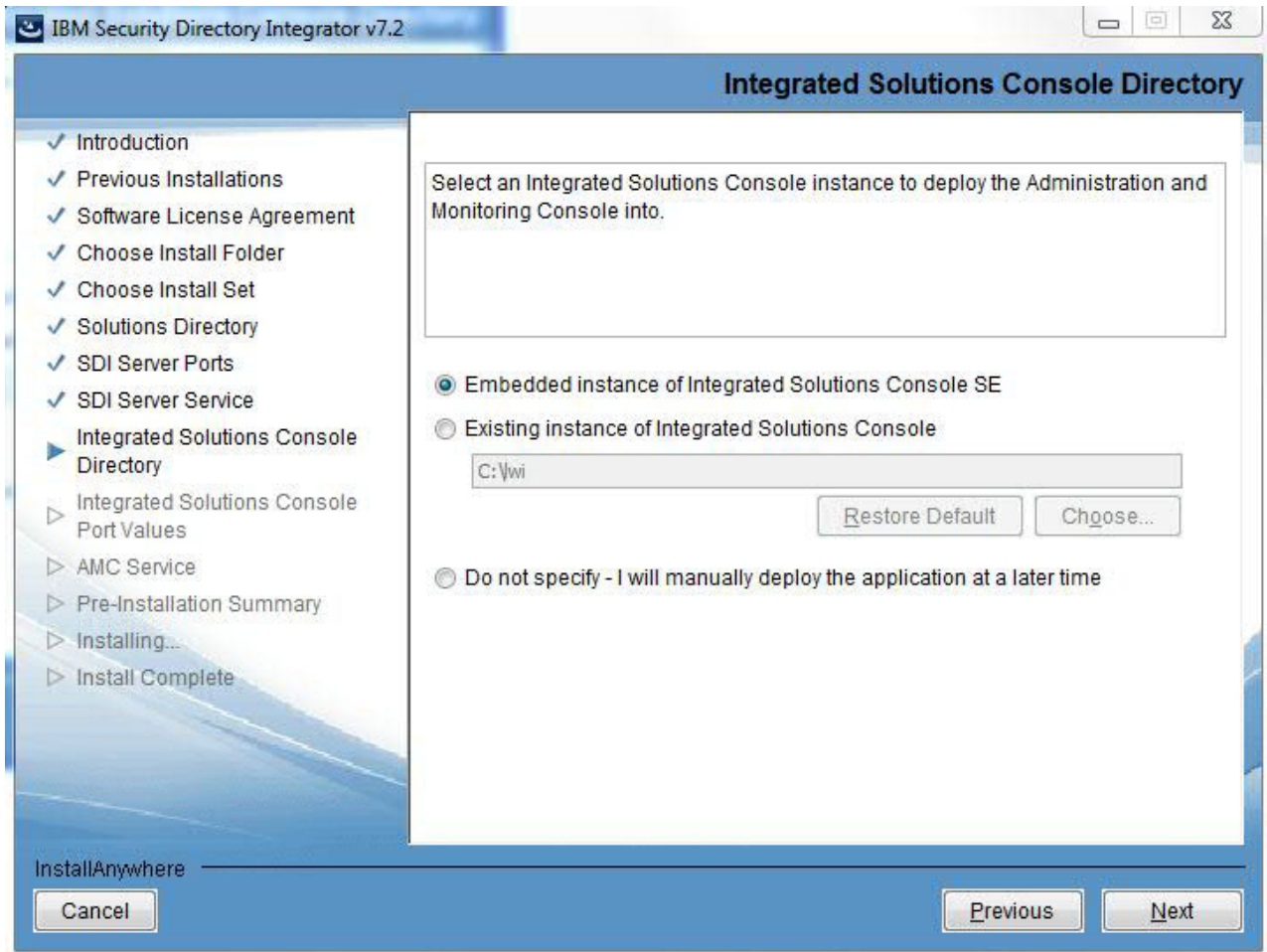
IBM Security Directory Integrator AMC Deployment Panel

This panel is only displayed if the Custom install set was chosen and you also chose to install the AMC feature. You must choose which ISC instance AMC will be deployed to. You may choose to deploy AMC to the bundled ISC that is shipped with IBM Security Directory Integrator, an ISC that is already installed on the target machine, or choose to deploy AMC at a later time. When choosing an ISC that is already installed, the user must select a directory that contains the embedded Web platform (LWI) or IBM WebSphere Application Server, for example C:\Program Files\IBM\WebSphere\AppServer or C:\dev\IBM\TDI\lwi.

If you did not choose to install the embedded web platform feature, then that choice will be grayed out.

Note:

1. If you are adding features and the AMC feature is already installed, this panel will be skipped.
2. When deploying AMC to IBM WebSphere Application Server, the Security Directory Integrator AMC Admin role is not assigned automatically as when deploying to the embedded Web platform. This role must be manually assigned by the ISC console administrator.



ISC Port panel

This panel is shown either during a typical install or custom install, when you choose to deploy AMC to an Embedded instance of ISC. The ISC instance could be the embedded ISC that is shipped with IBM Security Directory Integrator, or it can be an ISC that is already resident on the target system.

If you are deploying AMC to a custom SE, the default values that are used for the HTTP and HTTPS ports are found as follows:

Look in the *TDI_Selected_ISC/conf/overrides/*.properties* files for the first occurrence of the properties *com.ibm.pvc.webcontainer.port* and *com.ibm.pvc.webcontainer.port.secure* and use the associated values. If either of these properties is not defined in any of the *.properties* files in that directory, look in *TDI_Selected_ISC/conf/config.properties* for them. If the HTTP port is not found, it will default to port 80, and if the HTTPS port is not found, it will default to port 443. The help port will have the same value as the HTTP port.

If you are deploying AMC to a custom AE, the default values that are used for the HTTP and HTTPS ports are found as follows:

Look for files named *serverindex.xml* file in the following directory specification:

TDI_Selected_ISC\profiles\AppSrv01\config\cells\nodes**.

Inside those files, look for XML blocks similar to the following for the HTTP port:

```
<specialEndpoints xmi:id="NamedEndPoint_1200476459036"  
  endPointName="WC_adminhost">  
  <endPoint xmi:id="EndPoint_1200476459036" host="*" port="9060"/>  
</specialEndpoints>
```

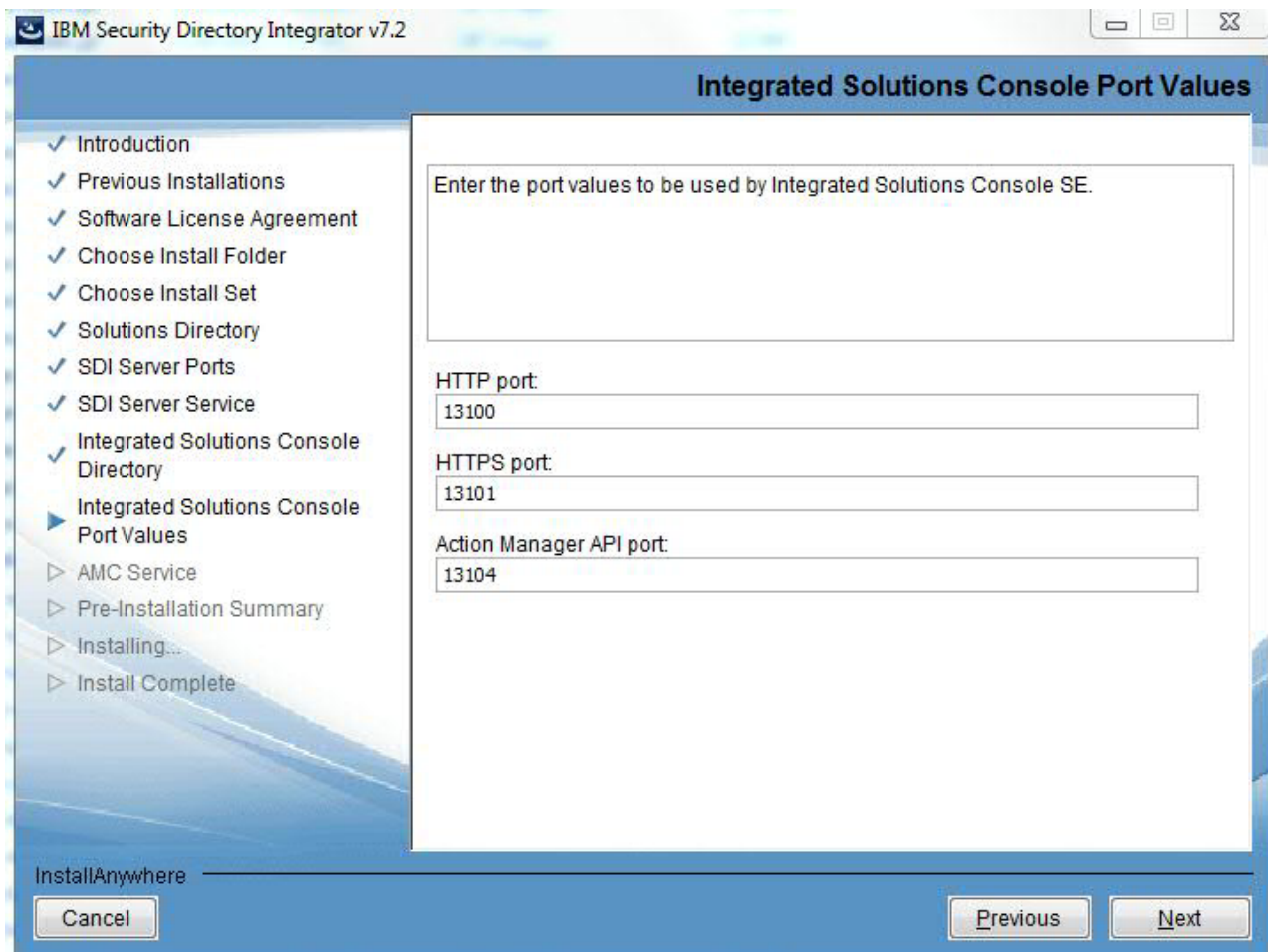
and similar to the following for the HTTPS port:

```
<specialEndpoints xmi:id="NamedEndPoint_1200476459039"  
  endPointName="WC_adminhost_secure">  
  <endPoint xmi:id="EndPoint_1200476459039" host="*" port="9043"/>  
</specialEndpoints>
```

The installer searches for a specialEndpoints tag that has an endPointName of **WC_adminhost** or **WC_adminhost_secure** and use the associated port values from the embedded endPoint tags. In the event the HTTP port is not found by this method, it 9060 and in the event the HTTPS port is not found, it will default to 9043. The help port will be set to the HTTP port value.

The values shown are the defaults for the embedded SE.

The panel will not allow ports to be entered that are already in use. A warning message will appear asking you to choose another port value.

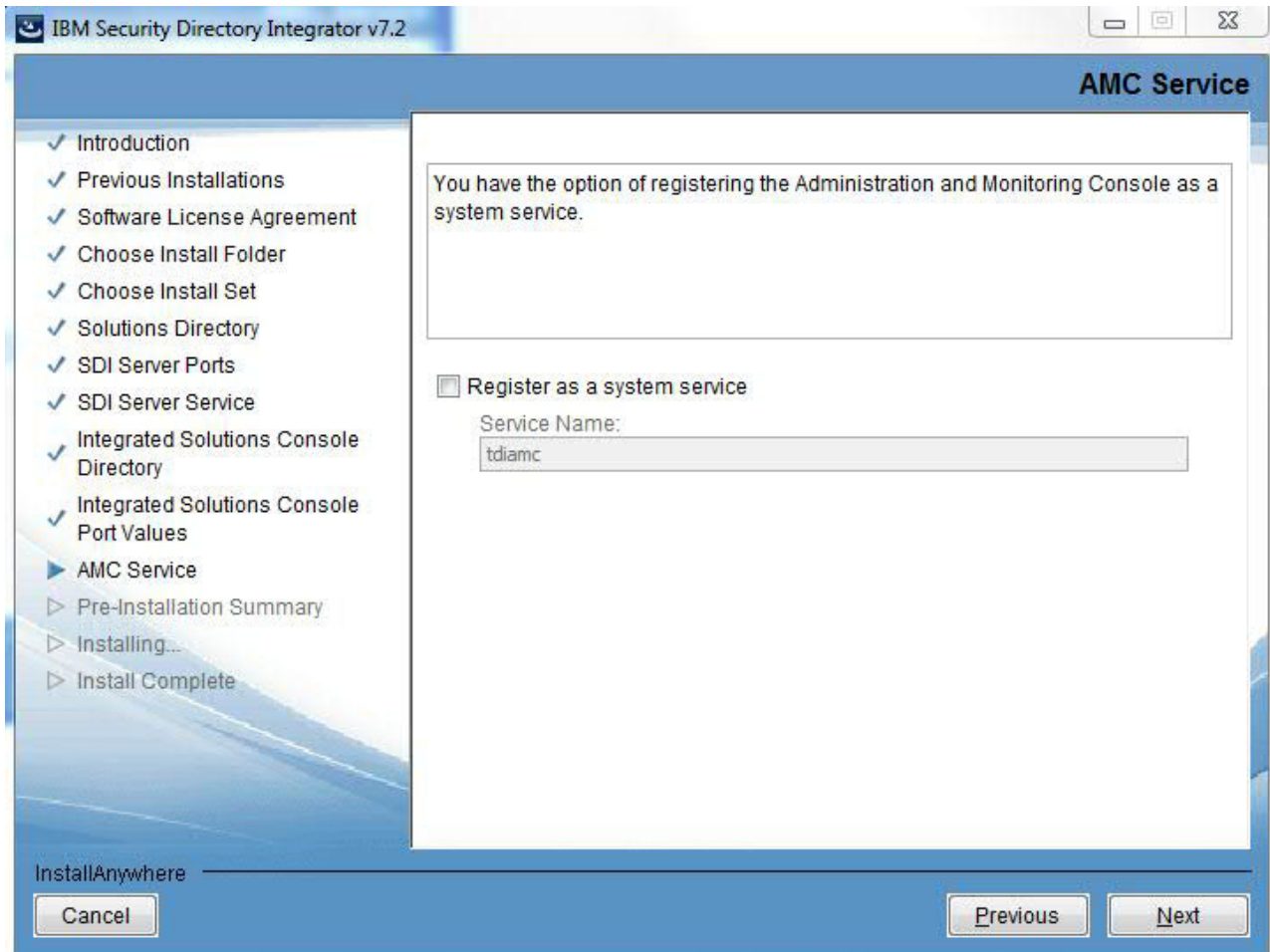


Register AMC as a service Panel

If the checkbox is checked, then AMC will be registered as a service for that OS.

The default is for the checkbox to be unchecked.

This panel is only shown if the embedded web platform and AMC features were selected and if you have administrative privileges.



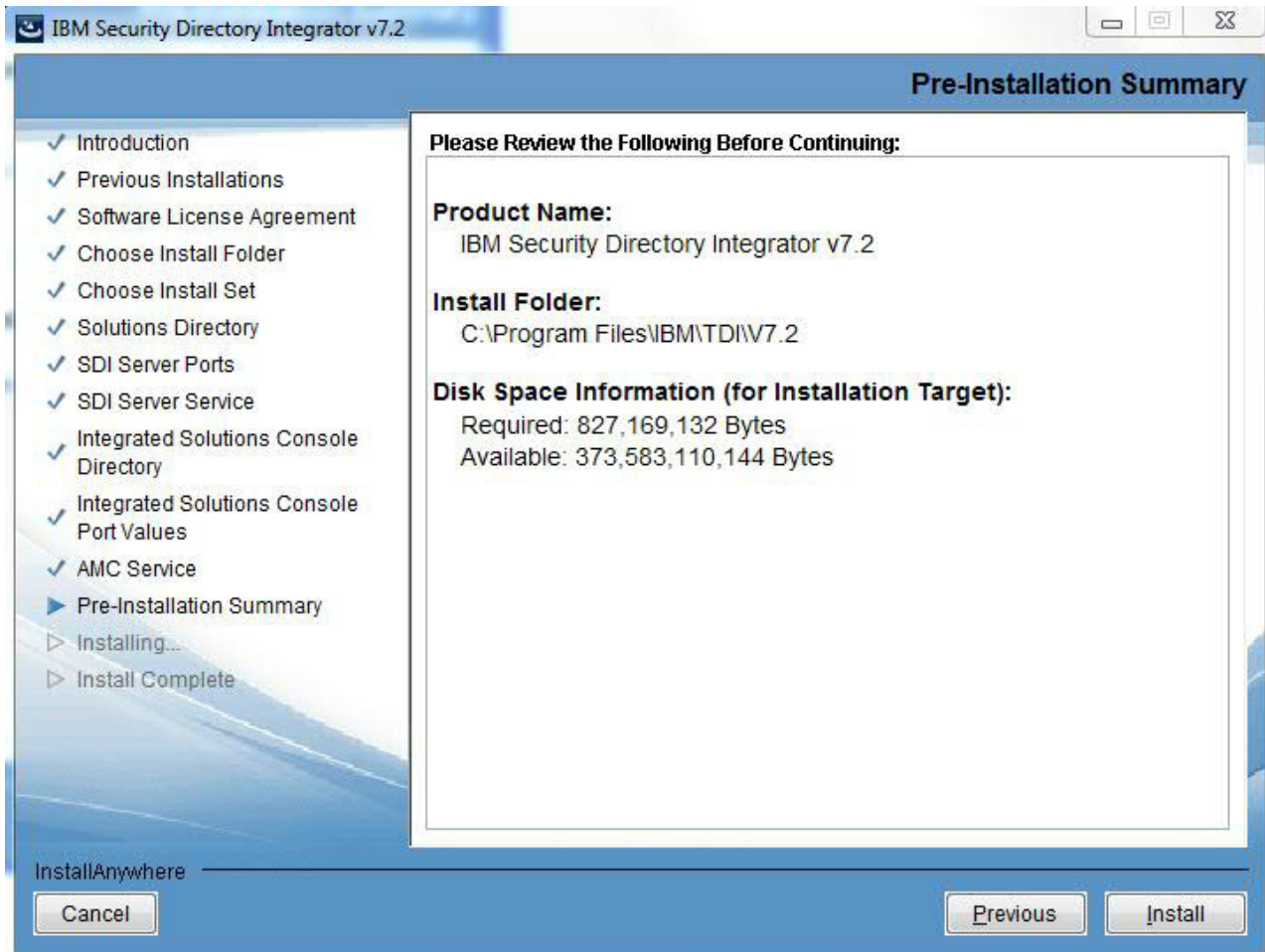
The installer will do its best to provide a valid default value for Service Name (see Registering AMC as a Windows service or Unix process for details on this process). If the installer is unable to determine a valid Service Name, the field will be blank. You cannot move forward until you enter a valid service name.

Note: If you are installing on an UNIX system, ensure that the Service Name does not exceed the maximum length of 4 characters. If it exceeds 4 characters, this limitation results in an error and you cannot proceed with the installation.

The AMC feature is deprecated and will be removed in a future version of IBM Security Directory Integrator.

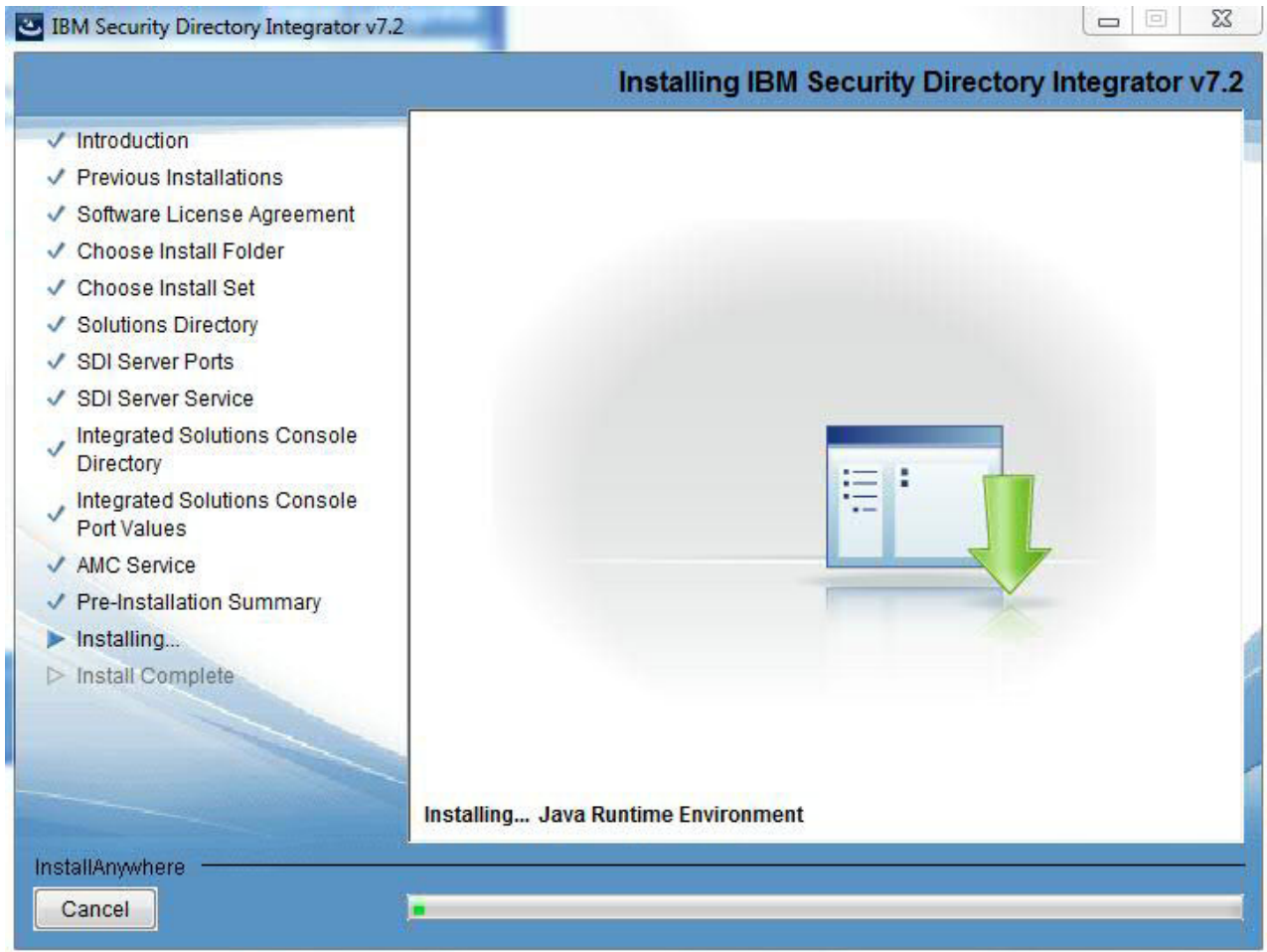
Pre-Install Summary Panel

This Summary panel gives you a summary of what features will be installed and where they will be installed to.



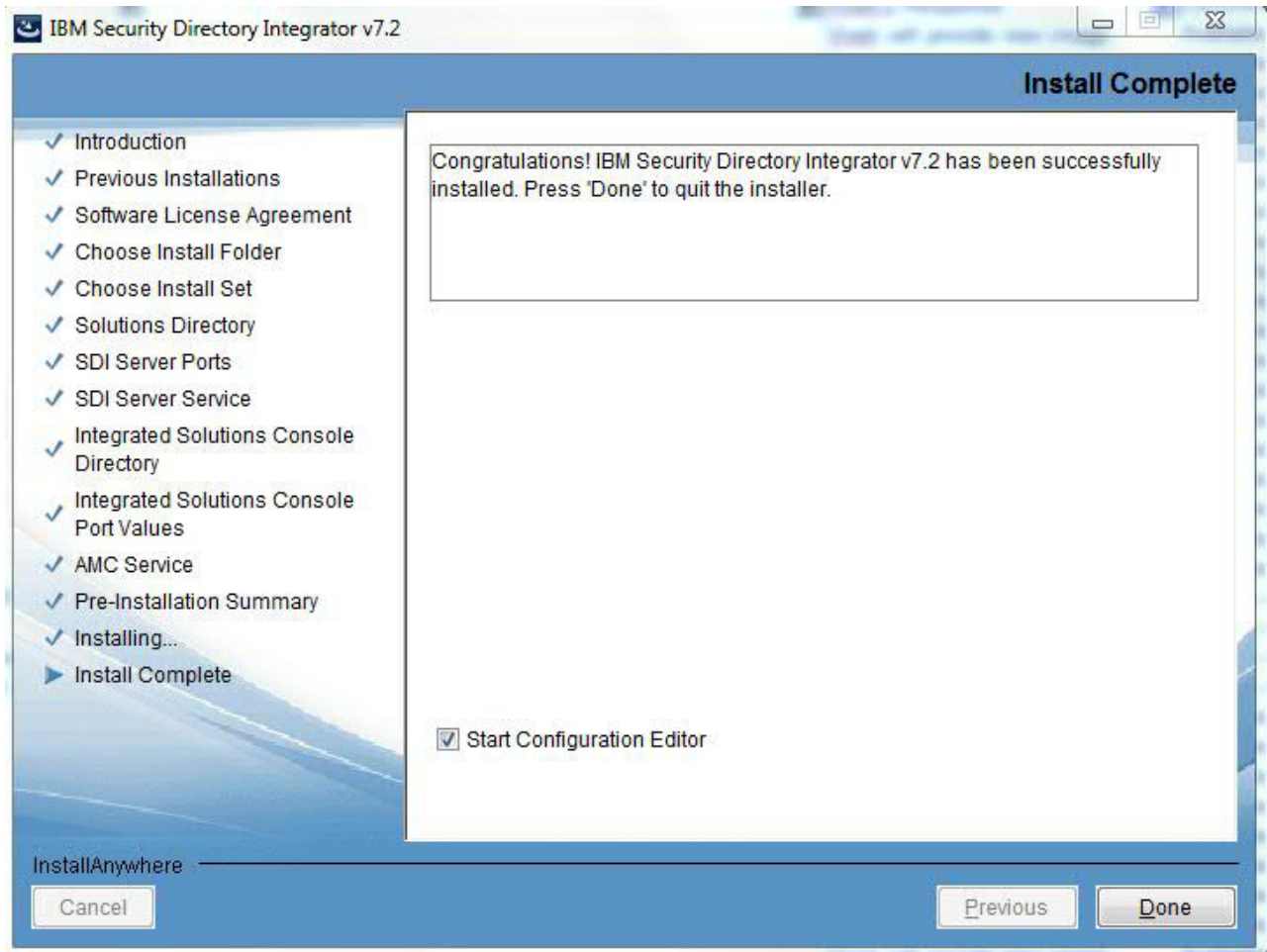
Installation Progress Panel

This panel is displayed while the actual install is occurring. This panel is the Progress Panel provided by InstallAnywhere. All of the features are installed while this is occurring.



Installation Complete Panel

This panel shows you that the install has completed successfully. When the Done button is pressed, the install is complete. **Start the Configuration Editor** is checked by default.

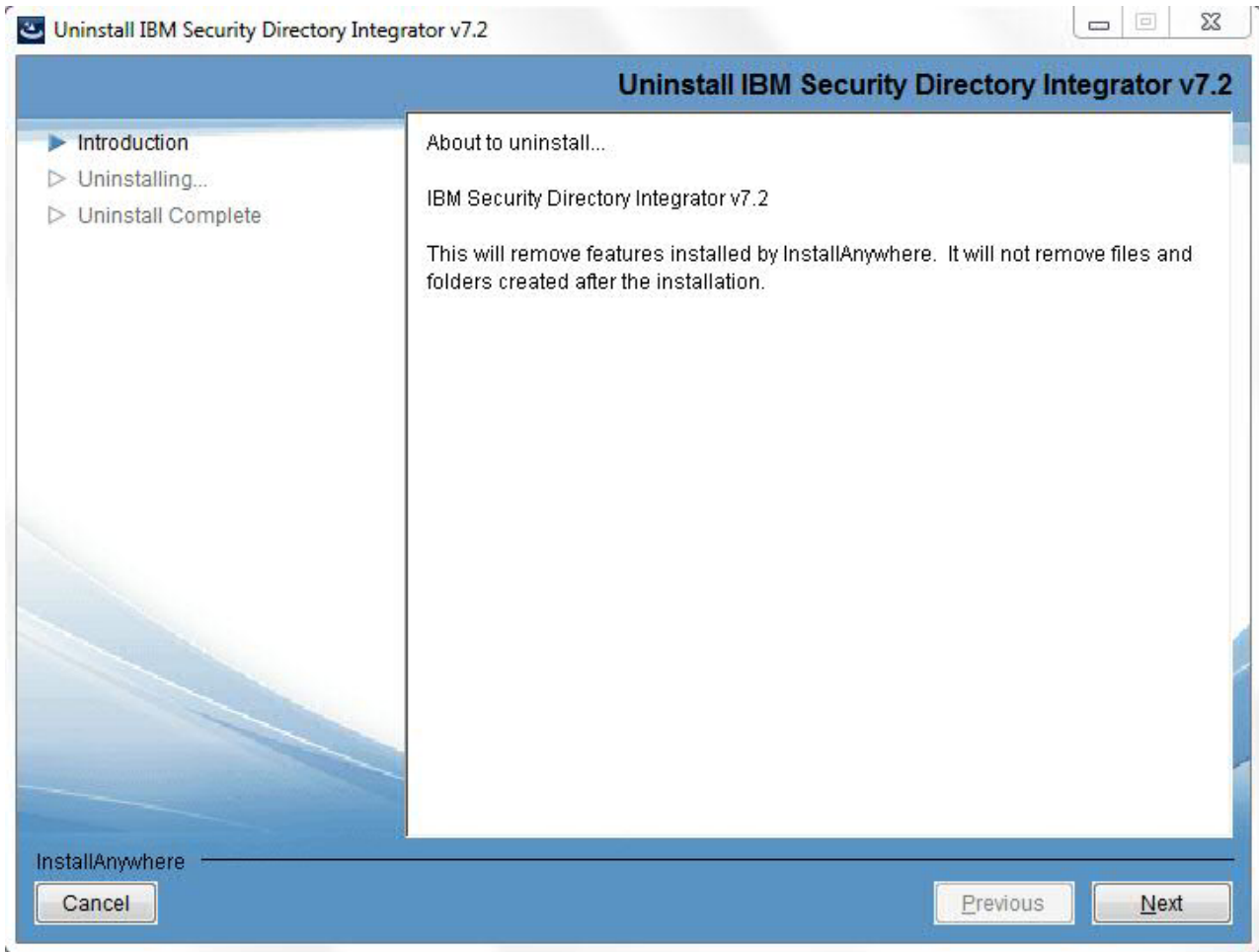


Uninstall Panel flow

Use the instructions provided here to uninstall the panel flow.

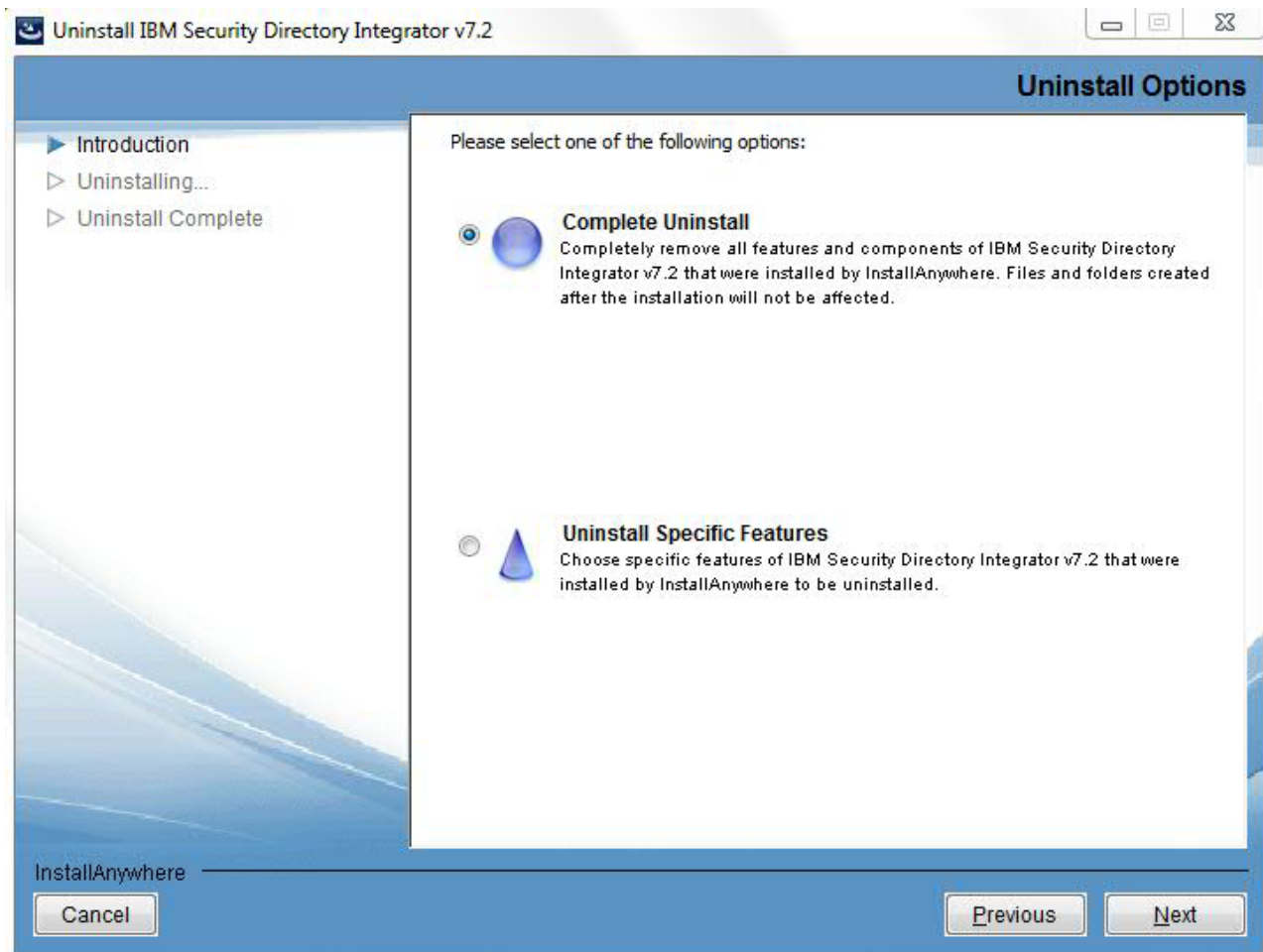
Uninstall Welcome Panel

This is an InstallAnywhere panel, with standard content.

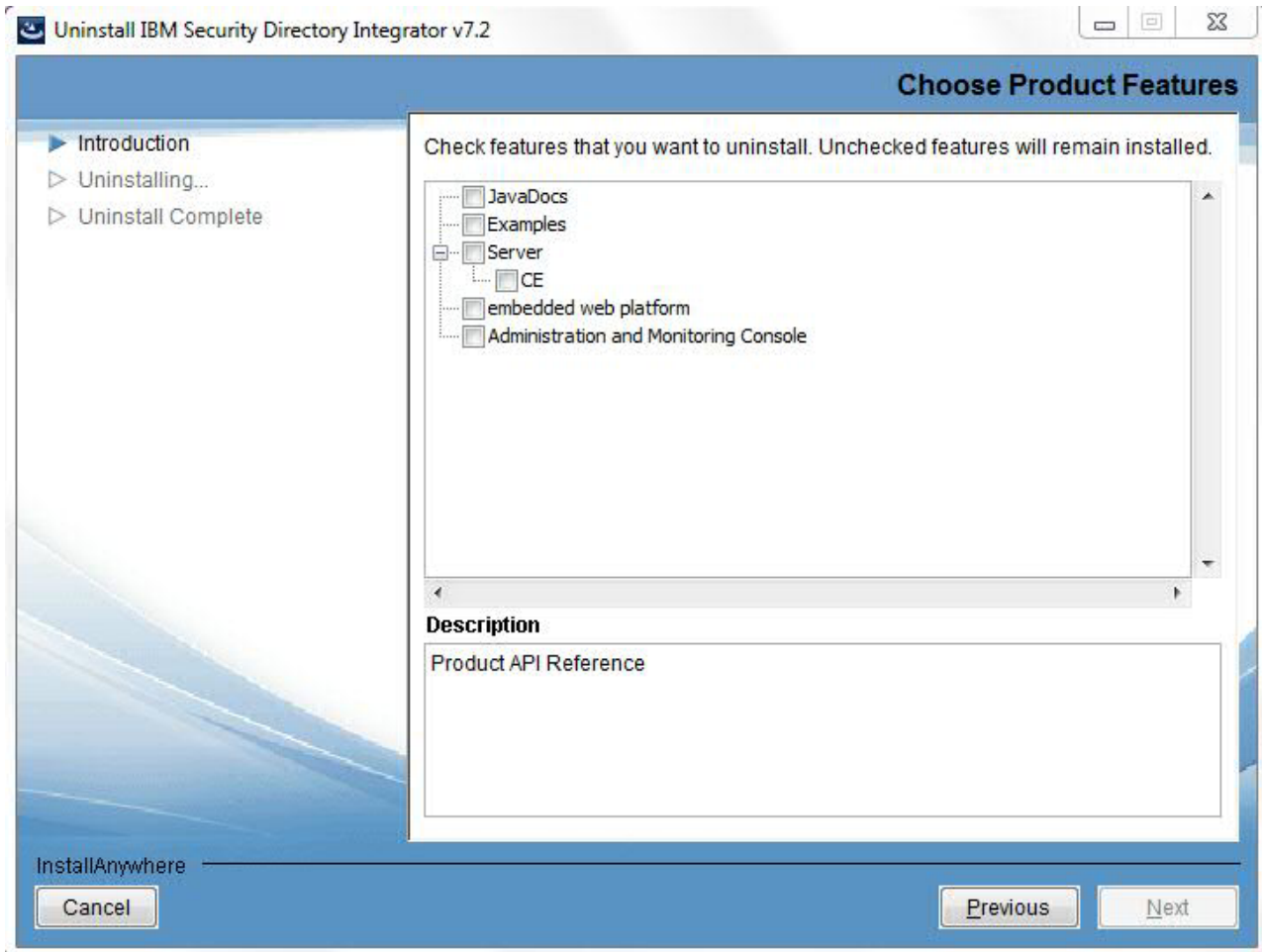


Choose Product Features Panel

This panel allows you to choose to uninstall the entire product, or only specific features.



If **Uninstall Specific Features** is chosen, the following panel is also displayed:



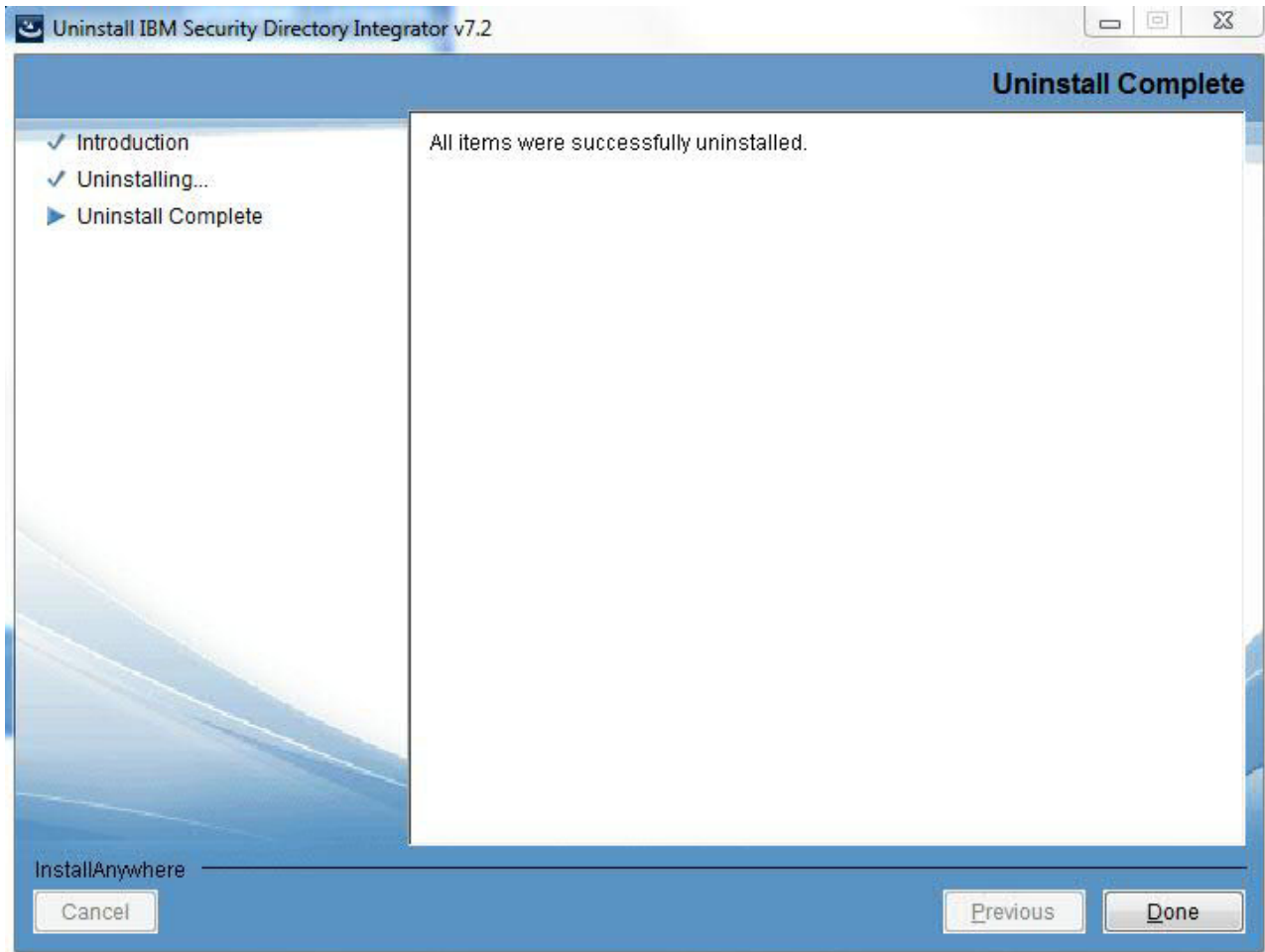
Uninstall Progress Panel

This panel is shown during uninstallation.



Uninstall Finish Panel

This panel shows you that the uninstallation has completed successfully. When the Done button is pressed, the uninstaller exits.



Add Feature Panel flow

You can know more about the add feature flow with information provided here.

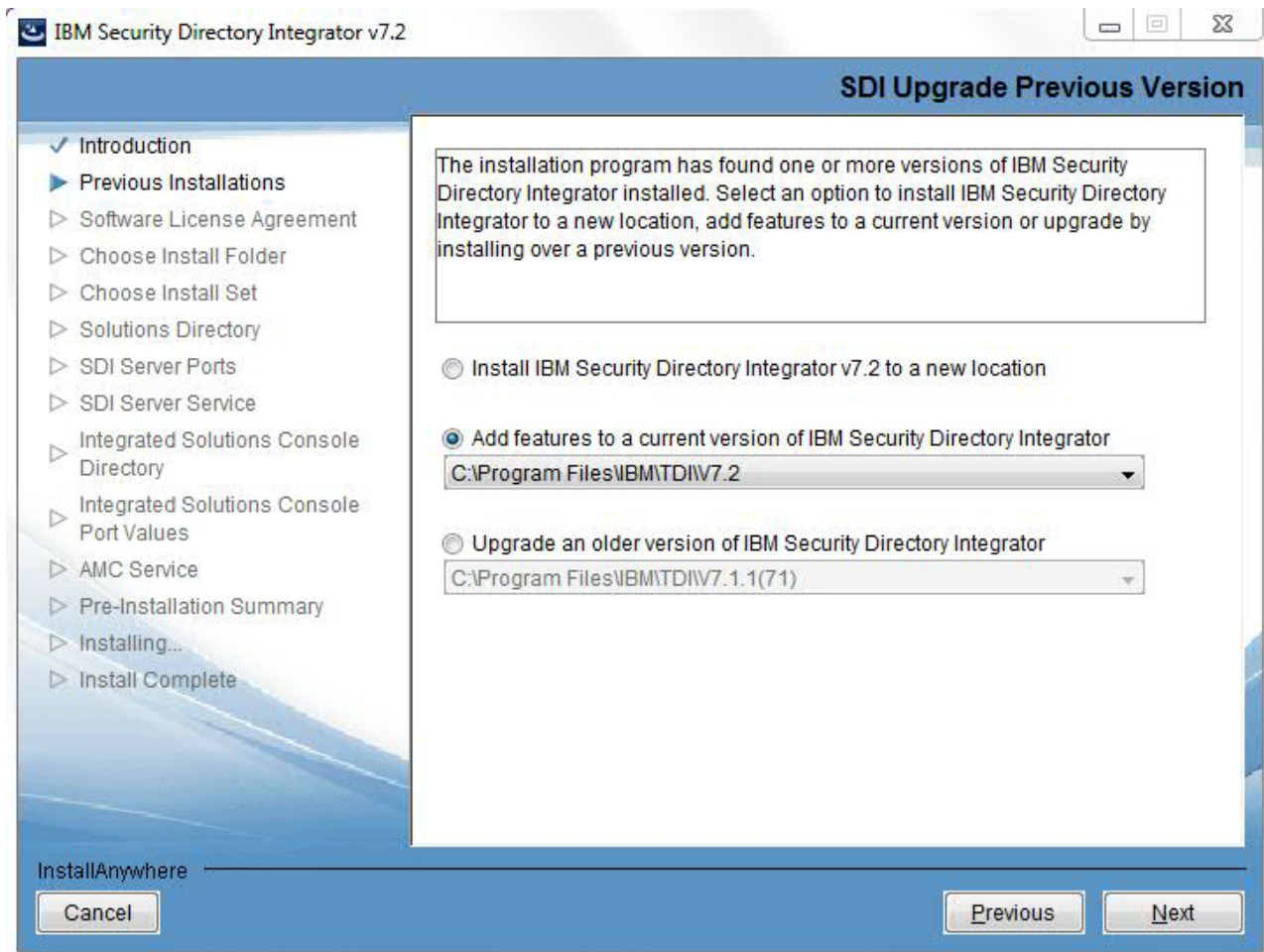
The Add Feature flow is similar to the new install flow. Only the unique panels will be shown here.

Pre-Initialization Panel

The Welcome Panel

Upgrade Panel

After the Welcome panel and the Previous IBM Security Directory Integrator information panel, if there is an instance of IBM Security Directory Integrator already installed on the box, you will see this panel.



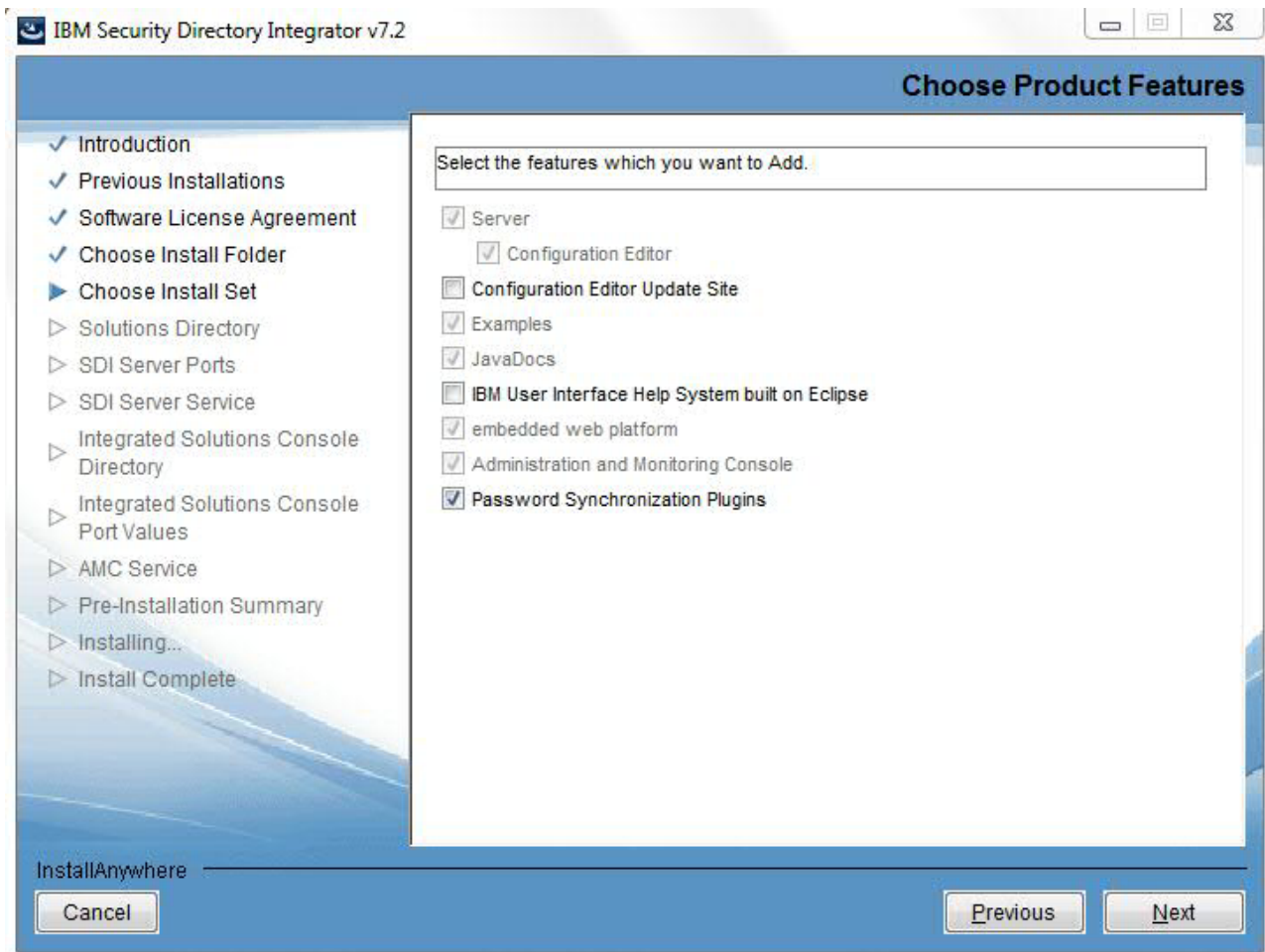
You are not able to choose the **Add Feature** button if there is no IBM Security Directory Integrator Version 7.2 instance available.

You are not able to choose the **Upgrade** button if there are no previous versions of IBM Security Directory Integrator available.

The IBM Security Directory Integrator drop down is enabled if the **Add Features** button is chosen.

Feature Selection Panel

The next panel in the Add Feature sequence will be the Feature Selection panel, with the already installed features selected and grayed out.



At this point you can add any additional features you choose.

You are not allowed to remove features.

From this point the panel flow matches the new install flow. Panels related to already-installed features will, however, be skipped.

If you select Configuration Editor, Server will also automatically be selected. Also if both features are selected and you deselect Server, then Configuration Editor will also be deselected.

Security Directory Integrator Solutions Directory Panel

Register Server as a System Service Panel

Security Directory Integrator AMC Deployment Panel

Register AMC as a service Panel

Pre-Install Summary Panel

Installation Progress Panel

Installation Complete Panel

Migration Panel flow

You can know more about the migration panel flow with information provided here.

The Migration flow is similar to the new install flow. Only the unique panels will be shown here.

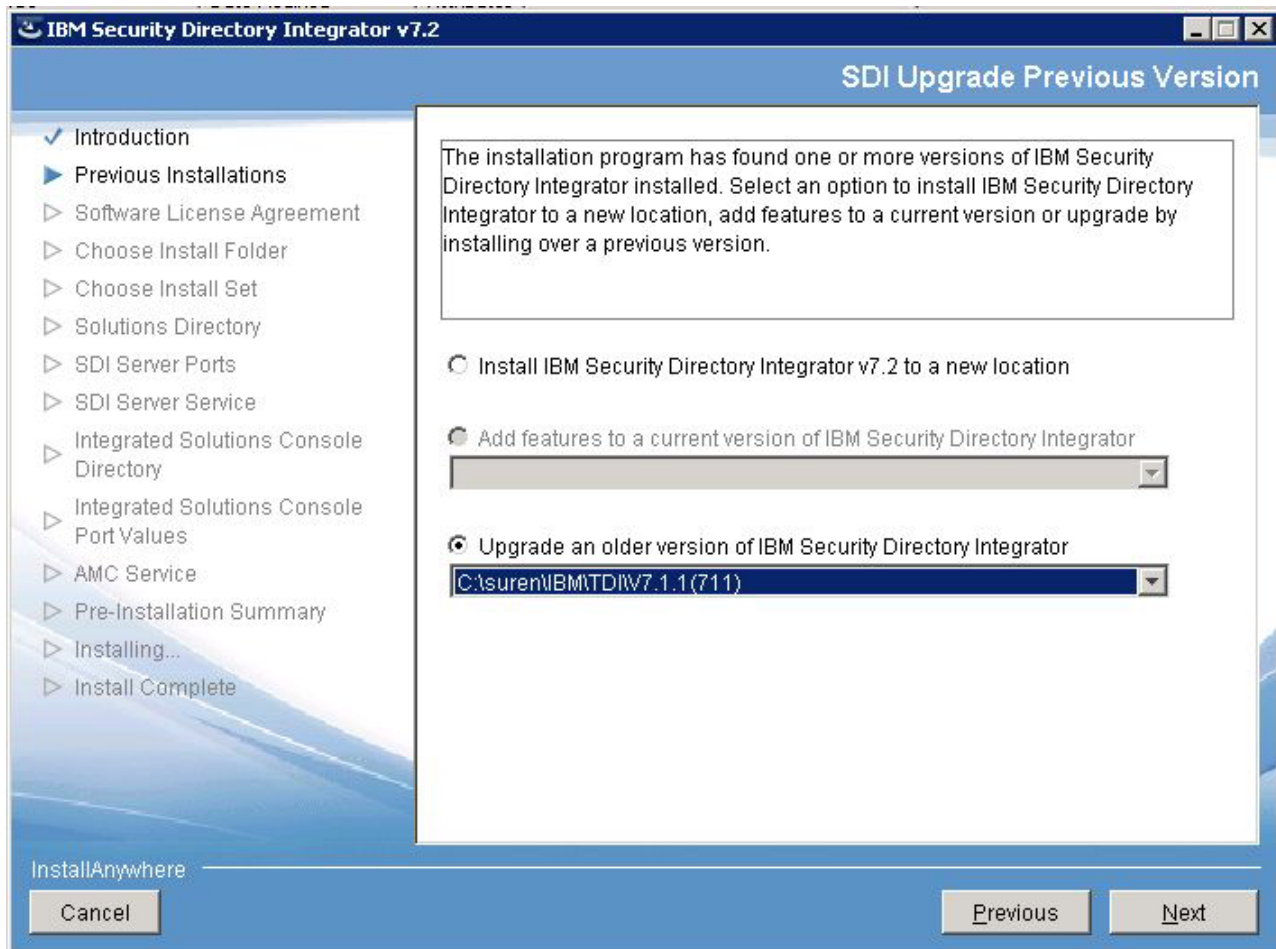
Pre-Initialization Panel

The Welcome Panel

Upgrade Panel

Note: If you are planning to upgrade IBM Security Directory Integrator Version 7.1.1 to Version 7.2 on Windows 2012 operating system, then ensure that **Windows 7 compatibility mode** is enabled on the Version 7.1.1 uninstaller.exe before you start the Version 7.2 installer. For more information, see the technical note at <http://www-01.ibm.com/support/docview.wss?uid=swg21634336>.

After the Welcome panel and the Previous SDI information panel, if there is an instance of IBM Security Directory Integrator already installed on the box, you will see this panel:



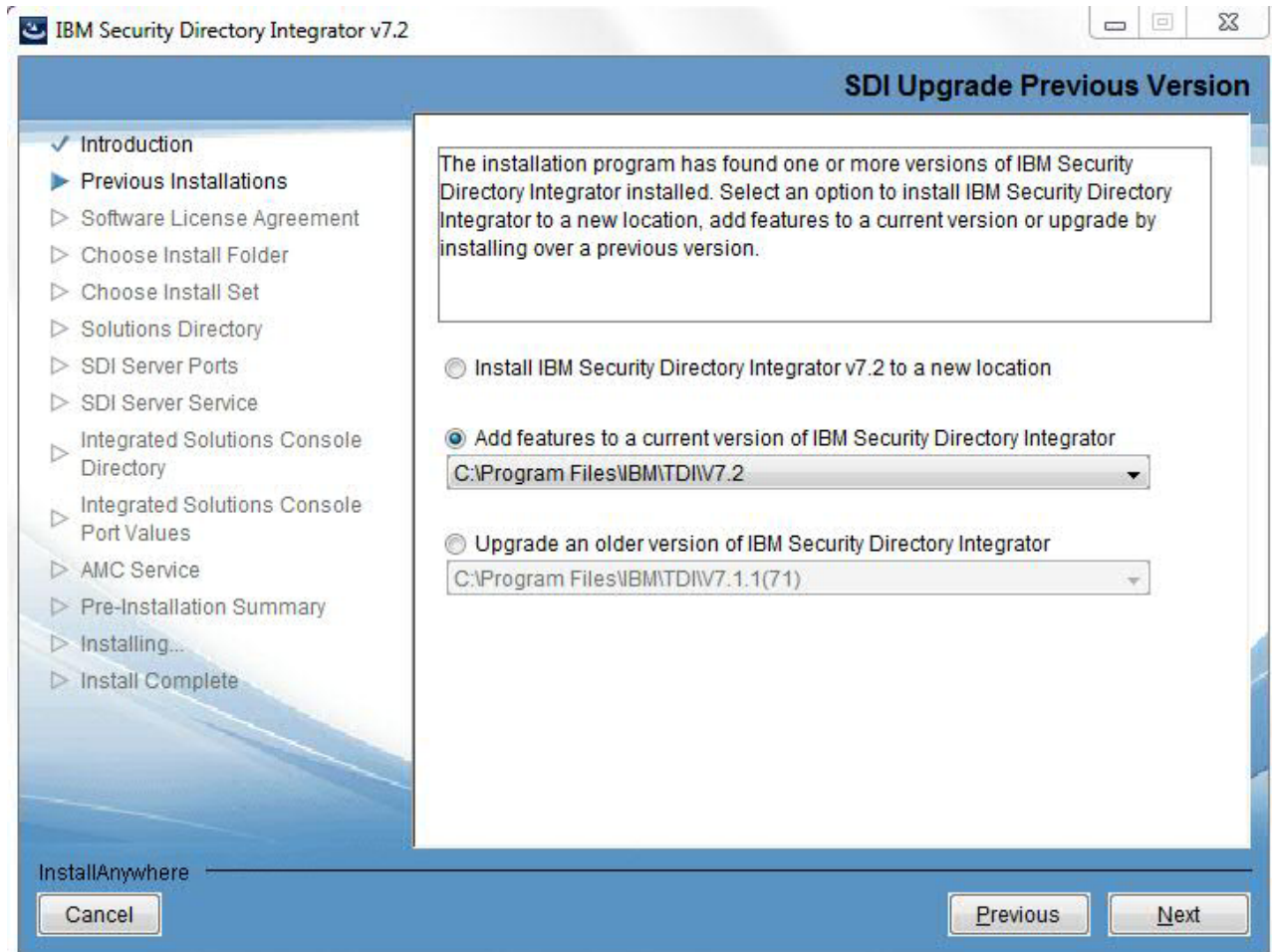
You are not able to choose the Add Feature button if there is no IBM Security Directory Integrator Version 7.2 instance available.

You are not able to choose the Upgrade button if there are no previous versions of IBM Security Directory Integrator available.

The previous IBM Security Directory Integrator version drop down is enabled if the Upgrade button is chosen.

IBM Security Directory Integrator 6.1.x, 7.0, 7.1.x Upgrade Flow

Note: Upgrading IBM Security Directory Integrator from versions 6.x or earlier, directly to version 7.2, is not supported. You must first upgrade from version 6.x to version 7.1.1 and then from version 7.1.1 to version 7.2. If you choose to upgrade from IBM Security Directory Integrator version 7.1.1, the next panel you will see will be the License panel and after accepting the license, the Feature selection panel:



It will show those features (selected) that were already installed in the previous version, and allow you to add new ones. You are not allowed to disable features that were previously installed.

If the Server was previously installed, the SDI Solutions Directory panel will be skipped. The installer will use the value from the previous installation.

If AMC was previously installed, you will still see the Choose ISC panel, in an IBM Security Directory Integrator 6.x.x migration.

The rest of the panel flow is the same as a new install. In an IBM Security Directory Integrator 7.0 migration, there is one new feature: **Register server as system service**, this panel is only shown after the feature selection panel.

If you select Configuration Editor, Server will also automatically be selected. Also if both features are selected and you deselect Server, then Configuration Editor will also be deselected.

Installing using the command line

Here is the list of commands you can use while doing the installation.

The following command line options are supported by the IBM Security Directory Integrator installer:

-i Sets the installer interface mode: silent, console or gui.

```
install_sdiv72_win_x86.exe -f Response File Name -i silent
```

```
install_sdiv72_win_x86.exe -i console
```

-f Sets the location of a response file (installer.properties file) for the installer to use.

```
install_sdiv72_win_x86.exe -f installer.properties
```

This path can be absolute or relative. (Relative paths are relative to the location of the installer.)

-r Creates a response file.

```
install_sdiv72_win_x86.exe -r myinstaller.properties
```

Note: The IBM Security Directory Integrator installer creates the `tdi_respfile72.txt` response file in the system's temporary files directory, even if `-r` option is not specified. For example:

- On Windows platform, the response file is created in the `C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp` directory.
- On non-Windows platform, the response file is created in the `/tmp` directory.

The `TDI_install_dir/examples/install` directory contains example response files for various installation and uninstallation scenarios.

-D Passes custom command-line arguments.

```
install_sdiv72_win_x86.exe -Dmyvar=myvalue
```

-l Uses the specified language code (and optional country code) to set the locale for the InstallAnywhere installer.

```
install_sdiv72_win_x86.exe -l en
```

```
install_sdiv72_win_x86.exe -l pt_BR
```

The required language code is a two-character (commonly lowercase) code defined by the ISO-639 standard. InstallAnywhere accepts both old (`iw`, `ji`, and `in`) and new (`he`, `yi`, and `id`) language codes.

The optional country code is a two-character (commonly uppercase) code defined by the ISO-3166 standard.

Locale options are only respected if the installer includes localizations for the locale you specify.

-? Shows help for the InstallAnywhere installer.

On Windows, `-help` only works from the console launcher. Make sure to set the LaunchAnywhere to **Console** on the Windows tab of the **Project >**

Platforms subtask. (For an installed LaunchAnywhere to provide this information, you need to make sure it is explicitly set to Console Launcher on the action.)

The following command line option is unique to the IBM Security Directory Integrator installation Wizard:

LAX_VM

The LAX_VM parameter is used to boot the installer from Java virtual machine, which is installed on the system.

You need to specify absolute path of the Java executable file that resides in the Java bin directory. For example,

```
install_sdiv72_win_x86.exe LAX_VM "Java_DIR/jre/bin/java.exe"
```

Use only the space characters between the arguments.

Note: Make sure that you use the absolute path of IBM JRE 7.0.4 and above, as parameter value. The IBM Security Directory Integrator installer may not work correctly with other JREs.

-D\$TDI_BACKUP\$="true"

This parameter should only be passed in on an uninstallation. This parameter is provided for future migration considerations; for example:

```
TDI_install_dir\uninst\uninstaller.exe -D$TDI_BACKUP$="true"
```

This instructs the uninstaller to run the *TDI_install_dir/bin/tdiBackup.bat* (.sh) script, which in turn will cause a directory *TDI_install_dir/backup_tdi* to be created. A backup of a number of files particular to your installation will be stored into this directory, including your global properties files, global certificates and the like.

Note: On a non-Windows system the \$ (dollar) must be escaped with a \ (backslash). For example:

```
TDI_install_dir\uninst\uninstaller -D\$TDI_BACKUP\$="true"
```

-D\$TDI_SKIP_VERSION_CHECK\$="true"

This parameter will cause the installer to skip any previous version checks. This essentially disables any migration from previous releases.

In a silent install, if this skip option is chosen and the install directory is same as an earlier installation of IBM Security Directory Integrator, it will cause the installer to stop.

Note: On a non-Windows system the \$ (dollar) sign must be escaped with a \ (backslash). For example:

```
./install_sdiv72_linux_x86_64.bin -D$TDI_SKIP_VERSION_CHECK\$="true"
```

-D\$TDI_NOSHORTCUTS\$="true"

This parameter is used to stop the installer from creating any shortcuts to the uninstaller, CE, or AMC.

Note: On a non-Windows system the \$ (dollar) must be escaped with a \ (backslash). For example:

```
./install_sdiv72_linux_x86_64.bin -D$TDI_NOSHORTCUTS\$="true"
```

Temporary file space usage during installation

Use the instructions provided here to make the best use of temporary file storage.

During installation, the installer may use a substantial amount of temporary file space in order to stage files. If your system is constrained in this regard, errors during installation might occur.

UNIX/Linux systems typically use `/tmp` or `/var/tmp` as temporary files storage, whereas on Windows, the temporary file storage area is found in the location pointed to by the environment variable `TEMP`.

InstallAnywhere installers can be instructed to redirect their temporary file usage by setting the environment variable `IATEMPDIR` before starting the installer. For example, on UNIX:

```
export IATEMPDIR=/opt/IBM/TDI/temp
```

Then, start your console mode installers from the session in which you have set the `IATEMPDIR` variable.

Performing a silent install

You can perform a silent installation with the help of instructions provided here.

To perform a silent installation you must first generate a response file. To generate this file, perform a non-silent installation with the `-r` option specified, for example:

```
install_sdiv72_win_x86.exe -r Response File Name
```

The response file is created in the directory that you specify during installation.

Note: The directory `TDI_install_dir/examples/install` contains a number of example response files for various installation and uninstallation scenarios. Once the response file is created, you can install silently using the following command:

```
install_sdiv72_win_x86.exe -i silent -f Response File Name
```

Note: The examples in this document use the Windows platform installation executable file. See “Launching the appropriate installer” on page 9 for a list of executable file names for each supported platform.

Service name limitation on UNIX systems

Consider the limitations while naming a service on UNIX system.

During silent installation of IBM Security Directory Integrator on UNIX systems, ensure that the service name for IBM Security Directory Integrator and AMC does not exceed the maximum length of 4 characters. This value is specified in the response file, `examples\install\TDICustomInstallRsp_Unix.txt`.

The silent installation fails if you specify a name that is longer than 4 characters, for example:

```
TDI_SERVER_SERVICENAME=tdisrv_silent  
TDI_AMC_SERVICENAME=tdiamc_silent
```

The log files, `/tmp/sdiv72install.log` and `/tmp/sdiv72debug.log`, report the following error:

tdisrv_silent must be 4 characters or less.
A service already exists with that name or the name is invalid.
UNIX based platforms, maximum 4 characters name is valid.

If this error occurs, you must edit the response file and change the values of the TDI_SERVER_SERVICENAME and TDI_AMC_SERVICENAME properties to be 4 characters or less.

Post-installation steps

Perform the steps listed here after the installation is done.

CE Update Site

Use the information provided here to manually deploy the Eclipse.

If the CE Update site was installed, you now have to manually deploy into Eclipse. See the section entitled “Installing or Updating using the Eclipse Update Manager” on page 48 for more information.

Plug-ins

Refer to the information provided here to access the documentation about password synchronization plug-ins.

If any of the password synchronization plug-ins were installed, see the *Password Synchronization Plug-ins* section of the IBM Knowledge Center for IBM Security Directory Integrator for information on how to deploy the plug-in code.

Administration and Monitoring Console (AMC)

You can know more about the general information, information about web platform deployment and deferred deployment of Administration and Monitoring Console.

General information

- For more information on AMC, see “Administration and Monitoring Console (AMC).”
- When you are ready to log into the console, browse to <http://hostname:port/ibm/console>. For more information, see section “Log in and logout of the console” on page 255.
- For more information on adding users and user roles, see the section “Console user authority” in “AMC in the Integrated Solutions Console” on page 242.

Bundled embedded web platform deployment

- If you installed AMC with the bundled embedded web platform and are ready to use AMC, you will need to run the commands to start AMC and Action Manager (AM) before you can log into the ISC console. For more information, see section “Starting and logging in the AMC and Action Manager” on page 238.

Note: On Windows, a shortcut to the launchAMC.html file is created in the start menu under Program Files.

- By default, the user who installs IBM Security Directory Integrator is the only one with access to log in to the console.

Customer or deferred deployment

- If you chose a custom to deploy AMC to, and are now ready to deploy, see “Deploying AMC to a custom ISC SE or IBM Dashboard Application Services Hub” on page 47. When deploying AMC in such a way the installer does not automatically assign the current user the SDI AMC Admin role. This right needs to be manually authorized by an administrator of the ISC console. This is typically done using the **Users and Groups -> Administrative User Roles** panel of the IBM Dashboard Application Services Hub console. Alternatively this role could be assigned using the `setAMCRoles` command.
- If you chose to defer deployment of AMC into an ISC and are ready to do so, see section “Deploying AMC to a custom ISC SE or IBM Dashboard Application Services Hub” on page 47.

Note: If you have done a custom ISC SE/AE deployment then at a minimum you will need to ensure that AM is started after you start the ISC SE/AE that AMC was installed into.

Documentation

You can access the documentation online or choose to deploy it manually. Read more about it in the information provided here.

The documentation system used by IBM Security Directory Integrator is the IBM Knowledge Center. After you have done a default installation, this means that IBM Security Directory Integrator documentation is made available to you online, on the Web, hosted by IBM. You may, however, choose to deploy the documentation locally. For more information, see “Installing local Help files.”

If you are new to IBM Security Directory Integrator, we recommend that you read and step through the *Getting Started* in order to get used to the concepts used.

If you have used earlier versions of IBM Security Directory Integrator, then section 3 of the *Configuring Directory Integrator* will be very beneficial to you in order to understand the new IDE framework and layout. It will also explain how you can import and open your existing configurations; and how the Server still uses the Config model at runtime.

Migration

Read more about migration through the link provided here.

If you have installed an earlier version of IBM Security Directory Integrator, then you will most likely need to migrate certain aspects of your previous deployment. More information on what to do in this case can be found under Chapter 5, “Migrating,” on page 61.

Installing local Help files

Use the instructions provided here to install the documentation locally.

Note: This feature is deprecated and will be removed in a future version of IBM Security Directory Integrator.

The IBM Security Directory Integrator installer does not contain any user documentation, other than the Java API documentation, which can be displayed by selecting the **Help -> Welcome** screen, **JavaDocs** link in the Configuration Editor. IBM provides the user documentation in online form in the IBM Knowledge Center for IBM Security Directory Integrator.

IBM Security Directory Integrator is equipped with code¹ to provide you with context-dependent online help that you can launch from the Configuration Editor (CE). By default, this code resolves the documentation from the online product documentation as referenced above. You can, however, install the documentation locally, such that you are not dependent upon the Internet to be able to read it.

These are the steps you must take to install documentation locally:

- The code to handle the documentation files, the IBM User Interface Help System built on Eclipse, is not installed by default. In order to install the Help system, you will need to do a custom install, and install the Help system feature into your existing IBM Security Directory Integrator installation.
- All the manuals are stored together in one compressed directory, which when uncompressed contains an Eclipse *Document plug-in*.
- All the manuals can be downloaded in their compressed form from the IBM Knowledge Center for IBM Security Directory Integrator. On the welcome page for the current release, click the *Information center plug-ins* link under the column *More information*.
- The entire documentation package, *di_plug-ins-7.2.0.1.zip*, should be uncompressed into the right place: *TDI_install_dir/ibm_help/eclipse/plugins* folder (or uncompress somewhere else, and move into the right place). The package contains the actual IBM Security Directory Integrator documentation in *com.ibm.IBMDI.doc_7.2.0.1*, alongside a number of other directories whose names end in *.doc*; all of those directories should be at the same aforementioned *plugins* level.
- The location of the documentation that the CE tries to access is set in the *global.properties* file, which resides in the *etc* folder in the installation directory of IBM Security Directory Integrator, or *solutions.properties* in the Solutions Directory. By default, this points to the online product documentation. If you change the following lines as shown here, then next time you run the CE and launch Help, it uses the local help system.

```
com.ibm.di.helpHost=publib.boulder.ibm.com
com.ibm.di.helpPort=80
```

to:

```
com.ibm.di.helpHost=localhost
com.ibm.di.helpPort=9999
```

- The location of the documentation server that AMC tries to access is set in its *web.xml* file. Open the *web.xml* file which is located in the *WEB-INF* folder of *tdiamc* webapp and list the IP address (or hostname) and port of the help server, for both occurrences of the following attributes: *InfocenterHostName* and *InfocenterPort*.

After you install the documentation in the *plugins* directory as outlined above, you can also decide to host the documentation on that computer for other installations of IBM Security Directory Integrator in your environment. In the *TDI_install_dir/ibm_help* directory there are a number of *.bat* files (Windows) or *.sh* files (Unix/Linux) that enable you to do this.

Starting and stopping the local documentation task

You must first start a local task to serve the documentation.

1. The help system is powered by Eclipse™ technology. (<http://www.eclipse.org>)

IC_start.bat or IC_start.sh

If you run this script, the script starts an information center on `http://your_IP_address:9999`

By editing this file, you can change the port number from the default, 9999; if you want to change this, to for example 80, change "-port 9999" to "-port 80". On those clients that are trying to access this information center, the port must match another property in the `global.properties` or `solution.properties` file, `com.ibm.di.helpPort` – its default is set to 80. Also, the `com.ibm.di.helpHost` property should read something like `infocenter_IP_address`, where `infocenter_IP_address` is the address of your local information center. In addition, in order for AMC to find this information center, you must update the parameters `InfoCenterHostname` and `InfoCenterPort` attributes in its configuration file, `web.xml`, to match the values above.

IC_stop.bat or IC_stop.sh

Stops the help system, a Java program, that serves the local information center.

help_start.bat or help_start.sh

Similar to `IC_start`, except the port used is a random one, and it also launches a local browser showing the start page. As the port is random, unsuitable for use other than on the local computer.

help_stop.bat or help_stop.sh

Stop the local Java task that was started by `help_start`.

Deploying AMC to a custom ISC SE or IBM Dashboard Application Services Hub

Use the instructions provided here to deploy the AMC after deferring it.

If you chose to defer deployment of AMC to ISC, and are now ready to deploy, follow these steps:

- Execute the following scripts:

```
TDI_install_dir/bin/setISCHome.bat(sh) ISC location
TDI_install_dir/bin/amc/install.bat(sh)
TDI_install_dir/bin/amc/setAMCRoles.bat(sh) username
```

Note:

1. Calling the `setAMCRoles` script is optional for both SE and AE. If executed, `username` should be an already existing one on the ISC/IBM WebSphere Application Server environment. As an alternative, you can use the ISC console ("Console User Authority" panel specifically) to manually assign one of the roles that came with AMC - "SDI AMC Admin" and "SDI AMC User" to a user.
 2. See the section "AMC in the Integrated Solutions Console" on page 242 for more information on AMC roles.
- Alter the `amc.properties` file so the lines specifying `am.api.port` and `amc.help.port` have appropriate port values. For ISC SE this file is located in `ISC location/runtime/isc/eclipse/plugins/AMC_7.2.0.0/` and for IBM Dashboard Application Services Hub this file is located in `ISC location/systemApps/isclite.ear/tdiamc.war`

If you chose a custom IBM Dashboard Application Services Hub to deploy AMC to, and are now ready to deploy, follow this step:

- Execute the following script:

```
TDI_install_dir/bin/amc/setAMCRoles.bat(sh) username
```

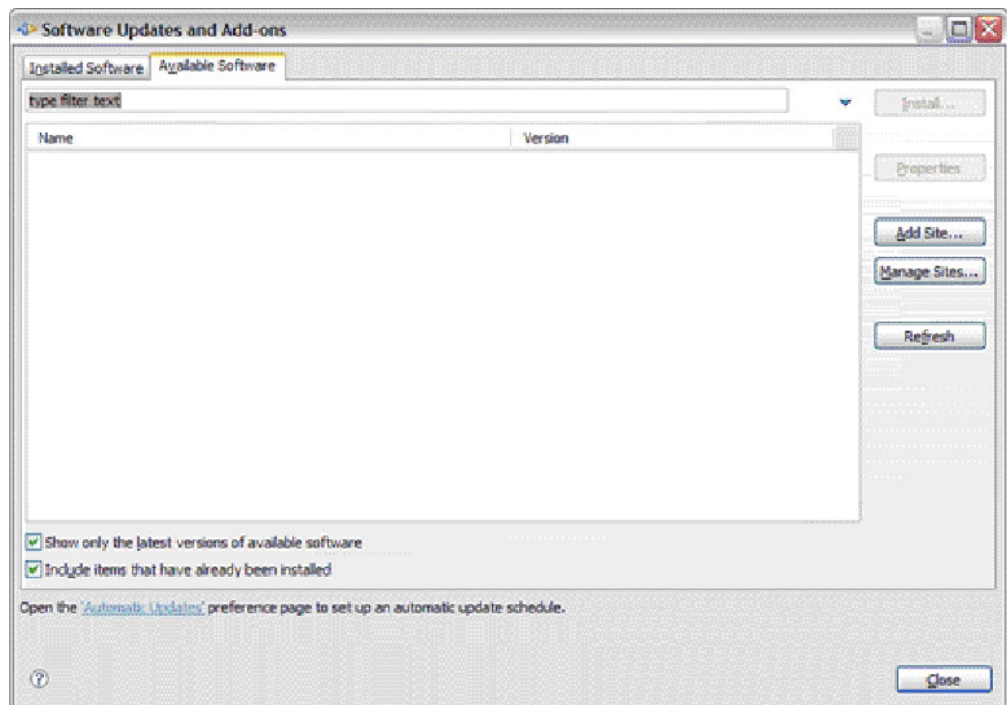
Note:

1. Calling the setAMCRoles script is optional for both SE and AE. If executed, *username* should be an already existing one on the ISC/IBM WebSphere Application Server environment. As an alternative, you can use the ISC console ("Console User Authority" panel specifically) to manually assign one of the roles that came with AMC - "SDI AMC Admin" and "SDI AMC User" to a user.
2. See the section "AMC in the Integrated Solutions Console" on page 242 for more information on AMC roles.

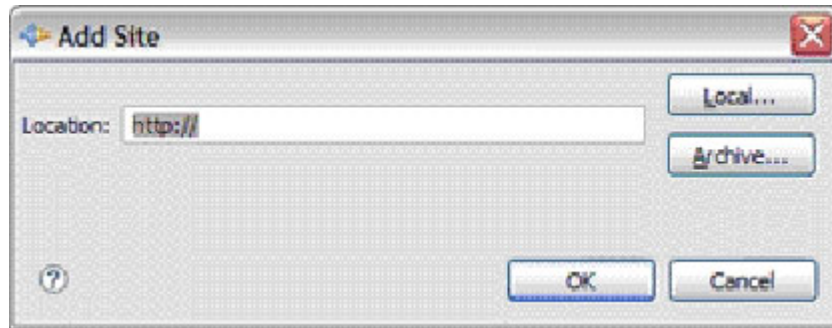
Installing or Updating using the Eclipse Update Manager

You can use the instructions provided here to use the Eclipse Update Manager.

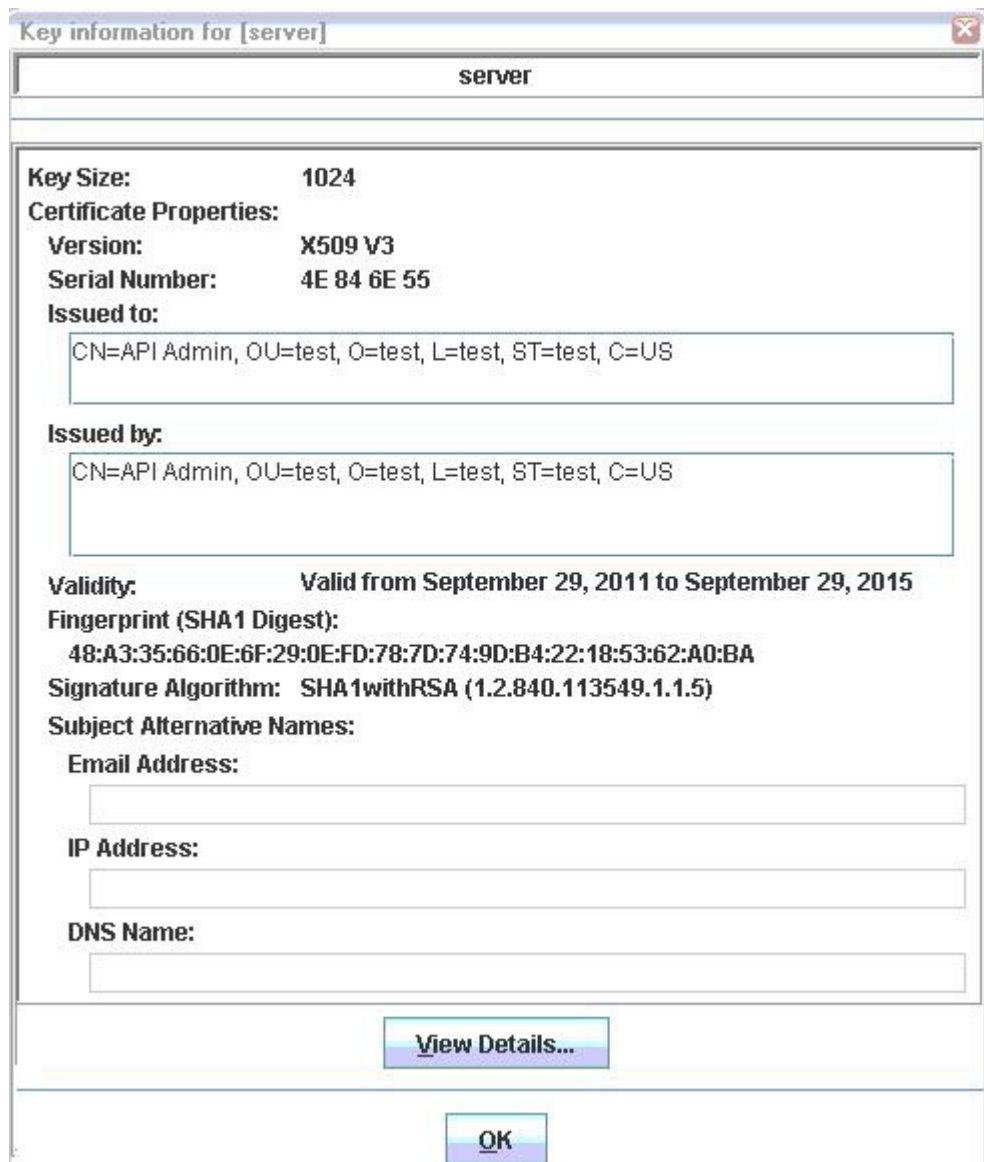
The IBM Security Directory Integrator Rich Client Platform contains a complete runtime environment to run the IBM Security Directory Integrator CE. However, it is possible to install the IBM Security Directory Integrator Eclipse plug-in into an existing Eclipse installation. This is done using the Eclipse Update Manager. In Eclipse, open the Eclipse Update Manager through the **Help** menu.



Before you have the IBM Security Directory Integrator plug-in installed you will want to add a new update site. Choose the **Add Site...** button and specify the location of the update site.



Depending on the location of the update site choose the appropriate action. In this example we choose a directory on the local file system. Using the **Local** button you are prompted to choose a directory which is then filled into the location input field. When you press **OK** the new update site and updates should be available:



Check the plug-ins you want to install and press **Install**. As the software update manager updates your installation you may be prompted to confirm the

installation and you are also usually encouraged to restart the workbench after installation. After installation is complete you should see IBM Security Directory Integrator in the **Installed Software** tab.

Post-installation steps

Perform some post installation tasks using the instructions provided here.

When the CE is installed as a plug-in in another Eclipse installation like in the procedure described above, a number of specific properties must be set to include the *SDI loader*. The IBM Security Directory Integrator loader is an `org.eclipse.osgi` fragment that provides class loading for the CE.

```
# TDI class loader
osgi.framework.extensions=com.ibm.tdi.loader
osgi.hook.configurators.include=com.ibm.tdi.loader.TDIClassLoaderHook
TDI_HOME_DIR=c:\Program Files\IBM\TDI\V7.2
```

Note that the property `TDI_HOME_DIR` needs to point to an existing IBM Security Directory Integrator Server installation, since the CE must be able to query many IBM Security Directory Integrator component Java classes in order to work correctly. This installation is also used to create the local development server that the CE uses. The fragment above shows the installation default for Windows; update this to reflect your environment.

There are several ways to set these properties. One is to update the `configuration/config.ini` file of the Eclipse installation.

Note: After installation and configuration of the CE into Eclipse, you may run into dependency problems. A Technote published about this issue may help you resolve such problems.

Uninstalling

You can uninstall IBM Security Directory Integrator in its entirety, or uninstall only certain components.

Launching the uninstaller

Launching the uninstaller requires some steps to be performed. Use the information provided here to do the same.

About this task

To uninstall IBM Security Directory Integrator, you must first launch the uninstaller:

Note: Before uninstalling, stop any component that you intend to remove, for example an instance of the IBM Security Directory Integrator Runtime, an AMC service that is running, or a Password Synchronization plug-in. Not stopping running components may cause some files to not be removed (to remain after the uninstallation). On Windows, a restart may be required and the IBM Security Directory Integrator Web Admin (AMC) service may remain in the Services list, requiring manual deletion.

Procedure

1. Navigate to the IBM Security Directory Integrator `_uninst` directory, for example: `install_path/_uninst`

2. Launch the uninstaller by executing the uninstall executable file.
For Windows platforms, the uninstall executable file is called `uninstaller.exe`.
For all other platforms, the uninstall executable file is called `uninstaller.bin`.
3. You will now enter the “Uninstall Panel flow” on page 31.

Results

Attention: During an uninstallation, a number of directories on the computer are emptied and removed. These are:

- `TDI_install_dir/lwi` - There is the possibility that some files are left over here, or files can get created by the embedded Web platform that the installer doesn't lay down. This directory is deleted on uninstallation.
- `TDI_install_dir/ce/eclipsece/features/com.ibm.tdi.*.jar`
- `TDI_install_dir/ce/eclipsece/plugins/com.ibm.tdi.*.jar`
- `TDI_install_dir/ce/eclipsece/configuration`
- `TDI_install_dir/ce/update_site/features/com.ibm.tdi.*.jar` - If any features have been added that match this wildcard, they will be deleted.
- `TDI_install_dir/ce/update_site/plugins/com.ibm.tdi.*.jar` - same as previous.
- `TDI_install_dir/maintenance/BACKUP` - This directory may be created by the update installer.
- `TDI_install_dir/_uninst/*`, `TDI_install_dir/amc/*`, `TDI_install_dir/osgi/*`, `TDI_install_dir/SCIM/*`, and `TDI_install_dir/LDAPSync/*` - These directories are deleted regardless of the changes made by the user.

Anything you may have put into these directories yourself that matches any of these criteria, will be removed as well during an uninstallation.

Performing a silent uninstallation

You can perform a silent uninstallation with the help of instructions provided here.

To perform a silent uninstallation of IBM Security Directory Integrator you must first generate a response file. To generate this file, you must perform a full GUI or console uninstallation with the `-options-record` option specified; for example:
`TDI_install_dir/_uninst/uninstaller.exe -r UninstallResponseFileName`
 The response file is created in the directory that you specify during uninstallation.

Note: The directory `TDI_install_dir/examples/install` contains a number of example response files for various installation or uninstallation scenarios. Once the response file is created, you can uninstall silently using the following command:
`TDI_install_dir/_uninst/uninstaller.exe -f UninstallResponseFileName`

Default installation locations

IBM Security Directory Integrator installs to the following default locations:

Windows platforms

`C:\Program Files\IBM\TDI\V7.2`

Linux and UNIX platforms

`/opt/IBM/TDI/V7.2`

Default solution directory

Use the `tdi_install_dir\bin\setDefaultSoldir.bat (sh)` to set a new default solution directory.

For example, if you run the following command:

```
setDefaultSoldir.bat C:\mysoldir
```

`TDI_SOLDIR=C:\mysoldir` is set in the `defaultSoldir.bat (sh)` file.

This value in the `defaultSoldir.bat (sh)` is used for setting the default directory when you choose the solution directory during IBM Security Directory Integrator installation.

The solution directory value in the `defaultSoldir.bat (sh)` script is set by the TDI installer.

This default value is changed based on the option you select on the Solutions Directory page of the installer.

The `defaultSoldir.bat (sh)` file is used by the IBM Security Directory Integrator server for getting the location of the default solution directory.

Chapter 3. Update Installer

Use the IBM Security Directory Integrator Update Installer, **applyUpdates.bat (sh)** to install fix packs to an existing IBM Security Directory Integrator installation.

The regular installer lays down a file that is named `.registry` in the installation directory that represents the current level of installed components. A script that is named `tdiSetBackupDir.bat` or `tdiSetBackupDir.sh` is created in the `bin` directory of the installation that sets the location of the backup directory. This directory is named `BACKUP` in the maintenance directory by default. You can change the backup location by running the `tdiSetBackupDir` script. For example, if a fix is named "ifix1", backup files and directories would be under `install dir/maintenance/BACKUP/ifix1` in this scenario. The update installer harvests the name of the backup directory during maintenance. The user who runs the maintenance procedure on IBM Security Directory Integrator must have write permission for the installation and backup directories. You must also be aware that during a complete uninstallation, the uninstaller attempts to delete the default backup directory.

The regular installer also handles maintenance of the `.registry` file during uninstalling and adding features.

- During a full uninstallation, the `.registry` file is deleted along with the other files.
- During a partial uninstallation, only the components that are being uninstalled are removed from the `.registry` file.
- When features are added, the `.registry` file is updated to contain the newly installed features.

After a feature is added, you must immediately install all of the fixes that are currently applied.

The update installer consists of several Java files. To avoid having to specify the Java executable file, a wrapper script is created in the `bin` directory, which is called **applyUpdates.bat (sh)**. This script uses existing scripts to find the right JRE to use and call the underlying code. The script's usage is shown here:

```
applyUpdates -update fix_file.zip [-clean [-silent]]
applyUpdates -rollback
applyUpdates -queryreg
applyUpdates -queryfix fix_file.zip
applyUpdate -enroll license_file.zip
applyUpdates -?
```

The options are as follows:

-update

This option is used to apply a fix pack.

The name of the compressed file that contains the fix pack is `fix_file.zip`. It can be a relative or absolute path.

The **-clean** option, which is only available for a fix pack, erases all of the files that were backed up before the current fix pack was applied. You are prompted to confirm that you want to delete the old data. The **-silent** option suppresses the confirmation prompt.

When a fix pack is being reapplied, for example, if new features that need the fix pack were added, the **-clean** option is ignored.

If you use the **-clean** option to cleaning the backup directories, the ability to roll back is limited to a single level.

-rollback

This option is used to roll back to the state IBM Security Directory Integrator was in before the most recent fix was applied. This data, by default, is stored in `tdi_install_dir/maintenance/BACKUP/FP##`.

-queryreg

This option shows the features that are in the current installation and all of the fixes that are applied.

The following output is an example:

```
Information from .registry file in: C:\Program Files\IBM\TDI\V7.2
Edition: Identity
Level: 7.2.0.1
```

```
Fixes Applied
=====
SDI-7.2-FP0001(7.2.0.0)
```

```
Components Installed
=====
BASE
SERVER
CE
CE UPDATE
JAVADOCS
EXAMPLES
IEHS
EMBEDDED WEB PLATFORM
AMC
    Deferred: false
PLUGINS
```

-queryfix

This option shows information about the fix that is contained in `fix_file.zip`.

The following output is an example:

```
Information from fix file: C:\fixes\SDI-7.2-FP0001.zip
Name: fixpack1
```

```
Minimum level required to apply fix: 7.2.0
Maximum level allowed to apply fix: 7.2.0.1
```

```
Prereq
-----
None
```

```
Components Affected
=====
BASE
CE
EXAMPLES
```

The compressed file with the fix `fix_file.zip` contains a manifest file `.manifest`, which contains information about applying the fix.

-enroll

This option is used to register an empty, trial, or full license. You can also use this option to upgrade the product from a trial version to a full version. However, this option is used by all of the installers. The license to be enrolled is contained in the compressed file and is passed as an argument.

To update a license for upgrading the product from a trial version to full version, run the following command:

```
applyupdates -enroll license_file.zip
```

Note: The *license_file.zip* file can be obtained from IBM sales or support team.

Complete the following steps if the IBM Security Directory Integrator server does not start because of an already enrolled license and due to hardware failure:

1. Take a backup of the *tdi-home.registry* file.
2. In the *.registry* file, remove the value that is defined in the license tag. For example, `<LICENSE>Full</LICENSE>`
3. Remove the existing nodelocked file from *tdi-home\license* directory.
4. Run the **applyUpdates** command.

```
tdi-home\bin\applyUpdates -enroll lumfile.zip
```

Depending on the number of licenses that are included in the compressed file, the resulting message indicates the licenses that are applied as shown in the following example.

```
./applyUpdates.sh -enroll /tmp/TDI_LUM_FULL.zip  
CTGDK0059I Trial license successfully enrolled.  
CTGDK0062I Full license successfully enrolled.
```

Note: The *lum_file.zip* file is deleted after the command is run.

-? This option is for usage information.

The .registry file

Use the *.registry* file to know the level of all IBM Security Directory Integrator components that are currently installed on the system in a particular installation directory.

The *.registry* file is in the installation directory. This file is initially created by the installer and is based on the options that were chosen at installation time.

When a fix is installed, the backed up files are stored in a directory with the name of the fix inside the backup directory. If the fix pack is installed successfully, extra entries are made to the *.registry* file, which represent the changes that are made to components by a fix. There is a *FIXES* section of the *.registry* file that represents the fixes were been applied, and each component has entries that represent which fixes were applied that altered them. However, if the fix pack is failed to install, the *.registry* file is not updated and contains the same entries that existed before the fix pack was applied.

The Update Installer recognizes the following components:

- BASE
- SERVER
- Configuration Editor (CE)
- CE_UPDATE
- JAVADOCS
- EXAMPLES
- IEHS
- embedded Web platform
- Administration and Monitoring Console (AMC)

- **PLUGINS:** The plug-ins component might require some steps that cannot be done by the update installer and must be done manually. If something needs to be changed in any of the `pwsync.props` files, you must do this change manually. Follow the steps in the `manual_readme.txt` file in the fix pack. You must complete these steps after installation of the fix pack, but before any of the post-installation steps that are listed in the following sections. The readme file warns you that the Update Installer updates only the files that the Installer installs. Files that you copied must be updated manually as described in the following post-installation steps. You need to do these steps before and after installation of the fix pack as specified. These steps are required only if you registered the corresponding Password Synchronizers into target systems.

Windows Password Synchronizer

Pre-installation

None

Post-installation

The following steps are required only if the fix pack contains an update of the DLL of the Password Synchronizer.

1. Remove the name of the DLL of the Password Synchronizer from the following registry key:

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\LSA\Notification Packages`

The DLL file is named `tdipwflt` on 32-bit Windows and `tdipwflt_64` on 64-bit Windows.

2. Reboot Windows, so that the Local Security Authority (LSA) process unloads the DLL of the Password Synchronizer.
3. Replace the DLL inside the `system32` Windows folder with the one from the installation of the Password Synchronizer. The DLL path after installation is either `install_dir/pwd_plugins/windows/tdipwflt.dll` or `install_dir/pwd_plugins/windows/tdipwflt_64.dll` depending on the version of Windows.
4. Add the name of the DLL (without the `.dll` extension) inside the registry key:
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\LSA\Notification Packages`
5. Reboot Windows again, so that the LSA loads the new DLL.

Afterward you restart, the Password Synchronizer must run normally by using the updated files.

IBM Security Directory Server Password Synchronizer

Pre-installation

1. Stop the Directory Server.
2. Stop the Proxy process of the Password Synchronizer by using the **stopProxy** command-line utility. This step is necessary because the IBM Security Directory Server Password Synchronizer does not automatically stop its Proxy when terminated.

Post-installation

None

Sun Directory Server Password Synchronizer

Pre-installation

Stop the Directory Server.

Post-installation

None

PAM Password Synchronizer**Pre-installation**

If possible, avoid any password changes while the update takes place. Otherwise, unregister the Password Synchronizer from the PAM configuration file.

Post-installation

If you unregistered the Password Synchronizer before the update, register it again.

For more information, see *Password Synchronization Plug-ins* section of the IBM Security Directory Integrator documentation.

Domino Password Synchronizer**Pre-installation****Post-installation**

Complete the post-installation instructions in the section for *Domino HTTP Password Synchronizer* in *Password Synchronization Plug-ins* section of the IBM Security Directory Integrator documentation.

Next, do a new setup of the Domino plug-in as described in the section *Deployment on a single Domino Server*.

Installation of fix packs

To install fix packs, follow the instructions in the readme file that is provided with the fix pack.

If a fix file contains a fix for a component and that component is installed on the system, a number of programmed actions are performed for the individual components.

If any manual steps must be performed outside of the update installer, instructions are included in the readme file for the fix.

Note: All IBM Security Directory Integrator process must be shutdown before any fix pack update is carried out.

Rollback

During a Rollback, the Update Installer uses information previously laid out during a fix, and files backed up, to restore a previous state.

Note: All IBM Security Directory Integrator process must be shutdown before any rollback is carried out.

The rollback operation does not roll back any files on which you took a manual action during the fix pack installation.

Troubleshooting

Use the Update Installer logs to troubleshoot errors related to installation of updates.

The Update Installer creates a log file named `updateinstaller.log` in the `install_dir/logs` directory. By default, the messages of INFO level are logged. You can change this by altering the `install_dir/logsinstall_dir/etc/updateinstaller-log4j.properties` file so that DEBUG messages are also logged.

Chapter 4. Supported platforms

For information about supported operating systems, web browsers, and virtualization support refer the link provided here.

See the “Software requirements” section in the IBM Security Directory Integrator documentation at http://pic.dhe.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDI.doc_7.2/sysreqs.html.

Chapter 5. Migrating

You can learn about migration, its types, scenarios, and various components that can be migrated through the information provided here.

In the context of IBM Security Directory Integrator, "migrating" can mean a number of things.

- Prepare relevant files (and their contents) to be used in a new location, on the same machine or a different one; or
- prepare relevant files to be used with a new version of the product.

The following table summarizes migration scenarios:

Table 1. Migration scenarios

Source & Destination versions equal?	Source & Destination install paths equal?	Scenario description
no	no	Migrate files to a new version, which is installed in a different location
no	yes	Migrate files to a new version, which will be installed in the same location
yes	no	Migrate files to a different installation of the same version
yes	yes	Restore backed up files to their original location

If you have to both migrate to a new version and a new location, you should do the version upgrade first, because here we will cover location migration only for the current release.

The IBM Security Directory Integrator Installer can assist in migrating from IBM Security Directory Integrator 7.0, 7.1, and 7.1.1 to IBM Security Directory Integrator Version 7.2.

Note: Upgrading IBM Security Directory Integrator from versions 6.x or earlier, directly to version 7.2, is not supported. You must first upgrade from version 6.x to version 7.1.1 and then from version 7.1.1 to version 7.2.

Migrate files to a different location

You can take care of the certain issues before selecting on the component / file to be migrated.

In this section we cover only IBM Security Directory Integrator.

Which files do not need to be modified to be used in another location?

You can refer to the list of file types that are not required to be modified.

- User configurations, data files (.xml, .xsd, .xsl, .txt ...), key store files (.jks, ...), certificate files (.der, ...), and so forth.

Also consider the implications of section "Maintaining encryption artifacts - keys, certificates, keystores, encrypted files" on page 183.

- Scripts (see "Which files should not be used in another location under normal circumstances?" on page 63 for exceptions)
- .bat, .sh and .vbs files

- JAR files
- Native binaries - .exe, .dll, .so, and so forth.
- Server API registry file
- Server Stash File
- Derby databases:
 For example the default System Store database "TDISysStore" and the default AMC database "tdiamcdb".
 Note that you must move the database as a whole (the whole folder). You should not merge the files of two databases.
 For more complicated scenarios, transfer data between databases using the JDBC Connector.
- Action Manager property files (located in the *TDI_install_dir/bin/amc/ActionManager* folder)
- Configuration files from the "etc" folder except these:
 - build.properties
 - global.properties
 - updateinstaller-log4j.properties
 - tdisrvctl-log4j.properties
- AMC configuration files:
 - amc.properties
 - amcdbhandler.properties
 - amcdbschema.xml
 - idiamc.sth
 - The AMC feature is deprecated and will be removed in a future version of IBM Security Directory Integrator.

Which files need to be modified before they can be used in another location?

You can refer to the list of file types that are required to be modified.

In general, location-sensitive files will contain the absolute path of the installation folder in one or more places. These occurrences need to be replaced with the new location path, so that the file becomes relevant to the new location.

Below is a list of the files that need migration and hints about which fields to update. These hints are based on the default content of these files. If you have modified the files, there may be other fields that are also location specific and need to be updated too.

- bin/amc/amcwinbservice.ini:
 This is the configuration file for AMC when registered as a Windows service. Update the "WorkingDirectory", "StartCommand" and "StopCommand" properties.
- global.properties/solution.properties:
 Update the "com.ibm.di.store.database" property.
 Also consider these properties: "api.config.folder", "systemqueue.jmsdriver.param.mqe.file.ini" and "com.ibm.di.loader.userjars".
 If you migrate the file to an installation that uses a different encryption key, see section "Maintaining encryption artifacts - keys, certificates, keystores, encrypted files" on page 183.

- etc/updateinstaller-log4j.properties:
Update the "log4j.appender.Default.file" property.
- etc/tdisrvctl-log4j.properties:
Update the "log4j.appender.Default.file" property.
- ibmdiservice.props:
This is the configuration file for the Server when registered as a Windows service.
Update the "path", "ibmdirroot" and "jvmRoot" properties.
- pwsync.props:
These are the configuration files of the Password Synchronizers.
Update the "proxyStartExe", "logFile", "javaLogFile" and "mqe.file.ini" properties.

Which files should not be used in another location under normal circumstances?

You can refer to the list of file types that should not be used after migration.

- Certain scripts
The sole purpose of the existence of these files is to convey location specific data.
There is virtually nothing else in them, so they would be of little value in another location.
Consider these scripts from the *TDI_install_dir/bin* folder: "javaHome", "defaultSolDir", "backupDir", "tdiISCHome".
- .reg files
These are used by the Windows Password Synchronizer.
- IBM WebSphere MQ Everyplace queue manager files
Although you cannot easily migrate IBM WebSphere MQ Everyplace files to another location, you can transfer data from one IBM WebSphere MQ Everyplace queue to another using the JMS Connector with an IBM WebSphere MQ Everyplace JMS driver.
- CE workspace
To reuse Directory Integrator projects from the Config Editor workspace, export them as Directory Integrator configurations and import them into the new workspace.
- etc/build.properties
This file contains time and version information about the release of the product.

Migrating files that contain encrypted data

For more information on migrating files that contain encrypted data, please refer the link provided here.

See "Maintaining encryption artifacts - keys, certificates, keystores, encrypted files" on page 183.

Migrate files to a newer version

You can migrate the files to a newer version through 3 ways. Learn more about those with the information provided here.

Installer-assisted migration

You can learn about how the installer works, which all files are migrated automatically and manually, through the information provided here.

The installer migrates certain files automatically during upgrade to a newer version. Note that the installer considers only the installation folder of the Directory Integrator.

All solution folders that are different than the installation folder must be migrated manually (or using some of the tools described in section “Tool-assisted migration” on page 65).

Which files does the installer migrate automatically?

- 6.0 to 7.1
 - global.properties
 - Cloudscape database (if used for System Store) is upgraded to Derby Version 10.5.3. See “Migrating Cloudscape database to Derby” on page 88.
 - pwsync.props (for each installed Password plug-in)
- 6.1.x to 7.1
 - global.properties
 - the AMC database
 - amc.properties
 - am_config.properties
 - pwsync.props (for each installed Password plug-in)
- 7.0 to 7.1
 - global.properties
 - pwsync.props (for each installed Password plug-in)
- 7.1 to 7.1.1
 - global.properties
 - solution.properties (if exists in default Solution Directory)
 - pwsync.props (for each installed Password plug-in)
- 7.1.1 to 7.2
 - etc\reconnect.rules
 - etc\derby.properties
 - etc\jlog.properties
 - etc\log4j.properties
 - etc\tdisrvctl-log4j.properties
 - etc\tdimigbl-log4j.properties
 - etc\updateinstaller-log4j.properties
 - etc\it_registry.properties
 - etc\tp.xml
 - etc\activemq.xml
 - etc\global.properties
 - solution.properties (if exists in default Solution Directory)
 - pwsync.props (for each installed Password plug-in)
 - The AMC database (use the following tools to back up AMC: bin/backupam.bat, bin/backupamc.bat, and bin/backupamcdb.bat)

- AMC_7.2.0.0\amc.properties
- AMC_7.2.0.0\conf\amcdbhandler.properties
- AMC_7.2.0.0\conf\logging.properties
- bin\amc\ActionManager\am_config.properties
- bin\amc\ActionManager\am_logging.properties

Which files need to be migrated manually?

Everything mentioned in section “Manual migration,” except those mentioned in its first subsection, *Property Files*.

Tool-assisted migration

Go through the list of tools that are used for installer-assisted migration.

These tool are used by the installer for the installer-assisted migration. You can use them for manual migration.

Property files migration

- global.properties:
Use the "tdimiggl" tool from *TDI_install_dir/bin*; see section “Migrating global and solution properties files using migration tool” on page 90.
- amc.properties:
Use the "tdimigamc" tool from *TDI_install_dir/bin/amc*; see “AMC and AM Command line utilities” on page 283. The AMC feature is deprecated and will be removed in a future version of IBM Security Directory Integrator.
- am_config.properties:
Use the "tdimigam" tool from *TDI_install_dir/bin/amc*; see “AMC and AM Command line utilities” on page 283.
- pwsync.props (for each installed Password plug-in):
Use the "migpwsync" tool from *TDI_install_dir/pwd_plugins/bin*; see “Migrating Password plug-ins properties files using migration tool” on page 91.

AMC database migration

Use the "backupamcdb"/"restoreamcdb" tools from *TDI_install_dir/bin/amc*; see “AMC and AM Command line utilities” on page 283. The AMC feature is deprecated and will be removed in a future version of IBM Security Directory Integrator.

Cloudscape System Store migration (only for 6.0)

See the more detailed instructions in section “Migrating Cloudscape database to Derby” on page 88.

Manual migration

You can start your manual migration with the information provided here. Further, you can go through a extensive list of properties that have been changed, added and deleted in IBM Security Directory Integrator versions.

Copy your Config files and any other custom files, including Derby databases from your old installation directory to the new installation directory. IBM Security

Directory Integrator supports a Solution Directory, and we recommend you copy the Config files, property files, Derby databases, and so on, to such a solution directory instead of to the installation directory of IBM Security Directory Integrator version.

Once you have copied the objects referenced above to a new location, you can set out to manually migrate their contents to adapt them for use with IBM Security Directory Integrator as described in the sections below:

1. Property Files
2. Configurations
3. Customized scripts
4. Added or replaced JAR files in the installation
5. Password Synchronizer configurations

Note: Sandbox data is version-specific; data recorded under any previous version does not play in Version 7.2.

Property files

- global.properties:

The following tables list the properties that have been deleted, changed, or added in different versions of IBM Security Directory Integrator:

Table 2. Deleted and changed properties

Properties in Global/Solution.properties	Modified, deleted, or added	Remarks
web.server.ssl.on	*MODIFIED*	For more information about this property, see the table titled <i>New properties in v7.1.1</i> . From IBM Security Directory Integrator version 7.2 onwards, the default value of this property is true. Example: web.server.ssl.on=true
{protect}-dashboard.auth.user.admin	*ADDED*	This property is added in IBM Security Directory Integrator version 7.2. It is used for providing Federated Directory Server username and password. The default value of this property is admin. Example: {protect}-dashboard.auth.user.admin=admin To specify multiple Federated Directory Server user login accounts, see the following example: {protect}-dashboard.auth.user.admin=admin {protect}-dashboard.auth.user.user1=user1passwd {protect}-dashboard.auth.user.user2=user2passwd
dashboard.auth.localhost	*MODIFIED*	From IBM Security Directory Integrator version 7.2 onwards, the default value of this property is properties.
dashboard.auth.remote	*MODIFIED*	From IBM Security Directory Integrator version 7.2 onwards, the default value of this property is properties.

Table 2. Deleted and changed properties (continued)

Properties in Global/Solution.properties	Modified, deleted, or added	Remarks
com.ibm.di.server.NIST.on	*ADDED*	<p>From IBM Security Directory Integrator version 7.2 onwards, this property is used to turn NIST mode in IBM Security Directory Integrator.</p> <p>If this property is set to true, then IBM Security Directory Integrator is enforced to run in NIST Compliant Mode.</p> <p>The default value is false, that is, it does not run in NIST Mode by default.</p>

Table 3. Deleted and changed properties in v7.1.1

Old property (pre-v7.0)	New property	Remarks
## Active Correlation Technology engine settings # act.engine.rule.set.file =myrules.acts	*DELETED*	Remove ACT Engine and ACT Connector
# Location of directory where the JRE that SDI will use is installed com.ibm.di.jvmdir=\$jvmRoot\$	*DELETED*	No longer possible to specify.
com.ibm.di.scriptengine .precompile=true	*DELETED*	No longer possible to specify; the current script engine does not have this functionality.
com.ibm.di.scriptengine .regex=java	*DELETED*	No longer possible to specify - Java syntax is always followed.
ibmjs.options=com.ibm.di.script .ScriptEngineOptions	*DELETED*	Related to previous property; this is no longer a valid option.
com.ibm.di.store.create .checkpoint.store =<multiple statements>	*DELETED*	Checkpoint/Restart functionality is removed; any System Store create table statements related to this should be removed too.
com.ibm.di.admin.library.dir=	*DELETED*	The current Config Editor does not use this, so no longer possible to specify.
api.remote.on=false	api.remote.on=true	RMI enabled by default in IBM Security Directory Integrator Server - Setting to true since it is enabled by default.
javax.net.ssl.trustStore= {protect}-javax.net.ssl .trustStorePassword= javax.net.ssl.trustStoreType=	javax.net.ssl.trustStore =serverapi\testadmin.jks {protect}-javax.net.ssl .trustStorePassword=administrator javax.net.ssl.trustStoreType=jks	RMI enabled by default in IBM Security Directory Integrator Server - empty values replaced by the default truststore.
javax.net.ssl.keyStore= {protect}-javax.net.ssl .keyStorePassword= javax.net.ssl.keyStoreType=	javax.net.ssl.keyStore =serverapi\testadmin.jks {protect}-javax.net.ssl .keyStorePassword=administrator javax.net.ssl.keyStoreType=jks	RMI enabled by default in IBM Security Directory Integrator Server - empty values replaced by the default keystore.
com.metamerge .securityTransformation =DES/ECB/NoPadding	com.ibm.di.securityTransformation =DES/ECB/NoPadding	FIPS 140-2 Certification - property name changed.
com.ibm.di.server.keystore =myKeyStore.jks com.ibm.di.server.key.alias =myKeyAlias	api.keystore=myKeyStore.jks api.key.alias=myKeyAlias	Server API keystore properties renamed.

Table 3. Deleted and changed properties in v7.1.1 (continued)

Old property (pre-v7.0)	New property	Remarks
com.ibm.di.store.database =TDISysStore com.ibm.di.store.jdbc.driver =org.apache.derby.jdbc .EmbeddedDriver com.ibm.di.store.jdbc.urlprefix =jdbc:derby: com.ibm.di.store.jdbc.user=APP	#com.ibm.di.store.database =TDISysStore #com.ibm.di.store.jdbc.driver =org.apache.derby.jdbc .EmbeddedDriver #com.ibm.di.store.jdbc.urlprefix =jdbc:derby: #com.ibm.di.store.jdbc.user=APP	The EMBEDDED MODE properties for the System Store have been commented out, since the System Store now runs in Network mode by default. The Installer never makes this change; if you have previously used Cloudscape/Derby in embedded mode you will need to make this change manually.
#com.ibm.di.store.database =jdbc:derby://localhost:1527 /TDISysStore;create=true #com.ibm.di.store.jdbc.driver =org.apache.derby.jdbc .ClientDriver #com.ibm.di.store.jdbc .urlprefix=jdbc:derby: #com.ibm.di.store.jdbc .user=APP #com.ibm.di.store.jdbc .password=APP #com.ibm.di.store.jdbc.start.mode =automatic #com.ibm.di.store.jdbc.host =localhost #com.ibm.di.store.jdbc.port=1527 #com.ibm.di.store.jdbc.sysIBM=true	com.ibm.di.store.database =jdbc:derby://localhost:1527 /TDISysStore;create=true com.ibm.di.store.jdbc.driver =org.apache.derby.jdbc.ClientDriver com.ibm.di.store.jdbc.urlprefix =jdbc:derby://localhost:1527/ com.ibm.di.store.jdbc.user=APP com.ibm.di.store.jdbc.password=APP com.ibm.di.store.jdbc.start.mode =automatic com.ibm.di.store.jdbc.host =localhost com.ibm.di.store.jdbc.port=1527 com.ibm.di.store.jdbc.sysIBM=true	These are the new, default properties for the System Store in IBM Security Directory Integrator v7.1.1. If you have migrated your installation, you will need to make these changes to your global.properties file as well, if you wish to run the System Store in Networked mode. The new architecture of the Configuration Editor in conjunction with other changes to the development process make that running System Store in embedded mode is very cumbersome. Therefore, we highly recommend that you run in Networked mode.
api.config.folder=\$change\$/configs	api.config.folder=configs	The configs folder is now always local to the Solution Directory.
##----- ## System Queue settings ##----- ## If set to "true" the System Queue is initialized on startup and can be used; ## otherwise the System Queue is not initialized and cannot be used. systemqueue.on=false ### MQe JMS driver initialization properties ## Specifies the location of the MQe initialization file. ## This file is used to initialize MQe on TDI server startup. systemqueue.jmsdriver.param.mqe.file.ini=\$change\$/MQePWStore/pwstore_server.ini	##----- ## System Queue settings ##----- ## If set to "true" the System Queue is initialized on startup and can be used; ## otherwise the System Queue is not initialized and cannot be used. systemqueue.on=true ### MQe JMS driver initialization properties ## Specifies the location of the MQe initialization file. ## This file is used to initialize MQe on TDI server startup. systemqueue.jmsdriver.param.mqe.file.ini=MQePWStore/pwstore_server.ini	The System Queue is now enabled by default in IBM Security Directory Integrator. Also, the IBM WebSphere MQ Everyplace initialization file is now located in a directory subordinate to the Solution Directory.

Table 4. New properties in v7.0

Property	Remarks
com.ibm.di.server.fipsmode.on=false	New property for enabling/disabling FIPS mode was added.
## To enable the built-in JAAS Authentication mechanism, ## set this property to "[jaas]". api.custom.authentication ## JAAS Authentication properties ## ----- ## java.security.auth.login.config=	Provide support for JAAS as a Server API Authentication provider; empty property is provided in which you can specify the JAAS Configuration file.

Table 4. New properties in v7.0 (continued)

Property	Remarks
<pre>## Encryption certificate properties com.ibm.di.server.encryption.keystore = <<value of com.ibm.di.server.keystore from 6.1.1 global.properties>> com.ibm.di.server.encryption.key.alias = <<value of com.ibm.di.server.key.alias from 6.1.1 global.properties >> ## Server API keystore passwords {protect}-api.keystore.password= << keystore password from idisrv.sth>> {protect}-api.key.password= << key password from idisrv.sth if present>></pre>	Provide separate configuration options for certificate to be used for PKI Encryption and SSL.
<pre>## TDI Logging com.ibm.di.logging.enabled=true</pre>	Provide mechanisms to completely disable logging - set to "false" if you want to disable all logging.
<pre>derby.connection.requireAuthentication=true derby.authentication.provider=BUILTIN derby.database.defaultConnectionMode=fullAccess</pre>	Additional parameters for the System Store (in Derby) in Networked mode.
<pre>##PKCS11 options ##Set the value of following properties to use PKCS11 enabled ## devices to store TDI servers private key / certificate. com.ibm.di.pkcs11cfg=etc\pkcs11.cfg com.ibm.di.server.pkcs11=false com.ibm.di.server.pkcs11.library= com.ibm.di.server.pkcs11.slot= {protect}-com.ibm.di.server.pkcs11.password =PASSWORD</pre>	Support IBM Security Directory Integrator Server's private key/certificate on PKCS 11 compliant crypto devices.
<pre>## Specify the unique ID for the TDI Server ## ----- ## This property helps a client connecting to the TDI server to identify different servers ## running on the same IP and the same port in different time. (Default is blank) com.ibm.di.server.id=</pre>	IBM Security Directory Integrator Server must provide a unique server ID available to remote server clients to detect the server being talked to.
<pre>## Timeout in minutes for loading configuration. api.config.load.timeout=2</pre>	Config initialization and Server API initialization need to be synchronized.
<pre>com.ibm.di.server.encryption.keystoretype = jks com.ibm.di.server.encryption.transformation = RSA</pre>	Symmetric Cipher Support (FIPS 140-2 compliance).
<pre>## Specifies a list of Server notification types, which will be suppressed. ## Notifications of suppressed types will not be propagated by the notifications framework. ## The notification types in the list are separated by spaces. Wildcards may be included. ## Example: ## api.notification.suppress=di.al.* di.ci.start ## The above example will suppress all AssemblyLine related notifications as well as ## notifications for starting a configuration instance. ## If the property is missing or is empty, no notifications will be suppressed. api.notification.suppress=di.server.api.authenticate di .server.api.authorize.*</pre>	Provide IBM Security Directory Integrator Audit Capabilities - Server notification suppression.
<pre>api.audit.on=false</pre>	Provide IBM Security Directory Integrator Audit Capabilities.

Table 4. New properties in v7.0 (continued)

Property	Remarks
<pre> ## This property specifies whether LDAP Group authentication is turned on. ## If it is set to 'true', the group membership of the authenticating user ## will be resolved and will be taken into account during authorization. ## If it is missing, the default value 'false' is used. api.custom.authentication.ldap.groupsupport= false ## Specifies the name of the attribute of a user in LDAP that contains a list of the groups of which the user is a member. ## It is taken into account only if 'api.custom.authentication.ldap.groupsupport' is set to true. api.custom.authentication.ldap.usermembershipattribute= ## Specifies how groups are named in the membership attribute of a user. ## For example, if the user's membership attribute contains values, ## which correspond to the 'objectSID' attributes of groups, set this property to 'objectSID'. ## If the user's membership attribute contains distinguished names of groups, then set this property to 'dn'. ## The property is required in case 'api.custom.authentication. ldap.groupsupport' is set to true. api.custom.authentication.ldap.usermembershipattributecontent= ## Specifies the name of a group's attribute in LDAP which corresponds to the way the group is named in the TDI User Registry. ## For example, if LDAP groups are addressed in the TDI registry by their common name, then set this property to 'cn'. ## If the User Registry contains the distinguished names of the groups, then set this property to 'dn'. api.custom.authentication.ldap.groupnameattribute= ## Represents the LDAP directory context, where groups will be searched. ## It is required only when LDAP group support is enabled api.custom.authentication.ldap.groupsearchbase= ## Optional property, which represents a list of space-separated attribute names. ## Specifies attributes which have non-string syntax. ## api.custom.authentication.ldap.binaryattributes= </pre>	<p>Enhance Authorization to support LDAP groups.</p>

Table 5. New properties in v7.1

Property	Remarks
<pre> # api.remote.server.ports=8700-8900 </pre>	<p>Commented out by default; this property is used to configure RMI ports. This is useful in case the default ports conflict with your firewall.</p> <p>The server will use these ports to listen for incoming RMI service requests, in addition to listening on the ports defined by other properties. For outgoing RMI service requests, random port numbers may be used.</p>

Table 5. New properties in v7.1 (continued)

Property	Remarks
<pre>## The properties determine the default bind address and the remote bind address for the Server API. ## * means bind to all network interfaces. The Remote Bind Address overrides the Default one. ## Only one IP address should be set. No hostnames are accepted. ## Mind that the java.rmi.server.hostname property is set implicitly to equal the Remote Bind Address property when used. This will cause the client stubs to create sockets on the specified Remote Bind Address. # com.ibm.di.default.bind.address=* # api.remote.bind.address=*</pre>	<p>Commented out by default; these two properties are used to configure the network interface (hostname or IP address) that the Remote API listens on.</p>
<pre>## Touchpoint Server properties tp.server.on=false tp.server.port=1098 tp.server.config=etc/tp.xml tp.server.auth=false tp.server.auth.realm=Tivoli Directory Integrator Touchpoint Server</pre>	<p>These properties configure the REST interface to IBM Security Directory Integrator connectors using a Service Control Management Protocol (SCMP) based Service.</p>
<pre>## ## Server API client properties ## api.client.ssl.custom.properties.on=true api.client.keystore=serverapi/testadmin.jks {protect}-api.client.keystore.pass=administrator api.client.keystore.type=jks {protect}-api.client.key.pass=administrator api.client.truststore=serverapi/testadmin.jks {protect}-api.client.truststore.pass=administrator api.client.truststore.type=jks</pre>	<p>These properties enable custom SSL properties for Server API clients. If <code>api.client.ssl.custom.properties.on=true</code>, then the <code>api.client.*</code> properties will be used by Server API clients. Otherwise the default <code>javax.net.ssl.*</code> properties will be used.</p>

Table 6. New properties in v7.1.1

Property	Remarks
<pre>## Web container web.server.port=1098 web.server.ssl.on=false web.server.ssl.client.auth.on=false # web.server.session.timeout=300</pre>	<p>These properties are the general settings for REST API and Dashboard. It specifies the port and security settings of the HTTP access point into IBM Security Directory Integrator REST and Dashboard.</p>

Table 6. New properties in v7.1.1 (continued)

Property	Remarks
<pre>## Dashboard properties ## dashboard.on=true dashboard.templates.folder=dashboard/templates ## Dashboard authentication properties ## ## The values for localhost and remotehost can be: ## none: No authentication is required ## deny: All connections denied ## ldap: Authentication is done by logging into ## an LDAP server and optionally validating ## group membership ## ## dashboard.ldap.url ## Specify the LDAP host port and optionally a ## search base ## (ldap://<host>:<port>[/<search base>]) ## ## dashboard.ldap.url.group ## Specify the LDAP host port and optionally a ## search base ## (ldap://<host>:<port>[/<search base>]) ## dashboard.auth=true dashboard.auth.localhost=none ## dashboard.auth.remote=deny # dashboard.auth.ldap.url # =ldap://localhost:389/ou=users,ou=system # dashboard.auth.ldap.url.group # =ldap://localhost:389/cn=group1, # ou=groups,ou=system</pre>	<p>These properties are specific to the Dashboard. The properties, which are set manually are, “dashboard.on” (enables/disables the Dashboard web application) and the “dashboard.templates.folder” (location of template solutions).</p> <p>All other properties are editable in the Dashboard.</p>
<pre>## REST API ## ----- api.rest.on=true api.rest.auth=false api.rest.auth.realm # =Tivoli Directory Integrator REST API api.rest.jmsdriver.name # =com.ibm.di.systemqueue.driver.ActiveMQ api.rest.jmsdriver.queue.sender.persistence=false api.rest.jmsdriver.queue.sender.timeToLive=60000 api.rest.jmsdriver.param.jms.broker # =vm://localhost?brokerConfig # =xbean:etc/activemq.xml # api.rest.jmsdriver.auth.username # api.rest.jmsdriver.auth.password</pre>	<p>These properties configure IBM Security Directory Integrator REST API enablement and security settings. The api.rest.jmsdriver property specifies the JMS queue to use in async log messages. Messages are used by the Dashboard.</p>

Table 7. Deleted/modified in v7.1.1

Old property	New property	Remarks
<pre>com.ibm.di.store.database =jdbc:derby://localhost :1527/\$change\$/TDISysStore; create=true</pre>	<pre>com.ibm.di.store.database =jdbc:derby://localhost :1527/\$soldir\$/TDISysStore; create=true</pre>	<p>From IBM Security Directory Integrator Version 7.1.1, \$soldir\$ is not replaced with <TDI_install_dir> by default. The directory is updated in runtime inside JVM with current Solution Directory of the user. Thus, IBM Security Directory Integrator System Store is unique for each Solution Directory.</p>
<pre>tp.server.port=1098</pre>	<pre>*DELETED*</pre>	<p>This property is redefined as web.server.port=1098.</p>

- amc.properties:
The table below lists which properties have been deleted or changed in BM Security Directory Integrator v7.1.1:

Table 8. Deleted and changed properties in AMC

Old property (pre-v7.0)	New property	Remarks
AMC.auth	*DELETED*	
monitor.refresh.rate	*DELETED*	The refresh rate for the Monitor Status panel. The rate was specified in minutes.
monitor.startup	*DELETED*	To set the Monitor Status panel as the first panel to be seen by the user when (s)he logs in.
LDAPHostName LDAPPort LDAPAdminUID LDAPAdminPwd LDAPServerType LDAPBindID LDAPBindPassword LDAPSuffix LdapUserPrefix LDAPUserSuffix LdapGroupPrefix LDAPGroupSuffix LDAPUserObjectClass LDAPGroupObjectClass LDAPGroupMember LDAPUserFilter LDAPGroupFilter LDAPsearchTimeout LDAPsslEnabled LDAPIgnoreCase	*DELETED*	LDAP Details.
com.ibm.di.amc.jdbc.start.mode	New default value: Automatic	
com.ibm.di.amc.jdbc.host	New default value: localhost	
com.ibm.di.amc.jdbc.port	New default value: 1528	
com.ibm.di.amc.jdbc.sysibm	New default value: True	

The table below lists which properties have been added in BM Security Directory Integrator v7.0:

Table 9. New properties in AMC

New property (default)	Remarks
am.logrotate (10)	Used to determine the maximum age of AM log files in days. The log files older than the specified value are deleted. The minimum value is 1 and can be increased up to 2147483647.
amc.session.timeout (20)	Used to determine the maximum time (in minutes) a user is inactive, before the AMC session expires and is automatically logged out of AMC. The value should be a positive integer number.
al.workEntries.cacheSize (100)	Used by AMC when the AssemblyLine is started in Synchronous mode. The cache size specified here is used for determining the size of the work entries cache.
amc.db.type (derby)	Specifies the database being used by the AMC.
am.api.host (localhost)	Action Manager RMI Details.
am.api.port (13104)	Action Manager RMI Details.
com.ibm.di.server.port.default (1099)	Default port for IBM Security Directory Integrator server. This property can be modified by the IBM Security Directory Integrator installer to have a value different from 1099. When AMC is started for the first time (or if its database was lost), this property is read and its value saved in the newly created AMC database. Later it will be used when AMC connects to the default IBM Security Directory Integrator server instance.

- am_config.properties:

The table below lists which properties have been deleted or changed in BM Security Directory Integrator v7.1.1:

Table 10. Deleted and changed properties in AM

Old property (pre-v7.0)	New property	Remarks
com.ibm.di.amc.am.serverapi .fail.interval.time=120 com.ibm.di.amc.am.queryProperty .interval.time=600 com.ibm.di.amc.am.healthAL.interval .time=5	*DELETED*	These properties should be commented out as the values for these properties will be configured by the user from AMC while creating each Server API Failure Trigger, On Property trigger and Configuring a Health AL respectively. Hence the properties mentioned in the am-config.properties file will not be used.
com.ibm.di.amc.am.queryAL.interval .time	*DELETED*	
javax.net.ssl.trustStore=\$change\$ /bin/amc/ActionManager/testadmin.jks javax.net.ssl.keyStore=\$change\$ /bin/amc/ActionManager/testadmin.jks	javax.net.ssl.trustStore=bin/amc /ActionManager/testadmin.jks javax.net.ssl.keyStore=bin/amc /ActionManager/testadmin.jks	Truststore files are now local to Solution Directory.

The table below lists which properties have been added in BM Security Directory Integrator v7.1.1:

Table 11. New properties in AM

New property (default)	Remarks
smtp.host= smtp.port= smtp.user= {protect}-smtp.password=	SMTP server details, added in BM Security Directory Integrator 7.0.
javax.net.ssl.trustStore= TDI_Install_dir/serverapi/testadmin.jks {protect}-javax.net.ssl.trustStorePassword=administrator javax.net.ssl.trustStoreType=jks javax.net.ssl.keyStore=TDI_Install_dir/serverapi/testadmin.jks {protect}-javax.net.ssl.keyStorePassword=administrator javax.net.ssl.keyStoreType=jks	Action Manager SSL Properties, added in BM Security Directory Integrator 7.1.
com.ibm.di.amc.am.encryption.keystore = TDI_Install_dir/testserver.jks com.ibm.di.amc.am.encryption.key.alias = server com.ibm.di.amc.am.encryption.keystoretype = jks com.ibm.di.amc.am.encryption.transformation = RSA com.ibm.di.amc.am.stash.file = TDI_Install_dir/idisrv.sth	Action Manager encryption properties, added in BM Security Directory Integrator 7.1. These properties are similar to the encryption properties used by the server. For convenience the location of the stash file has been added as a property: com.ibm.di.amc.am.stash.file. By default the AM will reuse the server's keystore and stash file encryption/decryption of AM protected properties.

Configurations

Certain Directory Integrator components/features have been modified or removed. Configurations that reference these need to be migrated manually. Here is a list of affected components/features:

- Checkpoint/restart functionality:

This functionality is removed in 7.0. This leaves Connectors that support Iterator mode with only the default ability to do a simple reconnect and automatically skip forward as many times as the number of successful reads. The assumption is that skipping forward this number of entries would get you back to where you last left off. Most BM Security Directory Integrator Connectors will not automatically attempt to do this, because the behavior can be indeterminate or

not appropriate. However, the default behavior is specific per Connector. The ability to automatically skip forward as many times as the number of successful reads is a new reconnect option available to each Connector and is configured in the Connection Errors panel, see **The Configuration Editor -> The Connector Editor -> Connection Errors** in the *Configuring* section of the IBM Knowledge Center for IBM Security Directory Integrator. If you require more than the ability to automatically skip entries processed, you need to use one of the following options in your solutions:

- Configure Delta for an Iterator mode for dynamically changing result sets.
- Override the `on_connection_failure` hook and do custom reconnect logic.
- Derby/Cloudscape in embedded mode as System Store used by multiple JVMs:
Default and recommended behavior in IBM Security Directory Integrator is running Derby in networked mode. If you continue to use Derby in embedded mode, considerations regarding multiple JVMs attempting to use the same database simultaneously still apply; see "Using Derby to hold your System Store" on page 198. For migrating databases, see "Migrating Cloudscape database to Derby" on page 88.
- Exchange Changelog Connector:
This Connector is removed in v7.0. You may consider using the unsupported Exchange Changelog Connector that is now provided as an "example" in `TDI_install_dir/examples/ExchangeChangelogConnector`.
- Btree Connector:
This Connector is removed from the default installation in v7.0. Use the System Store Connector instead as described in section "Migrating BTree tables and BTree Connector to System Store" on page 88; alternatively, use the (unsupported) Btree connector that is now provided as an "example" in `TDI_install_dir/examples/BTreeDBConnector`.
- Domino Change Detection Connector:
This applies to 6.0 and 6.1 only.
The **Delivery Mode** parameter is removed and **State Key Persistence** will be used instead. The behavior of old configurations which use this parameter will be as follows:
 - If the **Delivery Mode** parameter is set to "Assured once and only once delivery" mode then the **State Key Persistence** parameter will be set to "After read" which is the same behavior - the synchronization state is saved right after the notes document is read.
 - If the **Delivery Mode** parameter is set to "Normal assured delivery" mode then a check for a valid **State Key Persistence** parameter is made. If such is not found then the value of the **State Key Persistence** parameter is set to "After read". If the parameter is found in the configuration then the original value of it is used.
- IDS Changelog Connector:
The CRAM-MD5 option is no longer available in 7.0; you must manually choose another authentication mechanism.
In version 6.2 of IBM Security Directory Server the BEREncoder and BERDecoder classes have been moved from the `com.ibm.asn1` package to the `com.ibm.ldap.bp.asn1` package. Starting from IBM Security Directory Server v7.0 custom user solutions that directly use the old classes (`com.ibm.asn1.BEREncoder` and `com.ibm.asn1.BERDecoder`) need be updated to reflect this change.
- EMF XMLToSDO and EMF SDOToXML Function Components:
These are deprecated in 7.0. Consider other functionality in the future.

- DSMLv2 Parser:

This applies only to 6.0.

The "dsml.request" and "dsml.response" attributes have been removed. These attributes used to provide the raw request and response objects from the ITIM DSMLv2 library. If you have old configurations using any of these Attributes, you to edit your old configurations so that these Attributes are no longer used. All the data available through the raw request and response objects is also available through the other Attributes delivered by the DSMLv2 Parser.

- ITIM Agent Connector:

If you have used the ITIM Agent Connector in a previous version of IBM Security Directory Server, you may have to change the way you configure SSL connections. The ITIM Agent Connector in IBM Security Directory Server uses JSSE (Java based keystore or truststore) for SSL authentication, and this requires that you configure the SSL related certificate details in the global.properties or solution.properties file; instead of mentioning the certificate name in the old ITIM Agent Connector's "CA Certificate File" Parameter. These are the steps involved:

1. Import the ITIM Agent's certificate that was previously mentioned in the "CA Certificate File" parameter into the IBM Security Directory Integrator truststore with for example *keytool* (Ikeyman can be used too):

```
keytool -import -file servercertificate.der -keystore tim.jks
```

In this example the truststore is stored in the file tim.jks.

2. Configure this truststore in the "server authentication" section of the global.properties or solution.properties file:

```
## server authentication

javax.net.ssl.trustStore=serverapi\tim.jks
{protect}-javax.net.ssl.trustStorePassword=administrator
javax.net.ssl.trustStoreType=jks
```

Now, the ITIM Agent Connector uses the same JSSE-based secure communications architecture as the rest of IBM Security Directory Integrator.

If you already have a truststore file configured in global.properties or solution.properties, then import the certificate into that store instead of creating a new one.

- XML Parser:

The pre-v7.0 XML Parser has been renamed and is now called the Simple XML Parser; the current XML Parser is a new parser with more functionality, especially regarding hierarchical objects. Config files created under earlier versions of IBM Security Directory Server referring to the XML Parser will, when imported into v7.1 and later, refer to the Simple XML Parser (as the class name has not changed). If you want to use the new XML Parser instead, you will need to change that in your AssemblyLines and/or Connectors. In order to have the new XML Parser behave like the old one did you must set the both Entry Tag and Value Tag parameters to the values used in the Simple XML Parser.

Note: The Simple XML Parser exports a script variable named "xmldom", which is not exported by the new XML Parser. The new XML Parser represents the deeper hierarchy with the Entry itself. Any logic that relies on the "xmldom" variable and cannot be reworked to make use of the hierarchical structure provided by the Entry class, must not migrate to the new XML Parser.

- Castor Java to XML and XML to Java Function Components:

From IBM Security Directory Server v7.1 onwards, the location of the Castor mapping file has changed, from *TDI_install_dir/jars/functions/di_castor_mapping.xml* to *TDI_install_dir/etc/di_castor_mapping.xml*.

Consequently, the default value for the **Castor Mapping File** parameter now reflects the new location.

- **HTTP Client Connector:**

From IBM Security Directory Server v7.1 onwards, the HTTP Client Connector has been modified to automatically send an HTTP "Connection" header with value "close" when it does not intend to reuse the TCP connection for more HTTP requests. The reason for this modification is to comply with HTTP 1.1 recommendation (<http://tools.ietf.org/html/rfc2616#section-14.10>).

This behavior is mandatory according to the HTTP 1.1 spec and previously you needed to code this yourself in the AssemblyLine.

- **HTTP Server Connector:**

From IBM Security Directory Server v7.1 onwards, HTTP Server Connector has been modified to use persistent HTTP connections by default. This means that one TCP connection can be used by the same HTTP client for multiple HTTP requests (<http://tools.ietf.org/html/rfc2616#section-8.1>). HTTP clients may still send a "Connection" header with value "Keep-Alive", but it is no longer required in order to use a persistent connection. Idle TCP connections will be closed automatically after 20 seconds of inactivity.

Customized scripts

If you have customized any of the Directory Integrator scripts (for example, adding items to the PATH or the LD_LIBRARY_PATH environment variables in the startup scripts - *ibmdisrv*, *ibmditk*), you should apply these customizations to the corresponding scripts of the new version.

Previous versions of IBM Security Directory Server used the (MY)CLASSPATH variable in these scripts; the current version has the required path information built in and does not require this variable anymore. If you had tailored the aforementioned scripts before to include some libraries of your own, you do not have to do anything with the CLASSPATH variable; just make sure your library is in the correct place (typically in the *jars/* directory) so it is found by IBM Security Directory Server. Alternatively, use the *com.ibm.di.loader.userjars* property in *global.properties* to point to your own directory to be included in the loader path. In IBM Security Directory Server, the property may specify several directories or jar files, separated by the Java Property "path.separator", which is ":" on Linux and ";" on Windows. The *TDILoader* for jar files searches directories recursively for files that contain classes and resources. Only files with a ".zip" or ".jar" extension are searched.

Added or replaced JAR files in the installation

If you have added JAR files to the installation, you should copy them to the new version too.

IBM Security Directory Server now requires and includes a Java 7 compliant JVM (J2SE version 7.0.4). If you have developed your own code in Java, linked this code against the JVM libraries and integrated this with your IBM Security Directory Server solution, you might have to recompile and re-link your code.

If you have overwritten any of the original JAR files of the installation (for example, putting any required MQ jars in *TDI_install_dir*/jars/3rdparty/IBM), you should do the same with the new version.

A 64-bit Java Runtime Environment (JRE) is used now on Windows x86-64, Linux x86-64 and Linux s390. Compared to a 32-bit JRE, some performance degradation has been observed in some scenarios; you can still use the Windows x86-32 installer for non-password plug-in activities if you believe you will have potential issues with performance degradation.

If you do use the 64-bit JRE, you need to be aware that 64-bit shared libraries will be needed for any custom component (connector, parser, FC) that depends on JNI.

Password Synchronizer configurations

- Windows Password Synchronizer
Follow the steps described in "Migration from previous installations" in the Windows Password Synchronizer section of the *Password Synchronization Plug-ins* section of the IBM Knowledge Center for IBM Security Directory Integrator.
- Other Password Synchronizers
There are no specific migration steps. Uninstall the old version, install IBM Security Directory Integrator and configure it to suit your needs.

Backing up important data

Know everything about backing up data with the information provided here.

Files backed up by the Installer

Here is a comprehensive list of files backed up by the installer in different version upgrade.

Upgrade from version 6.0 to 7.1:

If the Server feature is being upgraded the listed files will be backed up.

```
TDI_install_dir\global.properties to TDI_install_dir\etc\global.properties.v60
TDI_install_dir\serverapi\testadmin.jks to TDI_install_dir\serverapi\testadmin.jks.v60
TDI_install_dir\serverapi\testadmin.der to TDI_install_dir\serverapi\testadmin.der.v60
TDI_install_dir\serverapi\registry.enc to TDI_install_dir\serverapi\registry.enc.v60
TDI_install_dir\serverapi\registry.txt to TDI_install_dir\serverapi\registry.txt.v60
TDI_install_dir\idisrv.ssth to TDI_install_dir\idisrv.ssth.v60
TDI_install_dir\testserver.jks to TDI_install_dir\testserver.jks.v60
TDI_install_dir\testserver.der to TDI_install_dir\testserver.der.v60
```

In addition, configuration files and solution.properties will be backed up.

Upgrade from version 6.1.x to 7.1:

If the Server feature is being migrated, the listed files will be backed up: (The new suffix will be .v61 or .v611 depending on the previous version.)

```
TDI_install_dir\etc\global.properties to TDI_install_dir\etc\global.properties.v61x
TDI_install_dir\serverapi\testadmin.jks to TDI_install_dir\serverapi\testadmin.jks.v61x
TDI_install_dir\serverapi\testadmin.der to TDI_install_dir\serverapi\testadmin.der.v61x
TDI_install_dir\serverapi\registry.enc to TDI_install_dir\serverapi\registry.enc.v61x
TDI_install_dir\serverapi\registry.txt to TDI_install_dir\serverapi\registry.txt.v61x
TDI_install_dir\idisrv.ssth to TDI_install_dir\idisrv.ssth.v61x
TDI_install_dir\testserver.jks to TDI_install_dir\testserver.jks.v61x
TDI_install_dir\testserver.der to TDI_install_dir\testserver.der.v61x
TDI_install_dir\etc\reconnect.rules to TDI_install_dir\etc\reconnect.rules.v61x
TDI_install_dir\etc\derby.properties to TDI_install_dir\etc\derby.properties.v61x
TDI_install_dir\etc\jlog.properties to TDI_install_dir\etc\jlog.properties.v61x
TDI_install_dir\etc\log4j.properties to TDI_install_dir\etc\log4j.properties.v61x
TDI_install_dir\etc\tdisrvctl-log4j.properties to TDI_install_dir\etc\tdisrvctl-log4j.properties.v61x
TDI_install_dir\etc\act-jlog.properties to TDI_install_dir\etc\act-jlog.properties.v611
(IBM Security Directory Integrator 6.1.1 only)
```

In addition, configuration files and solution.properties will be backed up.

Upgrade from version 7.0 to 7.1:

If the Server feature is being upgraded the listed files will be backed up.

```
TDI_install_dir\etc\global.properties to TDI_install_dir\etc\global.properties.v70
TDI_install_dir\serverapi\testadmin.jks to TDI_install_dir\serverapi\testadmin.jks.v70
TDI_install_dir\serverapi\testadmin.der to TDI_install_dir\serverapi\testadmin.der.v70
TDI_install_dir\serverapi\registry.enc to TDI_install_dir\serverapi\registry.enc.v70
TDI_install_dir\serverapi\registry.txt to TDI_install_dir\serverapi\registry.txt.v70
TDI_install_dir\idisrv.sth to TDI_install_dir\idisrv.sth.v70
TDI_install_dir\testserver.jks to TDI_install_dir\testserver.jks.v70
TDI_install_dir\testserver.der to TDI_install_dir\testserver.der.v70
TDI_install_dir\etc\reconnect.rules to TDI_install_dir\etc\reconnect.rules.v70
TDI_install_dir\etc\derby.properties to TDI_install_dir\etc\derby.properties.v70
TDI_install_dir\etc\jlog.properties to TDI_install_dir\etc\jlog.properties.v70
TDI_install_dir\etc\log4j.properties to TDI_install_dir\etc\log4j.properties.v70
TDI_install_dir\etc\tdisrvctl-log4j.properties to TDI_install_dir\etc\tdisrvctl-log4j.properties.v70
TDI_install_dir\etc\act-jlog.properties to TDI_install_dir\etc\act-jlog.properties.v70
```

In addition, configuration files, the workspace and solution.properties will be backed up.

Upgrade from version 7.1 to 7.1.1:

If the Server feature is being upgraded the listed files will be backed up.

```
TDI_install_dir\etc\global.properties to TDI_install_dir\backup_tdi\global.properties
TDI_install_dir\serverapi\testadmin.jks to TDI_install_dir\backup_tdi\testadmin.jks
TDI_install_dir\serverapi\testadmin.der to TDI_install_dir\backup_tdi\testadmin.der
TDI_install_dir\serverapi\registry.enc to TDI_install_dir\backup_tdi\registry.enc
TDI_install_dir\serverapi\registry.txt to TDI_install_dir\backup_tdi\registry.txt
TDI_install_dir\idisrv.sth to TDI_install_dir\backup_tdi\idisrv.sth
TDI_install_dir\testserver.jks to TDI_install_dir\backup_tdi\testserver.jks
TDI_install_dir\testserver.der to TDI_install_dir\backup_tdi\testserver.der
TDI_install_dir\etc\reconnect.rules to TDI_install_dir\backup_tdi\reconnect.rules
TDI_install_dir\etc\derby.properties to TDI_install_dir\backup_tdi\derby.properties
TDI_install_dir\etc\jlog.properties to TDI_install_dir\backup_tdi\jlog.properties
TDI_install_dir\etc\log4j.properties to TDI_install_dir\backup_tdi\log4j.properties
TDI_install_dir\etc\tdisrvctl-log4j.properties to TDI_install_dir\backup_tdi\tdisrvctl-log4j.properties
TDI_install_dir\etc\tdimigbl-log4j.properties to TDI_install_dir\backup_tdi\tdimigbl-log4j.properties
TDI_install_dir\etc\updateinstaller-log4j.properties to TDI_install_dir
\backup_tdi\updateinstaller-log4j.properties
TDI_install_dir\etc\it_registry.properties to TDI_install_dir\backup_tdi\it_registry.properties
TDI_install_dir\etc\tp.xml to TDI_install_dir\backup_tdi\tp.xml
```

In addition, configuration files from TDI_install_dir\configs folder, the workspace and solution.properties will be backed up.

Upgrade from version 7.1.1 to 7.2:

If the Server feature is being upgraded the listed files will be backed up.

```
TDI_install_dir\etc\reconnect.rules
TDI_install_dir\etc\derby.properties
TDI_install_dir\etc\jlog.properties
TDI_install_dir\etc\log4j.properties
TDI_install_dir\etc\tdisrvctl-log4j.properties
TDI_install_dir\ etc\tdimigbl-log4j.properties
TDI_install_dir\ etc\updateinstaller-log4j.properties
TDI_install_dir\ etc\it_registry.properties
TDI_install_dir\etc\tp.xml
TDI_install_dir\ etc\activemq.xml
TDI_install_dir\etc\global.properties
solution.properties (if exists in default Solution Directory)
pwsync.props (for each installed Password plug-in)
The AMC database
ISC location/runtime/isc/eclipse/plugins/AMC_7.2.0.0\amc.properties
ISC location/runtime/isc/eclipse/plugins/AMC_7.2.0.0\conf\amcdbhandler.properties
ISC location/runtime/isc/eclipse/plugins/AMC_7.2.0.0\conf\logging.properties
TDI_install_dir\bin\amc\ActionManager\am_config.properties
TDI_install_dir\bin\amc\ActionManager\am_logging.properties
```

Backup tools

These tools are used for backup/restore by the installer.

They can also be used for manual migration.

backupamc/restreamc

Use to backup/restore AMC configuration files.

backupamcdb/restreamcdb

Use to backup/restore the AMC database.

backupam/restream

Use to backup/restore the Action Manager (AM) database.

See “AMC and AM Command line utilities” on page 283 for more details.

Manual backup

In case you are performing a manual back up, take care of the instructions provided here.

Manual backup means copy the file to some dedicated backup folder. Conversely, restore means copy the file from the dedicated backup folder to its original location.

Note that in some cases you have to consider dependencies between files. You need to backup a group of interdependent files as a whole. Such groups of files are:

Derby database files

To backup a database, backup the whole folder that contains the database files. For example copy the *TDI_install_dir*/TDISysStore folder to backup the default System Store database or copy the *TDI_install_dir*/bin/amc/tdiamcdb folder to backup the default AMC database.

IBM WebSphere MQ Everyplace queue manager files

Backup the whole folder of the IBM WebSphere MQ Everyplace queue manager. For example copy the *TDI_install_dir*/MQePwStore folder to backup the default System Queue.

CE workspace files

Backup the whole workspace folder.

OSGI Backup the whole osgi folder.

LDAPSync

Backup the whole LDAPSync folder.

SCIM Backup the whole SCIM folder.

Migrating AMC 7.x configuration settings to another AMC deployment

You can migrate AMC configuration data to a new AMC deployment with the instructions provided here.

Note: The AMC feature is deprecated and will be removed in a future version of IBM Security Directory Integrator.

The section explains the steps that are to be followed for migrating AMC configuration data to a new AMC deployment. These instructions are useful when migrating AMC 7.0 and later from ISC SE to IBM Dashboard Application Services Hub, IBM Dashboard Application Services Hub to ISC SE, or to create a mirror instance of AMC on another machine.

1. Backup all of the AMC configuration files and data:

- a. Stop the AMC that you are migrating configuration data from using the `stop_tdiadc.bat (sh)` script. This script stops the server on which AMC is deployed.
 - b. Execute the `backupamc.bat (.sh)` script specifying the directory in which the AMC configuration files need to be backed up.
2. Migrate all of the AMC configuration data to the new AMC instance:
 - a. If the ISC you are migrating the AMC configuration to, is residing on another machine you will need to get the AMC backup directory copied to the new machine.
 - b. Ensure the AMC being migrated to is stopped. See step 1a for information on stopping AMC.
 - c. Execute the `restoreamc.bat (.sh)` script specifying the directory you have the AMC configuration information backed up to. This command will place the AMC configuration files at the correct location in the AMC you are migrating to. This completes the migration process.

Note:

1. The instructions are for migrating AMC 7.0 and later to any other ISC deployment.
2. The instructions assume that you already have AMC deployed already in both the system being migrated from and the system being migrated to using the IBM Security Directory Integrator installer. The systems can be either running AMC in ISC SE or IBM Dashboard Application Services Hub.
3. If you are trying to migrate the AMC configuration to another AMC deployment on the same machine and you want all of the AMC commands shipped with your IBM Security Directory Integrator deployment to use that AMC from that point on, you will need to update the configured ISC location in the AMC command line utility. This can be done using the following command: `setISCHome.bat (sh)`. The command takes as a parameter the location of the ISC installation directory, which is the installation location of IBM WebSphere Application Server for IBM Dashboard Application Services Hub and the location of the embedded Web platform for ISC SE. The command needs to be executed between steps 1 and 2 mentioned above.

Converting from EventHandlers to corresponding AssemblyLines

You can migrate certain parts of a typical EventHandler. Learn in detail about these with the instructions provided here.

EventHandlers do not exist in IBM Security Directory Integrator Version 7.2. Therefore, in order to replace the deleted functionality in old solutions, you need to migrate your EventHandler configurations to Server/Iterator or Changelog Connector configurations.

For each EventHandler a corresponding AssemblyLine must be created. Then a Server/Iterator Connector corresponding to the EventHandler must be inserted into the AssemblyLine "Feeds" section. Then the Connector parameters must be set - this is specific for each EventHandler/Connector pair, but generally the Connector parameters must be set to the same values as the corresponding EventHandler parameters (which usually have the same names).

Any processing configured in the EventHandler must be re-implemented in the AssemblyLine "Flow" section.

The functionality of the "enabled" EventHandler parameter (otherwise known as "Auto-start service") is also available for AssemblyLines. If you want your AssemblyLine to be started right after the IBM Security Directory Integrator Server is started, go to the **Solution Logging and Settings** section in the Navigator in your workspace in the Config Editor and add your AssemblyLine.

In general an EventHandler executes some piece of logic when a certain event occurs. "Event" has a different meaning for each EventHandler. For the HTTP EventHandler an "event" is an HTTP request. For the IBM Security Directory Integrator EventHandler an "event" is a change notification that comes from an IBM LDAP Directory.

Below are some general guidelines on migrating certain parts of a typical EventHandler. They are divided based on the titles of the UI tabs for an EventHandler in the pre-7.0 Config Editor:

Hooks

The "Prolog" hook of an EventHandler corresponds to the "Prolog - After Init" hook of an AssemblyLine. This hook is invoked for each incoming "event".

The "Epilog" hook of an EventHandler corresponds to the "Epilog - After Close" AssemblyLine hook. This hook is invoked once after each incoming "event" is processed.

In both the "Prolog" and "Epilog" EventHandler hooks the "event" Entry is accessible under the names "conn" and "event". However in the AssemblyLine hooks you should modify your script to use "work" instead of "conn" or "event".

The "Shutdown Request" hook of an EventHandler corresponds the "Shutdown Request" AssemblyLine hook.

Action Map

The Action Map of an EventHandler defines what actions should be taken when an "event" arrives. You should build the same actions into the logic of the AssemblyLine that you are preparing as a replacement of the EventHandler.

For example if the Action Map prescribed that a custom script should be executed if Attribute "x" of the event equals "3", then you could add an "IF" component to the AssemblyLine that checks for Attribute "x" being equal to 3 and executes a Script Component.

Logging

If you have configured custom log appenders for the EventHandler, you should configure the same appenders in the logging settings of the AssemblyLine(s) that you are preparing as a replacement for the EventHandler.

Config

These configuration parameters are specific to each EventHandler. See the subsections below for instructions on how to migrate them. The subsections are named after the corresponding Connectors.

TCP Server Connector

You can reproduce an old EventHandler's configuration into a new Connector's configuration with steps provided here.

1. Create a new AssemblyLine and insert the TCP Server Connector in it.
2. Set the **tcp.port** and **debug** Connector parameters to the values of the corresponding EventHandler parameters.
3. Set the **useSSL** and **requireClientAuth** Connector parameters to false (unchecked in the Config Editor).

Mailbox Connector

Configurations that use the Mailbox EventHandler, need to be migrated by using the steps provided here.

There is no need to migrate existing configurations that use the IBM Security Directory Integrator v6.0 Mailbox Connector, because the IBM Security Directory Integrator IBM Security Directory Integrator Mailbox Connector is compatible with the IBM Security Directory Integrator v6.0 Mailbox Connector.

1. Create a new AssemblyLine and insert the Mailbox Connector in it.
2. Copy the contents of the **mailServer** EventHandler parameter to the Connector parameter with the same name.
3. Set the **mailProtocol** Connector parameter to the value of the EventHandler parameter with the same name.
4. Copy the contents of the **mailUser** and **mailPassword** EventHandler parameters to the Mailbox Connector parameters with the same names.
5. Copy the contents of the **mailFolder** EventHandler parameter to the Connector parameter with the same name.
6. Copy the contents of the **pollInterval** EventHandler parameter to the Connector parameter with the same name.
7. If the enabled EventHandler parameter is true, add your AssemblyLine to the "Config -> AutoStart" folder in the Config Editor; thus the IBM Security Directory Integrator server will start your AssemblyLine on startup.
8. If the **debug** EventHandler parameter is true, set the Connector parameter with the same name to true.

JMX Connector

Existing configurations that use the IBM Security Directory Integrator v6.0 JMX EventHandler can be transformed into IBM Security Directory Integrator Version 7.2 configurations that use the JMX Connector using the instructions provided here.

1. Create a new AssemblyLine and insert the JMX Connector in it.
2. Copy the contents of the **eventTypes** JMX EventHandler parameter to the JMX Connector parameter with the same name.
3. Select "local" for the **mode** Connector parameter.
4. Leave the **url** Connector parameter blank.
5. Set the **allMBeans** Connector parameter to true.
6. Leave the **mBeanTypes** Connector parameter blank.

SNMP Server Connector

You can know more about SNMP Server Connector using the information provided here.

The IBM Security Directory Integrator SNMP Server Connector provides all features of the IBM Security Directory Integrator v6.0 SNMP EventHandler except

support for single-threaded mode. The IBM Security Directory Integrator SNMP Server Connector works in multi-threaded mode only. If you need to migrate an existing IBM Security Directory Integrator v6.0 configuration using the SNMP EventHandler to an IBM Security Directory Integrator v7.0 configuration, which uses an AssemblyLine with the SNMP Server Connector, you need to do the following:

1. Create a new AssemblyLine.
2. Insert into the AssemblyLine an instance of the SNMP Server Connector.
3. Set the **udp.port** Connector parameter to the value this parameter has in your SNMP EventHandler configuration.
4. Set the **snmp.community** Connector parameter to the value this parameter has in your SNMP EventHandler configuration.
5. If your SNMP EventHandler used to be configured to be "Auto-started" by the IBM Security Directory Integrator Server, add your new AssemblyLine to the "Config -> AutoStart" folder of the Config Editor.

IBM Security Directory Server Changelog Connector

Existing configuration that use the IBM Security Directory Server EventHandler can be migrated to use the IBM Security Directory Integrator Changelog Connector using the information provided here.

1. Set the following Connector parameters to the values of the EventHandler parameters with the same names: **ldapUrl**, **ldapUsername**, **ldapPassword**, **ldapAuthenticationMethod**, **ldapUseSSL**, **ldapSearchBase**.
2. Leave the **jndiExtraProviderParams** Connector parameter empty.
3. Set the **iteratorStateKey** Connector parameter to some unique identifier, one that has no corresponding state saved in the System Store.
4. Set the **nsChangenum** Connector parameter to the next change number that the EventHandler would process. The last change number that the EventHandler has processed is normally stored in an external properties file, referenced by its **ldapChangeNumberFileName** parameter.
5. Set the **stateKeyPersistence** Connector parameter to "After read" (the EventHandler writes the last received change number to its file backend after it reads a changelog entry and before it dispatches it for processing).
6. Set the **mergeMode** Connector parameter to "Merge changelog and changed data". This will ensure that the changelog attributes (changenumber, targetdn, ...) appear as attributes of the Entry.
7. Set the **useNotifications** Connector parameter to true.
8. Set the **batchRetrieval** Connector parameter to false.

Note: As opposed to the EventHandler, the Connector does not let you select a part of the directory tree, for whose notifications it will listen - it subscribes for changes in the whole directory tree (the Connector does not have equivalents of the **ldapEventBase** and **ldapSearchScope** EventHandler parameters). If this is critical for you, you can implement some custom filtering in your solution to overcome this limitation of the Connector.

HTTP Server Connector

A configuration that uses the HTTP EventHandler can be migrated to use the HTTP Server Connector using the steps provided here.

1. Set the **tcpPort** Connector parameter to the value of the **Port** parameter of the EventHandler.

2. Leave the **backlog** Connector parameter empty.
3. Set the **contentType** Connector parameter to "text/html".
4. Set the **tcpDataAsProperties** Connector parameter to true (the EventHandler always returns the TCP information as properties).
5. Set the **headersAsProperties** Connector parameter to the value of the **headersAsProperties** of the EventHandler.
6. Set the **httpAuth** Connector parameter to true, if the EventHandler uses HTTP basic authentication (that is if it has a configured authentication Connector).
7. If the EventHandler uses HTTP basic authentication, set the **authRealm** Connector parameter to the value of the **authrealm** EventHandler parameter. If the **authrealm** EventHandler parameter is missing or empty, set the **authRealm** Connector parameter to "IBM-Directory-Integrator".
8. Set the **authConnector** Connector parameter to the value of the **AuthConnector** parameter of the EventHandler.
9. Set the **useSSL** Connector parameter to the value of the **useSSL** parameter of the EventHandler.
10. Set the **needClientAuth** Connector to false (the EventHandler does not support SSL client authentication).
11. Set the **msgChunked** Connector parameter to false (the EventHandler does not support chunking of HTTP responses).

LDAP Server Connector

A configuration that uses the LDAP Server EventHandler can be migrated to use the LDAP Server Connector using the instructions provided here.

1. Set the **ldapPort** Connector parameter to the value of the **tcp.port** parameter of the EventHandler.
2. Leave the **backlog** Connector parameter empty.
3. Set the **ldapUseSSL** Connector parameter to the value of the **ldapUseSSL** parameter of the EventHandler.
4. Set the **charset** Connector parameter to the value of the **charset** parameter of the EventHandler.
5. Set the **ldapBinaryAttributes** Connector parameter to the value of the **binary** parameter of the EventHandler.

Sun Directory Change Detection Connector

You can know more about Sun Directory Change Detection Connector using the information provided here.

The LDAP EventHandler catches notifications about changes in a directory tree. The EventHandler does not use a changelog, so it receives only real-time notifications. The Sun Directory Change Detection Connector offers basically the same functionality when run in real-time delivery mode. There are a few differences though:

The Connector does not have equivalents for the **ldapSearchFilter** and **ldapSearchScope** EventHandler parameters. To achieve the same functionality as in the EventHandler, you should implement some custom filtering that limits the set of received notifications.

The schema of the returned data differs between the Connector and the EventHandler. The Connector applies delta tagging to each Entry it returns, while

the EventHandler provides the type of the change in the "ldap.operation" property. For details on the schema consult the documentation of each component.

Once the considerations above are resolved, you can migrate an existing configuration with the LDAP EventHandler to use the Sun Directory Change Detection Connector like this:

1. Set the following Connector parameters to the values of the EventHandler parameters with the same names: **ldapUrl**, **ldapUsername**, **ldapPassword**, **ldapAuthenticationMethod**, **ldapUseSSL**, **ldapSearchBase**.
2. Leave the **jndiExtraProviderParams** Connector parameter empty.
3. Set the **deliveryMode** Connector parameter to "Realtime" (the EventHandler does not use a changelog, it only catches real-time notifications).
4. Set the **mergeMode** Connector parameter to "Return only changed data" (no changelog is used in real-time delivery mode by the Connector).

Active Directory Change Detection Connector

You can know more about Active Directory Change Detection Connector through the information provided here.

The migration from the AD Changelog EventHandler to the Active Directory Change Detection Connector is straight forward in the most aspects since the EventHandler itself has incorporated the older version this connector - Active Directory Changelog Connector in order to obtain changes from the AD.

Similar to the EventHandler the corresponding Connector can also be interrupted any time during the synchronization process, in that case it will store its state in the User Property Store. Both the EventHandler and the Connector rely on the uSNChanged mechanism in this process, by storing the USN number in the property store. They also offer sn API for retrieving the current USN synchronization values. The difference is that the EventHandler `getUSNvalues` method returns an Entry with Attributes:

```
START_USN  
END_USN  
CURRENT_USN_CREATED  
CURRENT_USN_CHANGEDT
```

whereas the Connector returns the current synchronization value as *long*.

Another difference is that the AD EventHandler initializes internally an LDAP Connector in order to block and receive change notifications. This behavior can also be simulated in the ADCD Connector by enabling the **useNotifications** parameter.

The following steps should be performed in order to migrate from an EventHandler-based solution to Connector-based one:

1. Create a new AssemblyLine with an instance of an Active Directory Change Detection Connector in Iterator mode.
2. Set the **ldapUrl**, **ldapUsername**, **ldapPassword** and **ldapAuthenticationMethod** to the values these connection parameters have in the EventHandler configuration.
3. Specify whether SSL connection is used according to the value in the old configuration.
4. Copy the content of the **ldapSearchBase** EH parameter to the same in the Connector configuration

5. Copy the content of the **persistentParameterName** EH parameter to the **persistentStateKey** Connector parameter.
6. Set the parameter **useNotifications** to true.
7. Set the **startAt** parameter according to the value in EH.
8. Leave the other Connector parameters as they are.
9. Transfer any logic in the Action Map section of the EventHandler to be invoked from the new AL.

DSMLv2SOAPServerConnector

You can know more about DSMLv2SOAPServerConnector using the information provided here.

The migration from the DSMLv2 EventHandler to the DSML v2 SOAP Server Connector requires rework of the AssemblyLines that are previously used with the EventHandler, so that they can be integrated in the solution with the DSMLv2 SOAP Server Connector. This is because the core architecture as changed; now a single AssemblyLine processes all operations. Therefore, all the old AssemblyLines logic responsible for handling the different types of DSMLv2 operations should be incorporated into the new AssemblyLine containing the DSMLv2 Soap Server Connector, or should be invoked using a AssemblyLine Connector. For this purpose branching components can be used in order to separate the logic for the specific DSMLv2 operations (available in the `dsml.operation` Attribute).

The migration of a configuration with the DSMLv2 EventHandler to a similar one with the DSMLv2 SOAP Server Connector consists of the following steps:

1. Create a new AssemblyLine with an instance of a DSMLv2 SOAP Server Connector in Server mode.
2. Copy the content of the EH **port** parameter to the **dsmlPort** Connector parameter.
3. Set the **authRealm**, **useSSL**, **binaryAttributes** and **msgChunked** to the values these connection parameters have in the EventHandler configuration.
4. Create a branch component for each of the DSMLv2 operations listed as parameters in the EH configuration and apply in the branches the logic implemented in the corresponding old AssemblyLine, either by transferring there the appropriate AL components or by invoking the old AL itself using an AssemblyLine connector. In both cases the naming context will no longer be needed.
5. Copy the content of the twoEH **WaySSL** parameter to the **needClientAuth** Connector parameter.
6. The EH Attribute **headerAsProperties** cannot be passed to the Connector, since the HTTP parser it initializes internally is configured to always set this value to "false". Therefore, in case the solution accesses headers as properties, it should be modified to use Attributes for this purpose (`getAttribute()` instead of `getProperty()`).
7. For compliance, the **soapbinding** Connector Attribute should be set to "false" since the DSMLv2 parser internally used by the EH does not take advantage of it.
8. In case an **authConnector** is specified in the configuration of the DSMLv2 EventHandler, then the HTTP basic authentication of the Connector must be enabled and the appropriate logic must be implemented in the "After Accepting connection" hook (for example, initialize the authenticator Connector and call its `lookup()` method using an Entry with Attributes "username" and "password"

as search criteria. Similar to the EventHandler the authentication is to be considered successful in case an Entry is returned).

9. The **indentoutput** parameter of the DSMLv2 parser internally used by the Connector cannot be set in contrast to the one used by the EH.

Migrating BTree tables and BTree Connector to System Store

You can learn to migrate BTree tables and BTree Connector to System Store.

The BTree Connector is deprecated, and is now only provided as an unsupported example. Therefore, you might decide to move the way your Delta information is maintained from the old Btree objects to Delta Tables in the System Store. The best strategy for doing this is engineering a situation where your Delta information is empty (for example, establishing a new baseline) and then switch from the Btree objects to the System Store Delta Tables. Note that the parameter that used to hold the filename of the Btree objects now indicates a table name in a database, so some editing of this value might be required.

Changing a solution to use the System Store Connector instead of the BTree Connector for storing IBM Security Directory Integrator Entries is straight forward since both connectors follow the same logic when specifying Key Attribute Name and Selection Mode attributes. The only difference is that instead of the underlying BTree database, the System Store Connector has to use predefined a database (for example the embedded Derby database) and specify a table to store into.

Storing other Java objects using the System Store Connector differs significantly from storing them with BTree and will require more elaborate transformation. The following solution, which puts Java objects in the underlying BTree database, cannot be directly applied to the System Store Connector, since it does not provide direct access to the backend database:

```
scripts var bt = system.getConnector("btreedb");
bt.initialize (null); var db = bt.getDatabase();
db.insert ("my key", new java.lang.String("my value"));
var value = db.search ("my key"); value = value + " - modified";
db.replace ("my key", value);
```

Instead of this the standard methods (`put()`, `find()` and `modify()`) from the Connector API can be used, but the object should be first wrapped into an Entry object, which subsequently can be stored in the System Store.

Migrating Cloudscape database to Derby

You can migrate Cloudscape database to Derby using the instructions provided here.

IBM Security Directory Integrator Version 7.2 uses Apache Derby Version 10.8 as its bundled database, used by default by the System Store. You will need to migrate your existing Cloudscape or Derby databases (created using previous versions of IBM Security Directory Integrator) to be able to use IBM Security Directory Integrator Version 7.2. Apache Derby Version 10.8 drivers that are shipped with IBM Security Directory Integrator Version 7.2 cannot be used to communicate with older versions of Cloudscape.

For details, and information on differences between Cloudscape/Apache Derby Version 10.8 and its prior versions, refer to the following web page:<http://publibfp.boulder.ibm.com/epubs/html/c1894710.html>.

Notable differences that have an immediate impact are as follows:

- The long varbinary data type is no longer supported. Instead, BLOB datatype has been introduced (making Derby compatible with DB2®). For this reason, all SQL Statements that made use of long varbinary datatype must now be modified to use BLOB.
- JDBC Java package names have changed from com.ibm.db2j.* in previous releases to org.apache.derby.* in Derby Version 10.
- The JDBC URL for Derby (embedded/network mode access) Version 10 is different from Cloudscape 5.1. Hence the JDBC properties mentioned in global.properties / solution.properties have also been modified for the current version of IBM Security Directory Integrator.

Table 12. JDBC URL differences

Connection type	Cloudscape Version 5.1	Derby Version 10
Embedded Derby / Cloudscape	jdbc:db2j:	jdbc:derby:
DB2 JDBC Universal Database Driver (Network mode)	jdbc:db2j:net	jdbc:derby:net (Not recommended to use)
DerbyClient Driver	-	jdbc:derby (Recommended)

The Derby team has provided a migration utility that migrates a Cloudscape 5.1 database to a new Derby Version 10 database. It migrates all the tables and their corresponding data into a newly generated Derby Version 10 database. It modifies all tables with varbinary datatype to BLOB datatype, hence making the migration process quite painless.

This utility is bundled with IBM Security Directory Integrator, in the *TDI_install_dir/tools/CSMigration* folder, along with a wrapper script that invokes the migration tool, called migrateCS.bat(sh). To migrate a Cloudscape 5.1 System Store Database created using IBM Security Directory Integrator Version 6.0 to Derby Version 10, you have to invoke the migrate script in the following manner:

```
migrateCS [Path_of_CloudscapeV51_Database] [Path_of_new_DerbyV10_Database]
```

Note: This migration utility can be used for migrating only from Cloudscape 5.1 to Derby Version 10. Hence, the *TDI_install_dir/tools/CSMigration/migratCS.bat(sh)* file can be used for migrating system store database from IBM Security Directory Integrator Version 6.0 to Versions 6.1.1 and later. However, for migrating system store database from IBM Security Directory Integrator Version 6.1.1 to later versions, you must simply copy the old TDISysStore from the Version 6.1.1 installation directory to the new installation of the new version.

You may need to give some thought to the location of the new Derby database. In IBM Security Directory Integrator v6.0 and v6.1.x, the System Store database often was located in the installation directory of IBM Security Directory Integrator; this is an unfortunate location for many reasons. For IBM Security Directory Integrator Version 7.2 we strongly recommend you use a Solution Directory, away from the installation directory.

Besides migration of data, you also need to modify your global.properties / solution.properties files (using the migration tool or manually) to incorporate the new JDBC URL parameters.

Migrating global and solution properties files using migration tool

Use the `tdimiggb1` tool located in the `TDI_install_dir/bin` directory to migrate any `global.properties` file starting with IBM Security Directory Integrator 6.x to Version 7.2.

The filename is `tdimiggb1.bat` on Windows and `tdimiggb1.sh` on UNIX/LINUX. Use the `tdimiggb1-4log4j.properties` file to control logging for `tdimiggb1.bat(sh)`.

The usage if the command is as follows:

```
tdimiggb1 -f propfile [-b backfile] [-n newfile] [-v] [-?]
```

where:

- f propfile - The name of the file to migrate
- b backfile - Backup the original file with the specified name
- n newfile - Name to give the file that is migrated
- s dir - Working directory where the solution directory is located.
- v - Enable verbose mode
- ? - Prints the usage statement

During the installation of IBM Security Directory Integrator, the installer backs up the existing `global.properties` file; and then calls this command, in order to migrate the `global.properties`.

The migration tool tries to migrate a `global.properties` file (or `solution.properties` file if required) up to the latest IBM Security Directory Integrator version. The tool (`tdimiggb1`) makes no assumptions about which release the `global.properties` files starts from and can handle `global.properties` files starting at IBM Security Directory Integrator version 6.0. The tool also tries to apply all migration changes unless a particular migration step is specifically declared inappropriate for migration by the migration tool. For these cases, perform the migration steps manually.

The activities of the migration tool are broken down into stages. In sequence, the tool:

1. Checks whether you have to migrate your Derby (Cloudscape) database (IBM Security Directory Integrator 6.0 migration).
2. Performs all of the migration actions in the following order:
 - a. Delete actions.
 - b. Add actions.
 - c. Derby (Cloudscape) migration file changes (only if necessary and only for IBM Security Directory Integrator 6.0 migrations).
 - d. Migration modify actions.
3. Calls the Derby (Cloudscape) migration tool `migrateCS` to migrate the database up to the current Derby version (only for IBM Security Directory Integrator 6.0 migrations).

For each action set (migration modify actions for example), the migration tool tries to perform the migration actions starting from the earliest release to the latest release. For migration from IBM Security Directory Integrator 6.0, the caller must separately invoke the Derby (Cloudscape) migration tool to migrate the database up to the current Derby version. The `tdimiggb1` tool only makes the required Derby (Cloudscape) modifications to the properties file itself.

4. Uses `log4j` logging APIs for logging error messages.

The log4j configuration file is specified in the startup script (the bat or sh) file. The command uses a file called `tdimiggb1-log4j.properties` to set up the log4j logging. The command changes directory to the solution directory and therefore uses the `tdimiggb1-log4j.properties` file in the solution directory if the IBM Security Directory Integrator installation directory is not specified.

Migrating Password plug-ins properties files using migration tool

IBM Security Directory Integrator contains a migration utility to upgrade the `pwsync.props` files for each of the installed Password plug-ins. The utility is called `migpwsync` and is provided to migrate the `pwsync.props` files read by both the native plug-in and the JavaProxy.

The `migpwsync` utility is shipped in the `TDI_install_dir/pwd_plugins/bin` directory.

The utility has the following options:

- **-?** - using this option the utility will print the help information and will exit.
- **-v** - using this option the utility will print more verbose information to the standard output
- **-f** - this is a required option used to provide the location of the `pwsync.props` file.
- **-b** - this is the option that specifies the location of the file that will be used as backup. This is an optional field and if not provided the value of the `-f` option will be used with ".backup" appended.
- **-n** - this option specifies the file location where the migrated information will be written to. This is an optional field and if not provided the value of `-f` will be used as the place to output the migrated configuration.

Examples

- Migrating the configuration file of PAM plug-in :

```
# TDI_install_dir/pwd_plugins/bin/migpwsync.sh
-f TDI_install_dir/pwd_plugins/pam/pwsync.props
```

- Migrating the configuration file of Windows plug-in :

```
> TDI_install_dir\pwd_plugins\bin\migpwsync.bat
-f TDI_install_dir\pwd_plugins\windows\pwsync.props
```

Note:

1. The installer will update all the `pwsync.props` files setup by it in the `TDI_install_dir/pwd_plugins` directory during installation. If you have moved any of the `pwsync.props` files then you need to be manually migrate it using a command similar to the ones above.
2. The `migpwsync` utility changes the current directory to the plug-ins home directory (`TDI_install_dir/pwd_plugins`.) The provided file paths will be considered relative to that directory, if they are not absolute paths.
3. After migrating old `pwsync.props` file, add the following ActiveMQ related properties, if you want to configure ActiveMQ as default the JMS Password store:
 - `jmsDriverClass=com.ibm.di.plugin.pwstore.jms.driver.ActiveMQ`
 - `jms.broker=<JMS Server address>`. For example, `jms.broker=tcp://<activeMQhost>:61616` or `jms.broker=ssl://<activeMQhost>:61617`
4. To configure ActiveMQ as the default JMS Password store, set the `jms.clientId` property in the `pwsync.props` file.

Chapter 6. Security

You can work upon the security features, use those, and take care of the issues, with the detailed instructions provided here.

Security features are found throughout IBM Security Directory Integrator (IBM Security Directory Integrator). Some features secure access into remote systems from IBM Security Directory Integrator, others protect access into IBM Security Directory Integrator from remote systems, and yet others provide mechanisms to secure data, such as user credentials into remote systems.

Many of the features described in this section are not necessary when running IBM Security Directory Integrator in a stand-alone mode in a secured environment. However, the features come in handy when other systems must communicate with IBM Security Directory Integrator, such as through the remote Web Admin Console (AMC) management tool or the IBM Security Directory Integrator Remote Server API. Furthermore, if multiple people have access to the IBM Security Directory Integrator server it could be necessary to protect access to confidential data, as well as maintain the integrity of the integration rules that IBM Security Directory Integrator executes.

This section explains the following features:

1. "Manage keys, certificates and keystores"
2. "Secure Sockets Layer (SSL) Support" on page 99
3. "Remote Server API" on page 107
4. "IBM Security Directory Integrator Server Instance Security" on page 131
5. "Miscellaneous Config File features" on page 140
6. "Web Admin Console Security" on page 147
7. "Summary of configuration files and properties dealing with security" on page 144
8. "Miscellaneous security aspects" on page 147

This section does not describe all the security capabilities of the individual IBM Security Directory Integrator components. Some common elements are described in "Miscellaneous security aspects" on page 147, however for individual elements of security configuration in the individual IBM Security Directory Integrator components, consult the *Reference* section of the IBM Knowledge Center for IBM Security Directory Integrator.

Manage keys, certificates and keystores

You can learn about managing different type of keys, listing those in a keystore, and creating keys, through the information provided here.

Background

Refer to the links provided here to know more about keys in SSL, encryption, and security concepts.

The main uses of cryptographic keys in the product are SSL (see section “Secure Sockets Layer (SSL) Support” on page 99) and encryption (see section “IBM Security Directory Integrator Server Instance Security” on page 131).

For detailed information on security concepts and how they are used in the IBM JVM, see <http://www.ibm.com/developerworks/java/jdk/security/>.

Public/private keys and certificates

You can learn more about public/private keys, how do they work independently, and with each other through the information provided here.

SSL and asymmetric encryption algorithms such as RSA (which is the default encryption algorithm of the Server) use public/private keys. Public and private keys have a one-to-one correspondence - matching public and private keys are called a "key pair".

Normally inside a keystore a public key comes wrapped in an X.509 certificate. Most keystore operations actually involve the whole public key certificate and not only the public key.

Again in most cases inside a keystore a private key is accompanied by the corresponding public key certificate.

Secret keys

You can know about which all algorithms and keystores make use of secret keys with the knowledge provided here.

Secret keys are used by symmetric encryption algorithms such as DES, AES and RC4. Note that some keystore formats such as JKS and PKCS#12 do not support secret keys.

You cannot use secret keys for SSL (the SSL protocol actually generates secret keys on the fly, but normally you don't have control over them).

Keystores

Learn more about keystores, their file formats, origin and a comparison among various keystores with the information provided here.

A keystore, as the name implies, provides storage for keys. It can be a file or a hardware device. The most popular keystore file formats used by Java programs are JKS, JCEKS and PKCS#12. See the following table for comparison:

Table 13. Keystore file formats

Keystore file format	Origin	Store public/private keys and certificates	Store secret keys
JKS	Proprietary	Yes	No
JCEKS	Proprietary	Yes	Yes
PKCS#12	Standard	Yes	No

Note that the only one of the above keystore formats that can store secret keys is JCEKS. Also in general JCEKS offers greater protection than JKS. JKS, JCEKS and PKCS#12 keystores are protected by a password. Furthermore, each private or secret key inside a keystore can be protected by an individual password. Public key certificates do not have passwords, because normally there is no need to keep them secret.

Keys for SSL

You require a set of public/private keys to work with SSL. Learn about the procedure of setting up this through the information provided here.

About this task

For detailed information on using SSL with the IBM JVM see: <http://www.ibm.com/developerworks/java/jdk/security/60/secguides/jsse2Docs/JSSE2RefGuide.html>.

To use SSL you need to provide a set of public/private keys. You cannot use secret keys for SSL.

An SSL connection has two sides – the SSL server side and the SSL client side. Each side has two keystores – an SSL keystore and an SSL truststore. Note that the word "keystore" is used both to mean a store of keys and an SSL keystore. So SSL keystore and SSL truststore are both keystores. In fact, the SSL keystore and the SSL truststore are only logical roles and it is perfectly legal to use the same physical keystore file for both. The SSL keystore contains a private key that is used to prove the authenticity of this SSL side to the other side of an SSL connection. The SSL truststore contains public key certificates of trusted parties.

Procedure

1. To setup keys for your SSL server, you can: Generate a private key and a corresponding self-signed public key certificate and put it in your SSL keystore. (see section "Generate a public/private key pair and a self-signed certificate"). This step is needed only if your side of the SSL connection has to prove its authenticity to its peers – that is if you are the SSL server or if you are the SSL client and client authentication is required.
2. [Optionally] Obtain a certificate from a Certificate Authority and replace your self-signed certificate with it. (see section "Import public key certificate in a keystore")
3. [Optionally] Export the public key certificate of your private key and distribute it to the SSL parties that will interact with you. (see section "Export public key certificate from a keystore") If you are using a certificate from a Certificate Authority then it will be enough for others to have only the certificate of the Certificate Authority itself.
4. Import certificates of trusted parties in your SSL truststore (see section "Import public key certificate in a keystore"). This step is mandatory for SSL clients. For SSL servers it is necessary only if client authentication is required.

Results

Note: If you are using the default properties to configure SSL (javax.net.ssl.*), the SSL keystore should contain exactly one private key, because there is no way to specify which key will be used.

Keys for encryption

You can learn about encryption, keys used for encryption and the corresponding algorithms through the information provided here.

For encryption you have two alternatives:

- use a public/private key pair
- use a secret key

For public key encryption the most popular algorithm is RSA. Note that other popular public key algorithms such as DiffieHellman (key exchange) and DSA (digital signature) cannot be used for encryption.

Generally encryption with secret keys is much faster and much more secure than encryption with public keys. However, by default the Directory Integrator Server uses public key encryption with RSA to preserve compatibility with earlier versions.

Tools

You can use keytool and Ikeyman utilities for working with keys and certificates. Refer to the information provided here to learn more about these tools.

The IBM JVM provides two utilities for working with keys and certificates - keytool and Ikeyman. keytool is a command-line utility that is popular in the Java community. Ikeyman is a GUI tool from IBM, which provides many of the features of 'keytool'. Both tools are located in the *TDI_install_dir/jvm/jre/bin* folder. For detailed information about these tools see the documentation of the IBM JVM: <http://www.ibm.com/developerworks/java/jdk/security/>

Default keystores shipped with the product:

Table 14. IBM Security Directory Integrator keystores

Keystore location	Keystore password	Trusted public keys	Private keys
<i>TDI_install_dir/testserver.jks</i>	server	admin	server
<i>TDI_install_dir/serverapi/testadmin.jks</i>	administrator	server	admin

List the contents of a keystore

You can use the `list` command of keytool for listing the contents of a keystore.

For example the following command lists information about the keys (alias and type) inside the keystore file `mystore.jck`; the format of the keystore is JCEKS and its password is "mystorepass":

```
keytool -list -storetype jceks -keystore mystore.jck -storepass mystorepass
```

Create keys

You can learn about creating keys, managing keys in a keystore and using those through the information provided here.

Generate a public/private key pair and a self-signed certificate

For example the following keytool command generates an RSA public/private key pair with alias "myserverkey" and a X.509 self-signed public key certificate:

```
keytool -genkeypair -alias myserverkey -dname cn=myserver.mydomain.com
-validity 365 -keyalg RSA -keysize 1024
-keypass mykeypass -storetype jceks -keystore mystore.jck -storepass
mystorepass
```

The distinguished name of the owner of the certificate is "cn=myserver.mydomain.com", which should be the same as the DNS name of the server that will use the self-signed certificate for SSL (for public key encryption the content of the certificate does not matter much). The certificate is valid for 365 days. The size of the generated RSA key is 1024 bytes. The password of the private key is "mykeypass". The key pair is stored in a keystore file `mystore.jck` with format JCEKS (if the file does not exist, it will be created). The password of the keystore is "mystorepass".

The `mystore.jck` keystore can be used as an SSL keystore of a server program that runs on the "myserver.mydomain.com" host. The keystore also contains a public key certificate for the private key, so it can be used

as an SSL truststore for clients that connect to the server on "myserver.mydomain.com". (Although to give your private key to clients is completely unnecessary and generally a bad security practice.)

Obtain a certificate from a Certificate Authority

Normally the process of acquiring and using CA-signed certificates goes like this:

First a key pair and a self-signed certificate is generated (see section "Generate a public/private key pair and a self-signed certificate"). After that a certificate for the public key is requested from a Certification Authority. When the Certification Authority sends back the signed certificate, the certificate is imported into the appropriate truststore, replacing the self-signed certificate.

For example using `keytool` you can generate a Certificate Signing Request for the "myserverkey" key from the `mystore.jck` keystore like this:

```
keytool -certreq -file myreq.csr -alias myserverkey -keypass mykeypass
-storetype jceks
-keystore mystore.jck -storepass mystorepass
```

This command creates a Certificate Signing Request in the `myRequest.csr` file for the public key with alias "myserverkey". The created Certificate Signing Request now can be sent to a Certification Authority. When the new certificate arrives, you can import it in the keystore as described in section "Import public key certificate in a keystore". The following `keytool` command generates a 256 bit AES key with alias "myseckey":

```
keytool -genseckey -keyalg AES -alias myseckey -keysize 256 -keypass mykeypass
-storetype jceks
-keystore mystore.jck -storepass mystorepass
```

The new key is stored in a JCEKS keystore file `mystore.jck` with password "mystorepass". The password that protects the secret key is "mykeypass".

Copy key from one keystore to another

For example you can copy the key pair created in section "Generate a public/private key pair and a self-signed certificate" with the following `keytool` command:

```
keytool -importkeystore -srckeystore mystore.jck -destkeystore myotherstore.jks
-srcstoretype jceks
-deststoretype jks -srcstorepass mystorepass -deststorepass myotherstorepass
-srcalias myserverkey
-destalias myotherserverkey -srckeypass mykeypass -destkeypass myotherkeypass
```

The copy will be stored under alias "myotherserverkey" in the JKS keystore file `myotherstore.jks` (if it does not exist the file will be created).

Convert keystore from one format to another

For example you can convert the JCEKS keystore created in section "Generate a public/private key pair and a self-signed certificate" to a JKS keystore `myotherstore.jks` with the following `keytool` command:

```
keytool -importkeystore -srckeystore mystore.jck -destkeystore
myotherstore.jks -srcstoretype jceks
-deststoretype jks -srcstorepass mystorepass -deststorepass
myotherstorepass
```

The command will eventually ask for the password of each individual private or secret key inside the source keystore. Note that JKS and PKCS#12 keystores cannot hold secret keys. You should not try to convert a keystore that contains secret keys to either JKS or PKCS#12.

Export public key certificate from a keystore

The following command exports the public key certificate created in section "Generate a public/private key pair and a self-signed certificate" to a binary file `myserverkey.der`:

```
keytool -exportcert -alias mysserverkey -file mysserverkey.der
-storetype JCEKS -keystore mystore.jck
-storepass mystorepass
```

The resulting `.der` file contains the DER encoding of the X.509 certificate. It is a binary file. To get the same binary data in text form (base-64 encoded form of the DER encoding of the X.509 certificate) use the `"-rfc"` option of `keytool`:

```
keytool -exportcert -alias mysserverkey -file mysserverkey.arm
-storetype JCEKS -keystore mystore.jck
-storepass mystorepass -rfc
```

Import public key certificate in a keystore

To import a new trusted certificate in a keystore use a command like this:

```
keytool -importcert -alias mysserverkey -file mysserverkey.der
-storetype JCEKS -keystore mystore.jck
-storepass mystorepass
```

`keytool` will attempt to verify the signer of the certificate which you are trying to import. This means constructing a certificate chain from the imported certificate to some other trusted certificate. If a chain cannot be established, `keytool` will ask you whether you are certain that the certificate needs to be imported.

To import a certificate that is a response from a Certificate Authority to a Certificate Signing Request (this means you already have a private key in the keystore for that certificate) use a command like this:

```
keytool -importcert -alias mysserverkey -keypass mykeypass -file
mysserverkey.der -storetype JCEKS -keystore mystore.jck
-storepass mystorepass
```

Note that when you import a certificate for an existing private key, you have to specify the password of the private key. `keytool` will attempt to verify the signer of the certificate by constructing a certificate chain to a trusted certificate. If a chain cannot be established, the import will fail – you will not be asked to verify the authenticity of the certificate. To have a successful import of an answer to a Certificate Signing Request, you have to trust the Certificate Authority which issued the certificate. If your Certificate Authority is one of the popular ones (for example, VeriSign or Thawte) you could rely on the certificates in the default truststore of the JVM (`java.home/lib/security/cacerts`) by using the `"-trustcacerts"` option of `keytool`:

```
keytool -importcert -alias mysserverkey -keypass mykeypass -file
mysserverkey.der -storetype JCEKS -keystore mystore.jck
-storepass mystorepass -trustcacerts
```

Extend the validity of a certificate using keytool

Suppose you have a JCEKS keystore called `mystore.jck` that includes an expired (or about to expire) self-signed certificate whose alias name is `"mysserverkey"`. The keystore has the associated private key in it. Assume that the password for the keystore is `"mystorepass"` and the password for the private key is `"mykeypass"`. Now, if you want to extend the validity of this certificate by another 365 days, you can run the following command using `keytool`:

```
keytool -selfcert -v -alias mysserverkey -keypass mykeypass -validity 365
-storetype jceks -keystore mystore.jck
-storepass mystorepass
```

The above operation will generate a new self-signed certificate, that has the same DN, SIGALG, KEYS as the original certificate but has a new SERIAL NUMBER and VALIDITY period.

Note: The generated new certificate will automatically replace the original one.

So if you need the original one later for reference or for any reason, you must keep a copy of the original keystore before doing the certificate extension explained above.

Note that this works only for self-signed certificates. It actually generates a new self-signed certificate for the public key, so you need to export it and update the truststores of the SSL parties that you are going to communicate with.

Work with keys stored in PFX/PKCS#12 files

As far as Java is concerned PKCS#12 is just another type of keystore (like JCEKS and JKS). To work with PKCS#12 keystores just set the "-storetype" option of `keytool` to "pkcs12". For example the following command lists the content of a `mystore.p12` PKCS#12 file with password "mystorepass":

```
keytool -list -storetype pkcs12 -keystore mystore.p12 -storepass mystorepass
```

Create a keystore file

You don't need to create keystore files before you use them - `keytool` will automatically create a new keystore file, when it needs to write something to a file that does not exist. For example, if you generate a new key or import a certificate in a non-existing keystore, `keytool` will create the keystore file first.

Run keytool in FIPS mode

To run `keytool` in FIPS-compliant mode use the "-providerClass" option on each command like this:

```
keytool -list -storetype JCEKS -keystore mystore.jck -storepass mystorepass  
-providerClass com.ibm.crypto.fips.provider.IBMJCEFIPS
```

Secure Sockets Layer (SSL) Support

You can encrypt and authenticate the network traffic through SSL security feature. You can also view a list of supporting connectors and learn about the configuration through the information provided here.

SSL is an important foundation for many IBM Security Directory Integrator security features. You need a working-level knowledge of SSL in order to fully exploit the capabilities in IBM Security Directory Integrator.

The following Connectors support SSL with properly configured IBM Security Directory Integrator Servers:

- Connectors
 - AD Change Detection Connector
 - Axis Easy Web Service Server Connector
 - Axis2 Web Service Server Connector
 - Domino Change Detection Connector
 - Domino Users Connector
 - DSML v2 SOAP Server Connector
 - FTP Client Connector

- HTTP Server Connector
- IDS Changelog Connector
- IBM MQ Series Connector
- JMS Connector
- JMS Password Store Connector
- JNDI Connector
- LDAP Connector
- LDAP Group Connector
- LDAP Server Connector
- Lotus® Notes® connector
- Mailbox Connector
- Sun Directory Change Detection Connector
- LDAP Connector
- TADDM Change Detection Connector
- TADDM Connector
- TCP Connector
- TCP Server Connector
- TPAE IF Change Detection Connector
- Web Service Receiver Server Connector
- zOS LDAP Changelog Connector

SSL provides for encryption and authentication of network traffic between two remote communicating parties. Most production deployments of IBM Security Directory Integrator make use of SSL. That is why SSL support is one of the major security features of IBM Security Directory Integrator. More information on SSL as well as information on using SSL in Java programs from a development point of view can be found at <http://download.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html>

IBM Security Directory Integrator can be used as a client, as a server or as both at the same time. Configuring IBM Security Directory Integrator for SSL when used as a client is different from configuring IBM Security Directory Integrator when used as a server. That is why this section has been divided in two sub-sections – “Server SSL configuration of IBM Security Directory Integrator components” and “Client SSL configuration of IBM Security Directory Integrator components” on page 101.

Server SSL configuration of IBM Security Directory Integrator components

You need to define a keystore to enable SSL support for IBM Security Directory Integrator as a server. The steps provided here will help you perform this task.

About this task

When an IBM Security Directory Server component is used as a server (for example a Server mode Connector) SSL mandates that a keystore to be used by IBM Security Directory Integrator must be defined. For information on keystores and truststores, see the documentation at <http://download.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html> The following steps are required to enable SSL support for IBM Security Directory Integrator as a server:

Note: RMI is enabled by default in the IBM Security Directory Integrator server. Properties for server authentication carry the default keystore property values.

1. If you don't have a java (jks) keystore file already in IBM Security Directory Integrator create a keystore file using *keytool* (found in *TDI_install_dir/jvm/jre/bin*, or *TDI_install_dir/jvm/bin* depending on your platform). If you don't have a personal key to be used in IBM Security Directory Integrator get one from a Certificate Authority or create a self-signed key.
2. If the certificate in the IBM Security Directory Integrator is a self-signed certificate, export the certificate.
3. If the IBM Security Directory Integrator certificate is a self-signed certificate, using a key tool, import the exported IBM Security Directory Integrator certificate to the keystore file in the client as a root authority certificate.
4. Edit *TDI_install_dir/etc/global.properties* file for the keystore file location, keystore file password and keystore file type. ## client authentication

```
javax.net.ssl.keyStore=serverapi\testadmin.jks {protect}-  
javax.net.ssl.keyStorePassword=administrator  
javax.net.ssl.keyStoreType=jks
```
5. Enable SSL for the clients (for example, using https in the Web browser).
6. Restart IBM Security Directory Integrator

Note:

1. The IBM Security Directory Integrator server does not manage the keystores/truststores. All that the IBM Security Directory Integrator server provides to the IBM Security Directory Integrator components in terms of keystore support is the *global.properties* or *solution.properties* files, in which the standard Java keystore/truststore properties can be specified.
2. An IBM Security Directory Integrator component can choose to use the default configured keystore/truststore in *global.properties* or *solution.properties*, or it can choose to implement its own handling of SSL sockets (for example implementing a custom *SSLServerSocket* Java class) so that it can use keystores/truststores different from the default.
3. If IBM Security Directory Integrator needs to use both a client and a server certificate only the default certificate configured in *global.properties* or *solution.properties* is used, then this must be the same certificate. An alternative would be to write a custom implementation of the *SSLSocket* or the *SSLServerSocket* Java class and make it use a certificate different from the default.
4. See section “Certificates for the IBM Security Directory Integrator Web service Suite” on page 148 for specifics on the certificates for IBM Security Directory Integrator Web service components.

Client SSL configuration of IBM Security Directory Integrator components

You need to define a truststore to enable SSL support for IBM Security Directory Integrator as a client. The steps provided here will help you perform this task.

About this task

When an IBM Security Directory Integrator component is used as a client (for example the LDAP Connector) SSL mandates that a truststore to be used by IBM Security Directory Integrator must be defined. For information on keystores and truststores, see the documentation at <http://download.oracle.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html>

The following steps are required to enable SSL support for IBM Security Directory Integrator as a client:

1. Configure a server (such as IBM Security Directory Integrator) to enable SSL.
2. If the certificate in the server is a self-signed certificate, export the certificate.
3. If you don't have a Java (jks) keystore file already, create a keystore file using *keytool* (found in *root_directory/jvm/jre/bin*, or *root_directory/jvm/bin*, depending on your platform) for IBM Security Directory Integrator.
4. If the server certificate is a self-signed certificate, import the server certificate to the IBM Security Directory Integrator keystore file as a root authority certificate using *keytool*.
5. Edit *root_directory/etc/global.properties* file for the keystore file location, keystore file password and keystore file type.

Note: These four lines (comments starting with #) are no longer needed for client and server authentication to the IBM Security Directory Integrator server. Stores that belong to IBM Security Directory Integrator are set up to be used by default. This is part of enabling Remote Method Invocation (RMI) by default.

```
# Keystore file information for the server TDI authentication.  
# It is used to provide the public key of the TDI to the SSL enabled client.  
# javax.net.ssl.keystore=D:\test\clientStore.jks  
# javax.net.ssl.keystorePassword=secret  
# javax.net.ssl.keystoreType=jks
```

6. Enable SSL for the Connectors.
7. Restart IBM Security Directory Integrator.

Note: IBM Security Directory Integrator truststore and keystore do not play any part in SSL configuration for the Domino Change Detection connector. See section "Lotus Domino SSL specifics" on page 147 for more information.

SSL client authentication

In case you plan to use a IBM Security Directory Integrator component as a client and a server the SSL mandates to use a keystore and truststore both. Refer to the information provided here to learn more about it.

If an IBM Security Directory Integrator component is used as a client and the server with which it communicates requires SSL client authentication, then apart from a truststore, IBM Security Directory Integrator needs a keystore as well. In this case the keystore can be defined just like it is defined when IBM Security Directory Integrator is used a server – see the section "Server SSL configuration of IBM Security Directory Integrator components" on page 100.

Note: Client IBM Security Directory Integrator components which support SSL client authentication do not normally need a "SSL client authentication" check box as do IBM Security Directory Integrator server components. All such a client IBM Security Directory Integrator component needs in order to prove its identity to the server is to have its keystore generated and configured in *global.properties* or *solution.properties*. If the server requires an SSL client certificate then the client SSL library automatically sends the client's certificate from the keystore configured in *global.properties* or *solution.properties*.

IBM Security Directory Integrator and Microsoft Active Directory SSL configuration

You can follow the steps provided here to configure SSL for IBM Security Directory Integrator and Microsoft Active Directory.

About this task

1. Install Certificate Services on the Windows Server and an Enterprise Certificate Authority in the Active Directory Domain. Details are available at <http://windowsitpro.com/article/articleid/14923/how-do-i-install-an-enterprise-certificate-authority.html>. Make sure you install an **Enterprise Certificate Authority**.
2. Start the Certificate Server Service. This creates a virtual directory in Internet Information Service (IIS) that enables you to distribute certificates.
3. Create a Security (Group) Policy to direct Domain Controllers to get an SSL certificate from the Certificate Authority (CA).
 - a. Open the **Active Directory Users and Computers Administrative** tool.
 - b. Right-click, under the domain, **Domain Controllers**.
 - c. Select **Properties**.
 - d. Select the **Group Policy** tab, and click to edit the **Default Domain Controllers Policy**.
 - e. Go to **Computer Configuration->Windows Settings->Security Settings->Public Key Policies**.
 - f. Right click **Automatic Certificate Request Settings**.
 - g. Select **New**.
 - h. Select **Automatic Certificate Request**.
 - i. Run the wizard. Select the **Certificate Template for a Domain Controller**.
 - j. Select your **Enterprise Certificate Authority** as the CA. Selecting a third-party CA works as well.
 - k. Complete the wizard.

Note: All Domain Controllers automatically request a certificate from the CA, and support LDAP using SSL on port 636.

4. Retrieve the Certificate Authority Certificate to the computer on which you installed IBM Security Directory Integrator.

Note: You must install IIS before installing the certificate server.

- a. Open a Web browser on the computer on which you installed IBM Security Directory Integrator.
 - b. Go to `http://server_name/certsrv/` (where *server_name* is the name of the Windows 2000 server). You are asked to log in.
 - c. Select the task **Retrieve the CA certificate or certificate revocation list**.
 - d. Click **Next**.

The next page automatically highlights the CA certificate.
 - e. Click **Download CA certificate**

A new download window opens.
 - f. Save the file to the hard drive.
5. Create a certificate store using keytool. Use keytool.exe to create the certificate store and import the CA certificate into this store.

Note: keytool.exe is found in `root_directory\jvm\jre\bin`, or `root_directory\jvm\bin`, depending on your platform.

Use the following command:

```
jvm\jre\bin\keytool -import -file
certnew.cer -keystore keystore_name.jks
-storepass password-alias keyalias_name
```

For example, assume the following values:

```
Keystorename = idi.jks  
Password = secret  
Keyalias name = AD_CA
```

The command looks like this script:

```
C:\Program Files\IBM\TDI\V7.2\jvm\jre\bin\keytool -import  
-file certnew.cer -keystore idi.jks -storepass secret -alias AD_CA
```

To verify the contents of your keystore, type the following script:

```
C:\Program Files\IBM\TDI\V7.2\jvm\jre\bin\keytool  
-list -keystore idi.jks -storepass secret
```

The following lines result:

```
Keystore type: jks  
Keystore provider: SUN
```

Your keystore contains 1 entry:

```
ad_ca, Mon Nov 04 22:11:46 MST 2002, trustedCertEntry,  
Certificate fingerprint (MD5): A0:2D:0E:4A:68:34:7F:A0:21:36:78:65:A7:1B:25:55
```

6. Configure IBM Security Directory Integrator to use the keystore created in the previous step. Edit `root_directory/global.properties` file for the keystore file location, keystore file password and keystore file type. In the current release, only jks-type is supported.

```
#server authentication  
#example  
javax.net.ssl.trustStore=c:\test\idi.jks  
javax.net.ssl.trustStorePassword=secret  
javax.net.ssl.trustStoreType=jks  
#client authentication  
#example  
javax.net.ssl.keyStore=c:\test\idi.jks  
javax.net.ssl.keyStorePassword=secret  
javax.net.ssl.keyStoreType=jks
```

7. Enable SSL for your LDAP connector.
 - a. Go to the LDAP Connector configuration window.
 - b. Change **LDAP URL** to port 636.
 - c. Check **Use SSL**.
8. Restart IBM Security Directory Integrator.

Note: The IBM Security Directory Integrator Windows service wrapper permits you to start IBM Security Directory Integrator as multiple service instances.

Summary of properties for enabling SSL and PKCS#11 support

Refer to the link and information provided here to learn about configuring SSL properties.

You can configure SSL properties for server authentication, client authentication, and PKCS#11 support. See “Using cryptographic keys located on hardware devices” on page 182 on Public Key Cryptography Standards (PKCS).

Table 15. SSL Server Authentication

Property	Default value	Description
<code>javax.net.ssl.trustStore</code>	<code>serverapi\testadmin.jks</code>	Location of the truststore files.
<code>{protect}- javax.net.ssl.trustStorePassword</code>	administrator (encrypted by default)	truststore password.
<code>javax.net.ssl.trustStoreType</code>	jks	Type of the truststore.

Table 16. SSL Client Authentication

Property	Default value	Description
javax.net.ssl.keyStore	serverapi\testadmin.jks	keystore files location.
{protect}- javax.net.ssl.keyStorePassword	administrator (encrypted by default)	keystore password.
javax.net.ssl.keyStoreType	jks	Keystore type.

Table 17. PKCS#11 Support

Property	Default value	Description
com.ibm.di.pkcs11cfg	etc\pkcs11.cfg	Use this to specify the path of the configuration file required to initialize the IBM PKCS11 implementation provider. Added in IBM Security Directory Integrator 7.0.
com.ibm.di.server.pkcs11	false	Use pkcs11 compliant crypto devices for ssl. Added in IBM Security Directory Integrator 7.0.
com.ibm.di.server.pkcs11.library	none	Specify the path to the PKCS11 client library. Added in IBM Security Directory Integrator 7.0.
com.ibm.di.server.pkcs11.slot	none	Specify the slot number of the device.
{protect}- com.ibm.di.server.pkcs11.pass	none	Access the pkcs11 compliant crypto device using this password. Encrypted by default. Added in IBM Security Directory Integrator 7.0.

SSL example

In order to demonstrate how IBM Security Directory Integrator can be configured for SSL when used as a server and also when used as a client, two examples are provided – one deploying the LDAP Server Connector and one deploying the LDAP Connector.

IBM Security Directory Integrator component as a server

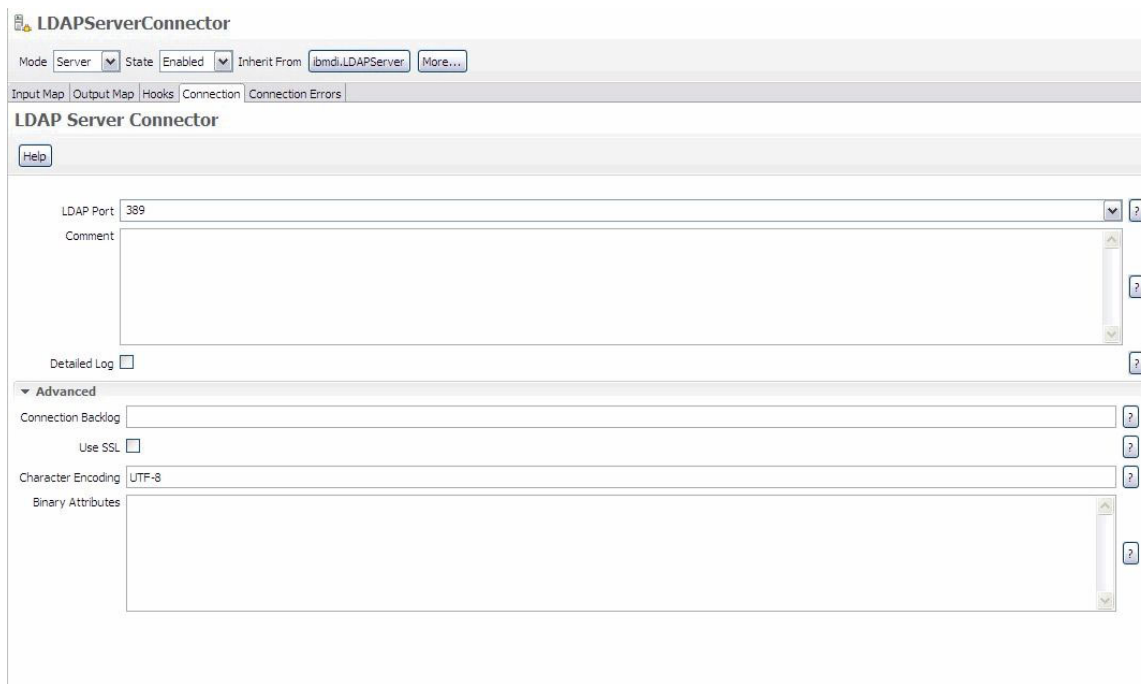
Refer to the steps listed here to configure IBM Security Directory Integrator component for SSL as a server.

About this task

This example uses the LDAP Server Connector. The LDAP Server Connector listens for LDAP requests. When an LDAP request arrives the Connector parses the request and provides the request data to the hosting AssemblyLine. The AssemblyLine then processes the request and provides the data for the response to the LDAP Server Connector, so that it can build the LDAP response and send it back to the LDAP client. The following guidelines explain step by step about how to configure IBM Security Directory Integrator for SSL when the LDAP Server Connector is used:

1. Obtain the server keystore either requesting it from a Certification Authority (CA) or creating a self-signed certificate as explained in the section called "Generate a public/private key pair and a self-signed certificate".

2. Set the keystore details in `global.properties` or `solution.properties` as described in the "Server SSL configuration of IBM Security Directory Integrator components" on page 100" section.
3. Select **Use SSL** on the Connector GUI configuration window. You may need to expand the Advanced section to make the parameter visible.



IBM Security Directory Integrator component as a client

Refer to the steps listed here to configure IBM Security Directory Integrator component for SSL as a client.

About this task

This example uses the LDAP Connector. The LDAP Connector connects to an LDAP Server and sends an LDAP request. After the Server returns the LDAP response the LDAP Connector provides that response to the AssemblyLine for further processing. The following guidelines explain step by step about how to configure IBM Security Directory Integrator for SSL when the LDAP Connector is used:

1. Generate the client truststore.
2. Import the LDAP server certificate into the client truststore.
3. Set the truststore details in `global.properties` or `solution.properties` as described in the "Client SSL configuration of IBM Security Directory Integrator components" on page 101" section.

The following command line imports an existing certificate into a keystore (the keystore is created if not already existing):

```
keytool -import -trustcacerts -file myLDAPServerCert.cer
        -keystore myClientTruststore.jks -storepass myclientTruststorePassword
        -alias myTrustedLDAPServerAlias
```

This command line imports a the `myLDAPServerCert.cer` certificate under alias `myTrustedLDAPServerAlias` into the `myClientTruststore.jks` keystore. The

password to access the keystore is `myclientTruststorePassword`.

Remote Server API

You can make a client application contact a server in IBM Security Directory Integrator. A client task can remotely invoke a server task. Learn more about this through the links and information provided here.

This section does not cover securing an instance of an IBM Security Directory Integrator Server; this is discussed in "IBM Security Directory Integrator Server Instance Security" on page 131. Instead, this section discusses how client applications can contact a server.

IBM Security Directory Integrator supports the concept of a Remote API (also known as just "Server API"), where client tasks can invoke tasks on a remote IBM Security Directory Integrator Server by means of an access layer called RMI.

Note: The "remote Server" could very well be running on the same computer as the client application, for example if you start up a Server instance on your local computer and then access it using the Remote API through the loopback address, 127.0.01. All concepts discussed below are still valid, even though the remote Server runs locally.

The Server API calls address the following areas:

- Getting Server information
- Getting information for components installed on the Server
- Reading and writing to configuration(s) loaded by the Server
- Loading new configurations into the Server
- Starting, querying and stopping AssemblyLines
- Cycling through AssemblyLines

Note: Increasing needs for remote server access for each running IBM Security Directory Integrator server have resulted in a change from local access by default to remote access by default. As of v7.1.1, the remote server API is enabled by default. Prior to v7.1.1, the server API was enabled only for local access by default, where local access means access from the same Java Runtime Environment (JRE). To ensure security, remote access requires SSL client authentication. SSL access using client authentication is provided with the sample keystore and truststore deployed with IBM Security Directory Integrator.

The Server API itself is documented in the IBM Security Directory Integrator Java API documentation (*TDI_install_dir/docs/api*; you can launch a browser to display this documentation by selecting **Help -> Welcome -> JavaDocs** in the CE). The package of interest in this context is `com.ibm.di.api`. Also see the section called "Using the Server API" under *Reference* section of the IBM Knowledge Center for IBM Security Directory Integrator.

The Configuration Editor uses the Remote API to talk to the server you use to test-run your solutions. If this IBM Security Directory Integrator server is running on the same machine, it is often called the "local development server". For setups where the deployment platform does not support the Configuration Editor, you can run the development server on the deployment server, and the Configuration Editor on a supported platform like Windows (this way of running we call the "Remote Configuration Editor"). This design provides a uniform interface for both

remote and local Config files. For some aspects of the Configuration Editor talking to a remote deployment server, see “Using the Remote Configuration Editor” on page 143.

The Server API is configured through a set of server properties (see “Configuring the Server API”). These properties are specified in the `global.properties` configuration file of the IBM Security Directory Integrator Server. Some of the properties, in turn, point to additional configuration files and keystore files.

The Server API provides a number of security-related features (which both IBM Security Directory Integrator Solution-based clients as well other client applications have to deal with). There are three aspects to Server API Access Security:

1. “Server API SSL remote access” on page 111 (which secures the transport channel to a remote IBM Security Directory Integrator Server),
2. “Server API authentication” on page 113 (which handles the client authentication to an IBM Security Directory Integrator Server),
3. “Server API Authorization” on page 123 (which handles the client authorization to an IBM Security Directory Integrator Server, that is, what the client is allowed to do once authenticated).

Configuring the Server API

You can refer to the list of properties that helps configure the server API.

The relevant properties are:

Property	Default value	Description
<code>api.on</code>	true	If set to true , the Server API is initialized on startup and can be used; otherwise the Server API is not initialized and cannot be used. All other properties whose names start with “api.” are only taken into account if <code>api.on</code> is set to true .
<code>api.audit.on</code>	false	If set to true , the audit feature is turned on. If it is set to true , an audit entry is created at each audit point, even if the audit notifications are suppressed.
<code>api.user.registry</code>	<code>serverapi/registry.txt</code>	If configured, specifies the Server User Registry file name.
<code>api.user.registry.encryption.on</code>	false	If set to true , the Server API decrypts the Server User Registry file on startup.
<code>api.remote.on</code>	true	If set to true , the remote RMI part of the Server API is initialized and can be used; otherwise, the remote RMI part of the Server API is not initialized and cannot be used.
<code>api.remote.ssl.on</code>	true	If set to true , SSL with client and server authentication is used on RMI connections of the Server API and its JMX layer; and the Server API uses the Server certificate and private key (the one specified through the <code>api.keystore</code> and <code>api.key.alias</code> properties) for SSL connections. RMI clients must trust that certificate. If set to false , no SSL is used for client connections and no authentication and authorization is performed; connections are accepted from the local host and from hosts listed in the <code>api.remote.nonssl.hosts</code> property; if <code>api.remote.nonssl.hosts</code> is empty, only connections from the local host are accepted.
<code>api.remote.ssl.client.auth.on</code>	true	If set to true , the SSL client authentication for remote Server API is switched on.
<code>api.remote.naming.port</code>	1099	If specified, the port on which the RMI registry listens for requests.
<code>api.remote.server.ports</code>	8700-8900	The range of ports that may be used by the various RMI services. Entered as a comma separated list allowing hyphen-marked intervals (for example 1, 3-5, 10). Port numbers must be > 0 and <= 65536. The server will use these ports to listen for incoming RMI service requests, in addition to listening on the ports defined by other properties. For outgoing RMI service requests, random port numbers may be used.

Property	Default value	Description
<code>com.ibm.di.default.bind.address</code>	*	The default bind address for the whole IBM Security Directory Integrator Server - the components and the Server API. * means bind to all available network interfaces. Missing or invalid value binds to all interfaces, too. Only one IP address should be provided as value. This property will affect all Server Connectors that create a Server Socket.
<code>api.remote.bind.address</code>	*	The bind address for the RMI Server API. Overrides the <code>com.ibm.di.default.bind.address</code> property. * means connect to all available network interfaces. Missing or empty value means fall back to <code>com.ibm.di.default.bind.address</code> . Only one IP address should be provided as value.
<code>api.truststore</code>	<code>testserver.jks</code>	If specified, the keystore file that contains the public certificates of all remote users of the Server API.
<code>{protect}-api.truststore.pass</code>	server (encrypted by default)	If specified, the password for the keystore file named through the <code>api.remote.server.truststore</code> property.
<code>api.remote.nonssl.hosts</code>		If specified, shows a list of IP addresses to accept non-SSL connections from (host names are not accepted). Use space, comma or semicolon as a delimiter between IP addresses. This property is only taken into account when <code>api.remote.ssl.on</code> is set to false .
<code>api.jmx.on</code>	false	If set to true , the JMX layer of the Server API is initialized on startup and can be used; otherwise, the JMX layer is not initialized and cannot be used.
<code>api.jmx.remote.on</code>	false	If set to true , the remote JMX interface (as defined by JSR160) is initialized and can be used; otherwise the remote JMX interface is not initialized and cannot be used.
<code>api.config.folder</code>	<code>configs</code>	If set to <code>TDI_root/configs</code> , only the configuration files placed in this folder can be edited through the Server API.
<code>api.config.lock.timeout</code>	0	If set to 0, there is no timeout.
<code>api.custom.method.invoke.on</code>	false	The ability to use <code>Session.invokeCustom()</code> methods can be turned on or off (the default is false, or off). If the value of this property is set to true then users can use these methods, otherwise an Exception will be thrown.
<code>api.custom.method.invoke.allowed.classes</code>		If specified, gives the list of classes that can be invoked directly by the Server API methods for custom method invocation (<code>Session.invokeCustom(...)</code>). This property is only taken into account if <code>api.custom.method.invoke.on</code> is set to true . The classes in this list must be separated by a space, a comma or a semicolon.
<code>api.custom.authentication.</code>	[ldap]	If specified, points to a JavaScript text file that contains custom authentication code. To enable the built-in LDAP/JAAS Authentication mechanism, set this property to [ldap]/[jaas].
<code>com.ibm.di.server.id</code>		If specified, contains the server ID. Assign a unique value for each server from the set of servers that are running on the same IP and Port.
<code>api.config.load.timeout</code>	2	If specified, contains the serverapi config load timeout value in minutes. Added in IBM Security Directory Integrator v7.0.
<code>api.notification.suppress</code>	<code>di.server.api.authenticate</code> <code>di.server.api.authorize.*</code>	If specified, gives a list of server notification types that you want to suppress. Notifications of suppressed types are not propagated by the notifications framework. Notification types in the list are separated by spaces. You can include wildcards. Example: <code>api.notification.suppress=di.al.*</code> <code>di.ci.start</code> The above example suppresses all AssemblyLine related notifications as well as notifications for starting a configuration instance. If the property is missing or is empty, no notifications are suppressed. Added in IBM Security Directory Integrator v7.0.
<code>api.client.ssl.custom.properties.on</code>	true	Enables custom SSL properties for Server API clients. If true, the <code>api.client.*</code> properties will be used by Server API clients. Otherwise the default <code>javax.net.ssl.*</code> properties will be used. Added in IBM Security Directory Integrator v7.1.
<code>api.client.keystore</code>	<code>serverapi/testadmin.jks</code>	Keystore for Server API clients.
<code>api.client.keystore.pass</code>	administrator	Password for the keystore specified by the "api.client.keystore" property.

Property	Default value	Description
<code>api.client.keystore.type</code>	jks	Type of the keystore file specified by <code>api.client.keystore</code> ; optional property, if not specified the default keystore format for the JVM will be used
<code>api.client.key.pass</code>	administrator	Password for the private key. The key is located in the keystore specified by the "api.client.keystore" property.
<code>api.client.truststore</code>	serverapi/testadmin.jks	Truststore for Server API clients.
<code>api.client.truststore.pass</code>	administrator	Password for the truststore specified by the "api.client.truststore" property.
<code>api.client.truststore.type</code>	jks	Type of the keystore file specified by <code>api.client.truststore</code> ; optional property, if not specified the default keystore format for the JVM will be used

Note: The Java system properties that the Server API uses for its configuration are the same, regardless of whether the client is a Java program or a different instance of the IBM Security Directory Integrator Server. What should be noted though is that the way these Java system properties are set might be different. In IBM Security Directory Integrator these properties are normally set by editing the `global.properties` or `solution.properties` files, whereas in a Java program they can be specified either at the command line using the `-D` Java command line switch or by using Java code within the Java program using the `java.lang.System.setProperty(key,value)` standard Java method.

Remote Server API access on a Virtual Private Network

You can learn to access Remote Server API from a client VPN through the information provided here.

When the Remote Server API is accessed from a client on a Virtual Private Network (VPN), the VPN assigns an IP address to the Server API client computer. This VPN-assigned IP address needs to be specified in an RMI Java system property. If the Server API client is the Remote Configuration Editor, then this property can be set in `global.properties` or `solution.properties` by adding the following line to the properties files:

```
java.rmi.server.hostname=<IP_address>
```

Where `IP_address` is the VPN-assigned IP address.

If the Server API client is a custom Java program, then this property can be set from within the Java code in the following way:

```
java.lang.System.setProperty("java.rmi.server.hostname", "IP_Address");
```

where `IP_address` is the VPN-assigned IP address.

Note that the RMI Java system property needs to be set before any Server API related RMI code.

Server API access options

You can learn different ways of accessing the server API through the information provided here.

The Server API can be used in a variety of ways:

- Access the Server API from the Remote Configuration Editor through a network connection

- Access the Server API from IBM Security Directory Integrator components running in a remote IBM Security Directory Integrator server (remote Server API access). Examples of such components are:
 - System Queue Connector
 - Server Notifications Connector
 and so on.
- Access the Server API from within the same Java Virtual Machine of the IBM Security Directory Integrator Server (local Server API access); in this case the Server API can be reached from JavaScript in hooks or from the Script Component in addition to the options above.
- Access the Server API from non-IBM Security Directory Integrator Java applications. For this to work:
 - Java 7.0.4 or higher is required on the client side.
 - The following jar files must be included in the CLASSPATH of the remote side:
 - jars/common/diserverapi.jar
 - jars/common/diserverapirmi.jar
 - jars/3rdparty/others/log4j-1.2.16.jar
 - jars/common/miconfig.jar
 - jars/common/miserver.jar
 - jars/common/mmconfig.jar
 - jars/common/tdiresource.jar
 - jars/3rdparty/IBM/icu4j-50_1_1.jar
 - jars/3rdparty/IBM/jlog.jar
 You can copy these jar files from the IBM Security Directory Integrator installation.
 - If custom non-IBM Security Directory Integrator objects are used in the solution being implemented with the Server API (for example as Attribute values of an Entry that is transferred over the wire) the corresponding Java classes have to be available on the client side as well. These classes must be serializable and they have to be included in the CLASSPATH of the client JVM.

Server API SSL remote access

The remote interface of Server API can use SSL. Learn more about the remote access through the information provided here.

The Server API provides two sets of interfaces – local and remote. It is only the remote interfaces that can use SSL. The local interfaces do not use SSL as the access is within the boundaries of the Java Virtual Machine. IBM Security Directory Integrator can act as a server, as a client; as well as both as a client and as a server in a Server API access scenario. When SSL is used with the Server API, then a keystore and a truststore must be configured. There are two options for configuring these. Which of them is used depends on whether the Java System property `api.client.ssl.custom.properties.on` exists and on its value.

Using Server API specific SSL properties

Use the listed set of properties to configure the SSL.

When the Java System property `api.client.ssl.custom.properties.on` is set to `true`, then SSL is configured through the following IBM Security Directory Integrator Server API-specific Java System properties:

- **`api.client.keystore`** – specifies the keystore file containing the client certificate
- **`api.client.keystore.pass`** – specifies the password of the keystore file specified by `api.client.keystore`
- **`api.client.keystore.type`** – specifies the type of the keystore file specified by `api.client.keystore`; optional property, if not specified the default keystore format for the JVM will be used
- **`api.client.key.pass`** – specifies the password of the private key stored in the keystore file contained in `api.client.keystore`; if this property is missing, the password specified by **`api.client.keystore.pass`** is used instead.
- **`api.client.truststore`** – specifies the keystore file containing the IBM Security Directory Integrator Server public certificate.
- **`api.client.truststore.pass`** – specifies the password for the keystore file specified by `api.client.truststore`.
- **`api.client.truststore.type`** – specifies the type of the keystore file specified by `api.client.truststore`; optional property, if not specified the default keystore format for the JVM will be used

Use the Server API specific SSL properties when your client application is using the standard Java SSL properties. The standard Java SSL properties are properties used to configure another SSL channel used by the same application.

You can specify these properties as JVM arguments on the command line, for example:

```
java MyTDIServerAPIClientApp
-Dapi.client.ssl.custom.properties.on=true
-Dapi.client.truststore=C:\TDI\serverapi\testadmin.jks
-Dapi.client.truststore.pass=administrator
-Dapi.client.keystore=C:\TDI\serverapi\testadmin.jks
-Dapi.client.keystore.pass=administrator
```

This example refers to the sample "testadmin.jks" keystore file shipped with IBM Security Directory Integrator. Note that it contains both the client private key and also the public key of the IBM Security Directory Integrator Server, so we use it both as a keystore and truststore.

You can specify these properties in `global.properties` or `solution.properties` when the client is an IBM Security Directory Integrator server.

Using the standard SSL Java System properties

Use the listed JVM commands to configure the SSL.

When the Java System property `api.client.ssl.custom.properties.on` is missing or when it is set to "false", then the standard JSSE system properties are used for configuring the SSL channel. Follow the standard JSSE procedure for configuring the keystore and truststore used by the client application.

You can specify these properties as JVM arguments on the command line, for example:

```
java MyTDIServerAPIClientApp
-Djavax.net.ssl.keyStore=C:\TDI\serverapi\testadmin.jks
-Djavax.net.ssl.keyStorePassword=administrator
-Djavax.net.ssl.trustStore=C:\TDI\serverapi\testadmin.jks
-Djavax.net.ssl.trustStorePassword=administrator
```

Also these properties can be specified in `global.properties` or `solution.properties` when the client is an IBM Security Directory Integrator server.

Server API authentication

You can use Server API authentication while trying to set a server API session.

Server API authentication is usually referred to in the context of a remote Server API client establishing a Server API session. This scenario represents the substance of the Server API authentication logic as the Server API provides several different kinds of client authentication. But before diving into the different authentication mechanisms let us discuss the scenario in which a local client establishes a local Server API session.

Local client session

Use the instructions provided here to work with a client locally. Read the information to know about the options.

A local client session is a session established by a client which runs in the same Java Virtual Machine as the IBM Security Directory Integrator server. Examples of such sessions are local sessions for access to the local Server API established from JavaScript code in hooks or in a Script component, from Connectors and Function Components which are executed as part of an AssemblyLine which runs in the same IBM Security Directory Integrator server, and so on. When a local client establishes a local Server API session, the client has two options:

- Do not provide a username and password pair – in this case the local Server API session is established and the client is authorized as having the "admin" role. For more information about Server API roles, see "Server API Authorization" on page 123.
- Provide a username and password pair – in this case the Server API session is established only after the "username" supplied in the username and password pair is authorized according to the Server API Authorization logic described in the "Server API Authorization" on page 123 section. This option would normally be used when a specific user ID is needed for authentication – for demos, prototyping, and so on.

Remote client session

Use the instructions provided here to workaroud with a client remotely. Read the information to know more about the types of authentications.

A remote client session is a session established by a client which does not run in the same Java Virtual Machine as the IBM Security Directory Integrator server. Examples of such sessions are sessions for access to a remote Server API established from the Configuration Editor, or a Java application wishing to access an IBM Security Directory Integrator Server. For access of this kind there are the following methods of authentication to the IBM Security Directory Integrator Server:

JAAS authentication

You can use JAAS authentication to validate the users access controls. Learn more about configuring the properties by reading the information provided here.

The Java Authentication and Authorization Service (JAAS) is supported as an authentication module for IBM Security Directory Integrator Server APIs. JAAS is a set of APIs that enables services to authenticate and enforce access controls upon

users. The JAAS authentication is facilitated by the IBM Security Directory Integrator Server API. No changes are required on the IBM Security Directory Integrator Server API clients such as CLI and AMC in order to use the JAAS authentication module.

In order to use JAAS authentication, you must configure the appropriate properties in `global.properties` or `solution.properties` and the JAAS Logon should be installed.

SSL-based authentication

You can use the two-stage verification of the client's credentials through the SSL-based authentication.

This is the only authentication mechanism available in IBM Security Directory Integrator 6.0. SSL-based authentication is based on a two-stage verification of the client's credentials.

1. First the IBM Security Directory Integrator server verifies that a client (represented by its SSL certificate) has the right to access the IBM Security Directory Integrator server by checking whether the client's SSL certificate is contained in the IBM Security Directory Integrator server's truststore, that is, checks whether the IBM Security Directory Integrator server trusts this client. Checking whether the client's certificate is contained in the server's truststore is part of the SSL handshake sequence.

Attention: A client certificate example, corresponding to the Server certificate example in file `testserver.jks` is provided in file `serverapi/testadmin.jks`; the certificate's password is "administrator". As with all default security parameters you should not rely upon these and generate your own client/server certificates and specify these in the properties files. See "Certificates for the IBM Security Directory Integrator Web service Suite" on page 148.

The truststore is kept in the file indicated by the `api.truststore` property.

2. If the truststore check is successful then the server verifies that the client SSL certificate distinguished name (DN) matches a user ID in the "Server API User Registry" on page 125. If the client certificate's DN does not match any of the user IDs in the Server API User registry file the connection request from the client is denied. This second step could be regarded as part of the authorization sequence as well.

The SSL-based authentication mechanism can be switched off in IBM Security Directory Integrator. An additional property is available in the IBM Security Directory Integrator Server configuration file `global.properties` or `solution.properties`: **`api.remote.ssl.client.auth.on`**. When this property is set to "true", the IBM Security Directory Integrator Server requires client authentication within the SSL handshake (the IBM Security Directory Integrator 6.0 mechanism for SSL-based authentication). SSL client authentication for IBM Security Directory Integrator Server API does not depend on whether a username and password pair is supplied. This means that if no username and password pair is supplied, the IBM Security Directory Integrator 6.0 mechanism for SSL-based authentication is used. And if a username and password pair is supplied then the client still needs to send its SSL certificate for authentication, but the User ID for authentication (and at a later step authorization) is taken from the username supplied.

When **`api.remote.ssl.client.auth.on`** is set to "false", SSL-based authentication cannot be used. When the property is not specified a value of "false" is assumed.

Username/password based authentication

You can use authentication hook to perform Username/password based authentication.

This mechanism requires a client to supply a username and password on the opening of his Server API connection to the IBM Security Directory Integrator server. In order to configure this authentication method an authentication hook is used.

Authentication hook

This hook allows the provision of custom JavaScript code that performs username and password based authentication. This hook allows bundlers/deployers to write customized JavaScript code, which given a username and password pair determines whether the authentication should succeed or not.

The property allowing for this custom JavaScript authentication is specified in the IBM Security Directory Integrator Server configuration file `global.properties` or `solution.properties`: **api.custom.authentication**. The `api.custom.authentication` property points to a JavaScript text file on the disk that contains custom authentication code. If this property is not specified then the IBM Security Directory Integrator 6.0 SSL-based authentication mechanism is used. When the `api.custom.authentication` property is specified, the JavaScript code contained in the specified file is executed for each username and password based authentication request.

The authentication script has access to the predefined script object `userdata`. This object provides the following two public members:

- **userdata.username** - contains the name of the user requesting authentication
- **userdata.password** - contains the password provided by the user

The script is free to perform whatever checks and authentication actions it needs. It returns whether the authentication is successful through the **ret** object:

- set **ret.auth = true** to specify that the authentication is successful
- set **ret.auth = false** to specify that the authentication is not successful; in this case the authentication script can provide additional information for why the authentication failed through the **ret.errordescr** attribute (for example `ret.errordescr = "Invalid user name"`) and **ret.errorcode** (for example `ret.errorcode = 1`).

The description and error code fields is provided by the `AuthenticationException` thrown by the `ServerAPI` on unsuccessful authentication.

The authentication script has access to the main script object. It can be used for logging custom messages in the IBM Security Directory Integrator Server log file (for example `main.logmsg("Authentication failed for user : " + userdata.username)`).

An example authentication hook

An example authentication hook JavaScript file is available (in `TDI_install_dir/examples`) in order to demonstrate what the JavaScript of an authentication hook could look like. This example JavaScript can also be used as the basis of real-world IBM Security Directory Integrator authentication hooks. The example JavaScript demonstrates how an authentication hook can use an LDAP server (IBM Security Directory Server, Active Directory, and so on) for authenticating client requests.

The JavaScript file is named "ldap_auth.js" and is installed in the examples/auth_ldap IBM Security Directory Integrator Server folder. To deploy this sample LDAP authentication mechanism users can copy that file to the IBM Security Directory Integrator solution folder and specify `api.custom.authentication=ldap_auth.js` in `global.properties` or `solution.properties`. The JavaScript code in "ldap_auth.js" tries to bind to an LDAP Server with the specified username and password. If the bind operation is successful, the script indicates a successful authentication, otherwise the authentication is rejected. The details for connecting to the LDAP Server like the server URL are specified in the "ldap_auth.js" script - this means that users have to edit this file and set the proper connection parameters before using the script. Here is the sample "ldap_auth.js" script:

```
env = new Packages.java.util.Hashtable();
env.put("java.naming.factory.initial", "com.sun.jndi.ldap.LdapCtxFactory");
env.put("java.naming.provider.url", "ldap://192.168.113.54:389");
env.put("java.naming.security.principal", userdata.username);
env.put("java.naming.security.credentials", userdata.password);
env.put(Packages.javax.naming.Context.SECURITY_AUTHENTICATION, "simple");

main.logmsg("Authentication request for user: " + userdata.username);

try
{
    mCtx = new Packages.javax.naming.directory.InitialDirContext(env);
    ret.auth = true;
}
catch(e)
{
    ret.auth = false;
    ret.errordescr = e.toString();
    // ret.errorcode = "49";
}
```

LDAP Authentication support

The IBM Security Directory Integrator Server API provides support for LDAP Authentication. This allows you to leverage your existing LDAP infrastructures that already hold User IDs and Passwords.

LDAP Authentication Configuration:

You can configure the LDAP authentication by working upon the listed properties.

In order to use LDAP authentication the appropriate properties must be configured in `global.properties` or `solution.properties`. The list of these properties along with their descriptions follows:

api.custom.authentication

This is the same property used for username and password authentication. For more information on username and password authentication see the "Username/password based authentication" section. This property points to a JavaScript text file on the disk that contains custom authentication code. The user may not specify this property, in which case he can only use the IBM Security Directory Integrator 6.0 SSL-based authentication mechanism. The IBM Security Directory Integrator Version 7.2 username and password authentication does not work. Set this property to "[ldap]" to enable the IBM Security Directory Integrator Version 7.2 built-in LDAP Authentication mechanism, like this: `api.custom.authentication=[ldap]` All properties starting with "`api.custom.authentication.ldap.`" are only be taken into account when `api.custom.authentication` is set to `[ldap]`.

api.custom.authentication.ldap.critical

This parameter specifies the Server API behavior when the LDAP

Authentication module cannot be initialized on startup. If this parameter is set to "true" the Server API initialization fails and the Server API is not started.

If this parameter is missing or is set to "false" the Server API logs the LDAP Authentication initialization error but the Server API is started. An attempt to initialize the LDAP Authentication module is made on each authentication request received by the Server API until the LDAP Authentication module is initialized.

api.custom.authentication.ldap.hostname

The LDAP Server hostname. If LDAP custom authentication is used, this is a required property.

api.custom.authentication.ldap.port

The LDAP Server port number. For example, 389 for non-SSL or 636 for SSL. If LDAP custom authentication is used, this is a required property.

api.custom.authentication.ldap.ssl

Specifies whether SSL is used to communicate with the LDAP Server. When set to "true" SSL is used, otherwise SSL is not used.

api.custom.authentication.ldap.searchbase

Specifies the LDAP directory location where user searches is preformed. When this property is not specified user searches is not performed.

api.custom.authentication.ldap.admin dn

Specifies an LDAP Server administrator distinguished name that is used for user searches. When this property is not specified anonymous bind is used for user searches.

api.custom.authentication.ldap.adminpassword

Password for the LDAP Server administrator distinguished name.

api.custom.authentication.ldap.userattribute

Specifies the user id attribute to be used in searches. When this property is not specified user searches are not performed. An example setting of this property would be: `api.custom.authentication.ldap.userattribute=cn`

If a required property is missing an exception is thrown on initialization.

If the value of either **api.custom.authentication.ldap.searchbase** or **api.custom.authentication.ldap.userattribute** is missing no search context is initialized and no searches is performed during the actual user authentication. (No search means that the bind to the LDAP Server is attempted directly with the username and password provided for authentication.)

When **api.custom.authentication.ldap.admin dn** is provided a search context is created using "simple" authentication. If an error occurs during the search context initialization, the initialization of the LDAP Authentication module fails and an exception is thrown.

When **api.custom.authentication.ldap.admin dn** is not provided a JNDI search context is created using JNDI "anonymous" bind.

Note: If the search context cannot be initialized using **api.custom.authentication.ldap.admin dn**, authentication fails directly - no anonymous bind is attempted.

LDAP Authentication Logic:

Use the listed paths to authenticate the credentials for LDAP authentication.

On each attempt to authenticate a user the LDAP Authentication module is passed the username and the password for the user to be authenticated. The following authentication paths are possible:

- Both **api.custom.authentication.ldap.searchbase** and **api.custom.authentication.ldap.userattribute properties** are specified :
 - If the username given for authentication ends with the value of the **api.custom.authentication.ldap.searchbase** property it is assumed that a full distinguished name is provided and no user search is performed. A bind to the LDAP Server is attempted directly with the username and password provided for authentication. If the bind succeeds the authentication is considered successful, otherwise the authentication is considered failed.
 - If the username does not end with the value of the **api.custom.authentication.ldap.searchbase** property, a search with a subtree search scope is executed against the search context created on initialization. The search query used is "*(<LDAPUserIDAttribute>=<username>)*" where *LDAPUserIDAttribute* is the value of the **api.custom.authentication.ldap.userattribute** property and *username* is the username given for authentication. If exactly one search result is returned, a bind to the LDAP Server is performed with the distinguished name of the returned entry and the password provided for authentication. The authentication succeeds only if the bind to the LDAP Server is successful. In all other cases it is considered that the authentication has failed. If multiple search results are returned, authentication fails.
- At least one of **api.custom.authentication.ldap.searchbase** or **api.custom.authentication.ldap.userattribute** properties is not specified.

In this case no searches are performed and a bind to the LDAP Server is attempted directly with the username and password provided for authentication. If the bind succeeds the authentication is considered successful, otherwise it is considered that the authentication failed.

LDAP Group Support:

You can set permissions in the User Registry for a group the same way as you would for a user. You can distinguish between users and groups through the listed properties.

To ease administration, IBM Security Directory Integrator allows permissions to be configured for groups the same way as they are configured for users. You can set permissions in the User Registry using exactly the same syntax as you would for a user. The fact is that the User Registry does not care whether a security entity is a group or a user. The distinction between users and groups is drawn during the authentication process.

Group membership is configured in the LDAP directory, against which IBM Security Directory Integrator authenticates users. If a user is a member of some LDAP group, all permissions for that group are automatically inherited by the user when the user is authenticated. Group support is disabled by default, so you must turn it on.

The system properties that are related to LDAP group support are:

api.custom.authentication.ldap.groupsupport

This is an optional property - a boolean flag. If this property is missing, the default value "false" is used. Specifies whether group membership is resolved when authenticating users. If the group membership is resolved, it is taken into account during authorization.

api.custom.authentication.ldap.usermembershipattribute

This property is required only if **api.custom.authentication.ldap.groupsupport** is set to true. Specifies the name of the attribute of a user in LDAP that contains a list of the groups of which the user is a member.

api.custom.authentication.ldap.usermembershipattributecontent

This property is required only if **api.custom.authentication.ldap.groupsupport** is set to true. Specifies how groups are named in the membership attribute of a user. For example, if the user's membership attribute contains values that correspond to the "objectSID" attributes of groups, set this property to "objectSID". If the user's membership attribute contains distinguished names of groups, then set this property to "dn".

api.custom.authentication.ldap.groupnameattribute

This property is required only if **api.custom.authentication.ldap.groupsupport** is set to true. Specifies the name of a group's attribute in LDAP which corresponds to the way the group is named in the IBM Security Directory Integrator User Registry. For example, if LDAP groups are addressed in the IBM Security Directory Integrator registry by their common name, then set this property to "cn". If the User Registry contains the distinguished names of the groups, then set this property to "dn".

api.custom.authentication.ldap.groupsearchbase

This property is required only if **api.custom.authentication.ldap.groupsupport** is set to true. Represents the LDAP directory context, where groups are searched.

api.custom.authentication.ldap.binaryattributes

This is an optional property - it represents a list of space-separated attribute names. Specifies attributes which have non-string syntax.

Active Directory example

This example shows how to configure group support to work with an **Active Directory** server:

```
api.custom.authentication.ldap.groupsupport=true
api.custom.authentication.ldap.usermembershipattribute=tokenGroups
api.custom.authentication.ldap.usermembershipattributecontent=objectSID
api.custom.authentication.ldap.groupnameattribute=SAMAccountName
api.custom.authentication.ldap.groupsearchbase=DC=mytestadsrver,DC=com
api.custom.authentication.ldap.binaryattributes=objectSID tokenGroups
```

The 'tokenGroups' attribute is a calculated attribute that exists for all users in Active Directory.

It contains a collection of the Security Identifiers (SIDs) for all security groups that the user is a member of.

This collection contains only security groups (distribution groups, used for e-mail, are not included) and it contains all security groups including nested and primary groups.

The Security Identifiers are binary attributes so they must be set in the `api.custom.authentication.ldap.binaryattributes` property.

In the above example, groups are named by their "sAMAccountName" LDAP attribute in the IBM Security Directory Integrator User Registry.

IBM Security Directory Server example

This example shows how to configure group support to work with IBM Security Directory Server:

```
api.custom.authentication.ldap.groupsupport=true
api.custom.authentication.ldap.usermembershipattribute=ibm-allGroups
api.custom.authentication.ldap.usermembershipattributecontent=dn
api.custom.authentication.ldap.groupnameattribute=dn
api.custom.authentication.ldap.groupsearchbase=ou=mytestou,c=mytestcountry
```

For a given user entry, the "ibm-allGroups" operational attribute enumerates all static, dynamic and nested groups, to which that user has membership.

Note:

1. IBM Security Directory Integrator determines group membership by directly examining the LDAP user entry (as opposed to indirectly determining membership by scanning through all groups). For this approach to work correctly, the user entry must have an attribute that enumerates the groups, of which the user is a member. The group support works only with LDAP Servers that do support such a membership attribute on each user entry.
2. If you modify the group membership of a user, this does not affect existing Server API sessions. It is, however, reflected in sessions established after the modification.
3. Group support is currently provided only for LDAP authentication. There is no group support for JAAS authentication or authentication with custom JavaScript.
4. When SSL client authentication is enabled in the Server API, clients that do not specify a username are to be authenticated and authorized based on the owner of the SSL client certificate. If LDAP authentication with group support is also enabled (along with the SSL client authentication), group membership is resolved for the owner of the SSL client certificate.

Host based authentication

Configure the host based properties in order to use the host based authentication. Learn more about it through the information provided here.

Host based authentication is used, when SSL is turned off by specifying `api.remote.ssl.on=false` in `global.properties` or `solution.properties` files. Host based authentication is configured using the `api.remote.nonssl.hosts` property. This property specifies the list of host IP addresses from which remote Server API clients can use the Server API without specifying a username and password.

The syntax of this list of hosts is: a list of IP addresses (host names are not accepted); use a space, a comma or a semicolon as a delimiter between IP addresses. An example value of this property would be:

```
api.remote.nonssl.hosts=192.168.111.222, 192.168.112.158
```

When a client using host based authentication is successfully authenticated, then the client is granted admin authorization authority. That is why adding IP addresses to this list must be done with great care. It is not advisable to use host

based authentication in production environment because of its security issues. Host based authentication would normally be used while developing a solution or when doing a demo.

Summary of Server API Authentication options

There are a number of server API authentication options. Here is a summarized list of the same.

The following authentication options are available:

SSL-based authentication (the mechanism available in IBM Security Directory Integrator 6.0)

Only works when *api.remote.ssl.client.auth.on=true* (you also need *api.on=true*, *api.remote.on=true*, *api.remote.ssl.on=true*). The user is authorized by the rights assigned to the SSL certificate user ID in the Server API User Registry.

Note: When SSL is used and the remote client application uses Server API listener objects, then the client application must have its own certificate that is trusted by the IBM Security Directory Integrator Server (this is analogous to the setup for SSL client authentication). If there is no client certificate trusted by the IBM Security Directory Integrator Server, the listener objects do not work and the remote client application cannot receive notifications from the IBM Security Directory Integrator Server.

Username/password based authentication

Only works when *api.custom.authentication* is set to a JavaScript authentication file. This authentication method works regardless of whether SSL is used and whether SSL client authentication is used. The user is authorized as per the rights assigned to the *username* user in the “Server API User Registry” on page 125.

LDAP authentication

This was described in section “LDAP Authentication support” on page 116, and is dependent on a number of *api.custom.authentication* settings in the *global.properties* or *solution.properties*.

Host-based authentication

Only works when *api.remote.ssl.on=false*. Then opening of Server API sessions without username and password supplied from all hosts specified by the *api.remote.nonssl.hosts* property are successfully authenticated and granted admin authority. The *api.remote.nonssl.hosts* property can be specified in the *global.properties* or *solution.properties*.

Server API JMX layer

Server API JMX layer does not support username and password authentication. Use the authentication steps listed here to authenticate the JMX layer.

The remote JMX layer of the Server API does not support username and password based authentication. It ignores the *api.custom.authentication* properties. Regardless of the value of these properties and whether custom authentication is enabled or not for the Server API, the remote JMX layer performs the following authentication:

- If SSL is turned on and SSL client authentication is turned on, the remote JMX layer performs SSL-based authentication (as in IBM Security Directory Integrator 6.0).
- If SSL is turned on and SSL client authentication is turned off, the remote JMX layer does not work.

- If SSL is turned off, the remote JMX client is successfully authenticated only if its host is specified on the `api.remote.nonssl.hosts` property, that is, host-based authentication is assumed. In this case the client is granted admin authority.

The net result is that the Server API JMX layer does not support username and password authentication:

Server API authentication setup examples

You can go through a list of Server API authentication examples. This will help you in configuring the server.

Authentication configuration examples:

1. Non-SSL configuration and custom authentication:

```
api.remote.ssl.on=false
api.remote.nonssl.hosts=192.168.113.51, 192.168.113.52
api.custom.authentication=ldap_auth.js
```

SSL is not used.

- Authentication requests with no username and password supplied succeed only if they are invoked from the localhost or from 192.168.113.51 or 192.168.113.52.
- Authentication requests with username and password supplied succeed only if the `ldap_auth.js` successfully authenticates the user specified with the username and password parameters.
- Remote JMX clients are authenticated only when the request comes from the localhost or from 192.168.113.51 or 192.168.113.52.

2. SSL (without client authentication) and custom authentication:

```
api.remote.ssl.on=true
api.remote.ssl.client.auth.on=false
api.custom.authentication=ldap_auth.js
```

SSL is used for remote Server API communication.

- Authentication requests with no username and password supplied fail because neither SSL client authentication, nor host-based authentication is switched on.
- Authentication requests with username and password supplied succeed only if the `ldap_auth.js` successfully authenticates the user specified with the username and password parameters.
- Host-based authentication is not available in this case regardless of the value of the `api.remote.nonssl.hosts` parameter, because `api.remote.ssl.on` is set to true.
- Remote JMX layer is not accessible. This is because SSL is turned on but SSL client authentication is not used.

3. SSL with client authentication and custom authentication:

```
api.remote.ssl.on=true
api.remote.ssl.client.auth.on=true
api.custom.authentication=ldap_auth.js
```

SSL is used for remote Server API communication and the Server requires SSL client authentication.

- Authentication requests with no username and password supplied succeed when the SSL certificate of the client is present in the Server's truststore (or verifiable using the certificates in the truststore).
- Authentication requests with username and password supplied succeed only when the SSL client authentication is successful (the SSL certificate of the

client is present in the Server's truststore) and the `ldap_auth.js` script successfully authenticates the user specified with the username and password parameters. In this case, authorization is performed based on the username parameter from the username and password supplied and not with the user identity from the SSL client certificate.

- Host-based authentication is not available in this case regardless of the value of the `api.remote.nonssl.hosts` parameter, because `api.remote.ssl.on` is set to true.
- Remote JMX clients are authenticated when the SSL certificate of the client is present in the Server's truststore (or verifiable using the certificates in the truststore).

4. SSL with client authentication & no custom authentication:

```
api.remote.ssl.on=true
api.remote.ssl.client.auth.on=true
api.custom.authentication=
```

(as an alternative, the "`api.custom.authentication`" property may be missing entirely)

SSL is used for remote Server API communication and the Server requires SSL client authentication.

- Authentication requests with no username and password supplied succeed when the SSL certificate of the client is present in the Server's truststore (or verifiable using the certificates in the truststore).
- Authentication requests with username and password supplied do not succeed because custom authentication is not configured.
- Host-based authentication is not available in this case regardless of the value of the `api.remote.nonssl.hosts` parameter, because `api.remote.ssl.on` is set to true.
- Remote JMX clients are authenticated successfully only when the SSL certificate of the client is present in the Server's truststore.

Server API Authorization

You can assign authorization to a user with the instructions provided here.

After a client Server API session request is authenticated it needs to be authorized.

Users of the Remote API can be assigned several roles; a role defines a list of Server API calls that can be executed by the user and also defines in what context these calls can be executed. A Server API method can be executed if there is at least one role assigned to the user that allows the execution of this method in the context the user tries to execute it. For example, a role can grant the user rights to execute only specific `AssemblyLines` from a specific configuration. Refer to "Server API User Registry" on page 125 for details on how to create the file that holds these user rights.

Authorization is based on the user id. Depending on the authentication mechanism used the user id is retrieved in a different way:

- SSL based authentication - the user id is the distinguished name (DN) of the client's SSL certificate.
- Username or password based authentication - the user id is the username supplied in the username and password pair.
- Host based authentication - no user id can be retrieved from the client using this authentication mechanism; in this case the client session is authorized with the `admin` role.

Authorization roles

You can authorize a user with several roles. Here is list of roles that can be applied to the Server API security model:

Users of the Remote API are assigned roles; a role defines a list of Server API calls that can be executed by the user and also defines in what context these calls can be invoked. For example, a role can grant the user rights to invoke only specific AssemblyLines from a specific configuration.

Several roles can be assigned to a user, including assigning the same role several times with different parameters. A Server API method can be invoked if there is at least one role assigned to the user that allows the execution of this method in the context the user tries to execute it.

There are no deny semantics - actions cannot be explicitly forbidden. The following roles apply to the Server API security model:

<p><i>Read</i> role: read [list_of(configuration)]</p>	<p>The <i>read</i> role allows the user read data from the Server's configuration(s).</p> <p>If no list of configurations is specified or the list is empty, the user is not allowed to read any configuration.</p> <p>A special value * (asterisk) can be specified for the list of configurations and this means that the user is allowed to read (through Server API calls) all configurations currently loaded by the Server.</p> <p>When the list of configurations is not null/empty and does not specify * the user is allowed to read only the configurations specified.</p> <p>The <i>read</i> role does not grant permission to start processes (AssemblyLines) or apply any changes to the Server and its configurations. For example:</p> <pre>[ROLE]:read [CONFIG]:*</pre>
<p><i>Execute</i> role: execute [list_of(configuration [list_of(AssemblyLines)])]</p>	<p>The <i>execute</i> role gives the user permissions to execute AssemblyLines.</p> <p>If no list of configurations is specified or the list is empty, the user is not allowed to execute any AssemblyLine from any configuration.</p> <p>A special value * (asterisk) can be specified for the list of configurations and this means that the user is allowed to execute all AssemblyLines from all configurations.</p> <p>When the list of configurations is present and does not specify * the user is only allowed to start the processes from the configurations specified in the list. For each configuration specified in the list:</p> <ul style="list-style-type: none"> • If a list of AssemblyLines is not specified, the user is not allowed to execute any AssemblyLine from this configuration. • If a special value * (asterisk) is specified for the list of AssemblyLines, the user is allowed to execute all AssemblyLines from this configuration. • If the list of AssemblyLines is present and does not specify * the user is allowed to execute only the AssemblyLines specified in the list. <p>For example:</p> <pre>[ROLE]:execute [CONFIG]:C:/TDI/rs.xml [AL]:* [CONFIG]:C:/TDI/prototype.xml [AL]:TestAssemblyLine</pre>

<i>Admin</i> role: admin	<p>The <i>admin</i> role allows the user to execute all Server API calls in every possible context.</p> <p>A user with <i>admin</i> role is allowed to read and modify configurations, to load new configurations, to execute AssemblyLines, to read and modify server parameters.</p> <p>For example: [ROLE]:admin</p> <p>Note:</p> <p>Admin role is required to use the Remote Configuration Editor. Also see “Using the Remote Configuration Editor” on page 143.</p>
---------------------------------	---

The values specified in a [CONFIG] tag can be either Config file names, or Solution Names if they have been specified in the Config file.

Server API User Registry

The User registry file encrypts the server certificate. Know more about its structure through the information provided here.

The User Registry, identified by the *api.user.registry* property in the *global.properties* or *solution.properties* file is a text file that maintains the information about all the users of the API and their roles. This file is encrypted with the Server's certificate specified by the *api.key.alias* property from the keystore specified by the *api.keystore* property. The encryption algorithm employed is Asymmetric RSA encryption or decryption; that is why the “Certificates for the IBM Security Directory Integrator Web service Suite” on page 148 specifying the RSA algorithm, which is the default algorithm of the “The IBM Security Directory Integrator Encryption utility” on page 136 provided with BM Security Directory Integrator that you can use for this purpose. On startup, the Server API engine decrypts and reads this file into its memory structures.

Note:

1. The entire user registry file is encrypted as it is, block by block, in a straightforward manner using the RSA algorithm and the server public key. A digital signature or some sort of hashing is not used.
2. The authorization against the user registry is not optional. Currently the BM Security Directory Integrator Server has no concept of a plug-in authorization mechanism.

The contents of the Identity Registry text file is structured as follows:

```
[USER]
[ID]:<user_identifier>
[ROLE]:<role_identifier>
  [CONFIG]:<config_identifier>
    [AL]:<assembly_line_name>
    [AL]:<assembly_line_name>
    ...
  [CONFIG]:<config_id>
  ...
[ROLE]:<role_identifier>
  ...
[ROLE]:<role_identifier>
  ...
[ENDUSER]

[USER]
[ID]:<user_identifier>
```

```
[ROLE]:<role_identifier>
...
[ENDUSER]
...
```

Each tag must span a single line and each tag must be on a separate line. Tabs and spaces do not matter. Empty lines may appear anywhere. The tags in the Identity Registry file and their arguments are as follows:

Tag	Argument
[USER]	This tag takes no arguments, and serves as an opening bracket for the tags below; a [USER] and [ENDUSER] pair of tags, each placed on a single line, provide the definition of a single user in the registry file. There can be multiple pairs of this type, each of which specify a user of the Server API.
[ID]:<user_identifier>	This tag is the first tag after the [USER] tag and its argument <user_identifier> is the unique identifier of the user of the Server API. This ID value is the 118 from the truststore file. The tag and the argument of the tag are placed on a single line, and there can be only one [ID]: tag included in a [USER] and [ENDUSER] pair.
[ROLE]:<role_identifier>	This tag specifies a role for the user. Possible roles are: read , execute or admin . Everything after the [ROLE]: tag and its argument and before another [ROLE]: tag or an [ENDUSER] tag (whichever comes first) specifies details of this user role. The tag and the argument of the tag are placed on a single line, and there can be multiple [ROLE]: tags included in a [USER] and [ENDUSER] pair, specifying multiple roles for that user.
[CONFIG]:<config_id>	This tag specifies the identifier of an IBM Security Directory Integrator configuration, the absolute file path of the configuration. Relative file paths are not recognized. This tag is subordinate to a [ROLE]: tag, and the tag specifies a configuration for the role given by the [ROLE]: tag. This tag and its argument are placed on a single line, and there can be multiple [CONFIG]: tags, all belonging to the superior [ROLE]: tag. If no [CONFIG]: tag is associated with a [ROLE]: tag, the list of configurations for the corresponding role definition is empty.
[AL]:<assembly_line_name>	This tag specifies an AssemblyLine name. This tag is subordinate to a [CONFIG]: tag. The tag and its argument are placed on a single line, and there can be multiple [AL]: tags, all belonging to the superior [CONFIG]: tag. If no [AL]: tag is associated with a [CONFIG]: tag, the list of AssemblyLines for the corresponding configuration ID is empty.

The following text is an example of an Identity Registry file:

```
USER]
[ID]:CN=Stan, OU=TDI, O=IBM, C=US
[ROLE]:admin
[ENDUSER]

[USER]
[ID]:CN=John, OU=TDI, O=IBM, C=US
[ROLE]:read
[CONFIG]:*
[ROLE]:execute
[CONFIG]:C:/TDI/rs.xml
[AL]:*
[CONFIG]:C:/TDI/prototype.xml
[AL]:TestAssemblyLine
[ENDUSER]

[USER]
[ID]:CN=Peter, OU=TDI, O=IBM, C=US
[ROLE]:execute
[CONFIG]:C:/TDI/rs.xml
[AL]:*
[ENDUSER]
```

This set of Identity Registry entries implies the following constraints:

- "Stan" is an administrator according to this registry file, and is allowed to perform each and every Server API operation.
- John is allowed to read all configurations loaded on the Server, but can only execute processes from two configurations:
 - From "rs.xml", John can execute all AssemblyLines.
 - From "prototype.xml" John is only allowed to execute the AssemblyLine named "TestAssemblyLine".
- Peter can only execute all AssemblyLines from the "rs.xml" configuration.

Note: The **keytool** and/or the **Ikeyman** utility can be used to obtain the user ID from the truststore file. The following command line prints all users from the truststore file:

```
keytool -v -list -keystore <trust_store_file> -storepass <trust_store_pass>
```

where <trust_store_file> is the keystore file that contains the certificates of all trusted users and <trust_store_pass> is the password for this keystore file. This command line prints something like the text below for each user certificate:

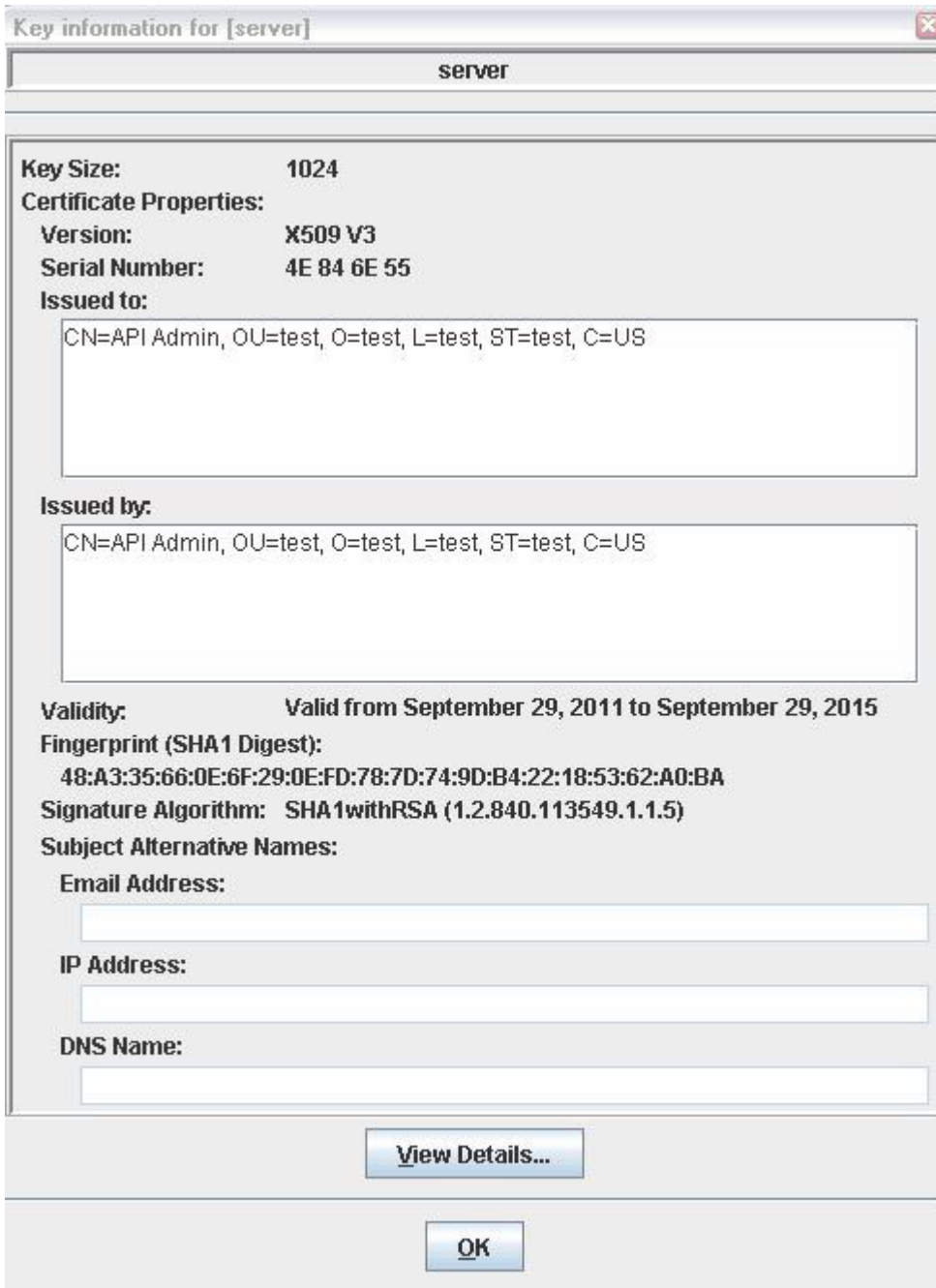
```
Owner: CN=Stan, OU=TDI, O=IBM, C=US
Issuer: CN=Stan, OU=TDI, O=IBM, C=US
Serial number: 408f6a34
Valid from: 4/28/04 11:24 AM until: 7/27/04 11:24 AM
Certificate fingerprints:
    MD5:  F6:EF:81:8B:4C:0F:10:E4:A0:16:99:AB:42:29:70:8B
    SHA1: FE:37:62:8B:42:2F:54:F8:F6:F3:FC:A1:DD:7D:2A:51:9A:85:09:02
```

The value of the **Owner** field **must** be specified as value for the [ID]: tag in the Identity Registry as is, including all white space and commas. For this example, the line with the ID tag looks like:

```
[ID]:CN=Stan, OU=TDI, O=IBM, C=US
```

An alternative way to obtain the user ID from the truststore file is to use Ikeyman in the following way:

1. Start Ikeyman (or select **Key Manager** from the toolbar).
2. From the **Key Database File** menu click **Open...**
3. In the **Open** field, set the appropriate values and click **OK**.
4. In the **Password** field, enter the password for the truststore file.
5. Click on the certificate you are interested in.
6. Click the **View/Edit...** button. A window opens which contains information on the subject's DN (user ID).



Server Audit Capabilities

You can audit IBM Security Directory Integrator events. Notifications are created for each event. You can know more about the audit capabilities through the information provided here.

The IBM Security Directory Integrator Audit Component enables the IBM Security Directory Integrator Server to audit events such as authentication and authorization in the Server API.

Notifications are generated when authentication and authorization (auth*) events occur. Audit data is packaged into an Entry and provided as user data in the notification. The "Audit Service" consists of a separate Audit config that is

automatically loaded by the IBM Security Directory Integrator server. The Audit config contains auto-started Audit AssemblyLines. The Audit ALs iterate on the notification connector using suitable filters. IBM Security Directory Integrator users can even generate "user defined notifications" if they want to create audit events from within their own code.

IBM Security Directory Integrator auditing contains two main parts:

- A way for generating the necessary audit information
- An "Audit service" for handling existing audit data

Generating necessary audit information is implemented by creating IBM Security Directory Integrator Entries on each audit point in the Server API, and by broadcasting these Entries wrapped in a notification. For this purpose a new class is presented in the Server API (com.ibm.di.api.APIAuditor), that generates the Entry, attaches the Entry as UserData to a notification, and sends it to all interested listeners.

The "Audit Service" is the main consumer of the audit notifications. The Audit Service is a config consisting of several ALs that iterate on the Notification Connector. Using different filters can register to a variety of notification types.

Auditing scope

Only those events can be considered for auditing which pass the listed criteria.

The IBM Security Directory Integrator audit capability follows only what people do, and does not follow Server events in general. There is a difference between a user being authorized to perform a task (stop an AL) and the task actually being performed (AL is terminated). Being authorized is an authorization event and the performing of a legal action, like stopping an AL, is a Server event. When a user instructs an AL to stop and the AL terminates, an authorization event is paired with a Server event. At other times, a Server event occurs by itself, as when an AL completes naturally. Only events which involve direct user interaction are audited. This limits the default audit points to authentication and authorization events inside the Server API. Almost every method exposed by the Server API is protected by its own piece of authorization code. The Audit component does not try to send notifications for all authorization events, but selects a reasonable subset of authorization-guarded Server API methods. The principles for the selection are to audit all events that:

- Delete logs or tombstones
- Start or stop IBM Security Directory Integrator entities such as configs, ALs, and the Server
- Replace the config instance configuration: replace the config instance configuration or the check-in configuration
- Allow the user to change vital IBM Security Directory Integrator data: set external property, post a message in the System Queue, call custom Java code inside the IBM Security Directory Integrator JVM

Suppression of notifications

You can know more about suppression of notifications, commands to generate the suppressed event types, method to do the same through the information provided here.

The IBM Security Directory Integrator Server API allows certain notification types to be suppressed for improved performance. The notification framework does not propagate suppressed events. If you try to broadcast an event of a type *suppressed*,

the Server API does not issue an error. However, the suppressed event cannot reach any of the registered notification listeners. The list of suppressed event types is configured by a system property named:

```
api.notification.suppress
```

By default, all authentication and authorization events are suppressed:

```
api.notification.suppress=di.server.api.authenticate di.server.api.authorize*
```

The event types in the list are separated by spaces. Wildcards matching multiple event types are allowed. If the event type property is missing or is empty, no events are suppressed. You can suppress all custom notifications by typing:

```
api.notification.suppress=user
```

Note: Suppression affects the whole IBM Security Directory Integrator Server and can result in suppression of all kinds of notifications. Even built-in notifications, such as an AssemblyLine starting or the Server shutting down, can be suppressed. Improper use of the suppression capabilities can interfere with the work of components that listen for notifications such as the Tombstone Manager and the Server Notifications Connector.

Sending notifications

You can deliver a notification to every registered Listener. Here is a list of notification delivery parameters.

Sending notifications uses a method in `com.ibm.di.api.APIEngine`:

```
public static void sendNotification
    (String type, String id, Object data, String configInstanceId)
```

This method creates a `DIEvent`. By this means, a notification is delivered to every Listener registered to receive the a particular type of notification. Notification delivery parameters include:

Table 18. Notification Delivery Parameters

Parameter name	Definition
<code>type</code>	Notification event type.
<code>id</code>	Notification event ID.
<code>data</code>	Notification event <code>UserData</code> object in the form of a Java object with additional information.
<code>configInstanceId</code>	Notification <code>ConfigInstance</code> ID to which the notification is bound.

The `com.ibm.di.api.APIEngine` method throws `DIEException` if the type parameter is null. Calls to the method can be invoked either:

- Locally, from the IBM Security Directory Integrator Server JVM. This type of access includes scripting in AssemblyLine hooks and also uses the API from new Connectors implemented in Java and deployed on the IBM Security Directory IntegratorServer
- Remotely, from another JVM (on the local or a remote network computer), through Remote Method Invocation (RMI). This type of access uses solutions that:
 - Connect remotely to IBM Security Directory Integrator
 - Manage processes within IBM Security Directory Integrator
 - Build business logic on top of IBM Security Directory Integrator
 - Are applications dedicated only to IBM Security Directory Integrator
 - Are applications that use IBM Security Directory Integrator to accomplish some of their goals

IBM Security Directory Integrator Server Instance Security

You can set up a server instance with help of encryption algorithms and the miscellaneous configuration files. Learn more about it through the information provided here.

This section does not deal with the specifics of client (IBM Security Directory Integrator-based or other) access to an IBM Security Directory Integrator Server, this is discussed in “Remote Server API” on page 107; instead, it focuses on the encryption algorithms used, and the miscellaneous configuration files needed to set up a server instance.

The IBM Security Directory Integrator Server requires a keystore containing both its private key and associated certificate/public key that is used for PKI encryption of Config Files, properties in Properties files, Server User registry files and other objects, as well as being used for SSL communication.

The system properties *api.keystore* and *api.key.alias* specify the keystore and the key alias of the Server's certificate/key within the keystore. The password of the keystore and the password of the key itself (if different from the keystore password) are specified in the Server's stash file. Access to a keystore is guarded by a password, defined at the time the keystore is created, by the person who creates the keystore, and changeable only when providing the current password. In addition, each private key in a keystore can be guarded by its own password. For more information on the stash file of the server, see section “Stash File.”

The RSA algorithm is used for encryption of files and property values. It is used as a block cipher where the block size is determined by the modulus component of the RSA key. Encryption is done in ECB (Electronic Codebook) mode. PKCS#1 Padding is applied separately on each block. Note that the same RSA key-pair, which is used for encryption of files, is also used for SSL communication with the Server. IBM Security Directory Integrator uses the RSA implementation from the IBMJCE security provider. All key sizes supported by that provider are also supported by IBM Security Directory Integrator. From IBM Security Directory Integrator v7.0, secret key ciphers can also be employed for encryption. RSA is used as the default for compatibility with earlier versions, but secret key ciphers are much faster and much more secure than public key ciphers.

DES and AES algorithms are used for encryption of password-protected configuration files. An encryption key (DES or AES) is derived from the UTF-8 binary representation of the password. The derived encryption key is 64 bit for DES and 128 bit for AES. ECB mode is used with no padding.

DES/AES keys are derived from passwords, when using password-protected configuration files. Apart from the above, the IBM Security Directory Integrator does not generate keys. Existing keys are loaded from an external key store. Key establishment and key store access are performed through the IBMJCE and IBMJSSE2 security providers. All key sizes and algorithms supported by those providers can be used with the IBM Security Directory Integrator.

Stash File

Stash file stores the password of the keystore and the password of the key itself. Learn to work with a stash file through the information provided here.

The stash file contains the Server keystore password values encrypted with AES128 with a fixed key. The Server stash file is named "idisrv.sth" (the name is not configurable) and it is loaded by the Server from the Solution Folder. A command line utility for creating a stash file is available in the IBM Security Directory Integrator bin folder: *createtash.bat* or *createtash.sh*:

```
createtash <keyStorePassword> [<keyPassword>] [<securityProviderClass>]]
```

where *keyStorePassword* is the password of the keystore file specified by the *api.keystore* system property and *<keyPassword>* is the password of the Server's private key specified by the *api.key.alias* system property.

keyPassword is an optional parameter if no *<securityProviderClass>* parameter is specified. If *<keyPassword>* is not specified it is assumed that the Server's private key password is the same as the keystore's password. To use the utility with the *<securityProviderClass>* parameter, you must specify both previous parameters: *keyStorePassword* and *keyPassword*. If a security provider is specified then this provider is used for the cryptography.

The utility creates a stash file named "idisrv.sth" with the specified password(s) in the current directory.

Attention: IBM Security Directory Integrator comes bundled with a sample stash file, with a password of "server". For improved security, we strongly advise you to generate your own stash file using the aforementioned utility. Also, the stash file must be kept inaccessible, except for the actual IBM Security Directory Integrator Server that needs it.

Server Security Modes

You can run the IBM Security Directory Integrator Server in two modes: **standard** and **secure**. Learn more about these through the information provided here.

Standard mode

When run in standard mode, the Server does not PKI encrypt configurations saved on disk, unless a specific Server API call that requests PKI encryption is invoked. When in this mode the Server is able to read both encrypted and unencrypted configurations.

Secure mode

When run in secure mode the Server encrypts all configurations it saves on the disk using PKI encryption. In secure mode the Server can only read and load encrypted configurations. When the system property *com.ibm.di.server.securemode* is set to "true", the Server runs in secure mode. (A system property for the use of the IBM Security Directory Integrator Server can be set by adding it in the *global.properties* or *solution.properties* file or directly specify it on the java command line when starting the IBM Security Directory Integrator server:
-Dcom.ibm.di.server.securemode=true)

If the command line option *-e* is specified on the java command line when starting the Server, it runs in secure mode regardless of the value of the *com.ibm.di.server.securemode* system property.

Note: Pre-IBM Security Directory Integrator 6.0 password-based encryption of configuration files is supported for compatibility with earlier versions. Password-based encryption is used when the user specifies a password when

creating the configuration. Pre-IBM Security Directory Integrator 6.0 password-based configuration encryption cannot be combined with PKI encryption. If you specify a password when the Server is run in secure mode, an error message is displayed.

Working with encrypted IBM Security Directory Integrator configuration files

You can perform a cryptographic transformation on the configuration files. You can know more about the usage of encrypted files and things to take care with the information provided here.

To provide confidentiality of data, IBM Security Directory Integrator can encrypt configuration files, property values in properties files, server user registry files and JavaScript files.

IBM Security Directory Integrator encryption involves a cryptographic transformation that uses a key or a key-pair. The key/key-pair needs to be hosted in a keystore file.

The cryptographic transformation can be either public-key encryption or secret key encryption. By default IBM Security Directory Integrator uses public key encryption. (The secret key encryption option has been introduced in IBM Security Directory Integrator 7.0. Before that only public key encryption was supported.)

See:

Public key encryption uses a key-pair that consists of a public key and a private key. The public key is used for encryption and the private key is used for decryption. Currently only the RSA cipher is supported for public key encryption. Public/private key pairs can be generated and managed using the standard JRE utilities `keytool` and `Ikeyman`. See “Manage keys, certificates and keystores” on page 93 for more information on managing certificates with associated public and private keys.

IBM Security Directory Integrator data encryption is configured by the following system properties (these can be set in `global.properties` or `solution.properties`):

- `com.ibm.di.server.encryption.keystore` : the keystore file that contains the key/key-pair for encryption
- `com.ibm.di.server.encryption.keystoretype` : the type of the keystore file
- `com.ibm.di.server.encryption.key.alias` : the alias of the key/key-pair in the keystore
- `com.ibm.di.server.encryption.transformation` : the name of the encryption transformation; see remarks below

The password of the keystore and the password of the key/key-pair itself (if different from the keystore password) are specified in the Server's “Stash File” on page 131. (Access to a keystore is guarded by a password, defined at the time the keystore is created, by the person who creates the keystore, and changeable only when providing the current password. In addition, each private key in a keystore can be guarded by its own password.)

The name of the transformation can be either RSA or some secret key transformation (for example, AES/CBC/PKCS5Padding). More detailed discussion of what is in a transformation name can be found at <http://www.ibm.com/developerworks/java/jdk/security/60/secguides/JceDocs/>

api_users_guide.html#trans; general information about Java Security (which is what IBM Security Directory Integrator uses) can be found at <http://www-128.ibm.com/developerworks/java/jdk/security/60/secguides/jsse2Docs/JSSE2RefGuide.html>.

Note:

1. The "com.ibm.di.server.encryption.*" properties affect not only encryption of configurations, but also encryption of property files, JavaScript files and the Server API User Registry.
2. If you change the encryption key and/or the encryption transformation, the Server cannot decipher previously encrypted files. To work around this problem, decrypt the old files with the old key (you must have the old key available in order to do so) and encrypt them with the new key. Encryption and decryption of files can be done with the cryptoutils tool.
3. The standard RSA algorithm has a restriction on the length of data it can work on. IBM Security Directory Integrator uses a custom scheme that splits input data into small enough equally sized blocks and encrypts each of them separately.
4. Data encrypted with RSA result in different cipher-texts on different encryption runs. This effect is a feature of the PKCS#1 padding scheme used with RSA.
5. A secret key (symmetric) cipher can be either a block cipher or a stream cipher. Stream ciphers encrypt the bits of the message one at a time, and block ciphers take a number of bits and encrypt them as a single unit (a block). Block ciphers (for example, AES) use a feedback mode (so that patterns in the plain-text are not preserved in cipher-text) and a padding scheme (to allow encryption of data, whose length is not multiple of the block size of the cipher). Stream ciphers (for example, RC4) do not use a feedback mode and a padding scheme.
6. If the transformation involves a block cipher, it must use some padding scheme (for example, "PKCS5Padding"), otherwise the Server not be able to encrypt data whose length is not multiple of the block size of the cipher. (Stream ciphers do not use padding, so they are not affected by this restriction.)
7. The algorithm of the key/key-pair must match the algorithm in the specified transformation. For example if the transformation is RSA then an RSA key-pair must be provided; if the transformation is DES/ECB/PKCS5Padding, you must provide a DES key. You can generate a new secret key using the keytool utility, see "Manage keys, certificates and keystores" on page 93.
8. JKS keystores do not support secret keys, so you should use some other keystore type such as JCEKS if you want to use a secret key encryption.
9. When using a block cipher in a feedback mode that requires an initialization vector (IV), encrypted data is prefixed with the initialization vector as plaintext. The IV does not have to be kept secret but must be unpredictable. That is why a random IV is generated for each piece of data that is being encrypted. Generation of random data can sometimes be resource-intensive, so you may wish to consider a non-IV feedback mode (ECB) if performance is an issue.
10. Which secret key transformations are supported for encryption depends on the capabilities of the Java security provider. By default IBM Security Directory Integrator uses the IBMJCE provider. Supported block ciphers are: DES, AES, DESede (Triple DES), Blowfish and RC2. They can be used in any of the following feedback modes: ECB, CBC, CFB, OFB, PCBC. The only

available padding scheme is "PKCS5Padding". The MARS block cipher should not be used for encryption, because it does not support padding (<http://www-128.ibm.com/developerworks/java/jdk/security/50/secguides/JceDocs/api/com/ibm/crypto/provider/Mars.html>). Supported stream ciphers are RC4 and ARCFOUR (basically the same cipher under two different names). The SEAL stream cipher requires large keys (160 bit) so it can be used only after configuring unrestricted IBM SDK policy on the IBM Security Directory Integrator JRE (<http://www.ibm.com/developerworks/java/jdk/security/60/#sdkpol>).

Separation of certificates for PKI Encryption and SSL

Creating an encrypted IBM Security Directory Integrator configuration file from scratch

You can create an encrypted IBM Security Directory Integrator configuration file from scratch using the `cryptoutils` command line tool.

This is how you create an encrypted IBM Security Directory Integrator configuration file from scratch.

Using the `cryptoutils` command line tool

1. Create a normal un-encrypted IBM Security Directory Integrator configuration file using the Configuration Editor.
2. Use the `cryptoutils` command line tool to encrypt this configuration file as described in the "The IBM Security Directory Integrator Encryption utility" on page 136" section.
3. In order to run this encrypted configuration file you must start the IBM Security Directory Integrator server in secure mode as described in the "Server Security Modes" section.
4. In order to edit this encrypted configuration file you can use one of two options described in the "Editing an encrypted IBM Security Directory Integrator configuration file" section.

Editing an encrypted IBM Security Directory Integrator configuration file

You can perform editing on an encrypted file using the steps discussed here.

You can first decrypt the encrypted configuration file using the `cryptoutils` command line tool as described in the "The IBM Security Directory Integrator Encryption utility" on page 136" section. Then you can edit the decrypted configuration using the Configuration Editor and finally you can encrypt back the modified configuration file using the `cryptoutils` tool.

Standard encryption of `global.properties` or `solution.properties`

Use the steps described here to encrypt the `global.properties` or `solution.properties` files.

The `global.properties` and `solution.properties` files store a number of properties, some of which can represent sensitive data such as passwords. In order to protect this sensitive data IBM Security Directory Integrator is capable of encrypting this data.

All properties whose names are prefixed with *{protect}*- are PKI encrypted by the Server using the Server's public key. The Server's key is specified by the *com.ibm.di.server.encryption.key.alias* property from the keystore specified by the *api.keystore* property. For example, if you want to encrypt a property *com.ibm.di.server.encryption.keystore* you can add the following line in the *global.properties* or *solution.properties* file:

```
{protect}-com.ibm.di.any.property=some_value
```

The next time the Server runs it detects that this property has to be encrypted and it immediately overwrites the file, writing the plain text value "some_value" in encrypted form.

Note: On some operating systems (Linux/UNIX systems if so configured) the *global.properties* file might not be accessible for writing. In this case the server outputs a warning message that the file has not been written/encrypted.

Protecting the properties in *global.properties* or *solution.properties* is also accessible from the "Global-Properties" and "Solution-Properties" Property Stores accessible from the **Browse Server Stores** option in the Configuration Editor.

Encryption of properties in external property files

Use a named certificate from the server's keystore to encrypt properties in external property files .

Properties stored in external property files can be protected by encryption in just the same way as properties in the *global.properties* or *solution.properties* can.

Instead of using the server's default certificate, it is possible to encrypt properties in external property files using a specifically named certificate from the server's keystore.

For more information on encrypting properties stored in these files, see the "Standard encryption of *global.properties* or *solution.properties*" on page 135 section. The syntax of properties in an external property file is as follows:

```
[{protect}-]keyword <colon | equals> [{encr}][{java}]value
```

- The optional *{protect}*- prefix signals that the value either is or should be encrypted. When the value starts with the character sequence *{encr}* it means that the value is already encrypted.
- The optional *{java}* value prefix signals that the value is a serialized java object. The value must be b64-encoded. For example:

```
{protect}-api.truststore.pass  
={encr}J8AKimpEutu3Bb10Vg55F/5d5v02kXWcNUWnCq3vINUc6K0719z9dEk3H430t2iTT1dZTI6FSSVin9KsCy  
BLmgv+n84w7He1K13ro2dFmZbTYKMXuxGooqN9nL2V0vZoptNqzoWvs6IN/p3Vk1IBt1ao/9mEPEKu1wRnKtkQ89Bg=
```

The IBM Security Directory Integrator Encryption utility

You can edit the files using *cryptoutils* utility. You should take care of the listed parameters while performing the same.

In the *TDI_install_dir/serverapi* directory you find a utility (*cryptoutils*) which enable you to decrypt and re-encrypt files, (for example, the Identity Registry file) such that you can edit the file manually.

The tool recognizes the following command-line parameters:

input {required} Specifies the file to be encrypted or decrypted.

output

{required} Specifies the new file that is created with the resulting data after the encryption or decryption is done. If the file exists, it is overwritten.

mode {required} Specifies the mode in which the tool operate; it can be one of the following modes:

- *encrypt*: encrypt user registry
- *decrypt*: decrypt user registry
- *encrypt_config*: encrypt an The IBM Security Directory Integrator Encryption utility configuration file or a JavaScript file
- *decrypt_config*: decrypt an The IBM Security Directory Integrator Encryption utility configuration file or a JavaScript file
- *encrypt_props*: encrypt the values of all protected properties in an The IBM Security Directory Integrator Encryption utility properties file
- *decrypt_props*: decrypt the values of all protected properties in an The IBM Security Directory Integrator Encryption utility properties file

Note: User Registry files are encrypted differently from configuration and JavaScript files.

keystore

{required} Specifies the keystore file which contains the key for encryption/decryption.

storepass

{required} Specifies the password of the keystore file.

alias {required} Specifies the alias of the encryption/decryption key in the keystore

keypass

{optional} Specifies the password of the encryption/decryption key; by default, the keystore password is used to access the key

transformation

{optional} Specifies the name of the cryptography transformation used for encryption/decryption; can be RSA or any secret key transformation (for example, AES/CBC/PKCS5Padding); the default is RSA.

storetype

{optional} Specifies the type of the keystore file (for example, JKS); this parameter is case-insensitive (JCEKS and jceks are equivalent); if this parameter is missing, the default keystore type of the JRE (configured by the "keystore.type" security property in the java.security file of the JRE) is used.

cryptoproviderclass

{optional} Specifies the Java security provider which is used for encryption/decryption (but not for keystore access); by default the providers from the security provider list of the JRE (configured in java.security JRE file) is used.

Examples:

Encrypt the User Registry

An IBM Security Directory Integrator Server running in secure mode requires that the User Registry is encrypted with the Server key.

You can encrypt a plaintext User Registry file like this:

```
cryptoutils -input registry.txt -output registry.enc -mode encrypt
-keystore ../testserver.jks -storepass server -alias server
```

Decrypt an IBM Security Directory Integrator configuration

```
cryptoutils -input myconfig.enc.xml -output myconfig.xml -mode decrypt_config -keystore ../testserver.jks
-storepass server -alias server
```

This command decrypts the "myconfig.enc.xml" configuration file (possibly created by an IBM Security Directory Integrator Server, which runs in secure mode). Now the decrypted configuration "myconfig.xml" can be easily modified using the Configuration Editor. After modifying the configuration, it can be encrypted again, so that an IBM Security Directory Integrator Server in secure mode can read and use it.

Encrypt an IBM Security Directory Integrator configuration using a symmetric cipher (rather than the default "RSA")

```
cryptoutils -input myconfig.xml -output myconfig.enc.xml -mode encrypt_config -keystore ../server.jck
-storepass server -alias server -transformation AES/CBC/PKCS5Padding -storetype jceks
```

The above command assumes that the keystore "server.jck" exists. That keystore is supposed to contain an AES secret key under alias "server".

Decrypt the global.properties file

The IBM Security Directory Integrator Server automatically encrypts the values of protected properties when reading the `global.properties` or `solution.properties` file.

You can decrypt all encrypted values in the `global.properties` file like this:

```
cryptoutils -input ../etc/global.properties -output ../etc/global.properties -mode decrypt_props
-keystore ../testserver.jks -storepass server -alias server
```

Note: When the `cryptoutils` tool is used to encrypt and decrypt the "Server API User Registry" on page 125, configuration files (see the "Server Security Modes" on page 132" section for details how the server treats encrypted configurations) or "Encryption of IBM Security Directory Integrator Server Hooks" on page 142, it encrypts and decrypts a file as a whole.

On the other hand, the encryption/decryption mode for property files encrypts/decrypts only the values of the protected properties and not the whole file. Thus after encrypting a `.properties` file using `encrypt_props` mode, the property keys and the comments in the file are still readable by humans. For more information on protected properties see sections "Standard encryption of `global.properties` or `solution.properties`" on page 135 and "Encryption of properties in external property files" on page 136.

IBM Security Directory Integrator System Store Security

Use the Derby to define the repository of users and passwords. Refer to the list provided here to set the property with appropriate value. Further you can also utilize User Authorization mechanism provided by Derby with the instructions provided here.

The IBM Security Directory Integrator System Store is the database or persistent layer where all the information which is required by an IBM Security Directory Integrator Server is persisted. Traditionally, this layer did not have any security around itself. Any user was able to access the System Store. However from IBM Security Directory Integrator 7.0, there is configurable security provided around the System Store.

In IBM Security Directory Integrator 7.0, the System Store by default is used in Network Mode. This way, a number of IBM Security Directory Integrator instances and other applications is able to access the System Store concurrently. In view of the System Store being available over the Network there is a need to have some security built around it in order to protect the data which is maintained by the IBM Security Directory Integrator Server.

Derby (previously known as Cloudscape) provides several ways to define the repository of users and passwords. To specify which of these services to use with your Derby system, set the property `derby.authentication.provider` to the appropriate value as discussed in the appropriate section listed below.

External Directory Service

A directory service stores names and attributes of those names. Derby uses the Java naming and directory interface (JNDI) to interact with external directory services that can provide authentication of users' names and passwords.

You can allow Derby to authenticate users against an existing LDAP directory service within your enterprise. LDAP (lightweight directory access protocol) provides an open directory access protocol running over TCP/IP. An LDAP directory service can quickly authenticate a user's name and password.

On configuring a set of properties defined by Derby you can start using the External Directory Service as a repository for user names and passwords.

User-defined class

The user defined class approach enables you to hook Derby to any other external authentication service other than LDAP.

Set `derby.authentication.provider` to the full name of a class that implements the public interface `org.apache.derby.authentication.UserAuthenticator`. By writing your own class that fulfills some minimal requirements, you can hook Derby up to an external authentication service.

The class that provides the external authentication service must implement the public interface `org.apache.derby.authentication.UserAuthenticator` and throw exceptions of type `java.sql.SQLException` where appropriate.

Built-in Derby Users

Derby provides a simple repository for storing the user names and passwords. For using this built-in repository the property `derby.authentication.provider=BUILTIN` should be set.

The IBM Security Directory Integrator System Store is using the Built-in repository for storing the user name and password. Since IBM Security Directory Integrator have only one user for accessing the System Store this is the most viable provider that can be used.

User Authentication

The user authentication details deal with the authentication of users. The user authentication mechanism only authenticates if the user name is present in the mentioned repository (it can any one of the repositories which are mentioned above) and if the password is correct for the specified user. However if you want to have more control over the access rights, you can use the User Authorization mechanism provided by Derby.

The master switch for requiring that users be authenticated against provided parameters is the property `derby.connection.requireAuthentication` - the default is *TRUE*.

The access modes can be set using the property `derby.database.defaultConnectionMode=fullaccess`. This property sets the default access mode for all the users in the Derby repository. This property also defines the access level for the System Store user. The different access levels supported by Derby are *fullAccess*, *readOnly*, and *noAccess*. However if you want to have specific access modes for specific users, you can assign access using the properties mentioned below:

- `derby.database.fullAccessUsers=<usernames>` for allowing full access to users.
- `derby.database.readOnlyAccessUsers=<usernames>` for allowing read only access to users.
- `derby.database.noAccessUsers=<usernames>` for not allowing users to access the database.

The *usernames* should be a comma separated list of users for example

```
derby.database.fullAccessUsers=sa, mary
```

In the current version of IBM Security Directory Integrator we have only one user accessing the System Store. This user is required to perform all the operations on the System Store hence we have set the access mode to *fullAccess*.

Miscellaneous Config File features

You can have a detailed understanding on the miscellaneous Config File features with the information provided here.

The "password" configuration parameter type

The configuration parameters of an IBM Security Directory Integrator component in a Config can be "string", "number", "boolean", and so on. One of the available types is "password". If a configuration parameter is of type password, then the Configuration Editor shows its value in the component configuration window as a sequence of '*' characters - both when typing in a new password, and when opening an existing configuration for editing or running.

Component Password Protection

You can define the component passwords in a default property store by using the instructions provided here.

IBM Security Directory Integrator saves configuration information in an XML file which contains clear text for all configuration values. This includes sensitive information like passwords. IBM Security Directory Integrator supports encryption of the entire configuration file but does not encrypt or protect sensitive information when the configuration file is saved in clear text.

IBM Security Directory Integrator provides a way to better protect passwords that are needed for its various components; it hides the passwords in a clear text configuration and provides default security for passwords that are stored. In order to do this component passwords are defined (stored and retrieved) in a default property store, instead of in the configuration file. In IBM Security Directory Integrator, a user defined property store can be any system for which there is a connector and the default property store most likely be an external properties file.

All component passwords will by default go to this default property store, instead of in the configuration file (as it is in older versions of the product). Thus, passwords can be isolated from the configuration file unless explicitly overridden by the user (may be appropriate for initial development).

Saving passwords to configured Properties

Use the instructions provided here to save the password using the password store.

About this task

The password protection mechanism is directly related to the configuration windows offered to the user. The configuration windows, or forms, contain descriptions of each parameter and its syntax. One type of syntax is *password* which causes the Configuration Editor to use a password text field for editing. Whenever the value for a password syntax component parameter is changed, the value of the password is saved in an external repository, called the *Password Store*. This external repository for passwords is configured in the *Properties* page in the configuration editor (*Password-Store*) and is specified in the configuration file for the current IBM Security Directory Integrator solution. If no such property store is configured the password is saved in clear text in the configuration file.

If a default password store is configured, a unique property name is generated the first time a protected/password parameter is saved. This key is used as the key in the password store. The same property name is written to the configuration file as a standard property reference. When the value is later retrieved, standard property resolution takes place to retrieve the actual value from the password store.

If a Password Store is specified, a unique key is generated for the password and the password is saved encrypted in the Password Store under that key. In the configuration file, the password is referenced only by that key.

If no Password Store is specified, the password appears in plain text in the configuration file.

For example:

1. Create a new project from the Configuration Editor
2. Right-click on the "Properties" folder in the navigation view and select "New Property Store" called "MyProps".
3. From the "Connector" tab of the newly created Property Store, type in "MyProps.properties" in the "Collection Path/URL" field.
4. Specify that the new Property Store is used as the Password Property Store (right-click on the new properties store in the navigation view and select **Password Property Store**).
5. Add a new assembly line with a FTP Client Connector.
6. Enter a password in the "Login Password" field of the FTP Client Connector.
7. Save the solution and close the Configuration Editor.

After the above procedure, in the configuration file of the created solution will contain lines that resemble the following text:

```
<parameter name="ftpPass">@SUBSTITUTE{property.MyProps:ftpPass-38ae53e8779cfd65}</parameter>
.....
<PasswordStore>MyProps</PasswordStore>
```

...and in the "MyProperties.properties" file there is a line like the following text:

```
{protect}-ftpPass-38ae53e8779cfd65={encr}GVJC01A7VUiW=
```

This means that the FTP password configuration in the solution file references an encrypted property from the current Password Store - "MyProps". The property key used is "ftpPass-38ae53e8779cfd65".

Protecting attributes from being printed in clear text during tracing

Use the methods provided here to protect the sensitive data during tracing.

IBM Security Directory Integrator solution builders need a way to protect sensitive data, such as passwords, from being printed in clear text when tracing on the solution is needed. Therefore in IBM Security Directory Integrator some of the methods dealing with the Attribute class have been enhanced to say whether an attribute is protected or not. If the attribute is marked as protected and tracing is on, a fixed number of stars '*' is output instead of the actual value.

When connection parameters are found in the TaskCallBlock (TCB), the values never be logged directly by IBM Security Directory Integrator. The fact that parameters were given is logged, but not the values themselves. If the solution needs to be debugged, those values can be dumped manually, for example using scripting.

Encryption of IBM Security Directory Integrator Server Hooks

You should encrypt the sensitive data before adding it to server hook directory. For details refer the information provided here.

Server Hook scripts are defined and made available by creating files in the "serverhooks" subdirectory of the solution directory. Scripts that contain sensitive information should be encrypted with the Server API before adding it to the directory. Scripts can be encrypted by using `cryptoutils` (see "The IBM Security Directory Integrator Encryption utility" on page 136). Note that the IBM Security Directory Integrator server only decrypts script files that have the ".jse" filename extension. The ".jse" extension indicates to the IBM Security Directory Integrator server that the script file is encrypted. That is why, after you encrypt a Server Hook script file, make sure to change its filename extension to ".jse".

Remote CE and SSL

You should take care of the listed points here while working with Remote Configuration Editor and SSL.

The Configuration Editor used to edit remote Config Files (that is, Config files on a remote system) is called the Remote Configuration Editor (Remote CE). The IBM Security Directory Integrator Remote CE is capable of starting AssemblyLines in configurations opened for editing. The Remote Configuration Editor is a client of the Server API of the remote IBM Security Directory Integrator Server. Consequently the Remote Configuration Editor is authenticated and authorized as a client of the Server API. In order for this to work when SSL is used:

1. The server to which the Remote Configuration Editor connects must be configured to require SSL client authentication. This is a configuration of the Server API – for details see "SSL-based authentication" on page 114.
2. The Remote Configuration Editor IBM Security Directory Integrator instance must be configured to supply SSL client authentication. This is configured in a "SSL client authentication" on page 102.

This SSL client authentication is needed because the Remote Configuration Editor uses listener objects so that it can be notified when an AssemblyLine has terminated and for this to work with SSL both the client must trust the server identity and server must trust the client identity.

Using the Remote Configuration Editor

You can take care of the limitations of using the remote configuration editor.

Using a Remote Configuration Editor is a little different from using a local CE. When running a remote Configuration Editor to manage a Config on a remote system, you must be mindful of restrictions that apply to the CE in remote mode. Notable restrictions include:

- When editing Config files locally, it is sufficient to have appropriate file system access (read and write) to the Config file. However, when editing a remote Config, you must have Admin privileges on the remote Config Instance.
- When connecting to a data source (using **Connect** buttons in mapping windows), these connections are evaluated locally.
For example: `ldap://localhost:389` results in the Configuration Editor (CE) attempting to connect to the local LDAP server, rather than to the LDAP server on the remote computer.
- When generating WebServices-related connectors results in function components that generate the WSDL file, jar files (using Complex Type Generator), and so on, you are generating them locally. These components are not generated on the remote system to which the CE is connected and must be uploaded manually to the remote system for deployment.
- The remote CE only allows editing and viewing of those Configs that are present in the folder specified by the `api.config.folder` property.
- When working with **System Store** operations (such as deleting the Iterator state key, and so on) that are available in the CE, work with the local system store and not with the remote IBM Security Directory Integrator computer's System Store. Only when the AssemblyLine (AL) is executed does the AL connect to the remote System Store, because at AL execution time, the AL is running inside the remote JVM.
- When you use the **Parameter Substitution** editor (available with Ctrl-E), the editor shows only the local properties, and not the properties set on the remote system. Similarly, creating and saving a new property store (file type) stores the property store (file) locally.
- When using the Configuration Editor to edit remote Config files, you are subject to Server API authentication and authorization, because the CE is acting as a client application. Therefore, in order to use the CE in this way, you must have *admin* access on the Remote Server.
- When using the Remote Server, the Remote Server itself must have sufficient access to the local file system where the Config files are stored. If the ConfigFiles are stored on a read-only file system or a file storage location where the user ID under which the Remote Server is running does not have write access, you cannot edit remote Configs.

Summary of configuration files and properties dealing with security

You can refer the summarized list of configuration files and properties dealing with security.

Table 19. The table of configuration files that were discussed above and what is contained in each.

Configuration file	Location	Description
global.properties	<i>TDI_home/etc</i>	This file is the primary configuration file for the server.
solution.properties	Solution folder	This file (<i>solution.properties</i>) is initially a copy of <i>global.properties</i> used by the current solution. After you make changes, values in this file override corresponding values in <i>global.properties</i> .
registry.txt	<i>TDI_home/serverapi</i>	This file is the User registry for the Server API, defined by the "api.user.registry" property in <i>global.properties</i>
build.properties	<i>TDI_home/etc</i>	This file contains the IBM Security Directory Integrator build information, build date, version, and so on; it is a text file, and by default the file is in the platform-native encoding.
tdisrvctl-log-4j.properties	<i>TDI_home/etc</i>	This file controls the logging strategy for the <i>tdisrvctl</i> command line utility.
Log4j.properties	<i>TDI_home/etc</i>	This file controls the logging strategy for the server (<i>ibmdisrv</i>) when started from the command line.
jlog.properties	<i>TDI_home/etc</i>	This file controls the tracing and First Failure Data Capture (FFDC) strategy
ibmdi.ico	<i>TDI_home/etc</i>	This file lists the icons for IBM Security Directory Integrator.
idisrv.sth	<i>TDI_home</i>	This file contains the IBM Security Directory Integrator server stash; it is a binary file that contains the encrypted password for the sample server keystore file (<i>testserver.jks</i>).
derby.properties	<i>TDI_home/etc</i>	This file contains the default configuration for the Derby System Store shipped with IBM Security Directory Integrator.
reconnect.rules	<i>TDI_home/etc</i>	This file contains text that defines reconnect rules for how IBM Security Directory Integrator should handle reconnect exceptions.
global.properties.v611	<i>TDI_home/etc</i>	This file serves as a sample place holder and is useful during migration.
TDI0701.SYS2	<i>TDI_home/etc</i>	This is the product signature (license) file used by the ITLM agent to recognize IBM Security Directory Integrator.
pkcs11.cfg	<i>TDI_home/etc</i>	This file is used for initializing the IBM PKCS11 implementation provider. For details refer to section PKCS11 Configuration File.
testadmin.der	<i>TDI_home/serverapi</i>	This file is the exported certificate from <i>testadmin.jks</i> .
testadmin.jks	<i>TDI_home/serverapi</i>	This file contains an example keystore and truststore for a Server API remote client.

Table 19. The table of configuration files that were discussed above and what is contained in each. (continued)

Configuration file	Location	Description
cryptoutils.bat(sh)	<i>TDI_home</i> /serverapi	This file is a command line utility (shell script) used for encrypting and decrypting IBM Security Directory Integrator configurations and the user registry file.
testserver.jks	<i>TDI_home</i>	This file is a sample server keystore and truststore, referenced as an example.
testserver.der	<i>TDI_home</i>	This file is an exported sample server certificate, ready to be imported in a truststore.
am_config.properties	<i>TDI_home</i> /ActionManager	This file configures the Action Manager.
am_logging.properties	<i>TDI_home</i> /ActionManager	This file configures Action Manager logging.
ibmdiservice.props	<i>TDI_home</i> /win32_service	This file configures the Windows service.
mqeconfig.props	<i>TDI_home</i> /jars/plugins/	This file allows configuration of the IBM WebSphere MQ Everyplace service. In IBM Security Directory Integrator, you can access IBM WebSphere MQ Everyplace using authentication for the IBM WebSphere MQ Everyplace Mini-Certificate Server to issue certificates; the certificates are then used for authentication. When authenticating, additional properties available in IBM Security Directory Integrator that must be added to the mqeconfig.props properties file.

Note: The file registry.txt can be encrypted and decrypted using the “The IBM Security Directory Integrator Encryption utility” on page 136. The cryptoutil tool should not be applied on global.properties or solution.properties. You can encrypt individual property values but not the whole properties file.

Table 20. The table of properties that are referenced above, characteristics about them, what they do, what their value can be, what they are used for.

Name	Possible values	Description
com.ibm.di.server.securemode	true/false	On or off switch for secure mode.
api.keystore	file name	Server keystore used for SSL certificates. Previously com.ibm.di.server.keystore.
api.key.alias	Key alias	Key alias from keystore for SSL certificates. Previously com.ibm.di.server.key.alias.
{protect}-api.keystore.password	SSL keystore password	Keystore password for SSL. Added in IBM Security Directory Integrator 7.0.
{protect}-api.key.password	SSL key password	Key password for SSL. Added in IBM Security Directory Integrator 7.0.
com.ibm.di.server.encryption.keystore	file name	Data encryption for the keystore that hosts the key used by the Server. Added in IBM Security Directory Integrator 7.0.
com.ibm.di.server.encryption.key.alias	Key alias	Encryption keystore key alias. Added in IBM Security Directory Integrator 7.0

Table 20. The table of properties that are referenced above, characteristics about them, what they do, what their value can be, what they are used for. (continued)

Name	Possible values	Description
com.ibm.di.server.encryption.keystoretype	Keystore type, that is, "JKS", "JCEKS", and so on.	Keystore type that hosts the encryption key of the Server. Added in IBM Security Directory Integrator 7.0.
com.ibm.di.server.encryption.transformation	"RSA" or some secret key transformation	Server transformation used for encryption. Can be set to either "RSA" (public key encryption) or to some secret key transformation. Added in IBM Security Directory Integrator 7.0.
api.on	true/false	On or off Server API switch.
api.user.registry	file name	Server API users registry file
api.user.registry.encryption.on	true/false	User registry switch for encrypted or not encrypted.
api.remote.on	true/false	On or off switch for remote Server API. The default setting is true.
api.remote.ssl.on	true/false	On or off switch requiring, or not requiring, SSL for the remote Server API.
api.remote.ssl.client.auth.on	true/false	On or off switch requiring, or not requiring, SSL client authentication for the remote Server API
api.truststore	file name	Server truststore.
api.truststore.pass	*	Truststore password.
api.remote.nonssl.hosts		Non-SSL addresses for accepting non-SSL IP connections.
api.custom.method.invoke.on	true/false	Server API methods for custom method invocation =true when allowed to be used, and =false when disallowed.
api.custom.method.invoke.allowed.classes		Server API classes that can be directly invoked by the Server API methods for custommethod invocation.
api.custom.authentication	Script file name or "[ldap]/[jaas]" for built in LDAP or JAASAuthentication	Custom authentication method.
api.custom.authentication.ldap.*		LDAP authentication configuration set of properties.
javax.net.ssl.*		Standard JSSE set of properties for keystore, truststore and their passwords
com.ibm.di.server.pkcs11	false	pkcs11 compliant crypto devices for SSL, required or not required. Added in IBM Security Directory Integrator 7.0

Table 20. The table of properties that are referenced above, characteristics about them, what they do, what their value can be, what they are used for. (continued)

Name	Possible values	Description
{protect}-com.ibm.di.server.pkcs11.pass	administrator	Access password for pkcs11 compliant crypto device. Added in IBM Security Directory Integrator 7.0
com.ibm.di.server.pkcs11.accl	false	Hardware cryptographic devices to be used for encryption when this property is set to true.

Note: All properties listed in the above table can be set in the configuration file `global.properties`, and can be protected by encryption using the {protect}- prefix (see section “Standard encryption of global.properties or solution.properties” on page 135” for details).

Web Admin Console Security

Refer the link provided here to know more about Web Admin Console Security.

See “AMC and Action Manager security” on page 251.

Miscellaneous security aspects

You can refer to the various security aspects listed here.

HTTP Basic Authentication

You can authenticate a component using HTTP basic authentication. Make sure to check the listed parameters.

Some IBM Security Directory Integrator components give you the opportunity to use HTTP Basic Authentication as authentication mechanism. As the name says it is basic (simple) authentication. HTTP Basic Authentication should not be considered secure for any particularly rigorous definition of secure, because the credentials are base64 encoded and they can be easily decoded by someone. You should use more complex schemes to protect their data (for example a combination of turned on SSL and HTTP Basic Authentication). If the component supports HTTP Basic Authentication, then you see the following parameter:

authenticationMethod

Specifies the type of HTTP authentication. If the type of HTTP authentication is set to `Anonymous`, then no authentication is performed. If HTTP basic authentication is specified, HTTP basic authentication is used with user name and password as specified by the `username` and `password` parameters.

Lotus Domino SSL specifics

You can refer to the Lotus Domino SSL specifications provided here.

The Domino APIs for SSL do not use JSSE, and are instead Domino-specific. This means that the IBM Security Directory Integrator truststore and keystore (see section “Client SSL configuration of IBM Security Directory Integrator components” on page 101) do not play any part in SSL configuration for the Domino Change Detection connector. For SSL configuration of the Domino Change

Detection connector, a `TrustedCerts.class` file is used. This file is generated every time the `DIOP` process starts (in the Domino Server) and must be in the classpath of IBM Security Directory Integrator (that is, the `ibmdisrv` or `ibmditk` shell scripts which start the IBM Security Directory Integrator server and IBM Security Directory Integrator Configuration Editor respectively). You must copy the `TrustedCerts.class` to a local path included in the `CLASSPATH` or have the `Lotus\Domino\Data\Domino\Java` of your Domino installation in the classpath. Whether the IBM Security Directory Integrator truststore or keystore are set or not in `global.properties` (or `solution.properties`) is of no consequence to this connector.

Note: The above is related to the configuration of SSL for the Notes Connector and the Domino Change Detection Connector since they use SSL over IOP.

Certificates for the IBM Security Directory Integrator Web service Suite

You can use the instructions and example provided here to name the certificate for the IBM Security Directory Integrator web service suite.

The `cn=` portion of the distinguished name (`dn`) of a certificate to be used with the IBM Security Directory Integrator Web services Server Connectors must match the DNS name or IP address of the host computer on which IBM Security Directory Integrator is running. Otherwise an Exception is thrown, because the client not be able to establish an SSL connection to the IBM Security Directory Integrator Web services Server Connector. An example of the `cn=` portion of the distinguished name of a certificate follows: `cn=www.myserver.com`. (This constraint about the distinguished name in the server's certificate comes from the HTTPS protocol - see `rfc2818` "HTTP over TLS.")

Note: If IBM Security Directory Integrator needs to use both a client and a server certificate only the default certificate configured in `global.properties` or `solution.properties` is used, then this must be the same certificate. An alternative would be to write a custom implementation of the `SSLSocket` or the `SSLServerSocket` Java class and make it use a certificate different from the default.

Example Server certificate creation

The following command line creates a self-signed server certificate in the keystore named `"MyServerKeyStore.jks"`.

```
keytool -alias MyServerCertAlias -keyalg RSA -genkey -dname cn=<server_ip_address>
        -validity 365 -keystore MyServerKeyStore.jks -storepass mystorepass -keypass mykeypass
```

The alias of the created certificate is `"MyServerCertAlias"`. The RSA algorithm is used to create the key pair. The distinguished name of the certificate is the IP of the server. The certificate is valid for 365 days (one year). The password of the keystore is `"mystorepass"`. The password of the created private key is `"mykeypass"`. The created certificate can then be configured for use by setting the following properties in the `global.properties` or `solution.properties` file:

```
api.key.alias=MyServerCertAlias
api.keystore=MyServerKeyStore.jks
```

IBM WebSphere MQ Everyplace authentication with mini-certificates

You can refer to the instructions provided here to know more about IBM WebSphere® MQ Everyplace® authentication with mini-certificates.

IBM Security Directory Integrator IBM WebSphere MQ Everyplace components can be deployed to take advantage of IBM WebSphere MQ Everyplace Mini-Certificate authenticated access. To use these IBM WebSphere MQ Everyplace features, it is necessary to download and install IBM WebSphere MQ Everyplace 2.0.1.7 and IBM WebSphere MQ Everyplace Server Support ES06. Use of certificate authenticated access prevents an anonymous IBM WebSphere MQ Everyplace client Queue Manager or application submitting a change password request to the IBM WebSphere MQ Everyplace Password Store Connector.

For more information on configuring IBM WebSphere MQ Everyplace authentication with Mini-Certificates, see "Authenticated IBM WebSphere MQ Everyplace Access" in the *Password Synchronization Plug-ins*.

Chapter 7. Reconnect Rule Engine

You can refer to the information provided here to understand more about Reconnect rule engine.

Introduction

The IBM Security Directory Integrator Server supports reconnect rules that apply to certain error situations during the life of a Connector. The server takes measures, laid out in rules, based on conditions occurring when communicating with target systems.

The AssemblyLine polls the Reconnect Rule Engine every time a Connector raises an exception and the engine recommends a course of action for the current situation. The AssemblyLine code then acts in the proposed way. The possible actions to attempt are:

- to reconnect
- to leave the exception unhandled and let further error mechanisms like error hooks process it.

The *reconnect action* leads to a reconnect attempt only if reconnect is enabled by means of the options available in the Connector's "Connection Errors" tab in the CE. If reconnect is not enabled in this configuration, reconnect is not attempted in case of error regardless of the decision of the Reconnect Rule Engine.

Reconnecting basically involves automatic restart of the Connector and bringing it to its previous position (if so configured). This is done by executing terminate for the Connector, then executing initialize for the Connector and in case of Iterator Connectors, optionally skipping entries until the position before the reconnect is reached. On each reconnect attempt the corresponding reconnect hook is invoked. The script in the hook may eventually change the configuration so that a subsequent reconnect would be successful. If the user has specified failover, an automatic failover or fallback is attempted when the reconnect attempts fail.

The *error action* implies that no automatic reconnect is attempted and that the corresponding error hooks are invoked. The hooks can eventually perform some custom recovery or error reporting.

Reconnect Rules

You can understand about type of rules, part of rules, part of error situation and nested exceptions through the information provided here.

The Reconnect Rule Engine makes decisions based on configured rules. Each rule describes what should be done when a given kind of error situation ensues. The engine uses two types of rules:

- **Built-in rules**, which are stored in the `tdi.xml` files of each connector file and are packaged in the connector's jar file; as a result these rules are always specific to the particular connector class and match all connector names; this list of rules is the default list of the Reconnect Rule Engine when working on an error situation for a given Connector; if you have programmed your own Connectors in Java, then for information about how to construct your own built-in rules see

section "Connector Reconnect Rules definition" in the "Implementing your own Components in Java" appendix under the *Reference* section of IBM Knowledge Center for IBM Security Directory Integrator.

- For compatibility with previous releases of IBM Security Directory Integrator, when the Reconnect Rule Engine is set up it implicitly adds to the built-in rules, a set of rules that prescribe to attempt reconnect on all `IOException`-s and all `CommunicationException`-s (`java.io.IOException` and `javax.naming.CommunicationException`);
- **User-defined rules**, which are loaded from an external text file named `etc/reconnect.rules`; this list of rules overrides the built-in rules. See "User-defined rules configuration" on page 153.

Each rule applies to certain connectors and certain error situations.

A rule has the following parts:

- **Connector Class**: the Java Class of the connectors to which the rule applies
- **Connector Name**: the name of the connector component as it is specified in the configuration file of the currently executed solution
- **Exception Class**: the base class of the exceptions to which the rule applies
- **Regular Expression**: a regular expression that matches the messages of the exceptions to which the rule applies
- **Action**: the action, prescribed by the rule. Can be *error* or *reconnect*.

An error situation is described by the following parts:

- **Connector Class**: the class of the connector that raised the exception
- **Connector Name**: the name of the connector that raised the exception
- **Exception**: the exception raised by the connector - a subclass of `java.lang.Throwable`.

A rule applies to an error situation if all of the following conditions are fulfilled at the same time:

- the rule applies to the connector in the error situation (subclasses of the connector class, described in the rule are also matched)
- the rule applies to the name of the connector which caused the error situation
- the exception is an instance of the exception class, to which the rule applies
- the rule does not have a regular expression to match the exception message or the regular expression matches the message of the exception.

When a given error situation occurs, the reconnect rule engine finds the most specific rule that matches the error situation. First the engine searches through the user-defined rules and if no matching rule is found, it searches the built-in rules. If still no matching rule is found, the engine prescribes the default action, which is "error". If a matching rule is found in the user-defined rules, then the built-in rules are not searched, even if there exists a more specific rule in the built-in rules.

Note: If two or more rules match an error situation, the most specific rule is selected; if there are several most-specific rules and none of them is more specific than the rest, then the first rule in the list is selected. That is why the order of the rules in the rule lists matters. For example: suppose the following rules exist (this is pseudo-syntax used for clarity only):

```
...exceptionClass = "java.io.IOException", exceptionMessageRegExp = ".*", action = "error"...  
...exceptionClass = "java.io.IOException", exceptionMessageRegExp = "\\w*", action = "reconnect"...
```

If an exception of type `java.io.Exception` with message "problem" is raised, then the first rule is selected, although both rules match the error and no rule is more specific than the other (the outcome of the regular expression match is not considered for weighting purposes.)

Nested Exceptions

Some exceptions are nested inside other exceptions. When the reconnect rule engine searches through a list of rules (for example the built-in rules), the engine searches for a rule that matches the top-level exception first. If no matching rule is found, then the engine searches again the same list of rules but this time it searches for a match for the nested exception (if the top-level exception has no nested exception this search is skipped). Note that only the first-level nested exception is attempted to be matched by the reconnect rule engine; if there are more levels of nested exceptions they are ignored.

Note: Automatic Failover is not possible for Server mode Connectors.

User-defined rules configuration

You should take care of the format of the rule while defining it and should also take care of some important information listed here. Further you can also refer to some examples.

The list of user-defined rules is configured in a text file named `reconnect.rules` in the "etc" subfolder of the IBM Security Directory Integrator solution folder (or the IBM Security Directory Integrator installation folder, if no solution folder has been defined). Each rule is placed on a single line. The format of a rule is as follows:

```
<connector_class>:<connector_name>:<exception_class>:<action>:<regular_expression>
```

where

- `<connector_class>` is the fully qualified name of the Java class of the Connector
- `<connector_name>` is the name of the Connector as inserted in the `AssemblyLine`
- `<exception_class>` is the fully qualified name of the Java class of the exception
- `<action>` can be either 'error' or 'reconnect'
- `<regular_expression>` is a Java regular expression as described in the JavaDoc of the `java.util.regex.Pattern` class at <http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html>.

Note:

1. Each part except the action can be empty. If a part is empty that means "match-all".
2. Each part is mandatory - even if it is empty the surrounding colons must be present. (Consequently on each line there must be at least 4 colons - each colon separating two adjacent parts of the rule. At least 4, because the regular expression may contain colons too. These colons do not interfere with the rule parsing because the regular expression comes last in a rule.)
3. No redundant white space is allowed.
4. The regular expression starts just after the fourth colon and spans until the end of the line.
5. The user-defined rules file is not a Java properties file. The main reason is that a key for a rule must include all rule parts, except the reconnect action, in order to be unique. So the only value from using the Java properties mechanism

would be the separation of the action from the other rule parts. However, it would come at the price of escaping white-spaces, colons and equal signs (requirements for a valid property key). Even if the Java property framework was used, custom parsing of the property key would still be required in order to extract the rule parts from it.

6. The regular expression (not the reconnect action) comes last on each line. This pattern is chosen such that it is unnecessary to escape colons (which are considered rule part delimiters) in the regular expression.
7. The regular expression must match the entire message text: Suppose the message text you want to match contains the words "Some Error" somewhere in the message text. A suitable regular expression might then be:

```
.*Some Error.*
```

The character "." matches any character except new line, and the * modifier specifies 0 or more. Now suppose the message ends with a new line. If that is the case, the previous regular expression does not match. You can try a regular expression like this instead:

```
.*Some Error.*\r?\n?
```

"\r" and "\n" specify return and new line characters, and the ? modifier specifies 0 or 1 occurrence.

8. You must still configure reconnect in the Connector's configuration; see "General reconnect configuration" on page 155.

Examples

An example, consisting of two rules:

```
com.ibm.di.connector.ReconnectTestConnector:myconnname:java.io.IOException:error:.*\Wfatal\W.*  
::java.io.IOException:reconnect:
```

Reconnect with the JDBC Connector

IBM Security Directory Integrator's JDBC connector is configured in Iterator mode to iterate a table from DB2 and is enabled for the reconnect feature. However, at the time of running the solution, DB2 instance is not started yet. In order to have reconnect working, the following exception details need to be mentioned in the reconnect.rules file:

```
com.ibm.di.connector.JDBCConnector::com.ibm.db2.jdbc.DB2Exception:reconnect:
```

Reconnect with the RAC Connector

This connector is deprecated and will be removed in a future version of IBM Security Directory Integrator.

IBM Security Directory Integrator's RAC connector is configured in Iterator mode and is enabled for the reconnect feature. In case the Agent Controller server is down, in order for the RAC connector to try to reattempt (reconnect), the following exception details need to be mentioned in the reconnect.rules file:

```
com.ibm.di.connector.RACConnector::org.eclipse.tptp.platform.execution.exceptions  
.AgentControllerUnavailableException:reconnect:
```

Exception considerations

You can refer to the considerations discussed here while working on exceptions.

Every environment and solution created for a particular environment using IBM Security Directory Integrator is typically unique. User-defined rules are custom-built and the functionality is made available so solutions can automatically

attempt to reconnect based on the exceptions specific to the environment or solution. Refer to the IBM Security Directory Integrator Java API documentation for information about specific exceptions that are returned by the IBM Security Directory Integrator APIs for each Connector.

Additionally, some IBM Security Directory Integrator components rely on underlying libraries and the APIs of these libraries throw exceptions for specific situations. Below we list a few core IBM Security Directory Integrator components where you can look for additional information on exceptions and what may be the cause of the exceptions. This information is helpful when deciding if you want to attempt to create custom reconnect rules for specific exceptions that may be encountered:

- LDAP Connector - The LDAP Connector depends on the JNDI libraries shipped with the JRE. For more information on the JNDI interface, its APIs, and the exceptions it may throw, see <http://java.sun.com/j2se/1.5.0/docs/api/javax/naming/package-summary.html>.
- JDBC Connector - The JDBC Connector depends on the configured JDBC Driver. The Java API documentation or reference material for the configured JDBC driver should be consulted for more information on the possible exceptions that may be thrown. The "Understanding JDBC Drivers" subsection in the JDBC Connector section in *Reference* contains links to the JDBC Driver documentation for a set of commonly used JDBC drivers.

General reconnect configuration

Refer to the configuration options listed here with respect to the reconnect rule.

Specifying a reconnect rule is necessary for a reconnect to be attempted. However it is not sufficient by itself. The other requirement is enabling reconnect in the general reconnect configuration. This can be done under the **Connection Errors** tab in the Configuration Editor . If reconnect is not enabled in this configuration, reconnect is not attempted in case of error regardless of the decision of the Reconnect Rule Engine. Here is a list of Configuration options:

Number Of Retries

The number of times a reconnect attempt is made when a problem occurs, before giving up. If a new problem occurs later on, the same number of attempts is made.

Delay Between Retries

The number of seconds to wait between each reconnect attempt, and before the first reconnect attempt.

Retry Connect on Initial Connection Failure

If this flag is set, and a connection cannot be established when the connector is being initialized, a "reconnect" attempt is made. Not really reconnect, since a connection was not established in the first place, but generally the same mechanism.

Auto Reconnect on Connection Loss

If this flag is set, and the connection is lost after the connector is initialized, a reconnect attempt is made.

Auto Skip Forward

After a reconnect, automatically skip forward as many times as the number of successful reads.

Automatic Failover

If this flag is set, an automatic failover is attempted after an automatic reconnect fails.

FailOver Connector

Name of the Connector in the Resources Library that is used for automatic failover.

Failback After

If this field is set to a positive value, after the specified seconds have passed, an automatic failback is attempted. If the failback fails, it is not attempted again before the specified seconds have passed again.

Note: For both the **Retry Connect on Initial Connection Failure** and **Auto Reconnect on Connection Loss** flags, the reconnect engine is determine if the exception leads to a reconnect attempt, or is a more general error.

Chapter 8. System Queue

You can store and forward messages between IBM Security Directory Integrator Servers and AssemblyLines with System Queue.

The System Queue is an IBM Security Directory Integrator JMS messaging subsystem similar to the IBM Security Directory Integrator System Store. The System Queue simplifies the development of IBM Security Directory Integrator solutions in which asynchronous communication is required to share work amongst multiple AssemblyLines. The System Queue can use either the IBM WebSphere MQ or IBM WebSphere MQ Everyplace as its underlying JMS messaging system, as well as any other JMS system provided the JMS Script Driver can properly address this JMS system.

Note: The System Queue Connector (see *Reference*) does not talk directly to the System Queue, but rather uses the Server API as an intermediary.

In IBM Security Directory Integrator, the System Queue is enabled by default by the install process.

System Queue Configuration

You can know more about System Queue Configuration and the required properties using the information provided here.

The System Queue is configured using the following driver-specific Java properties specified in the IBM Security Directory Integrator `global.properties` or `solution.properties` file:

systemqueue.on

This parameter specifies whether the System Queue is to be started and initialized on IBM Security Directory Integrator Server startup. The valid values are `true` and `false`. The default value is `true`.

systemqueue.jmsdriver.name

This parameter specifies the fully qualified name of the Java class to be used as a JMS Driver for the System Queue. This value can be the name of a user-provided class or one of the following standard IBM Security Directory Integrator JMS Driver implementations:

- `com.ibm.di.systemqueue.driver.ActiveMQ` (“Apache ActiveMQ parameters” on page 158, default JMS Provider)
- `com.ibm.di.systemqueue.driver.IBMMQe` (“IBM WebSphere MQ Everyplace parameters” on page 160)
- `com.ibm.di.systemqueue.driver.IBMMQ` (“IBM WebSphere MQ parameters” on page 160)
- `com.ibm.di.systemqueue.driver.IBMMB` (“Microbroker parameters” on page 161)
- `com.ibm.di.systemqueue.driver.JMSScriptDriver` (other JMS system by way of the “JMSScript Driver parameters” on page 161)

The default value is `com.ibm.di.systemqueue.driver.ActiveMQ`.

Depending on the `systemqueue.jmsdriver.name` parameter, one of the following sections is applicable:

Apache ActiveMQ parameters

You can know more about Apache ActiveMQ parameters using the information provided here.

To use ActiveMQ as the JMS provider for the System Queue, set the `systemqueue.jmsdriver.name` property in `global.properties/solution.properties` to `com.ibm.di.systemqueue.driver.ActiveMQ`. The ActiveMQ driver has the following parameter.

- **jms.broker** - the ActiveMQ server address (Protocol, IP address, and TCP port number). For example, `tcp://localhost:6161` or `ssl://localhost:616171` to use SSL connection.

The default value is:

```
vm://localhost?brokerConfig=xbean:etc/activemq.xml
```

This value runs the ActiveMQ in embedded mode. The `etc/activemq.xml` file holds the default ActiveMQ configuration.

Note:

1. The path to the ActiveMQ configuration XML file (after `xbean:`) cannot hold spaces. For more information, see <https://issues.apache.org/activemq/browse/AMQ-1385>.
If the path contains spaces, each space character must be URL encoded three times, thus transforming it to `%2520`.
2. The System Queue initializes ActiveMQ at startup if the `systemqueue.on=true` parameter is set to true in the `solution.properties` file.

Configuration

You can know more about ActiveMQ configuration and the required parameters using the information provided here.

ActiveMQ configuration relies on the `activemq.xml` file, located at `TDI_install_folder/etc`. The ActiveMQ configuration parameters are as follows.

broker

This ActiveMQ message broker consists of transport connectors, network connectors and properties that are used to configure the broker. The attributes are:

- `brokerName="localhost"` - the name of the broker.
- `dataDirectory="./ActivemqDataStore"` - the directory, which is used to store the data of ActiveMQ.
- `useShutdownHook="true"` - sets whether or not a shutdown handler is used to close the broker if the JVM is terminated.
- `useJmx="true"` - sets whether or not the services of the broker to be exposed into JMX.

managementContext

This parameter configures how the ActiveMQ is exposed in JMX. The attributes are:

- `createConnector="true"` - sets whether or not the ActiveMQ creates its own JMX connector.

- `o connectorPort="1099"` - the port of the Connector. The value is 1099 by default.

persistenceAdapter/kahaDB

This parameter configures message persistence for the broker. The attributes are:

- `journalMaxFileLength="32mb"` - sets the maximum size of the message data logs.
- `checksumJournalFiles="true"` - creates a checksum for a journal file to enable checking for the corrupted journals.
- `checkForCorruptJournalFiles="true"` - if enabled, checks for corrupted journal files on startup and try and recover them.

transportConnectors

This parameter consists of transport connectors that the ActiveMQ listens to. The attributes are:

- `name="openwire"` - the name of the transport connector.
- `uri="tcp://localhost:61616"` - the address of the transport connector.

Note: For more information about the XML objects used in the XML configuration file, refer to the ActiveMQ's XBean XML Reference 5.0 at <http://activemq.apache.org/xbean-xml-reference-50.html>.

Logging

The ActiveMQ relies on log4j to log information in the broker client and the broker. The following listed lines inside `log4j.properties` configure the ActiveMQ logging by setting the default logging categories of ActiveMQ items.

- `log4j.logger.org.apache.activemq=INFO`
- `log4j.logger.org.apache.activemq.spring=WARN`
- `log4j.logger.org.apache.activemq.web.handler=WARN`
- `log4j.logger.org.springframework=WARN`
- `log4j.logger.org.apache.xbean=WARN`
- `log4j.logger.org.apache.camel=ERROR`

Using SSL with ActiveMQ

You can configure the ActiveMQ to use SSL connection by specifying the `<sslContext>` element and the correct transportConnector's URI in the XML configuration file of ActiveMQ.

ActiveMQ relies on certificates to use SSL connection. By default ActiveMQ is configured to reuse the IBM Security Directory Integrator Server API certificates located at the `TDI_install_folder/serverapi` folder as `keyStore` and `trustStore`. To reuse, the names of the client and server keystore files must be specified in the `<sslContext>` element of the configuration file of ActiveMQ. For example:

```
<sslContext>
  <sslContext
    keyStore="file:./serverapi/testadmin.jks" keyStorePassword="administrator"
    trustStore="file:./serverapi/testadmin.jks" trustStorePassword="administrator"/>
</sslContext>
```

Where, `testadmin.jks` is the name of the IBM Security Directory Integrator certificate and `password` is the password of the IBM Security Directory Integrator certificate.

Note: The `<sslContext>` element and all the parameters are taken into account only when the `javax.net.ssl` properties are not set in the IBM Security Directory

Integrator `solution.properties` file. By default, the ActiveMQ reuses the `javax` properties from the IBM Security Directory Integrator API instead of the properties set in the `<sslContext>` tag.

IBM WebSphere MQ Everyplace parameters

You can know more about IBM WebSphere MQ Everyplace and the required parameters using the information provided here.

In order to be able to use IBM WebSphere MQ Everyplace as the JMS provider for the System Queue an IBM WebSphere MQ Everyplace Queue Manager needs to be created. This can be done using the “IBM WebSphere MQ Everyplace Configuration Utility” on page 164 bundled with IBM Security Directory Integrator.

systemqueue.jmsdriver.param.mqe.file.ini

This is an IBM WebSphere MQ Everyplace-specific parameter that specifies the relative file system file name of the IBM WebSphere MQ Everyplace initialization file. This property is required and takes effect only if the IBM WebSphere MQ Everyplace JMS driver is specified in the `systemqueue.jmsdriver.name` property. The default value is `MqePWStore/pwstore_server.ini`. This is the default location for the IBM WebSphere MQ Everyplace initialization file created by the “IBM WebSphere MQ Everyplace Configuration Utility” on page 164.

The system queue is turned on by default. If you want to use IBM WebSphere MQ Everyplace as a system queue you then an abridged enabling procedure is as follows:

1. Set the `systemqueue.on` property in the `global.properties` or `solution.properties` file to `true`.
2. Configure IBM WebSphere MQ Everyplace by invoking:

```
cd solution_dir (if using the installation directory, use cd TDI_install_dir)
TDI_install_dir/jars/plugins/mqeconfig.sh
TDI_install_dir/jars/plugins/mqeconfig.props create server (one line)
```

IBM WebSphere MQ parameters

You can know more about IBM WebSphere MQ parameters using the information provided here.

These are IBM WebSphere MQ-specific parameters; for more information about these parameters, see the MQ JMS driver initialization properties in the “System Queue Configuration Example” on page 163 section.

systemqueue.jmsdriver.param.jms.broker

(IP address and TCP port number)

systemqueue.jmsdriver.param.jms.serverChannel

(server channel defined for the MQ server instance)

systemqueue.jmsdriver.param.jms.qManager

(name of the Queue Manager defined for the MQ server instance)

systemqueue.jmsdriver.param.jms.sslCipher

(cipher suite name corresponding to the cipher selected when configuring the MQ server channel, for example, `SSL_RSA_WITH_RC4128_MD5`)

systemqueue.jmsdriver.param.jms.sslUseFlag

(true for SSL connection requested, false if not)

Microbroker parameters

You can know more about Microbroker parameters using the information provided here.

In order to use Microbroker (MB) as the JMS provider for the System Queue, the `systemqueue.jmsdriver.name` property in `global.properties` or `solution.properties` must be set to `com.ibm.di.systemqueue.driver.IBMMB`.

The Microbroker driver has the following parameters (listed here without the "systemqueue.jmsdriver.param." prefix):

jms.broker

the MB server address (IP address and TCP port number); an example value would be "9.126.6.120:1883"

jms.clientID

the client ID; it is required.

Note: In order to be able to use Microbroker as the JMS provider for the System Queue, some Microbroker jars are needed. A sample list of the required jars is available in section External System Configuration, Microbroker of the JMS Connector in *Reference*.

JMSScript Driver parameters

You can know more about JMSScript Driver parameters using the information provided here.

The JMS Driver allows you to provide connectivity to any JMS provider through scripting in JavaScript, without writing and building Java code. The JMS Driver acts as a bridge between the System Queue and a user-specified piece of JavaScript, residing on the local file system, which is responsible for creating a `javax.jms.QueueConnectionFactory` object or a `javax.jms.TopicConnectionFactory` object. These objects are obtained in a provider-specific way.

systemqueue.jmsdriver.param.js.jsfile

This is a JMS Script Driver specific parameter (that is, taken into account when the `systemqueue.jmsdriver.name` is set to `com.ibm.di.systemqueue.driver.JMSScriptDriver`) that specifies the name of the file that contains the user-supplied JavaScript code to handle your JMS system of choice. For more information about this parameter, see the JMS driver settings in the "System Queue Configuration Example" on page 163 section. Note that the names of the Java properties do not have the `systemqueue.jmsdriver.param.` prefix.

systemqueue.jmsdriver.param.js.jsscript

The script body which contains JavaScript code for interfacing with the corresponding JMS provider. If this parameter is not provided, then the `systemqueue.jmsdriver.param.js.jsfile` parameter is used for loading the JavaScript to execute.

systemqueue.jmsdriver.param.user.xxxxx

These are user-defined properties which are passed by the System Queue to the configured JMS Driver implementation. For example if the following property is set:

```
systemqueue.jmsdriver.param.user.my.prop1=myvalue1
```

the configured JMS Driver get a property with a name of `user.my.prop1` and a value of `myvalue1`.

systemqueue.auth.username

This is the user name used by the System Queue for authentication to the configured JMS system. If this parameter has not been set then the System Queue does not use authentication to the configured JMS system.

systemqueue.auth.password

This is the password used by the System Queue for authentication to the configured JMS system. This parameter is only used when the `systemqueue.auth.username` parameter has been specified.

The env JavaScript object

You can learn more about env JavaScript object using the information provided here.

The piece of JavaScript executed by the JMS Driver needs access to a JavaScript object named *env*. This is an object of type `java.util.Hashtable`, which contains provider-specific parameters for connecting to the JMS provider. These parameters are intended to be used by the JavaScript code in order to access the specific JMS system server instance.

These parameters can be specified in `global.properties` or `solution.properties` using the `systemqueue.jmsdriver.param` prefix. For example, if a URL param is needed for some JMS system, then the following property can be set in `global.properties` or `solution.properties`:

```
systemqueue.jmsdriver.param.myjmssystem.url=myjmsserver.mydomain.com:12345
```

This definition would cause the System Queue to pass it to the JavaScript code as an entry in the env Hashtable, whose key would be "myjmssystem.url" (the System Queue removes the prefix) and whose value would be "myjmsserver.mydomain.com:12345".

The ret JavaScript object

You can know more about ret JavaScript object and the required parameters using the information provided here.

The piece of JavaScript executed by the JMS Driver has access to a JavaScript object named *ret*. This is an object of type `com.ibm.di.systemqueue.driver.JMSScriptDriver.Ret`. It is an instance of the *Ret* inner class of the JMS Script driver class. This *ret* object is to be used to return to the JMS Script driver and eventually to the System Queue the provider-specific objects which the JavaScript code obtains from the JMS system. The *ret* object can also be used to return any error information to the JMS Driver and the System Queue.

This ret object has the following members which can be set from JavaScript:

- `queueConnectionFactory` - an object of type `javax.jms.QueueConnectionFactory`. This is the place where the `javax.jms.QueueConnectionFactory` object obtained from the specific JMS system should be stored.
- `topicConnectionFactory` - an object of type `javax.jms.TopicConnectionFactory`. This is the place where the `javax.jms.TopicConnectionFactory` object obtained from the specific JMS system should be stored.

- `errorcode` - an object of type `java.lang.Object`. This is the place where any error information object should be stored. An example of such an object would be a `java.lang.Exception` object.
- `errordescr` - an object of type `java.lang.String`. This is the place where any textual error description should be stored.

JavaScript example for Fiorano MQ

You can use the example provided here to know more about Fiorano MQ.

An example configuration and JavaScript code to use the third-party Fiorano MQ system is provided in the `TDI_install_dir/examples` folder, and reproduced below:

```
var ctx = new Packages.java.util.Hashtable();
ctx.put("jms.username", "anonymous");
ctx.put("jms.password", "anonymous");
ctx.put("jms.broker", "http://192.168.113.220:1856");
ctx.put("jms.qManager", "fiorano.jms.runtime.naming.FioranoInitialContextFactory");

var ic = new javax.naming.InitialContext(ctx);

var queueFactory = ic.lookup("primaryQCF");
var topicFactory = ic.lookup("primaryTCF");

ret.queueConnectionFactory = queueFactory;
main.logmsg("driverFiorano.js : QueueConnectionFactory : " + queueFactory);

ret.topicConnectionFactory = topicFactory;
main.logmsg("driverFiorano.js : TopicConnectionFactory : " + topicFactory);
```

Note: This piece of JavaScript demonstrates how the parameters can be hard-coded in the JavaScript code. An alternative is to use the `env` JavaScript object to get any user-supplied parameters from `global.properties` or `solution.properties`. Using the `env` object for parameter retrieval would make changing the configuration easier, because only properties in `global.properties` or `solution.properties` would have to be changed, and no JavaScript code editing would be necessary. This means that users without JavaScript skills would be able to change the configuration.

System Queue Configuration Example

You can refer to the System Queue Configuration example provided here.

```
##-----
## System Queue settings
##-----
## If set to "true" the System Queue is initialized on startup and can be used;
## otherwise the System Queue is not initialized and cannot be used.
systemqueue.on=true

## Specifies the fully qualified name of the class that will be used as a JMS Driver.
# systemqueue.jmsdriver.name=com.ibm.di.systemqueue.driver.JMSScriptDriver
# systemqueue.jmsdriver.name=com.ibm.di.systemqueue.driver.IBMMQ
systemqueue.jmsdriver.name=com.ibm.di.systemqueue.driver.ActiveMQ

### MQe JMS driver initialization properties
## Specifies the location of the MQe initialization file.
## This file is used to initialize MQe on TDI server startup.
# systemqueue.jmsdriver.param.mqe.file.ini=MQePWStore/pwstore_server.ini

### MQ JMS driver initialization properties
systemqueue.jmsdriver.param.jms.broker=192.168.113.54:1414
systemqueue.jmsdriver.param.jms.serverChannel=S_s04win
systemqueue.jmsdriver.param.jms.qManager=QM_s04win
systemqueue.jmsdriver.param.jms.sslCipher=SSL_RSA_WITH_RC4128_MD5
systemqueue.jmsdriver.param.jms.sslUseFlag=true

### JMS Javascript driver initialization properties
## Specifies the location of the script file
# systemqueue.jmsdriver.param.js.jsfile=driver.js

### ActiveMQ driver initialization properties
```



```

## Specifies the location of the ActiveMQ initialization file.
## This file is used to initialize ActiveMQ on TDI server startup.
systemqueue.jmsdriver.param.jms.broker=vm://localhost?brokerConfig=xbean:etc/activemq.xml

## This is the place to put any JMS provider specific properties needed by a JMS Driver,
## which connects to a 3rd party JMS system.
## All JMS Driver properties should begin with the 'systemqueue.jmsdriver.param.' prefix.
## All properties having this prefix are passes to the JMS Driver on initialization after
## removing the 'systemqueue.jmsdriver.param.' prefix from the property name.
# systemqueue.jmsdriver.param.user.param1=value1
# systemqueue.jmsdriver.param.user.param2=value2
# ...

## Credentials used for authenticating to the target JMS system
# {protect}-systemqueue.auth.username=<username>
# {protect}-systemqueue.auth.password=<password>

```

Security and Authentication

You can know more about Security and Authentication and their respective methods through the information provided here.

Encryption

Of the standard JMS Drivers, only the driver for MQ supports SSL. The IBM WebSphere MQ Everyplace JMS Driver works only with a local Queue Manager - this is mandated by the IBM WebSphere MQ Everyplace architecture. The JMS Script Driver is a generic driver which supports whatever the corresponding user-provided JavaScript supports.

Authentication

Some JMS systems, such as IBM WebSphere MQ, can use or even require the use of user name and password authentication. The System Queue provides two standard properties in `global.properties` or `solution.properties` which can be used to configure and supply a user name and password to the System Queue. These properties are `systemqueue.auth.username` and `systemqueue.auth.password`. These two properties are protected by the standard IBM Security Directory Integrator server encrypting of properties which are marked as `{protect}-`. In this way after these properties are set and the IBM Security Directory Integrator server is started the properties' values get encrypted. For more information about these two properties, see the "System Queue Configuration" on page 157 section.

IBM WebSphere MQ Everyplace Configuration Utility

You can know more about IBM WebSphere MQ Everyplace Configuration Utility using the information provided here.

To configure and use IBM WebSphere MQ Everyplace as the default system queue, set up MQ Queue Manager in the new Solution Directory using IBM WebSphere MQ Everyplace Configuration Utility. The IBM WebSphere MQ Everyplace Queue Manger setup has two predefined queues:

- **_default** – serves as a general purposes queue
- **passwords** – serves as a general purposes queue passwords. This queue is used by the JMS Password Store components for storage of password changes. This makes the System Queue more usable.

The IBM Security Directory Integrator IBM WebSphere MQ Everyplace Configuration Utility (a command line utility) creates a default IBM WebSphere MQ Everyplace Queue when initially setting up the IBM WebSphere MQ Everyplace Queue Manager. This default IBM WebSphere MQ Everyplace Queue is

named "_default". This default Queue is created for convenience only – so that a user can use the IBM WebSphere MQ Everyplace Configuration Utility to set up IBM WebSphere MQ Everyplace (using the appropriate IBM WebSphere MQ Everyplace Configuration Utility command) and then start using the System Queue and the System Queue Connector right away.

Additionally the IBM Security Directory Integrator IBM WebSphere MQ Everyplace Configuration Utility can be used to create and delete user IBM WebSphere MQ Everyplace Queues to be used by the System Queue and the System Queue Connector.

Creating an IBM WebSphere MQ Everyplace Queue using the IBM WebSphere MQ Everyplace Configuration Utility

Typing the following command line creates an IBM WebSphere MQ Everyplace Queue named "queue_name" using the mqeconfig.props configuration file:

```
mqeconfig mqeconfig.props create queue queue_name
```

Deleting an IBM WebSphere MQ Everyplace Queue using the IBM WebSphere MQ Everyplace Configuration Utility

Typing the following command line delete the IBM WebSphere MQ Everyplace Queue named "queue_name" using the mqeconfig.props configuration file:

```
mqeconfig mqeconfig.props delete queue queue_name
```

If your solution needs any special configuration, then you can use the IBM WebSphere MQ Everyplace Explorer to fine tune your IBM WebSphere MQ Everyplace configuration. The IBM WebSphere MQ Everyplace Explorer is not bundled with IBM Security Directory Integrator, but can be downloaded as part of the IBM WebSphere MQ Everyplace Server Support ES06 pack at http://www-1.ibm.com/support/docview.wss?rs=0&dc=D400&q1=MQ&q2=MQ+Everyplace&uid=swg24007943&loc=en_US&cs=utf-8&cc=us&lang=en.

Authentication of IBM WebSphere MQ Everyplace messages to provide Queue Security

You can authenticate IBM WebSphere MQ Everyplace messages to provide Queue Security.

In IBM Security Directory Integrator access to IBM WebSphere MQ Everyplace can be secured by means of authentication using the IBM WebSphere MQ Everyplace Mini-Certificate Server to issue certificates to be used for authentication. For that purpose several additional properties available in IBM Security Directory Integrator must be added to the mqeconfig.props properties file, which contains the configuration properties of the IBM WebSphere MQ Everyplace Configuration Utility.

The certificates issued by the IBM WebSphere MQ Everyplace Mini-Certificate server have a configurable validity period. The default validity period is 12 months. The IBM WebSphere MQ Everyplace documentation states that issued certificates should be renewed before the period expires. To enable this, the IBM WebSphere MQ Everyplace configuration utility include an option to renew certificates. Typing the following command renews the certificates:

```
mqeconfig mqeconfig.props renewcert {client | server}
```

1. When the last command option is "client", the following values must be set in the mqeconfig.props file:

- **clientRootFolder** - The directory where IBM WebSphere MQ Everyplace configuration instance is located.
 - **certServerReqPin** - This value is used as a one time authentication PIN for the given authenticatable entity when requesting certificate renewal from the IBM WebSphere MQ Everyplace Mini-Certificate server.
 - **certServerIPAndPort** - This value is used as the destination address for IBM WebSphere MQ Everyplace Mini-Certificate server requests. The format of the value is "FastNetwork:<host>:<port>", where host must be the computer name or TCP IP address or hostname where the IBM WebSphere MQ Everyplace Mini-Certificate server is running.
 - **certRenewalEntityName** - The IBM WebSphere MQ Everyplace authenticatable entity name requiring certificate renewal. Typical entity names include those below, however, any entity name configured in the IBM WebSphere MQ Everyplace Mini-Certificate may be used assuming the entity does indeed exist in the queue manager registry referred to by the value of "clientRootFolder":
 - PWStoreClient – client side IBM WebSphere MQ Everyplace queue manager
 - PWStoreServer+passwords – remote queue proxy on the client side.
2. When the last command option is "server", the following values must be set in the mqeconfig.props file:
- **serverRootFolder** - The directory where IBM WebSphere MQ Everyplace configuration instance is located.
 - **certServerReqPin** - This value is used as a one time authentication PIN for the given authenticatable entity when requesting certificate renewal from the IBM WebSphere MQ Everyplace Mini-Certificate server.
 - **certServerIPAndPort** - This value is used as the destination address for IBM WebSphere MQ Everyplace Mini-Certificate server requests. The format of the value is "FastNetwork:<host>:<port>", where host must be the computer name or TCP IP address or hostname where the IBM WebSphere MQ Everyplace Mini-Certificate server is running.
 - **certRenewalEntityName** - The IBM WebSphere MQ Everyplace authenticatable entity name requiring certificate renewal. Typical entity names include those below, however, any entity name configured in the IBM WebSphere MQ Everyplace Mini-Certificate may be used assuming the entity does indeed exist in the queue manager registry referred to by the value of "serverRootFolder":
 - PWStoreServer – server side IBM WebSphere MQ Everyplace queue manager
 - PWStoreServer+passwords – real queue on the server side.

Support for DNS names in the configuration of the IBM WebSphere MQ Everyplace Queue

There is no additional coding required to support this feature.

It should be noted that DNS support is really an IBM WebSphere MQ Everyplace feature, since the IBM Security Directory Integrator component implementations simply pass the configuration properties from mqeconfig.props through to the IBM WebSphere MQ Everyplace APIs. The mqeconfig.props properties which can accept DNS name or IP address values are:

- serverIP
- certServerIPAndPort

Configuration of High Availability for IBM WebSphere MQ Everyplace transport of password changes

You can learn to configure High Availability for IBM WebSphere MQ Everyplace transport of password changes using the information provided here.

To support high availability deployments, you have the possibility to deploy and configure multiple instances of the IBM Security Directory Integrator IBM WebSphere MQ Everyplace components. In some deployments, it may be necessary to configure multiple IBM WebSphere MQ Everyplace Password Store components for IBM Security Directory Integrator. For example, if password change plug-ins have been configured for multiple Windows Domain Controllers—in this case, it is likely that there separate instances of IBM WebSphere MQ Everyplace client side Queue Managers with the name "PWStoreClient". Additionally, for each of the client Queue Managers, there is a remote queue proxy connection to the IBM WebSphere MQ Everyplace server side Queue Manager queue used by the IBM WebSphere MQ Everyplace Password Connector for IBM Security Directory Integrator. The remote queue proxy name is "PWStoreServer+passwords". When you use this type of deployment scenario, the authentication certificates associated with these two IBM WebSphere MQ Everyplace entities (that is, "PWStoreClient", "PWStoreServer+passwords") is requested and issued multiple times. This happens each time the mqeconfig utility is executed. Before executing the second and each subsequent instances of the mqeconfig utility, it necessary to re-enable certificate issue for each of the IBM WebSphere MQ Everyplace entities mentioned above.

For some deployments, you may prefer to configure the IBM WebSphere MQ Everyplace Password Connector for IBM Security Directory Integrator such that it supports a particular high availability requirement. You may expect that an implementation supporting this type of requirement would employ multiple instances of the IBM WebSphere MQ Everyplace Password Connector for IBM Security Directory Integrator, each with its own associated IBM WebSphere MQ Everyplace Queue Manager configuration. In this case you would deploy multiple identical IBM WebSphere MQ Everyplace server side configurations, allowing a network load balancer to route requests from the IBM Security Directory Integrator IBM WebSphere MQ Everyplace Password Store client to an available server instance. Each IBM WebSphere MQ Everyplace Queue Manager on the server side is configured using the mqeconfig utility. When this utility executes it automatically request authentication certificates from the IBM WebSphere MQ Everyplace Mini-Certificate server for the entities named "PWStoreServer" and "PWStoreServer+passwords". These represent the Queue Manager and Queue names respectively. Before executing the second and each subsequent instance of the mqeconfig utility, it necessary to re-enable certificate issue for the two IBM WebSphere MQ Everyplace entities mentioned above.

Providing remote configuration capabilities in the IBM WebSphere MQ Everyplace Configuration Utility

You can use the instructions provided here to provide remote configuration capabilities in the IBM WebSphere MQ Everyplace Configuration Utility.

Creating a remote IBM WebSphere MQ Everyplace Queue using the Configuration Utility

Typing the following command line create a remote IBM WebSphere MQ Everyplace Queue named "queue_name" using the mqeconfig.props configuration file:

```
mqeconfig mqeconfig.props create remotqueue queue_name targetQMname [QM_ip_or_hostname comm_port]
```

In the above command line QM_ip_or_hostname and comm_port parameters are optional; if they are missing only a remote queue definition is created. If you provide these two parameters, a Connection definition also be created before creating the remote queue definition.

Note: A remote queue is not usable without a Connection definition. In addition several remote queues can be defined to share a single Connection. The targetQMname parameter specifies the name of the remote IBM WebSphere MQ Everyplace Queue Manager.

Deleting a remote IBM WebSphere MQ Everyplace Queue using the IBM WebSphere MQ Everyplace Configuration Utility

Typing the following command line delete a remote IBM WebSphere MQ Everyplace Queue named "queue_name" using the mqconfig.props configuration file:

```
mqconfig mqconfig.props delete remotequeue queue_name targetQMname
```

In the above command line the targetQMname parameter specifies the name of the remote IBM WebSphere MQ Everyplace Queue Manager.

Chapter 9. Encryption and FIPS mode

You can learn about encryption through the information and links provided here.

To provide confidentiality of data, IBM Security Directory Integrator can encrypt:

- configuration files
- property values in properties files
- server user registry files
- JavaScript files

Encryption is the process of selecting some humanly readable text, called *plaintext*, and hiding its content and meaning to make the data in the plaintext format more secure. Plaintext is written in lowercase letters. Encrypted text is called *ciphertext*. Ciphertext is written in capital letters.

Note: In Config files, if the `{protect}-` prefix precedes the name of a property, then the property value is, or should be, encrypted. The prefix, `{protect}-` is optional. The values that are already encrypted values start with `{encr}`.

See “Working with encrypted IBM Security Directory Integrator configuration files” on page 133 and “Encryption of properties in external property files” on page 136.

For example:

```
[[protect]-]keyword <colon | equals> [[encr]][{java}]value
```

The `{java}` value must be b64-encoded. For example:

```
{protect}-api.truststore.pass={encr}J8AKimpEutu3Bb10Vg55F/5d5v02kXWcNUWnCq3vINUc6K0719z9dEk3H430t2iTT1dZTI6FSSV  
in9KsCyBlmgv+n84w7He1K13ro2dFmZbTYKMXuxGoqN9nL2V0vZoptNqzoWvs6IN/p3VkiIBt1ao/9mEPEKuIwRnKtKQ89B8g=
```

Configuring IBM Security Directory Integrator to run FIPS mode

You can configure IBM Security Directory Integrator to run FIPS mode.

The Federal Information Processing Standard (FIPS) Publication 140-2, FIPS PUB 140-2, is a U.S. government computer security standard used to accredit cryptographic modules.

When the IBM Security Directory Integrator server is configured to run in FIPS mode, that Server is using the FIPS 140-2 certified cryptographic modules. IBM Security Directory Integrator does not generate cryptographic keys – keys are created using external utilities such as *keytool* and *Ikeyman*). For information on IBM Security Directory Integrator use of encryption, see Chapter 6, “Security,” on page 93. In order to create, edit, export and overall manage keystores and truststores the Ikeyman GUI utility or the *keytool* command line utility can be used. The executable file, *keytool.exe* is found in *root_directory/jvm/jre/bin*, or *root_directory/jvm/bin*, depending on your platform.

Symmetric cipher support

You can use Symmetric cipher support for achieving a FIPS 140-2 compliance.

The reason for encrypting a message is to change the message into a meaningless form of text called cipher text that is meaningless to whoever intercepts the message. There are many different encryption algorithms called ciphers. One of the

most widely known ciphers is the *symmetric* cipher. The symmetric cipher has a key that both the sender and the receiver can keep. The sender uses that key to encrypt the message. The receiver uses the same key to decrypt the message.

An optional configuration is provided to use a symmetric cipher (specifically, the Advanced Encryption Standard, or AES). The symmetric cipher encoded using AES allows customers that need FIPS-compliant solutions to use a supported cipher around encryption.

The following property defines the cipher:

```
com.ibm.di.securityTransformation=DES/ECB/NoPadding
```

This property defines a cipher for the password-based encryption or decryption of IBM Security Directory Integrator configurations.

FIPS encryption

You can run IBM Security Directory Integrator and the IBM Security Directory Integrator server in a secure way using FIPS. You can also configure additional properties when you want to operate IBM Security Directory Integrator in a specific mode, for example, FIPS mode.

Connectors, Function Components, Parsers:

You can know more about Connectors, Function Components, Parsers using the information provided here.

FIPS 140-2 is concerned only with cryptographic functionality such as SSL, digital signing, encryption, cryptographic hashing and random number generation.

SSL

FIPS 140-2 requires TLS to be the protocol for SSL communication. SSLv3 and its predecessors are not allowed. When FIPS mode is turned on, IBM Security Directory Integrator components that use SSL will fail to communicate with external systems that do not support TLS.

JDBC and the System Store

The DB2 Type 4 JDBC driver (`com.ibm.db2.jcc.DB2Driver`) that is shipped with IBM Security Directory Integrator, supports SSL in a FIPS conformant way.

The Apache Derby drivers, network and embedded, do not support SSL in version 10.5.3 (which is the one bundled with IBM Security Directory Integrator versions prior to 7.2).

However, the Apache Derby Version 10.8 database engine can perform database encryption. By default IBM Security Directory Integrator uses Derby for its System Store. If you use database encryption functionality of Apache Derby Version 10.8 in FIPS mode, be sure to specify the IBM certified cryptographic provider `IBMJCEFIPS` as the provider used for encryption and also choose a FIPS approved encryption cipher. Here is an example of how to configure the System Store to use Derby with FIPS compliant database encryption:

```
com.ibm.di.store.database=jdbc:derby://localhost:1527/C:\TDI\TDISysStoreEnc;create=true;
  dataEncryption=true;encryptionKey=c566bab9ee8b62a5ddb4d9229224c678;encryptionAlgorithm=AES/CBC/NoPadding;
  encryptionProvider=com.ibm.crypto.fips.provider.IBMJCEFIPS
com.ibm.di.store.jdbc.driver=org.apache.derby.jdbc.ClientDriver
com.ibm.di.store.jdbc.urlprefix=jdbc:derby:
com.ibm.di.store.jdbc.user=APP
```


JMS and the System Queue

IBM WebSphere MQ Everyplace Mini-Certificates involve cryptography that is not FIPS compliant, so this security feature of IBM WebSphere MQ Everyplace should not be used in FIPS mode.

The IBM WebSphere MQ 5.3 JMS provider is not capable of running SSL in a FIPS compliant mode. In FIPS mode SSL should not be used with that provider.

To use FIPS compliant SSL communications between IBM Security Directory Integrator and IBM WebSphere MQ:

1. Ensure that the IBM WebSphere MQ installation is of Version 7.1 or higher.
2. Ensure that the corresponding Queue Manager on the MQ side requires FIPS compliant SSL communications.
3. Ensure that the corresponding SSL channel of the Queue Manager uses a FIPS compliant SSL Cipher Spec.
4. Turn on FIPS mode for IBM Security Directory Integrator. When FIPS mode is enabled for IBM Security Directory Integrator, it automatically enables FIPS mode on all JMS SSL connections to IBM WebSphere MQ.
5. Copy the JMS client jars from the IBM WebSphere MQ installation to IBM Security Directory Integrator; refer to the JMS Connector documentation in *Reference* for a list of necessary client libraries for MQ 7.1 and how to deploy them in IBM Security Directory Integrator.
6. On the IBM Security Directory Integrator side, configure a FIPS compliant SSL Cipher Suite that is compatible with the SSL Cipher Spec configured on the SSL channel of the MQ Queue Manager. You can do this using the `jms.sslCipher` parameter of the JMS Connector and the `systemqueue.jmsdriver.param.jms.sslCipher` system property of the MQ driver for the System Queue. For more information, see the section about *SSL CipherSpecs and CipherSuites* mapping and their FIPS compliance in the WebSphere MQ documentation.

The IBM Security Directory Integrator server and FIPS:

You can take care of the implications listed here while working on IBM Security Directory Integrator server and FIPS.

When run in this mode the IBM Security Directory Integrator Server is forced to use FIPS 140-2 cryptographic modules

Note: If the Server is running with FIPS and SSL enabled, then do not use clients with SSL for secure sockets communication. In this case the Server uses TLS and a connection will not succeed. Instead of using SSL make sure you are using TLS like the Server does for secure sockets communication.

Running the IBM Security Directory Integrator Server in FIPS mode has the following implications:

- Only FIPS compliant crypto algorithms are allowed for encryption and decryption of configurations, properties, and so forth.
- Auxiliary tools which use encryption/decryption should be used in FIPS compliant way - Ikeyman, createtash, cryptoutils, keytool, and so forth.
- Components will not be able to communicate with external systems that do not use TLS for socket communication.

- Some components should not be used when the Server is in FIPS mode because they will break the FIPS compliancy. Refer to Table 21 on page 173 for a list detailing component compliance.

Enabling FIPS mode:

When using FIPS, many IBM Security Directory Integrator configuration options are changed, so you must keep in mind several rules in order to maintain FIPS compliancy.

Some of the rules are mentioned in this document and others can be found in <https://w3.webahead.ibm.com/w3ki/download/attachments/370821/FIPS+140+Guidelines.pdf?version=1> and <http://www.ibm.com/developerworks/java/jdk/security/60/FIPShowto.html>.

Enabling FIPS mode in IBM Security Directory Integrator

1. Set the `com.ibm.di.server.fipsmode.on` property to **true** in `global.properties` or `solution.properties`.
2. Make sure the `com.ibm.di.securityTransformation` property value is in an algorithm which is FIPS compliant, for example, `AES/ECB/NoPadding`. This algorithm is used when you attempt to open an encrypted configuration.
3. Hardware cryptography can not be used along with SSL in FIPS mode. The underlying SSL module – IBMJSSE2 does not support hardware cryptography in FIPS mode as stated here: <http://www-128.ibm.com/developerworks/java/jdk/security/60/secguides/jsse2Docs/JSSE2RefGuide.html#runfips>. You cannot use hardware-based SSL keys for the Server API in FIPS mode; the `com.ibm.di.server.pkcs11` property must be absent or set to `false` in `global.properties` and `solution.properties`.
4. Make sure Server encryption uses a transformation that is FIPS 140-2 compliant.

By default the Server uses public key encryption with the RSA algorithm. However, the RSA encryption option is not compliant with FIPS 140-2. That is why you must manually configure another cryptographic transformation that is FIPS allowed. Below are sample steps that setup IBM Security Directory Integrator to use the AES cipher for encryption:

- Generate an AES secret key and put it in a keystore. This can be done using the `keytool` utility located in the `bin` folder of an IBM Security Directory Integrator installation like this:

```
keytool -genseckey -alias server -keyalg AES -keysize 128 -keystore server.jck -storepass mypass -storetype jceks
-keypass mykeypass -providerClass com.ibm.crypto.fips.provider.IBMJCEFIPS
```

This command creates a new keystore file `server.jck` of type JCEKS (JKS keystores cannot host secret keys) with an AES key of size 128 under alias `server`. The password for the created keystore is `mypass`. Pay special attention to the `keygenproviderclass` parameter – it is absolutely necessary to specify the FIPS certified provider if you strive for FIPS 140-2 compliance. Note that this is just an example, you can use whatever file names, passwords and aliases you wish.

- Change the IBM Security Directory Integrator settings to use secret key encryption with the newly generated key. For example in `global.properties` or `solution.properties` file, set the following properties:

```
com.ibm.di.server.encryption.keystore=server.jck
com.ibm.di.server.encryption.keystoretype=jceks
com.ibm.di.server.encryption.key.alias=server
com.ibm.di.server.encryption.transformation=AES/CBC/PKCS5Padding
```

- Migrate all existing files that have been encrypted with the old key:

All encrypted files that existed prior to the introduction of the new key, need to be migrated. Migration involves decryption with the old key and (optionally) re-encryption with the new one (see “Maintaining encryption artifacts - keys, certificates, keystores, encrypted files” on page 183). For example you can migrate `global.properties` as follows:

```
cryptoutils -input ../etc/global.properties -output ../etc/global.properties
-mode decrypt_props -keystore ../testserver.jks -storepass server -alias server
-transformation RSA -storetype jks -keypass server
```

```
cryptoutils -input ../etc/global.properties -output ../etc/global.properties
-mode encrypt_props -keystore ../server.jck -storepass mypass -alias server
-transformation AES/CBC/PKCS5Padding -storetype jceks -keypass mykeypass
```

- Regenerate the IBM Security Directory Integrator Server “Stash File” on page 131 to reflect the new passwords of the encryption keystore and the encryption key. This is done using the `createtash` utility found in the `bin` folder of an IBM Security Directory Integrator installation. For example:

```
createtash mypass mykeypass com.ibm.crypto.fips.provider.IBMJCEFIPS
```

- Use only FIPS compatible IBM Security Directory Integrator components in your solutions, as listed in the table below:

Table 21. FIPS compatible components

Directory Integrator Component	Allowed in FIPS mode?	Remarks
Connectors		
Active Directory Change Detection Connector	yes	Uses default JSSE factories for SSL
AssemblyLine Connector	yes	Operates as a Server API client
Axis Easy Web Service Server Connector	yes	Uses default JSSE factories for SSL
Command line Connector	yes	Provides no cryptography features
Domino/Lotus Notes Connectors	no	Domino/Notes 7 cryptographic capabilities are not FIPS conformant. (Some FIPS enablement may be included in Notes 8.0.1.)
ITIM DSMLv2 Connector	yes	Uses default JSSE factories for SSL
DSMLv2 SOAP Connector	yes	Uses default JSSE factories for SSL
DSMLv2 SOAP Server Connector	yes	Uses default JSSE factories for SSL
Exchange Changelog Connector	yes	Uses default JSSE factories for SSL
File Connector	yes	Provides no cryptography features
FTP Client Connector	yes	Provides no cryptography features
GLA Connector	yes	Provides no cryptography features. This connector is deprecated and will be removed in a future version of IBM Security Directory Integrator.
HTTP Client Connector	yes	Uses default JSSE factories for SSL
Old HTTP Client Connector	yes	Uses default JSSE factories for SSL
HTTP Server Connector	yes	Uses default JSSE factories for SSL
Old HTTP Server Connector	yes	Provides no cryptography features

Table 21. FIPS compatible components (continued)

Directory Integrator Component	Allowed in FIPS mode?	Remarks
IBM Security Directory Integrator Changelog Connector	yes	Uses default JSSE factories for SSL
ITIM Agent Connector	yes	Provides no cryptography features
JDBC Connector	depends	<p>If no cryptography features are used (SSL, encryption), the Connector is FIPS conformant.</p> <p>Otherwise FIPS conformance depends on the FIPS conformance of the cryptographic functionality of the JDBC driver that is used.</p> <p>See “Connectors, Function Components, Parsers” on page 170 for a discussion on the FIPS conformance of JDBC drivers.</p>
JMS Connector	depends	<p>If no cryptography features are used (SSL, encryption), the Connector is FIPS conformant.</p> <p>Otherwise FIPS conformance depends on the FIPS conformance of the cryptographic functionality of the JDBC driver that is used.</p> <p>See “Connectors, Function Components, Parsers” on page 170 for a discussion on the FIPS conformance of JMS providers.</p>
JMX Connector	yes	Provides no cryptography features
JNDI Connector	yes	Uses default JSSE factories for SSL
LDAP Connector	yes	Uses default JSSE factories for SSL
LDAP Server Connector	yes	Uses default JSSE factories for SSL
Mailbox Connector	yes	Uses default JSSE factories for SSL
Memory Queue Connector	depends	<p>Depends on the FIPS compliance of the JDBC driver used for the System Store.</p> <p>(The Memory Queue uses the System Store for persistence.)</p> <p>See “Connectors, Function Components, Parsers” on page 170 for a discussion on the FIPS conformance of JDBC drivers.</p>
Memory Stream Connector	yes	Provides no cryptography features
IBM WebSphere MQ Everyplace Password Store Connector	depends	<p>Only PKCS#7 is allowed in FIPS mode for message protection.</p> <p>The RSA encryption option must not be used. The IBM WebSphere MQ Everyplace Mini Certificates are not FIPS compliant, so they must not be used in FIPS mode.</p>
Sun Directory Change Detection Connector	yes	Uses default JSSE factories for SSL

Table 21. FIPS compatible components (continued)

Directory Integrator Component	Allowed in FIPS mode?	Remarks
Properties Connector	depends	<p>If encryption is turned off, the Connector is FIPS conformant.</p> <p>Otherwise FIPS conformance depends on the cipher used for encryption.</p> <p>An example of a FIPS 140-2 approved cipher is AES. Other approved ciphers can be found at: http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexa.pdf</p> <p>The Server encryption option will always be FIPS conformant as long as IBM Security Directory Integrator is configured correctly for FIPS mode. (See “Enabling FIPS mode” on page 172.)</p>
Server Notifications Connector	yes	Operates as a Server API client
System Queue Connector	depends	<p>If no cryptography features are used by the System Queue (SSL, encryption), the Connector is FIPS conformant.</p> <p>Otherwise FIPS conformance depends on the FIPS conformance of the JMS provider that is used by the System Queue.</p> <p>See “Connectors, Function Components, Parsers” on page 170 for a discussion on the FIPS conformance of JMS providers.</p>
Windows Users and Groups Connector	yes	Provides no cryptography features
System Store Connector	depends	Depends on the FIPS compliance of the JDBC driver used by the System Store.
RAC Connector	yes	Provides no cryptography features. This connector is deprecated and will be removed in a future version of IBM Security Directory Integrator.
RDBMS Changelog Connector	depends	Same as the JDBC Connector
SNMP Connector	yes	Provides no cryptography features
SNMP Server Connector	yes	Provides no cryptography features
TAM Connector	yes	IBM Security Directory Integrator Runtime for Java is FIPS conformant
TCP Connector	yes	Uses default JSSE factories for SSL
TCP Server Connector	yes	Uses default JSSE factories for SSL
Timer Connector	yes	Provides no cryptography features
URL Connector	yes	Provides no cryptography features
Web Service Receiver Server Connector	yes	Uses default JSSE factories for SSL
z/OS® Changelog Connector	yes	Uses default JSSE factories for SSL

Table 21. FIPS compatible components (continued)

Directory Integrator Component	Allowed in FIPS mode?	Remarks
Function Components		
Castor Java to XML FC	yes	Provides no cryptography features
Castor XML to Java FC	yes	Provides no cryptography features
EMF XMLToSDO	yes	Provides no cryptography features
EMF SDOToXML	yes	Provides no cryptography features
AssemblyLine FC	yes	Operates as a Server API client
Java Class Function Component	depends	<p>Depends on the FIPS compliance of the Java class, whose method will be invoked by the Function Component.</p> <p>If the Java class does not use cryptography (SSL, encryption, signing, cryptographic hash functions, and so forth) it can be safely used in FIPS mode.</p>
Parser FC	depends	Depends on the FIPS compliance of the Parser that is configured for the Function Component
CBE Generator Function Component	yes	Provides no cryptography features
SendEMail Function Component	yes	Uses default JSSE factories for SSL
Memory Queue FC	depends	<p>Depends on the FIPS compliance of the JDBC driver used by the System Store.</p> <p>(The Memory Queue uses the System Store for persistence.)</p> <p>See “Connectors, Function Components, Parsers” on page 170 for a discussion on the FIPS conformance of JDBC drivers.</p>
Axis Java To Soap FC	yes	Provides no cryptography features
WrapSoap FC	yes	Provides no cryptography features
Invoke Soap WS FC	yes	Uses default JSSE factories for SSL
Axis Soap To Java FC	yes	Provides no cryptography features
Axis EasyInvoke Soap WS FC	yes	Uses default JSSE factories for SSL
Complex Types Generator Function Component	yes	Provides no cryptography features
Remote Command Line Function Component	depends	<p>The cryptographic capabilities of the RXA toolkit are not FIPS compliant.</p> <p>If no cryptography features are used, the component can be used in FIPS mode.</p>
z/OS TSO/E Command Line FC	depends	Depends on the FIPS compliance of the cryptography involved in the TSO command that is invoked by the Function Component

Table 21. FIPS compatible components (continued)

Directory Integrator Component	Allowed in FIPS mode?	Remarks
SAP ABAP Application Server Component Suite	no	The SAP cryptographic module has not been FIPS 140-2 certified. If no cryptography features are used, the components can be used in FIPS mode.
Parsers	yes	None of the IBM Security Directory Integrator Parser components use cryptography so all of them can be used in FIPS mode.

Setting `com.ibm.di.server.fipsmode.on`

To enable FIPS mode in IBM Security Directory Integrator you must specify it in a property in `global.properties` or `solution.properties`. The property is named `com.ibm.di.server.fipsmode.on` and can be set to either **true** or **false**. When this property is set to **true**, the IBM Security Directory Integrator Server runs in FIPS mode. In this mode, the IBM FIPS security provider is set in the IBM Security Directory Integrator JVM before the IBM JCE security provider in the providers list. When the IBM Security Directory Integrator FIPS enabling property is true, it also enables FIPS mode in the IBM JSSE2 provider and sets the default JSSE SSL socket factories to be the ones from the IBM JSSE2 provider. By default FIPS mode is not enabled in IBM Security Directory Integrator, that is, the `com.ibm.di.server.fipsmode.on` property is set to **false**.

Using crypto algorithms in FIPS mode

Only FIPS-compliant crypto algorithms can be used. This means that you must use only FIPS-compliant algorithms in order to stay in FIPS-compliant mode. Using other algorithms violates FIPS compliancy.

Setting `com.ibm.di.securityTransformation`

When opening an encrypted configuration, IBM Security Directory Integrator uses the `com.ibm.di.securityTransformation` property to get the algorithm that decrypts the configuration. If this property is set to an algorithm which is not FIPS-compliant, and the IBM Security Directory Integrator Server FIPS mode is turned on, then an Exception is thrown. The Exception message which is shown would look something like this:

```
CTGDIC012E Could not load file<FILE_PATH>. No such algorithm: <ALGORITHM_NAME>.
```

In order to avoid this Exception, always set FIPS-compliant algorithms for this property when running in FIPS mode. By default the `com.ibm.di.securityTransformation` property is set to `DES/ECB/Nopadding` which is **not** a FIPS-compliant algorithm. This property also defines a cipher for the password-based encryption and decryption of IBM Security Directory Integrator configurations.

Setting properties automatically when running in FIPS mode

- IBM Security Directory Integrator sets a relevant System property which is not present in the `global.properties` file by default. This property is called `com.ibm.di.cryptoProvider` and is set to the `IBMJCEFIPS` security provider

when run in FIPS mode. You should note that if this property is set in `global.properties` then that particular value is used; if this property is set to a non-FIPS compliant provider, then even if IBM Security Directory Integrator is run in FIPS mode, IBM Security Directory Integrator is **not** FIPS-compliant.

- When in FIPS mode, specific JSSE Socket Factories are used. These are the IBMJSSE2 Socket Factories. This is done automatically by the IBM Security Directory Integrator Server which sets the `ssl.SocketFactory.provider` and the `ssl.ServerSocketFactory.provider` properties to the JSSE implementation classes in IBMJSSE2 provider.

Using the create stash file command line tool in FIPS mode

To create a stash file that conforms to FIPS 140-2 standards you must provide the IBMJCEFIPS provider class as the third parameter when using the `createstash` file tool. For example:

```
TDI_install_dir\bin\createstash Password Password com.ibm.crypto.fips.provider.IBMJCEFIPS
```

Using alternatives to RSA encryption in FIPS mode

In FIPS mode, configure IBM Security Directory Integrator to use the Advanced Encryption Standard (AES) instead of the RSA encryption algorithm. A secret key cipher that is FIPS 140-2 compliant is required. As an acronym, RSA stands for Rivest, Shamir, and Adelman, the inventors of the algorithm. The RSA algorithm is a strong encryption algorithm used for sending data over the internet. The RSA cipher is allowed only to encrypt and decrypt keys for transport (SSL, TLS) to stay within the boundaries of the Approved Mode of FIPS 140-2 Level 1, as stated at: <http://www.ibm.com/developerworks/java/jdk/security/60/FIPShowto.html>.

Running auxiliary tools in FIPS mode:

You can use the command line syntax for identifying the appropriate crypto provider, and when generating a secret key.

createstash

Pass the FIPS 140-2 certified crypto provider IBMJCEFIPS as an explicit provider parameter on the command-line:

```
createstash mypass mykeypass com.ibm.crypto.fips.provider.IBMJCEFIPS
```

cryptoutils

Pass the FIPS 140-2 certified crypto provider IBMJCEFIPS as an explicit provider on the command-line using the `cryptoproviderclass` option like this:

```
cryptoutils -input registry.txt -output registry.enc -mode encrypt -keystore ../testserver.jks -storepass server -alias server -cryptoproviderclass com.ibm.crypto.fips.provider.IBMJCEFIPS
```

Configuring FIPS properties for IBM Security Directory Integrator:

You can configuring FIPS properties for IBM Security Directory Integrator using the instructions provided here.

Running keytool/Ikeyman in FIPS mode

To use the keytool and Ikeyman utilities in FIPS mode, edit the java.security file in *TDI_install_dir/jvm/jre/lib/security*. In the first two lines in the java.security file, set the IBMJCEFIPS provider first and the IBMJCE security provider second. For example:

```
security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.2=com.ibm.crypto.provider.IBMJCE
```

However, on Solaris and HP-UX, the SUN provider should always be the first provider in the security providers list.

Configuring SSL and PKI certificates

You can know in detail about the SSL and PKI certificates through the information provided here.

IBM Security Directory Integrator uses both Secure Socket Layer (SSL) and Public Key Infrastructure (PKI) encryption methods. SSL and PKI provides an important foundation for many of the IBM Security Directory Integrator and IBM Security Directory Integrator server features. SSL provides for encryption and authentication of network traffic between two remote communicating parties. Similarly, PKI (public key infrastructure) enables users of unsecured networks to securely and privately exchange data by using a public and a private cryptographic key pair that is obtained and shared through a trusted authority. See “Configuring SSL and PKI certificates.”

SSL certificate

An SSL certificate resides on a secure server and is used to encrypt the data that identifies the server. The SSL certificate helps to prove the site belongs to the entity who claims it and contains information about the certificate holder, the domain that the certificate was issued to, the name of the Certificate Authority who issued the certificate, and the root and the country it was issued in.

PKI certificate

A PKI certificate enables users of an unsecured network to add security and privacy to data exchanges. PKI uses a cryptographic key pair that it gets and shares through a trusted authority called a Certificate Authority (CA). Using PKI, you can obtain a certificate that can identify an individual or an organization and directory services that can store the certificates. The CA can also revoke the certificates when necessary. The most common use of a digital certificate is to verify that a user sending a message is who the sender claims to be, and to provide the receiver with the encryption of the reply.

Follow these steps to provide separate configuration options for certificates to be used for PKI Encryption and SSL:

1. Add the following properties:

```
com.ibm.di.server.encryption.keystore
com.ibm.di.server.encryption.key.alias
api.keystore.password
api.key.password
```

2. Rename the following properties as shown:

```
com.ibm.di.server.keystore ----> api.keystore
com.ibm.di.server.key.alias ----->api.key.alias
```

Note: The `idisrv.sth` file now holds the password only for the encryption file.

Encrypting and decrypting using CryptoUtils

You can learn to encrypt and decrypt using CryptoUtils through the methods discussed here.

Using IBM Security Directory Integrator, you can PKI encrypt sensitive properties in the `global.properties` or in the `solution.properties` file. One method of decrypting PKI-encrypted properties is to use the Configuration Editor (CE) properties editor. The CryptoUtils command-line utility is another method of decrypting PKI-encrypted properties files. Decryption requires you to give your PKI credentials so that unauthorized users cannot access sensitive information. You can properties files that contain PKI encrypted properties using CryptoUtils. see "The IBM Security Directory Integrator Encryption utility" on page 136.

Working with certificates

You can learn to work the certificates using the information provided here.

Someone who wishes to send an encrypted message applies for a digital certificate from a CA. The CA issues an encrypted digital certificate that contains the applicant's public key and a other identification data. The CA makes reveals its own public key through printed media or perhaps on the Internet. The recipient of an encrypted message uses the CA's public key to decode the digital certificate attached to the message, verifies the certificate as issued by the CA, and then gets the sender's public key and identification data from the certificate. With this information, the recipient can send an encrypted reply.

Digital certificates are of two types:

- CA-signed certificates
- Self-signed certificates

CA-signed certificates are signed by a Certificate Authority such as VeriSign and thawte. A self-signed certificate is an identity certificate that is signed by its own creator.

IBM Security Directory Integrator provides separate configuration options for certificates to be used for public key infrastructure (PKI) encryption and Secure Socket Layer (SSL) connection. Independent configuration of PKI and SSL certificates allows you to migrate your encrypted properties separately from the process of upgrading your SSL certificates.

Under PKI, a Certificate Authority (CA) binds public keys to user identities. The user identity must be unique for each CA. Public Key certificates collect each user, user identity, public key, their binding, validity conditions, and other attributes that are made unforgetable in public key certificates issued by the CA.

The certificates used for SSL may expire, or for security reasons, SSL certificates may have to be refreshed frequently. Certificates used for PKI encryption can be persisted longer than it is appropriate to persist SSL certificates. PKI certificates should be maintained in case there is data that has been encrypted using the public key certificate. As a result, IBM Security Directory Integrator allows you to configure PKI and SSL certificates separately. Each server for an SSL connection and each client performing PKI authentication must issue a request for a certificate to the local CA, and must add the resulting certificate into its keystore.

These properties are added to the `global.properties` file:

```
com.ibm.di.server.encrypted.keystore
com.ibm.di.server.encrypted.key.alias
```

These properties variables are set to the same values as the ones already in `global.properties`:

```
api.keystore=truststore
api.key.alias=server
```

Comparing CA-signed and Self-signed certificates

You will be able to compare CA-signed and Self-signed certificates after reading the information provided here.

Certificate authority signed certificates

Self-signed certificates

Certificate authorities such as VeriSign require a procedure whereby applicants can prove their identities and obtain certificates that authenticate both the identity of the certificate applicants and its own identity as a signer of a certificate.

Typically there is a local certification authority (CA), that is, the certificates do not come from any of the well known CAs like VeriSign, and so on. The local CA itself should have a root certificate issued by a well-known CA, but even this is not always true. If the local CA's root certificate is self-signed, you must import it into the truststore of each server or client that is using SSL.

In this case, each server for an SSL connection, and each client doing PKI authentication, generates its own self-signed certificate. It is then necessary to export the certificate to a file and to import it into various truststores. If a client **C** connects to a server **S**, **C** must have **S**'s self-signed certificate in its truststore. If a client **C** does PKI authentication (symmetric SSL) to a server **S**, **S** must have **C**'s self-signed certificate in its truststore. Note: Self-signed certificates can be used for either a client or a server certificate. See the "Manage keys, certificates and keystores" on page 93 on information how to do this. Each server for an SSL connection and each client doing PKI authentication must then issue a request for a certificate to the local CA, and must add the resulting certificate into its keystore.

Configuring certificates using PKI and SSL

You can learn to configure certificates using PKI and SSL using the information provided here.

IBM Security Directory Integrator provides separate configuration options for certificates to be used for public key infrastructure (PKI) encryption and Secure Socket Layer (SSL) connection. Independent configuration of PKI and SSL certificates allows you to migrate your encrypted properties separately from the process of upgrading your SSL certificates.

Under PKI, a Certificate Authority (CA) binds public keys to user identities. The user identity must be unique for each CA. Public Key certificates collect each user, user identity, public key, their binding, validity conditions, and other attributes that are made unforgettable in public key certificates issued by the CA.

The certificates used for SSL may expire, or for security reasons, SSL certificates may have to be refreshed frequently. Certificates used for PKI encryption can be persisted longer than it is appropriate to persist SSL certificates. PKI certificates should be maintained in case there is data that has been encrypted using the

public key certificate. As a result, IBM Security Directory Integrator allows you to configure PKI and SSL certificates separately. Each server for an SSL connection and each client performing PKI authentication must issue a request for a certificate to the local CA, and must add the resulting certificate into its keystore.

These properties are added to the `global.properties` file:

```
com.ibm.di.server.encryption.keystore  
com.ibm.di.server.encryption.key.alias
```

These properties variables are set to the same values as the ones already in `global.properties`:

```
api.keystore=truststore  
api.key.alias=server
```

Using cryptographic keys located on hardware devices

You can learn to use cryptographic keys located on hardware devices using the instructions provided here.

The RSA signing and encryption algorithm (developed by Ron Rivest, Adi Shamir, and Leonard Adleman) is a well-known public key cipher. RSA Laboratories (Part of EMC Corp.) have published the PKCS#11 standard, which defines a platform-independent API to hardware cryptographic tokens, such as Hardware Security Modules and smart cards. The PKCS#11 API defines most commonly used cryptographic object types, including:

- RSA keys
- X.509 Certificates
- Data Encryption Standard (DES)DES/Triple DES keys
- All the functions required for using, creating or generating, modifying, and deleting the above objects

Public-Key Cryptography Standards (PKCS) PKCS#11 is a standard that provides a common application interface to cryptographic services on various platforms using various hardware cryptographic devices. Hardware Cryptographic key storage devices allow keys to be stored on hardware devices. IBM Security Directory Integrator supports private keys and certificates on crypto devices that are PKCS#11 compliant. Support is provided on all hardware devices supported by the IBM Java PKCS libraries shipped with the Java Runtime Environment (JRE). PKCS standards are a set of common protocols that allow secure information exchange over networks using a public key infrastructure (PKI). IBM Security Directory Integrator can store Secure Socket Layer (SSL) keys on the hardware devices. For the requirement to store keys on hardware devices, the following new properties are available in the `global.properties` file:

```
##PKCS11 options  
##Set the value of following properties to use PKCS11 enabled devices to store TDI servers  
##private key /certificate.  
com.ibm.di.pkcs11cfg=etc\pkcs11.cfg  
com.ibm.di.server.pkcs11=false  
com.ibm.di.server.pkcs11.library=  
com.ibm.di.server.pkcs11.slot=  
{protect}-com.ibm.di.server.pkcs11.password=PASSWORD
```

The default value of the property `com.ibm.di.server.pkcs11` is `false`. The value corresponding to the property `com.ibm.di.server.pkcs11.password` is encrypted.

Using IBMPCS11

You can use IBMPCS11 to access devices and to store SSL keys and certificates.

IBM Security Directory Integrator uses IBMPCS11 to access crypto hardware devices that store the SSL keys and certificates. Support is provided for all hardware devices supported by the IBM Java PKCS libraries and shipped with the Java Runtime Environment.

Table 22. SSL supported properties

Property	Default value	Description
com.ibm.di.pkcs11.cfg	etc\pkcs11.cfg	Use CFG file to point to the path of the configuration file required to initialize the IBM PKCS11 implementation provider.
com.ibm.di.server.pkcs11	false	Use PKCS#11 compliant crypto devices for ssl.
com.ibm.di.server.pkcs11.library		Use this property to specify the path to the PKCS11client library.
com.ibm.di.server.pkcs11.slot		Specify the slot number of the device.
{protect}-com.ibm.di.server.pkcs11.pass		Use this password to access the pkcs11 compliant crypto device.
com.ibm.di.server.pkcs11.accl	false	Use =true to set hardware cryptographic devices for cryptographic operations.

Enabling or disabling padding

You can enable or disable padding using the information and links provided here.

Padding means adding extra bits to a transmission so that the transmission is the exact, required, size. Some encryption and decryption algorithms require their input to be an exact multiple of the block size. If the plaintext to be encrypted is not an exact multiple, you must *pad* before encrypting by adding a padding string. When decrypting let the receiving party know how to remove the padding.

Note: All properties listed in the `global.properties` file can be set in the configuration file by the same name; it is recommended that you edit your `solution.properties` file instead if you have one. These properties can be protected by encryption using the `{protect}-` prefix (see section “Standard encryption of `global.properties` or `solution.properties`” on page 135 for details). When setting the property for padding, the default value is `DES/ECB/NoPadding`. The padding property defines an algorithm or cipher for password-based encryption and decryption of IBM Security Directory Integrator configurations. The property is: `com.ibm.di.securityTransformation`.

Maintaining encryption artifacts - keys, certificates, keystores, encrypted files

You can learn about maintaining encryption artifacts - keys, certificates, keystores, encrypted files using the information and links provided here.

Note: The default SSL certificates of IBM Security Directory Integrator are changed in 7.1.1. Therefore, IBM Security Directory Integrator 7.1.1 fails to decrypt the following items, if encrypted using the default IBM Security Directory Integrator certificates:

- encrypted passwords in `global` or `solution.properties`
- protected properties in external property stores

- encrypted IBM Security Directory Integrator configuration XML files in previous versions

Therefore, decrypt all the encrypted passwords using your previous version of IBM Security Directory Integrator. For more information about encryption utility, see “The IBM Security Directory Integrator Encryption utility” on page 136. Once the passwords are retrieved as text, use them in the latest version. The passwords are encrypted with the new default certificates once the server or Configuration Editor is started.

Changed encryption key

Any change of the key that the Server uses for encryption leads to a need for migration of existing encrypted files. To migrate an encrypted file, you should decrypt it with the old encryption key and encrypt it with the new one. Encryption and decryption can be done using the “The IBM Security Directory Integrator Encryption utility” on page 136 tool.

Files which are often encrypted or contain encrypted parts are: “Working with encrypted IBM Security Directory Integrator configuration files” on page 133, the “Server API User Registry” on page 125 and “Standard encryption of global.properties or solution.properties” on page 135 (IBM Security Directory Integrator properties files can contain encrypted properties, although the files are usually not encrypted as a whole).

Note: By default all sensitive properties (such as passwords) inside `global.properties` or `solution.properties` are encrypted. As a rule of thumb you should always migrate `global.properties` and `solution.properties` files when you change the Server encryption key.

Changed password for encryption key or keystore

The Server reads the password for the keystore that holds the encryption key and the password for the encryption key itself from the Server “Stash File” on page 131. Thus if any of those passwords is changed, the stash file must be updated. This can be done using the `createtash` tool.

Expired encryption certificate

If the Server uses public-key encryption, the certificate associated with the encryption key-pair can potentially expire at some point in time. If this happens, the certificate can be renewed using the procedure described in section “Extend the validity of a certificate using keytool”. That procedure preserves the underlying keys, so no migration of existing encrypted files is necessary.

Chapter 10. Configuring the IBM Security Directory Integrator Server API

You can use the information and links provided here to configure the IBM Security Directory Integrator Server API.

The IBM Security Directory Integrator Server API provides a set of programming calls that can be used to develop IBM Security Directory Integrator solutions and interact with the server locally and remotely. It also includes a management layer that exposes the Server API calls through the Java Management Extensions (JMX) interface. This section provides information about properties you can use to configure the Server API.

- For information on using the Server API, see "Appendix C. Server API" in the *Reference*.
- For additional information on configuring the server API, see "Configuring the Server API" on page 108.
- For information on IBM Security Directory Integrator server security, see "Remote Server API" on page 107.

Server ID

You can know more about server ID and their working using the information provided here.

Remote clients can identify the server they are talking to if the server can be distinguished using a unique ID. IBM Security Directory Integrator allows you to specify unique server IDs that allow remote clients, such as the Administration and Monitoring Console (AMC), to connect to different IBM Security Directory Integrator servers at different times using the same IP and port. In order to connect using the same ID and port but at different times, IBM Security Directory Integrator client applications must be able to register these client applications as different IBM Security Directory Integrator servers that you can associate with different data and databases.

Users are to assign the unique IDs manually, ensuring that any remote client (such as AMC) can connect to an IBM Security Directory Integrator server based on the IP address, the port, and the unique ID of the IBM Security Directory Integrator server. AMC registers every server with a unique ID, so that an IBM Security Directory Integrator server cannot be registered more than once by mistake or intentionally. When assigning IDs manually, users must ensure that different IBM Security Directory Integrator servers have distinguishable IDs.

You can configure the unique server ID property using `com.ibm.di.server.id`. To give a server a unique server id for a given server, provide a unique ID string for this property in the `global.properties` or `solution.properties` file on the server you are identifying. The default value for `com.ibm.di.server.id` is blank.

Exception for password protected Configs

The server API throws an exception if you use a password protected Config without using a password. You can know more about it through the information provided here.

The IBM Security Directory Integrator server API can detect and handle server problems when the server is faced with clients trying to access password protected configurations without supplying a password. A message displays, notifying the user about the problem. The error message is invoked when no password is supplied or the when the password entered is wrong. See “AMC and encrypted configs” on page 255.

Server RMI

You can know more about Server RMI using the information and links provided here.

Due to increasing needs for remote access to each IBM Security Directory Integrator server, Remote Method Invocation (RMI) is enabled by default. To ensure adequate security, default remote access requires Secure Socket Layer (SSL) for client authentication. The SSL access is facilitated with the sample keystore and truststore that are deployed with IBM Security Directory Integrator. See “SSL client authentication” on page 102 and “Summary of Server API Authentication options” on page 121.

Config load time-out interval

Use the `api.config.timeout` property to add a time-out interval.

If a config instance does not completely load the configuration file when the server API makes a call, the server API returns a null object. Add the `api.config.timeout` property in the `global.properties` file to add a time-out interval. The interval for loading the configuration file is set to two minutes by default. If the config file does not load within the time interval, an exception is thrown.

Chapter 11. Properties

You can use Properties to configure IBM Security Directory Integrator components and the IBM Security Directory Integrator Server.

Properties are simple `keyword:value` pairs of parameters kept outside your configuration files (configs), stored in External Properties files. This enables you to keep confidential information like passwords outside of your Config files. The `global.properties` file is the main configuration file for IBM Security Directory Integrator. Properties are defined in the `global.properties` file or in the `solution.properties` file. The `solutions.property` file is a writable copy of the `global.properties` file and is used when the server is started from the solution directory. If a solution directory different from the installation directory is specified during installation, then a copy of the `global.properties` file named `solution.properties` is created in the IBM Security Directory Integrator solution directory. Both files are text files, and are written so that they can be understood by the operating system that is running on the platform.

Properties are single-valued data containers that hold parameter information, for example, `true` or `5000`. You can access properties from script using entry functions like `getProperty()` and `setProperty()`. Get and set methods work directly with property values. Entry objects can also contain properties. Like Attributes, properties are data containers. Attributes are used to store data content, but properties hold parametric information. Property values and Attributes can be any type of Java Object. Properties do not show up in:

- Attribute map selection.
- Work entry lists.

Working with properties

You can learn to work with the Solution Directory in multiple ways using the information provided here.

This section introduces the primary concepts you need when working with properties. Properties set in any `properties` files form a baseline for the entire IBM Security Directory Integrator installation for all users on that computer. However, if your Solution Directory is different from the installation directory, you can have a set of text files in your Solutions Directory that mirror their counterparts in the installation directory. A property listed in any of those files override anything set in any of the global installation property files, including the `global.properties` and `solution.properties` files. Furthermore, a Java property set inside a Config file takes the highest precedence, and overrides anything in a global property file or the property files in the Solution Directory.

You can specify the Solution Directory in multiple ways:

- Set the environment variable `TDI_SOLDIR` before starting the Configuration Editor or the Server.
- Specify the `-s` parameter to the `ibmditk` script to start the Configuration Editor or the `ibmdisrv` script to start the IBM Security Directory Integrator Server. This takes precedence over setting `TDI_SOLDIR`. If `TDI_SOLDIR` equals the installation directory, all property files is read from there, and the remarks about property files in the Solutions Directory do not apply.

In any other case, the first time you run the IBM Security Directory Integrator Server, it makes a copy of all the property files into your Solutions Directory (it does not overwrite these files if they already exist). You can now tailor these files to your particular needs, without affecting the property files in the installation directory. The files remaining in the installation directory continue to form a baseline configuration for other instances of IBM Security Directory Integrator.

Note: The file `global.properties` is copied to a file called `solutions.properties` in your Solutions Directory. Other files, like `Log4J.properties` and the files in the `amc` and `serverapi` folders are copied under their own names.

For documentation purposes, the original `global.properties` file from the installation directory is copied to the `<Solution directory>/etc` folder; this file is **not** used for any other purpose.

Migrating using properties and the `tdimigbl` tool

The `tdimigbl` tool helps you to migrate your `global.properties` file from one version of IBM Security Directory Integrator to a higher version. See the following section Chapter 5, “Migrating,” on page 61.

Global properties

You can use Global properties to configure the IBM Security Directory Integrator Server settings that are kept in a file called `global.properties` in the `etc` folder of your installation directory.

All properties included in the `global.properties` file are listed with their default values and explained in this section. A reference to more detailed documentation is provided, where possible, in the beginning of the groups of properties. The Configuration Editor (CE) (`ibmditk`) and the IBM Security Directory Integrator server (`ibmdisrv`) read the `global.properties` file on startup. This file is read and applied before a file called `solution.properties` from your Solution Directory is read.

Table 23. Some important IBM Security Directory Integrator global properties

Use of the property	Property	Default value	Description
Add your own .jar or .zip files	<code>com.ibm.di.loader.userjars</code>	<code>c:\myjars</code>	Specifies directories or jar files, separated by the Java Property "path.separator", which is ":" on Linux and ";" on Windows. Directories are searched recursively by the <code>TDILoader</code> for jar files containing classes and resources. Only files with a <code>.zip</code> or <code>.jar</code> extension are searched.
Define cipher	<code>com.ibm.di.securityTransformation</code>	<code>DES/ECB/NoPadding</code>	Defines a cipher for the password-based encryption or decryption of IBM Security Directory Integrator configurations. Changed in IBM Security Directory Integrator 7.0.
Enable config autoload	<code>com.ibm.di.server.autoload</code>	<code>autoload.tdi</code>	Looks for <code>*.xml</code> files in the directory specified by the <code>"ibmdisrv -d"</code> command. Executes each <code>*.xml</code> file found in the directory defined by <code>-d</code> .

Solution properties

Solution properties typically override global properties and are found in a file in your solution directory called `solution.properties`. You can know more about it through the information provided here.

The `solution.properties` file is by default a copy of the `global.properties` file, and you should edit the `solutions.properties` file when configuring IBM Security Directory Integrator, because it is read last out of all the properties files. If you want to, you can delete properties in your `solution.properties` file and add property configuration statements that you specifically want to override the `global.properties` defaults.

Java properties

You can know more about Java properties using the information provided here.

Java properties are variables and settings of the Java Virtual Machine (JVM). Java log (Jlog) file properties are shown in “Useful JLOG parameters” on page 235.

Note: A Java property set inside a Config file takes the highest precedence, and overrides anything in a global property file or the property files in the Solution Directory.

Table 24. Java properties

Property	Default value	Description
<code>javax.net.debug</code>	none	Sets debug mode for the JSSE provider.
<code>com.ibm.di.javacmd</code>	none	Overrides the Java interpreter.
<code>com.ibm.di.installdir</code>	none	Uses this path to the Java executable file when running AssemblyLines from the Configuration Editor.
<code>com.ibm.di.jvmdir</code>	<code>TDI_root/jvm</code>	Defines the directory path where the JRE that IBM Security Directory Integrator uses is installed.
<code>com.ibm.di.server.maxThreadsRunning</code>	500	Sets this number of threads IBM Security Directory Integrator. Must be set higher than 3 to have any effect.
<code>com.ibm.di.server.securemode</code>	false	Sets the mode in which IBM Security Directory Integrator is running. (standard or secure)
<code>com.ibm.di.server.keystore</code>	<code>testserver.jks</code>	Names the keystore of the Server’s SSL certificate. Renamed in IBM Security Directory Integrator 7.0.
<code>com.ibm.di.server.key.alias</code>	server	Names the key alias of the Server’s SSL certificate. Renamed in IBM Security Directory Integrator 7.0.
<code>{protect}-api.keystore.password</code>	server (encrypted by default)	Provides the password for the server API keystore. Added in IBM Security Directory Integrator 7.0.
<code>{protect}-api.key.password</code>		Provides the key password. If not specified, uses server keystore password. Added in IBM Security Directory Integrator 7.0.
<code>com.ibm.di.server.encryption.keystore</code>	<code>testserver.jks</code>	Names the keystore of the server encryption key. Added in IBM Security Directory Integrator 7.0.
<code>com.ibm.di.server.encryption.key.alias</code>	server	Provides the key alias of the server encryption key. Added in IBM Security Directory Integrator 7.0.
<code>com.ibm.di.server.encryption.keystoretype</code>	jks	Provides the type of the keystore that hosts the key used by the server for encryption. Added in IBM Security Directory Integrator 7.0.
<code>com.ibm.di.server.encryption.transformation</code>	RSA	Names the cryptographic transformation used by the server for encryption. Can be set to either "RSA" (public key encryption) or to some secret key transformation. Added in IBM Security Directory Integrator 7.0.

Table 24. Java properties (continued)

Property	Default value	Description
com.ibm.di.server.fipsmode.on	false	Enables or disables FIPS standards in IBM Security Directory Integrator. If this property is set to true, IBM Security Directory Integrator runs in FIPS-compliant mode. For more information on FIPS mode, see Added in IBM Security Directory Integrator 7.0.
com.ibm.di.default.bind.address	*	The default bind address for the whole IBM Security Directory Integrator Server - the components and the Server API.

System properties

System properties are stored in the System Store instead of being stored in an external properties file such as `solution.properties`. You can know more about this through the information and link provided here.

Certain system properties and Java properties are read-only. These system properties are shown in the respective Property Stores (for example, System Store). Attempting to modify these read-only properties has no effect. See also Chapter 12, "System Store," on page 191.

Chapter 12. System Store

You can know more about System Store and its working using the information provided here.

IBM Security Directory Integrator supports persistent storage (that is, storage of objects that survive across JVM restarts), by means of a tightly-coupled relational database, the System Store.

The product deployed by default to implement the system store is a relational database implemented fully in Java, known as IBM Security Directory Integrator, and previously known as Cloudscape.

The System Store stores the following objects:

- Delta Tables
- User Property Store
- Password Store

Default location of System Store

The default location of IBM Security Directory Integrator System Store database, in network mode, is your Solution Directory. Therefore, you can have a System Store for each of your Solution Directories.

To share a single System Store across all the Solution Directories, replace `$soldir$` value with the actual `TDI_install_dir` in the `com.ibm.di.store.database` property of `global.properties` file and `solution.properties` file, if already created.

When you create a Solution Directory, update the following properties in the `solution.properties` file with unique values to avoid conflicts with other Solution Directory settings:

- `com.ibm.di.store.port=1527`
- `api.remote.naming.port=1099`
- `web.server.port=1098`
- System Queue port or Active MQ port in `<soldir>/etc/activemq.xml`

Note:

The following example describes the effect of specifying the same value for `com.ibm.di.store.port` property across multiple Solution Directories.

There are two solution directories (`soldir1`, `soldir2`) with the same `com.ibm.di.store.port` values (1527, 1527), and with unique `api.remote.naming.port` values (1099, 41099).

When you start server on `soldir1`, the server starts on port 1099 and System Store on port 1527, inside `soldir1`.

When you start Server on `soldir2`, the server starts on port 41099 and connects to the System Store, which is already listening on port 1527 inside `soldir1`.

The remainder of this section discusses the operational aspects of using IBM Security Directory Integrator, in particular in conjunction with using IBM Security Directory Integrator to hold your System Store.

Note: With regards to third party RDBMSs, in order to hold encrypted password values you may have to size the fields that hold them quite large. A typical small password might use as much as 178 characters. It depends on both your server's key, and the length of the unencrypted data you try to store (in bytes). Since this is a blocked encoding a larger password might use the same space, or double or triple that amount. Also, the size of the block depends on the server's key. One way to find the size you need, is to store the password (protected) to a file first, and then look at that file to see how many characters were used.

IBM Security Directory Integrator can run in either of two modes: *embedded* and *networked*. By default, as specified in the `global.properties` file, IBM Security Directory Integrator runs in *networked* mode.

The System Store used by IBM Security Directory Integrator releases before V7.0 was IBM Security Directory Integrator (then called Cloudscape) in *embedded* mode. There are drawbacks to the way IBM Security Directory Integrator runs in embedded mode. In embedded mode, IBM Security Directory Integrator runs as a separate thread within the JVM when required. Startup and shutdown of IBM Security Directory Integrator is automatic in embedded mode. However, when run this way, this IBM Security Directory Integrator thread claims exclusive access to the database files. This can become problematic when different JVMs, each with its own IBM Security Directory Integrator thread, try to access the same System Store.

In embedded mode, these actions cause a new, independent JVM to be started, triggering an access conflict when more than one JVM is active at any given time:

- A command line invocation of the IBM Security Directory Integrator Server with a config file, causing one or more AssemblyLines to run.
- Startup of the Configuration Editor (GUI)
- Startup of an AssemblyLine from within the Configuration Editor

None of these actions by themselves causes the IBM Security Directory Integrator thread to start. However, the IBM Security Directory Integrator thread does start if access to any of the objects in the System Store is required (for example, Objects supported by the System Store such as Delta Tables and the User Property Store).

The solution to the access conflicts as outlined previously is to run IBM Security Directory Integrator in *networked* mode, which enables concurrent access to the System Store. Also enable user authentication in IBM Security Directory Integrator to avoid security concerns in networked mode. To provide security at the database level, IBM Security Directory Integrator uses the BUILTIN security provider for IBM Security Directory Integrator. BUILTIN ensures that only valid users are able to access the IBM Security Directory Integrator database. When you have IBM Security Directory Integrator configured in networked mode, you can work with multiple instances of IBM Security Directory Integrator databases booted as System Stores. You can also configure a IBM Security Directory Integrator instance to work with a specific Configuration file instance.

Note: Depending on how IBM Security Directory Integrator was started, instances of IBM Security Directory Integrator can be left running in networked mode, even after all other IBM Security Directory Integrator processes have terminated.

When you set the property `derby.drda.startNetworkServer` to true (by default, this is the case, in `global.properties`), the Network Server automatically starts when you start IBM Security Directory Integrator (in this context, IBM Security Directory Integrator starts when the embedded driver is loaded). You may have to terminate IBM Security Directory Integrator manually, if desired.

Cloudscape command-line utility

To make working with the IBM Security Directory Integrator database more convenient, consider creating a script ("dbserver") with the following line (this example is for Unix/Linux):

```
export DB_JAR_DIR=jars/3rdparty/IBM
export DB_CLASSPATH=$DB_JAR_DIR/derby.jar:$DB_JAR_DIR/derbyclient.jar:\
$DB_JAR_DIR/derbynet.jar:$DB_JAR_DIR/derbytools.jar
java -classpath $DB_CLASSPATH org.apache.derby.drda.NetworkServerControl "$@"
```

You may have to join the middle two lines together at the "\" point.

The equivalent `dbserver.bat` file for Windows would be:

```
set DB_JAR_DIR=jars/3rdparty/IBM
set DB_CLASSPATH=%DB_JAR_DIR%\derby.jar;%DB_JAR_DIR%\derbyclient.jar;\
%DB_JAR_DIR%\derbynet.jar;%DB_JAR_DIR%\derbytools.jar;
java -classpath %DB_CLASSPATH% org.apache.derby.drda.NetworkServerControl %*
```

Note: The script must be started from within the IBM Security Directory Integrator installation path as the working directory, as the following classpath is relative to this directory.

The following is an example of usage of this utility script:

```
Show all available commands: ./dbserver
Start DBServer ./dbserver start -p 1527
Stop DBServer ./dbserver shutdown
```

The full list of sub-commands that you can specify to the `dbserver` script, and which are sent to IBM Security Directory Integrator is:

- `start [-h <host>] [-p <portnumber>]`: This starts the network server on the port/host specified or on localhost, port 1527 if no host/port is specified and no properties are set to override the defaults. By default Network Server will only listen for connections from the machine on which it is running. Use `-h 0.0.0.0` to listen on all interfaces or `-h <hostname>` to listen on a specific interface on a multiple IP machine.
- `shutdown [-h <host>] [-p <portnumber>]`: This shutdowns the network server on the host and port specified or on the local host and port 1527 (default) if no host or port is specified.
- `ping [-h <host>] [-p <portnumber>]`: This will test whether the Network Server is up.
- `sysinfo [-h <host>] [-p <portnumber>]`: This prints classpath and version information about the Network Server, the JVM and the Cloudscape server.
- `runtimeinfo [-h <host>] [-p <portnumber>]`: This prints extensive debugging information about sessions, threads, prepared statements, and memory usage for the running Network Server.
- `logconnections {on | off} [-h <host>] [-p <portnumber>]`: This turns logging of connections and disconnections on and off. Connections and disconnections are logged to `derby.log`. Default is off.

- `maxthreads <max> [-h <host>] [-p <portnumber>]`: This sets the maximum number of threads that can be used for connections. Default 0 (unlimited).
- `timeslice <milliseconds> [-h <host>] [-p <portnumber>]`: This sets the time each session can have using a connection thread before yielding to a waiting session. Default is 0 (no yield).
- `trace {on | off} [-s <session id>] [-h <host>] [-p <portnumber>]`: This turns drda tracing on or off for the specified session or if no session is specified for all sessions. Default is off .
- `tracedirectory <tracedirectory> [-h <host>] [-p <portnumber>]`: This changes where new trace files will be placed. For sessions with tracing already turned on, trace files remain in the previous location. Default is `cloudscape.system.home` .

When running in networked mode, the IBM Security Directory Integrator database is of course reachable over the network, not only by IBM Security Directory Integrator instances but also by other applications using the appropriate drivers. The credentials required for such access are defined in the `global.properties` file, and might have to be tailored for your particular site needs. Pay particular attention to the username and password parameters as these govern integrity and security of the data.

If you often alternate between running IBM Security Directory Integrator in dedicated mode and in networked mode, consider having two different "prototype" `global.properties` files on your file system, one each with the correct set of parameters for each of the two modes. Just before starting a server instance, copy in place the appropriate `global.properties` file, according to your needs. Alternatively, use separate Solution Directories; in a Solution Directory you can have a file called `solution.properties`, which property values defined in there override the ones defined system-wide in `global.properties`.

Property stores

Password store and User property stores are types of system stores.

Password Store

The *Password Store* is an external repository that stores a value which results from changing the value for a password syntax component. The password protection mechanism is directly related to the configuration windows offered to the user. The configuration windows, or forms, contain descriptions of each parameter and its syntax. One type of syntax is *password* which causes the Configuration Editor to use a password text field for editing. This external repository for passwords is configured in the *Properties* page in the configuration editor (*Password-Store*) and is specified in the configuration file for the current IBM Security Directory Integrator solution. If no such property store is configured the password is saved in clear text in the configuration file.

If a default password store is configured, a unique property name is generated the first time a protected/password parameter is saved. This key is used as the key in the password store. The same property name is written to the configuration file as a standard property reference. When the value is later retrieved, standard property resolution takes place to retrieve the actual value from the password store.

If a Password Store is specified, a unique key is generated for the password and the password is saved encrypted in the Password Store under that key. In the

configuration file, the password is referenced only by that key.

User property stores

The User Property Store is a System Store table used for maintaining serialized Java objects associated with a key value. This is where persistent component parameters and properties (such as the **Iterator State Store**) are maintained, as well as any data you store. The System Store implements User property stores as one of its three types of persistent stores for IBM Security Directory Integrator components. For information on IBM Security Directory Integrator user interfaces that allow you to select properties from a property store, see “Add a Solution View” on page 264.

Third-party RDBMS as System Store

You can configure the System Store to use other multi-user RDBMS systems, as opposed to using the bundled database, Apache Derby.

This is done by specifying appropriate SQL Data Definition Language (DDL) statements and driver parameters as system properties in `global.properties` or `solution.properties`. Example statements, commented out, are present in the distribution version of `global.properties` in the `TDI_install_dir/etc` directory, for the supported configurations of IBM DB2, Oracle and MS SQL*Server.

It is also possible to take advantage of suitable templates built in to the Configuration Editor, by going to the appropriate IBM Security Directory Integrator Server document. Right-click on the Server in the Servers pane, and select **Edit system store settings**. The **Server System Store** header in the window is a context-sensitive menu; it has selections for Derby Embedded, Derby Networked, Oracle, DB2, MS SQL*Server 2005+ and IBM solidDB.

Note: A System Store can also be configured on a per-project basis in the Configuration Editor; these settings are then stored in the Config file when the project is exported, and take precedence over the System Store defined for the Server.

JDBC Driver parameters provide a path to the database; additional properties are used to specify tailored SQL for certain operations IBM Security Directory Integrator must be able to perform in the System Store. Multiple SQL statements can be specified per property. Each separate statement should be terminated with a semicolon. An example property could be (note that for display purposes, the statements in this document are broken up in multiple lines; however, in your property file all statements for a given property should be on one line):

```
com.ibm.di.store.create.delta.systable=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,  
SEQUENCEID int, VERSION int);  
ALTER TABLE {0} ADD CONSTRAINT IDI_DS_{UNIQUE} PRIMARY KEY (ID)
```

where {0} => is replaced by the Table name; and

{UNIQUE} => is a special variable which can be used to generate a unique name based on the current system time.

The following section lists example connection parameters and statements for each of the supported RDBMS systems.

Oracle

You need to drop the JDBC driver client library, `ojdbc14.jar`, in the `TDI_install_dir/jars` directory for using Oracle.

JDBC connection parameters

```
com.ibm.di.store.database=jdbc:oracle:thin:@itdidev.in.ibm.com:1521:itimdb
com.ibm.di.store.jdbc.driver=oracle.jdbc.OracleDriver
com.ibm.di.store.jdbc.urlprefix=jdbc:oracle:thin:
com.ibm.di.store.jdbc.user=SYSTEM
{protect}-com.ibm.di.store.jdbc.password=password
```

Where `itimdb` is the SID of the database to be used as System Store.

Create table statements

```
com.ibm.di.store.create.delta.systable=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
SEQUENCEID int, VERSION int);ALTER TABLE {0} ADD CONSTRAINT IDI_CS_{UNIQUE} PRIMARY KEY (ID)
com.ibm.di.store.create.delta.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
SEQUENCEID int, ENTRY BLOB );ALTER TABLE {0} ADD CONSTRAINT IDI_DS_{UNIQUE} Primary Key (ID)
com.ibm.di.store.create.property.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
ENTRY BLOB );ALTER TABLE {0} ADD CONSTRAINT IDI_PS_{UNIQUE} Primary Key (ID)
com.ibm.di.store.create.sandbox.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
ENTRY BLOB )
com.ibm.di.store.create.recal.conops=CREATE TABLE {0} (METHOD varchar(VARCHAR_LENGTH), RESULT BLOB,
ERROR BLOB)
```

MS SQL Server

You should install a number of Microsoft client libraries in the `TDI_install_dir/jars` directory to use MS SQL Server.

JDBC connection parameters

```
com.ibm.di.store.database=jdbc:Microsoft:sqlserver://localhost:1433;
DatabaseName=master;selectMethod=cursor;
com.ibm.di.store.jdbc.driver=com.microsoft.jdbc.sqlserver.SQLServerDriver
com.ibm.di.store.jdbc.user=sa
com.ibm.di.store.jdbc.password=passw0rd
```

The above connection parameters are used with these Microsoft JDBC jars:

1. Msutil.jar
2. MsBase.jar
3. MSsqlserver.jar

Note: For Microsoft SQL Server 2008, the driver jar file to be placed in the `TDI_install_dir/jars` directory is `sqljdbc.jar` (only one file is required) and it can be obtained from your SQL Server 2008 installation at `<Microsoft SQL Server 2005-Install-Dir>/sqljdbc_<version>/<language>/sqljdbc.jar`; the JDBC connection parameters need to be specified as follows:

```
com.ibm.di.store.database=jdbc:sqlserver://localhost:1433;DatabaseName=name;selectMethod=cursor;
com.ibm.di.store.jdbc.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
com.ibm.di.store.jdbc.user=sa
com.ibm.di.store.jdbc.password=passw0rd
```

The `selectMethod` property is optional to the jdbc URL. When this property is set to "cursor", a database cursor is created. This is useful when reading very large result sets that cannot be contained in the clients memory.

The default behavior of `selectMethod` is not "cursor", but "direct", which keeps result sets in clients memory, thus providing much faster performance. So unless memory is a problem, it is better to use the default "direct" behavior. For more information: [http://msdn.microsoft.com/en-us/library/ms378988\(SQL.90\).aspx](http://msdn.microsoft.com/en-us/library/ms378988(SQL.90).aspx).

JDBC connection parameters (for JSQLConnect driver)

```
com.ibm.di.store.database=jdbc:JSQConnect://itdidriver/database=reqpro
com.ibm.di.store.jdbc.driver= com.jnetdirect.jsql.JSQLDriver
com.ibm.di.store.jdbc.urlprefix= jdbc:JSQConnect:
com.ibm.di.store.jdbc.user=administrator
{protect}-com.ibm.di.store.jdbc.password=password
```

These connection parameters are used with JSQConnect drivers. You must download the JSQConnect.jar file and copy it into the *TDI_install_dir/jars* directory.

Create table statements

The DATA TYPE for MS SQL is IMAGE.

```
com.ibm.di.store.create.delta.systable=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
SEQUENCEID int, VERSION int);
ALTER TABLE {0} ADD CONSTRAINT IDI_MYCONSTRAINT_{UNIQUE} PRIMARY KEY (ID)
com.ibm.di.store.create.delta.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
SEQUENCEID int, ENTRY IMAGE );
ALTER TABLE {0} ADD CONSTRAINT IDI_DS_{UNIQUE} Primary Key (ID)
com.ibm.di.store.create.property.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
ENTRY IMAGE );
ALTER TABLE {0} ADD CONSTRAINT IDI_PS_{UNIQUE} Primary Key (ID)
com.ibm.di.store.create.sandbox.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
ENTRY IMAGE)
com.ibm.di.store.create.recal.conops=CREATE TABLE {0} (METHOD varchar(VARCHAR_LENGTH),
RESULT IMAGE, ERROR IMAGE)
```

IBM DB2

You can refer to the parameters and statements listed here to know about DB2 working.

JDBC connection parameters

```
com.ibm.di.store.database=jdbc:db2:net://localhost:50000/ididb
com.ibm.di.store.jdbc.driver=com.ibm.db2.jcc.DB2Driver
com.ibm.di.store.jdbc.urlprefix= jdbc:db2:net:
com.ibm.di.store.jdbc.user=db2admin
{protect}-com.ibm.di.store.jdbc.password=db2admin
```

Where *ididb* in the database URL is the DSN for a DB2 instance.

Create table statements

```
com.ibm.di.store.create.delta.systable=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
SEQUENCEID int, VERSION int);
ALTER TABLE {0} ADD CONSTRAINT IDI_MYCONSTRAINT_{UNIQUE} PRIMARY KEY (ID)
com.ibm.di.store.create.delta.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
SEQUENCEID int, ENTRY BLOB );
ALTER TABLE {0} ADD CONSTRAINT IDI_DS_{UNIQUE} Primary Key (ID)
com.ibm.di.store.create.property.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
ENTRY BLOB );ALTER TABLE {0} ADD CONSTRAINT IDI_PS_{UNIQUE} Primary Key (ID)
com.ibm.di.store.create.sandbox.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL,
ENTRY BLOB )
```

IBM solidDB

IBMsolidDB[®] requires that the SolidDriver2.0.jar file is put in the *TDI_install_dir/jars* directory.

This JAR can be obtained from the IBMsolidDB installation (from *SolidDB_install_dir/jdbc/SolidDriver2.0.jar*).

JDBC connection parameters

```
com.ibm.di.store.database=jdbc:solid://localhost:1315
com.ibm.di.store.jdbc.driver=solid.jdbc.SolidDriver
com.ibm.di.store.jdbc.urlprefix=jdbc:solid:
com.ibm.di.store.jdbc.user=dba
{protect}-com.ibm.di.store.jdbc.password=dba
```

Create table statements

```
com.ibm.di.store.create.delta.systable=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH)
PRIMARY KEY NOT NULL, SEQUENCEID int, VERSION int)
com.ibm.di.store.create.delta.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH)
PRIMARY KEY NOT NULL, SEQUENCEID int, ENTRY BLOB)
com.ibm.di.store.create.property.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH)
PRIMARY KEY NOT NULL, ENTRY BLOB)
```

```
com.ibm.di.store.create.sandbox.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH)
NOT NULL, ENTRY BLOB)
com.ibm.di.store.create.recal.conops=CREATE TABLE {0} (METHOD VARCHAR(VARCHAR_LENGTH),
RESULT BLOB, ERROR BLOB)
```

Using Derby to hold your System Store

You can use Derby to hold your System Store.

The remainder of this section discusses the operational aspects of using Derby, in particular in conjunction with using Derby to hold your System Store.

Note: With regards to third party RDBMSs, in order to hold encrypted password values you may have to size the fields that hold them quite large. A typical small password might use as much as 178 characters. It depends on both your server's key, and the length of the unencrypted data you try to store (in bytes). Since this is a blocked encoding a larger password might use the same space, or double or triple that amount. Also, the size of the block depends on the server's key. One way to find the size you need, is to store the password (protected) to a file first, and then look at that file to see how many characters were used.

Derby can run in either of two modes: *embedded* and *networked*. By default, as specified in the `global.properties` file, Derby runs in *networked* mode.

The System Store used by IBM Security Directory Integrator releases before V7.0 was Derby (then called Cloudscape) in embedded mode. There are drawbacks to the way Derby runs in embedded mode. In embedded mode, Derby runs as a separate thread within the JVM when required. Startup and shutdown of Derby is automatic in embedded mode. However, when run this way, this Derby thread claims exclusive access to the database files. This can become problematic when different JVMs, each with its own Derby thread, try to access the same System Store.

In embedded mode, these actions cause a new, independent JVM to be started, triggering an access conflict when more than one JVM is active at any given time:

- A command line invocation of the IBM Security Directory Integrator Server with a config file, causing one or more AssemblyLines to run.
- Startup of the Configuration Editor (GUI)
- Startup of an AssemblyLine from within the Configuration Editor

None of these actions by themselves causes the Derby thread to start. However, the Derby thread does start if access to any of the objects in the System Store is required (for example, Objects supported by the System Store such as Delta Tables and the User Property Store).

The solution to the access conflicts as outlined previously is to run Derby in networked mode, which enables concurrent access to the System Store. Also enable user authentication in Derby to avoid security concerns in networked mode. To provide security at the database level, IBM Security Directory Integrator uses the BUILTIN security provider for Derby. BUILTIN ensures that only valid users are able to access the Derby database. When you have Derby configured in networked mode, you can work with multiple instances of Derby databases booted as System Stores. You can also configure a Derby instance to work with a specific Configuration file instance.

Note: Depending on how Derby was started, instances of Derby can be left running in networked mode, even after all other IBM Security Directory Integrator processes have terminated. When you set the property

derby.drda.startNetworkServer to true (by default, this is the case, in global.properties), the Network Server automatically starts when you start Derby (in this context, Derby starts when the embedded driver is loaded). You may have to terminate Derby manually, if desired.

Configuring Apache Derby Instances

You can learn to configure Apache Derby Instances using the information provided here.

To configure and manage multiple Derby instances and to provide facilities to start, stop and restart Derby servers in networked mode a menu option called **System Store** is provided in the IBM Security Directory Integrator Configuration Editor, as part of **Solution Logging and Settings** configuration of a project. Many of the configuration options listed take default values from the global.properties file, which was the configuration base for previous versions of IBM Security Directory Integrator; now.

The **System Store** menu option also provides ways to configure the System Store to use other databases like IBM DB2 as the backend RDBMS. For more information, refer to "System Store settings" under **The Configuration Editor -> Solution Logging and Settings** in *Configuring Directory Integrator*.

Starting Apache Derby in networked mode

You can start Apache Derby in networked mode with the instructions listed here.

If the com.ibm.di.store.hostname property is set to localhost then remote connections are not allowed. If the com.ibm.di.store.hostname property is set to the IP address of the local computer running IBM Security Directory Integrator, then remote clients can access this Derby instance by using the IP address. You can only start the network server for the local computer.

Table 25. Starting Apache Derby in networked mode

Property	Default value	Description
com.ibm.di.store.start.mode	automatic	The mode for starting up the Derby server process when required – set to automatic or manual.
Com.ibm.di.store.hostname	localhost	The URL of the Derby server.
Com.ibm.di.store.port	1527	The port for connecting to the Derby server.
Com.ibm.di.store.sysibm	true	The state for using the SYSIBM schema or not; values true or false.
com.ibm.di.store.varchar.length	512	The varchar(length) for the ID columns used in system store and System Store (PES) connector tables.
com.ibm.di.store.database	jdbc:derby:// localhost: 1527/\$soldir\$ /TDISysStore; create=true	Sets your Solution Directory as default location of the System Store database. Note: Do not replace \$soldir\$ value with absolute path of Solution Directory. The path is automatically updated at runtime in JVM

Enabling user authentication in System Store

You can add these properties to the global.properties file after the System Store network mode properties.

Table 26. Enable user authentication in System Store

Property	Default value	Description
derby.connection.requireAuthentication	true	Enables user authentication for the System Store.

Table 26. Enable user authentication in System Store (continued)

Property	Default value	Description
derby.authentication.provider	BUILTIN	Sets the user authentication provider to BUILTIN. This is the most basic and simple authentication provider that Derby has.
derby.database.defaultConnectionMode	fullAccess	Defines the access level to the System Store user. The different access levels supported by Derby are "fullAccess", "readOnly" and "noAccess".

Create statements for System Store tables

You can configure create table SQL statements for the list provided here.

- Delta systable
- Delta table
- Property table
- Sandbox tables
- Record AssemblyLine table
- Tombstone manager table
- `ibmsnap_commitseq` column name

Table 27. Create statements for System Store

Property	Default value	Description
com.ibm.di.store.create.delta.systable	CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL, SEQUENCEID int, VERSION int); ALTER TABLE {0} ADD CONSTRAINT IDI_CS_{UNIQUE} PRIMARY KEY (ID)	Create table SQL statements for the delta systable.
com.ibm.di.store.create.delta.store	CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL, SEQUENCEID int, ENTRY BLOB); ALTER TABLE {0} ADD CONSTRAINT IDI_DS_{UNIQUE} Primary Key (ID)	Create table SQL statements for the delta table.
com.ibm.di.store.create.property.store	CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL, ENTRY BLOB); ALTER TABLE {0} ADD CONSTRAINT IDI_PS_{UNIQUE} Primary Key (ID)	Create table SQL statements for the property table.
com.ibm.di.store.create.sandbox.store	CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL, ENTRY BLOB)	Create table SQL statements for the Sandbox tables.
com.ibm.di.store.create.recal.conops	CREATE TABLE {0} (METHOD varchar (VARCHAR_LENGTH), RESULT BLOB, ERROR BLOB)	Create table SQL statements for Record AL.

Table 27. Create statements for System Store (continued)

Property	Default value	Description
com.ibm.di.store.create.tombstones	<pre>CREATE TABLE IDI_TOMBSTONE (ID INT GENERATED ALWAYS AS IDENTITY, COMPONENT_TYPE_ID INT, EVENT_TYPE_ID INT, START_TIME TIMESTAMP, CREATED_ON TIMESTAMP, COMPONENT_NAME VARCHAR(1024), CONFIGURATION VARCHAR(1024), EXIT_CODE INT, ERROR_DESCR VARCHAR(1024), STATS LONG VARCHAR FOR BIT DATA, GUID VARCHAR(1024) NOT NULL, USER_MESSAGE VARCHAR(1024), UNIQUE (ID, GUID))</pre>	Specify the SQL statement for creating the Tombstone Manager table. Keep the same table names and field names.
com.ibm.di.conn.rdbmschlog.cdcolname	ibmsnap_commitseq	Provide the ibmsnap_commitseq column name to be used by the RDBMS changelog connector.

Backing up Apache Derby databases

You can take a back up of Apache Derby databases by using the method provided here.

Another matter that needs to be given some thought is **backup** of the data contained in a Derby database. The recommended (and simplest) way of doing this is to

- Shutdown the Derby database (if running in embedded mode, shut down all IBM Security Directory Integrator instances and Configuration Editor instances)
- Copy the entire Derby directory in your IBM Security Directory Integrator home directory (or whatever Derby directory your `global.properties` file is pointing to) to a different location, and ensure that this data is safe
- Restart the Derby database (if running in networked mode).

To restore a database, reverse source and destination of the copy operation in the above list of steps.

Troubleshooting Apache Derby issues

This section does not attempt to provide comprehensive troubleshooting guidelines for Derby, but there are a number of symptoms that are observed sometimes in the context of usage of Derby as the underlying database in IBM Security Directory Integrator.

These are:

Schema 'SYSIBM' does not exist error

Question

I'm trying to use Derby in networked mode and having issues. I've figured out how to start it up and I'm able to query it with sysinfo and testconnection, but when I run IBM Security Directory Integrator and try to open the system store I get an error stating:

```
[com.ibm.db2.jcc.a.SQLException: Schema 'SYSIBM' does not exist]
```

How do I fix this?

Explanation

The reason you get this error is because you are trying to boot a database that was created in embedded mode into a networked mode server without starting the server using the `-ld` flag. Note that for a networked mode Derby server to open an embedded mode database, the SYSIBM schema MUST be loaded. The SYSIBM schema is a special schema loaded by the Derby server. The SYSIBM contains stored prepared statements that return result sets to determine metadata information.

Corrective action

To solve this problem start the Derby networked server with the `"-ld"` flag, like:

```
./dbserver start -p 1527 -ld
```

Another Instance of Derby may already be booted

You may get the following error sometimes, especially when using Derby in embedded mode:

```
[ERROR XSDB6: Another instance of Derby may have already booted the database D:\tdi60\Derby.]
```

Explanation

Derby try to prevent two instances of Derby from booting the same database (in this case `D:\tdi60\Derby`). This can happen if you are running two AssemblyLines which are trying to update the same Derby database running in embedded mode. This error might also crop up if you have an unclosed connection to the database.

Corrective Action

If you want two AssemblyLines to update the same Derby database, then the correct mode of Derby should be networked mode; this mode of operation does not have that limitation.

You can work around this by closing the database using the **Browse Server Stores** option and then clicking on the **Close** button. Even if the database is not open, just opening and closing again through the **Browse Server Stores** option help solve this problem.

Future versions of IBM Security Directory Integrator attempt to handle this situation automatically, and stop and start Derby as required.

Can I use DB2 as a system store?

In IBM Security Directory Integrator it is possible to use DB2 as a system store, instead of the bundled Derby database system. However, some modification of system properties files is required for this to function correctly. You must replace the section on

Derby networked mode with a section similar to the following (insert the correct parameters for your installation).

If you look at the default `global.properties` file, there are some `CREATE_TABLE` statements for using and setting up the system store. If you use the right syntax, you can use non-Derby databases as system store. Here is the DB2 syntax:

```
## Location of the DB2 database (networked mode)
com.ibm.di.store.database=jdbc:db2://168.199.48.4:3700/tdidb
com.ibm.di.store.jdbc.driver=com.ibm.db2.jcc.DB2Driver
com.ibm.di.store.jdbc.urlprefix=jdbc:db2:
com.ibm.di.store.jdbc.user=db2inst1
com.ibm.di.store.jdbc.password=*****
com.ibm.di.store.start.mode=automatic
com.ibm.di.store.port=3700
com.ibm.di.store.sysibm=true

# the varchar(length) for the ID columns used in system store and PES Connector tables
com.ibm.di.store.varchar.length=512

# create statements for DB2 system store tables
com.ibm.di.store.create.delta systable=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH)
NOT NULL, SEQUENCEID int, VERSION int)
com.ibm.di.store.create.delta.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH)
NOT NULL, SEQUENCEID int, ENTRY BLOB )
com.ibm.di.store.create.property.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH)
NOT NULL, ENTRY BLOB )
com.ibm.di.store.create.sandbox.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH)
NOT NULL, ENTRY BLOB )
```

Note: Each `com.ibm.di.store.create.xxx` statement must be specified on one line, even though they are broken up in this example for illustration purposes.

Why can't remote connections be made to my Derby network server?

This may be because the Derby Server has been started by passing "localhost" as the hostname. This disallows any remote connections to be made to Derby. Stop the Derby server and start it with hostname parameter specified as the computer's IP address. This can be done by going to the Configuration Editor's **Server System Store** Server Settings window (available from the context menu on a server in the Servers view).

For more details, check <http://db.apache.org/derby/docs/10.5/adminguide/tadmincbdjhhd.html>

Chapter 13. Command-line options

Command-line options must have their value followed immediately after the option. Do not use a space between the option and the value.

There are options for:

- “Configuration Editor”
- “Server” on page 206
- “Command Line Interface – tdisrvctl utility” on page 209

Configuration Editor

You can know more about Configuration Editor and the required properties using the information provided here.

The CE is launched using the `ibmditk` wrapper script. This script invokes the Eclipse launcher for IBM Security Directory Integrator (`ce/eclipsece/miadmin`) with the proper settings for the Java VM and the IBM Security Directory Integrator install location property, both of which are required to run the current CE.

The Eclipse launcher (`ce/eclipsece/miadmin`) is a standard Eclipse launcher that takes command line parameters of its own. See Eclipse Command Line Options for a complete description of the Eclipse command line options.

```
"%TDI_HOME_DIR%\ce\eclipsece\miadmin" -vm "%TDI_JAVA_BIN_DIR%\javaw"  
-vmargs -Dcom.ibm.di.loader.IDILoader.path="%TDI_HOME_DIR%" %*
```

The above is a fragment of the `ibmditk` script showing the two required parameters (eclipse command line parameters) that the CE needs.

Of notable interest is the `"-data"` command line option that specifies the location of the workspace to use. If you are going to run multiple instances of the CE, you have to specify a different workspace for each instance of the CE since the workspace is locked by each instance. For example:

```
ibmditk -data c:/instance1_workspace
```

The above command launches the CE using `c:/instance1_workspace` as its workspace location.

Shutdown Servers option

This is a command line option that attempts to stop all running servers that uses the same installation directory as the CE. When this option is given on the command line like this:

```
ibmditk -tdishutdown
```

then the CE will start and look at every defined server in the IBM Security Directory Integrator servers project, filtering out those that do not use the same installation directory as the CE and attempt to stop it. When this is done the CE will exit the Java VM with an exit code of zero. There is no guarantee that the servers the CE tried to stop actually did stop. Some servers may linger beyond the time it takes the CE to complete this command and some servers may simply refuse to stop for various reasons.

Perspective option

This is a command line option that instructs the CE to open in an alternative perspective. Currently, the only perspective other than the default one is the Easy ETL perspective, and the CE is started with it using the following option:

```
ibmditk -perspective com.ibm.tdi.rcp.perspective.etl
```

Server

You can learn to work on server using the information provided here.

The following command-line options are for the IBM Security Directory Integrator server (`ibmdisrv [options]`):

Example:

```
ibmdisrv -c"C:\demos\rs.xml" -r"Access2LDAP" -l"c:\metamerge\mydemo.log"
```

Note:

1. There is no space between the option letter and the value. Use quotes to save against possible spaces or commas in the values.
2. The Windows Shell executive allows a maximum of nine (9) arguments, from the list below. There aren't any limitations on other platforms.
3. Do not use comma (,) in the configuration file name.

-s <dir>

Specifies the working directory where the solution is located; this directory is known as the Solution Directory. All relative file references in IBM Security Directory Integrator and in your Configs and so forth will be relative to this location. Must be the first parameter specified.

If the specified directory does not exist, it will be created by the IBM Security Directory Integrator server, and populated with a number of properties files (based upon those in the installation directory) that you can tailor to your needs. See Appendix A, "Example Property files," on page 339 for more information.

-c <file...>

Configuration file(s). If you don't specify this option, the items in the Autostart folder will be loaded and started (unless suppressed by specifying **-D**). Wildcards, as in *.xml, are allowed too.

Note: Submitting multiple configuration files is only allowed if the **-d** option is also specified.

-n <encoding>

Encoding to be used to write Config files. This must be a valid character set identifier valid in Java2; refer to the IANA Charset Registry (<http://www.iana.org/assignments/character-sets>) for the full list of values. Note that Java2 only supports a subset of those.

-r <al...>

List of AssemblyLine names to start. To start AssemblyLine **a** and **b**, use the command **-r a b**. Other syntaxes are supported as well: **-ra,b**; **-ra -rb**.

Note: If you use includes and namespaces, the AssemblyLine can be `myNamespace:/AssemblyLines/alName` (assuming namespace **myNamespace** and AssemblyLine name **alName**).

- T<name>
Enable JLOG-style tracing to file trace<name>.log, in directory <Tivoli_Common_Dir>/TDI/logs/. Default is trace to memory (from which it can be retrieved by the traceback routines of JFFDC in case of an unhandled exception.)
- D Flag to disable startup of items in the Autostart folder.
- w If -r (or -t) is specified then this flag causes IBM Security Directory Integrator to wait for each AssemblyLine to complete before starting the next. If this flag is not specified then IBM Security Directory Integrator starts all AssemblyLines specified by the -r parameter in parallel. When the last AssemblyLine has finished, the server stops.
- e Specifying this option causes the server to run in Secure mode. Using the master password specific to this server, it will decrypt and encrypt all Config files as well as the server API Registry.
- v Show version information and exit. This is logged in the log file only.
- P <password>
Password if configuration file(s) is/are encrypted.
- p Dump Java properties on startup. Note that you still must provide a configuration file, which is read before Java properties are dumped.
- d Start a "daemon", or *Config Instance* on this system.

If you start with -d, you will start one anonymous instance (the daemon), which will start one config instance for each config file specified on the command line; this allows you to start multiple config instances on the fly. You may specify 0 or more config files on the command line. It does not make sense to specify any AssemblyLines to run in this mode, since it is impossible to state which config file the AL will be in. You can autostart AssemblyLines, though, since those belong to the config instance that specifies the autostart.

If you start without -d, you will get one config instance that loads the config file specified on the command line. You must specify exactly one config file on the command line. (If you must use multiple config files, they may be piped in on standard input.) In this mode, you can specify any number of AssemblyLines to run. This is the traditional way of running the server.
- q Takes 1 argument, mode. Mode=1 means run in record mode, mode=2 means run in playback mode.
- l <file>
Log file (default console output). Does very little as few messages go to the console. To change the log file for most of the logging, change log4j.properties.
- R Disables the Remote API, regardless of the setting in global.properties.
- W Start all Configs in the same thread; they do not terminate but wait forever.
- M Start AssemblyLines in simulation mode.
- S This option is for internal use for communication between the Configuration Editor and a server only; it is used to pass Config Files between them. Do not use this option yourself.
- f extProp1=file1, extProp2=file2
Where extProp is the name of the external Property Store. file specifies from

where to read the properties. This option specifies a user-defined, external Property Store that can be entered when starting an IBM Security Directory Integrator server. This optional command-line parameter **-f** can be used with the "ibmdisrv" server startup scripts. *extProp* is the name of the external Property Store. *file* specifies from where to read the properties. When the **-f** option is used to specify a properties file from the command line, the server changes the Property Store configuration in memory only, i.e. the server does not make this change permanent by changing the IBM Security Directory Integrator Config file on disk - this change is valid for the current run of the IBM Security Directory Integrator server.

If any property files are specified at the command line, they are valid only for the Config Instances specified with the **-c** command-line option (which are loaded on IBM Security Directory Integrator server startup). The property files specified at the command line do not have any impact on Config Instances which have not been explicitly named with the **-c** command-line option (these can be Config Instances loaded by remote server API client for example).

If a Property Store whose name is specified with the **-f** command-line switch cannot be found in a Config Instance, an error message is logged in the server log (ibmdi.log in the Install-directory). When a Property Store name is specified more than once with the **-f** command-line switch then there are two effects: (1) a warning message is logged, and (2) the file specified last will take effect. This feature is implemented in the com.ibm.di.server.RS Java class (referenced by way of the main variable when scripting). After the reload() method is called, the MetamergeConfig object is loaded, and for each Property Store specified on the command line, the corresponding PropertyStoreConfig object is updated.

Note:

Although Copy/Paste of Config objects (ALs, Connectors, FCs, and so forth) are fully supported. You can easily copy ALs and components and then paste them into another Config. You can also exchange ALs and components using IM chats, e-mails and text files, because the copy-buffer is filled with the IBM Security Directory Integrator Config XML definition of the selected item. This makes passing stuff around simple and easy, and is a great tool for support and online assistance (for example, ICT/NotesBuddy, forums, ...).

Note:

Make sure you select the entire <MetamergeConfig> node in your copy command, including the start and end tags.

- i** This option specifies that the IBM Security Directory Integrator server ignores any properties from the global.properties file, and reads only the solution.properties file. This option can be used when the global.properties file is unreadable - for example, when the encoding the IBM Security Directory Integrator server is started with is different from the encoding of global.properties.
- ?** Prints a *usage* message, showing all options in brief.
- j <file>**
This option is used to read regression information from the specified file, and is compared with the details produced by the AssemblyLine. If there

are variations, warning messages are written to the log file. This option is useful only when running a single AssemblyLine.

-j<file>

This option is used to write the AssemblyLine regression information to the specified file.

-k<file>

This option is used to ignore the work Entry when reading the regression information.

Note:

1. If a property is used as a parameter for a Connector, Parser, or Function Component, and that property does not exist in the property store, a warning message is logged.
2. You can load a configuration file into the server without starting the AssemblyLines. Warning messages are logged for the non-existing properties that are used.

When IBM Security Directory Integrator ends it returns one of the following exit codes:

- 0 No error. The operation has been successful.
- 1
 - Cannot open log file (-l parameter)
 - Cannot open Config file
 - An AssemblyLine failed. Applicable only if the Server is run in non-daemon mode, that is, without the "-d" option. For example: "ibmdirsv -c rs.xml -r al" or "ibmdirsv -c rs.xml -r al1,al2,al3".
- 2 (Obsolete) Exit after auto-run. When you start IBM Security Directory Integrator specifying -w, the Server runs the AssemblyLines specified by the -r parameter and then exits.

Note: AssemblyLines run from the Configuration Editor are started in a different way and will not exit with status 2.
- 9 (Obsolete) License expired or invalid.

Note: If the Server is shutdown by an administration request and a custom exit code is specified, that custom code will be used as the exit code of the Server.

Command Line Interface – *tdisrvctl* utility

The Command Line Interface (CLI) to IBM Security Directory Integrator, called the *tdisrvctl* utility, is designed for remotely managing Configs, AssemblyLines, and so on.

This utility connects to a remote IBM Security Directory Integrator server using the Remote Server API, and performs the requested operations. As it is a client application interfacing to a Remote Server, it is subject to the same connection, authentication and authorization issues described in Chapter 6, "Security," on page 93.

It exposes various command line options for the following functions:

- Start, stop, or reload IBM Security Directory Integrator Configs.

- Start or stop AssemblyLines in a particular config.
- Display a list of configs loaded on the server.
- Shutdown server.
- Display config report.
- Manage config properties through TDI-p, the IBM Security Directory Integrator properties framework
- Send custom notification events.
- View exposed AL Operations.
- View tombstones for terminated Configs and AssemblyLines
- View IBM Security Directory Integrator Server details.

Note:

1. The command line utility is shipped in the *TDI_install_dir/bin* folder.
2. Remote Method Invocation (RMI) is enabled in the global.properties by default (`api.remote.on=true`), with security enabled (`api.remote.ssl.on=true`). When `api.remote.ssl.on=true`, the keystore and truststore `general_options` must be included in the command. For example:

```
tdisrvctl.bat -h mytdisserver.com -p 1099 ..\serverapi\testadmin.jks -P administrator
-T ..\testserver.jks -W server -op srvinfo
```

If RMI security is disabled (`api.remote.ssl.on=false`), then the IP Address of the client executing the `tdisrvctl` utility must be defined in the property `api.remote.nonssl.hosts`. In this case, the `tdisrvctl` command can be issued from the identified client without the keystore and truststore parameters. For example:

```
tdisrvctl.bat -h mytdisserver.com -p 1099 -op srvinfo
```

3. The remote IBM Security Directory Integrator server must be running.

Command Line Reference

You can know more the usage of Command Line Reference using the information provided here.

The command has the following usage:

```
tdisrvctl [general_options] -op operation [operation_specific_options]
```

where **general_options** can be:

-h host	Enter the remote server IP address or hostname (default is localhost).
-K keystore	Enter the name of the SSL key database file.
-p port	Enter the port number (default is 1099).
-P key_pwd	Enter the key file password.
-s	Specify the working directory where the solution directory is located.
-T truststore	Enter the name of the SSL truststore database file.
-u userID	Enter the username (for custom authentication).
-v	Run in verbose mode.
-w user_pwd	Enter the user password (for custom authentication).
-W trust_pwd	Enter the trust file password.
-?	Display command usage.

And **operation** can be:

event	Send custom notification events
prop	Manage Config properties
queryop	Query for AssemblyLine (AL) operations
reload	Reload running Configs
report	Generate Config report or list Configs on remote server
shutdown	Shutdown the server
srvinfo	View IBM Security Directory Integrator server information
status	View status of Configs or ALs
start	Start specific Config or ALs
stop	Stop specific Config or ALs
tombstone	View tombstone entries for specific Config or AL.
deletetombstone	delete a tombstone entry
debug	debug components of a running AssemblyLine

You can display help for any particular option like this:

```
tdisrvctl -op operation -?
```

Operations

You can refer to the list of operations provided here.

event Use this option to send custom notification events to a particular server. All listeners registered for the particular event receive this notification. This allows IBM Security Directory Integrator administrators to trigger listener applications based on planned custom events.

The usage for the event operation is:

```
tdisrvctl [general_options] -op event -e event_name [-s source ] [-d data]
```

where:

-e event_name	The name of the event to send.
-s source	The name of the source invoking the event (default "tdisrvctl").
-d data	The data to be passed to an event listener (default is null).

Example:

To send an event "user.process.X.completed" from "admin".

```
tdisrvctl -h itditest -op event -e "process.X.completed" -s admin -d "Admin triggered event"
```

Note: All events sent from tdisrvctl using the -e option are prefixed by "user."

prop The "prop" option exposes the properties of a config via the TDI-p. It allows the user to get / set / view the properties of a particular config.

The usage for the prop operation is:

```
tdisrvctl [general_options] -op prop -c config_name
[ [-l ] |
[-o property_store]
[-g key | all] |
[-s key=value] [-e] |
[-d key] ]
```

where:

-c config_name	Name the config to work with.
-l	List all the property stores configured.

-o property_store	Name the property store to work with.
-g key	Get the value of the specified key (or keyword 'all' implying get all keys).
-s key=value	Set the "key" to the specified "value."
-e	Encrypt the value when putting in the store (can be used with -s option only).
-d key	Delete the specified "key" from the store.

Note:

1. The '-l', '-g', '-s', '-d' options are mutually exclusive, and cannot be used together.
2. The '-e' option can only be used with the '-s' option.
3. Managing properties stored in the **password store** is NOT supported.
4. While specifying the "-c" option specify the COMPLETE configuration file path on the remote server, or give a path relative to the "configs" folder. To see the relative paths use the "report" option of tdisrvctl:

```
tdisrvctl -op report -l
```

Examples:

To see a list of all the property stores for config C1.xml

```
tdisrvctl -op prop -c C1.xml -l
```

To get a list of all the properties for config C1.xml

```
tdisrvctl -op prop -c C1.xml -g all
```

To get a list of all the properties for config C1.xml from store MyStore

```
tdisrvctl -op prop -c C1.xml -o MyStore -g all
```

To set a property MY_PROP to value MY_VALUE for config C1.xml in store MyStore and mark it as protected:

```
tdisrvctl -op prop -c C1.xml -o MyStore -s MY_PROP=MY_VALUE -e
```

queryop

The queryop option returns the list of AL operations exposed in an AssemblyLine.

This option is useful in a scripting environment. An IBM Security Directory Integrator solution developer can develop a script to automatically query for exposed operations and then use the result to start an AssemblyLine with a specific operation using the start operation's **-r -alop** flag. The output of this operation is such that it can be grepped for or tokenized easily in a scripted environment.

The usage for the **queryop** operation is:

```
tdisrvctl [general_options] -op queryop -c <configFile> -r <ALName>
```

where

configFile	Config file name
ALName	Name of the AssemblyLine

Output:

```
ALOp:{attr_1;attr_2...attr_n;}
```

Note: While specifying the "-c" option specify the COMPLETE configuration file path on the remote server, or give a path relative to the "configs" folder. To see the relative paths use the "report" option of tdisrvctl:

```
tdisrvctl -op report -l
```

Examples:

To query for *operations* exposed in an AL:

```
tdisrvctl -h itditest -T trust.kdb
-W secret -op queryop
-c examples/ADCustomConnector.xml
-r ADAssemblyLine
```

Example Output:

```
$initialize: {ldapur1;loginPasswd;loginUserName}
```

reload This option can be used to reload running Configs on a particular server.

The usage for reload operation is:

```
tdisrvctl [general_options] -op reload -c [config_list]
```

where:

config_list Comma separated list of Configs to reload.

Note: While specifying the "-c" option specify the COMPLETE configuration file path on the remote server, or give a path relative to the "configs" folder. To see the relative paths use the "report" option of tdisrvctl:

```
tdisrvctl -op report -l
```

Example:

To reload Configs C1.xml, C2.xml and C3.xml on remote host itditest:

```
tdisrvctl -h itditest -T trust.jks -W secret -op reload -c C1.xml,C2.xml,C3.xml
```

report This option can be used for generating a report for a particular config or for listing the configs available on the remote server's config folder.

The config report lists details of the particular config. The details are AssemblyLines, Connectors and Parsers in each AssemblyLine, Connector library, Parser library, Script library, Function Library. This option gives a one shot view of all the details of a particular config.

The config listing option helps the user in finding out the list of configs available on the remote server and what their exact names are. Of course, only those configs can be seen that are in the "config" folder of the remote server (see `global.properties` file for property **api.config.folder**). This command cannot obtain list of configs located "anywhere" on the system.

The usage for the report operation is:

```
tdisrvctl [general_options] -op report [-c config | -l]
```

where:

-c config Name of the Config whose report is to be generated.
-l The Configs in the remote server's config folder.

The displayed details for each connector or function component part of an AssemblyLine look like this:


```

Name      : count
Mode      : Iterator
State     : Enabled
Debug     : Disabled
Template  : system:/Connectors/ibmdi.Timer
Parser    : [parent]
Comment   : None

```

Note:

1. The specified config must be already loaded on the remote server.
2. Only one of the '-c' or '-l' option is allowed. Not both.
3. While specifying the "-c" option specify the COMPLETE configuration file path on the remote server, or give a path relative to the "configs" folder.
4. The argument to the -c option is case-sensitive, and must match the name of the config file exactly as known by the server instance, reported by for example "tdisrvctl -op status".

Examples:

To get a complete listing of the details of C1.xml on remote server:

```
tdisrvctl -h remoteserver -op report -c C1.xml
```

To get a list of the configs available in the "config" folder of the remote server:

```
tdisrvctl -h remoteserver -op report -l
```

shutdown

This option can be used to shutdown the IBM Security Directory Integrator server.

The format for this command is:

```
tdisrvctl [general_options] -op shutdown [-o return_code] [-f]
```

where:

- o return_code The return code with which the remote IBM Security Directory Integrator server should exit.
- f Force a controlled shutdown and exit all AssemblyLines.

Examples:

To shutdown the local IBM Security Directory Integrator server:

```
tdisrvctl -op shutdown
```

To shutdown the local IBM Security Directory Integrator server, with a controlled shutdown of all AssemblyLines:

```
tdisrvctl -op shutdown -f
```

To shutdown the server running on remote host itditest which is configured for SSL (server-auth only)

```
tdisrvctl -h itditest -T trust.kdb -W secret -op shutdown
```

srvinfo

This option is used to display the information of an IBM Security Directory Integrator server.

The usage of the command is:

```
tdisrvctl [general_options] -op srvinfo
```

Example:

To view the server information for an IBM Security Directory Integrator server running on localhost

```
tdisrvctl -h localhost -op srvInfo
```

status This option can be used to view status of AssemblyLines.

The usage for status operation is:

```
tdisrvctl [general_options] -op status -c [config_list | all]
-r [AL_list | all]
-listen
```

where:

config_list	Comma-separated list of Configs or keyword "all".
AL_list	Comma-separated list of ALs or keyword "all".
-listen	indicates to start receiving the logs of a running Config or AssemblyLine.

Note:

1. At least one of the options ('-c' or '-r') must be specified. -
2. The keyword "all" indicates all configs or AssemblyLines.
3. The -listen option requires exactly one Config or AssemblyLine to be specified.
4. While specifying the "-c" option specify the COMPLETE configuration file path on the remote server, or give a path relative to the "configs" folder. To see the relative paths use the "report" option of tdisrvctl:

```
tdisrvctl -op report -l
```

Examples:

To see the status of all configs and ALs:

```
tdisrvctl [general_options] -op status -c all -r all
```

You can also write

```
tdisrvctl [general_options] -op status
```

To see the status of AL1, AL2:

```
tdisrvctl -h itditest -op status -c c1.xml -r AL1,AL2
```

Output:

```
(Component Type # Component Name # RUNNING / STOPPED # Statistics):
1 # AL1 # RUNNING # [get:571] [add:571] [del:3] [requests:2333]...
1 # AL2 # STOPPED #
```

The Component Types are:

- 0 for Config
- 1 for Assembly line

The Statistics contain the following details (valid for AssemblyLines only):

- Attribute "add" - total number of "add" operations performed
- Attribute "mod" - total number of "modify" operations performed
- Attribute "del" - total number of "delete" operations performed
- Attribute "get" - total number of "getNext" (Iterations®) performed
- Attribute "request" - total number of requests accepted when there is a Server mode Connector in the AssemblyLine.

- Attribute "callReply" - total number of "callReply" operations performed
- Attribute "err" - total number of errors encountered
- Attribute "skip" - total number of 'skip' operations performed
- Attribute "lookup" - total number of "lookup" operations performed
- Attribute "ignore" - total number of "ignore" operations performed
- Attribute "reconnect" - total number of "reconnect" operations performed
- Attribute "exception" - the exception text if the component terminated with an exception

To see the details of Configs (running and stopped) on a particular server:

```
tdisrvctl -h itditest -op status -c all
```

To see the details of a running AssemblyLine on a particular server and start receiving its logs:

```
tdisrvctl -h itditest -op status -c rs.xml -r all -listen
```

start This option can be used to start a config or AssemblyLines.

The usage for the start operation is:

```
tdisrvctl [general_options] -op start -c [config]
-e [password]
-r [AL_list | all] -alop <alop_Name> [[requiredAttr_1;
requiredAttr_2; ... requiredAttr_n]] | [-f filename]
-s [Simulate mode]
-m [run name] -o [propStore1=filename1,propStore2=filename2...]
-t [temp config instance]
-listen
-sync
```

where

-c config	Name of config to start.
-e password	Password of config file if it is encrypted.
-r AL_list	Comma-separated list of ALs to start or keyword 'all'.
-o property file list	comma separated list of property store names and values
-alop operName	The specific AL operation and list of list required attributes for the specified operation.
-f filename	Name of the file where the input attributes and their values are configured for the operation.
-s Simulate mode	Run the specified AssemblyLines in simulate mode
-m multi-instance	Run multiple instances of same Config with different run names
-t temp config instance	Start temp config instance from the XML in the config file specified
-listen	receive the logs of the specified Config or AssemblyLine
-sync	execute AssemblyLine synchronously

Note:

1. The '-c' option is mandatory. -
2. The keyword "all" indicates all AssemblyLines.
3. Required attributes list is mandatory with -alop option.
4. -alop option cannot be used with -r all option. It works only with a specific AL.

5. When running a temp config with solution or run name it is not possible to check if another config with the same name is already running on the server. If this happens an exception will occur. You could check the running config instances using the **status** command.
6. The **-t** option expects the Config specified in the **-c** option to be located on the client machine.
7. If the **-t** option is used and the config specified in the **-c** option is relative then it will be searched in the current folder.
8. The **-listen** option requires exactly one Config or AssemblyLine to be specified.
9. The **-listen** option executes an AssemblyLine synchronously. There is no need to combine it with the **-sync** option.
10. The **-sync** option requires exactly one AssemblyLine to be specified.

Examples:

1. To start assembly line AL1 and AL2 of config C1 on remote server itditest:

```
tdisrvctl -h itditest -T trust.kdb -W secret -op start -c C1.xml -r AL1,AL2
```

The **-r** option requires that **-c** option should also be specified. This is because the AssemblyLines mentioned in the command *must* belong to one of the Configs in the **-c** option.

2. To start assembly line AL1 on remote server itditest with AL operation:

```
tdisrvctl -h itditest -T trust.kdb -W secret -op start  
-c examples/ADCustomConnector.xml  
-r ADAssemblyLine  
-alop $initialize {ldapurl:ldap://9.182.190.149:390;loginPasswd:password;loginUsrname:cn=root}
```

3. To start AssemblyLine AL1 on remote server itditest with AL operation update:

```
tdisrvctl -h itditest -T trust.kdb -W secret -op start  
-c examples/ADCustomConnector.xml -r ADAssemblyLine  
-alop search {$init.ldapurl:ldap://9.182.190.149:390;$init.loginPasswd:password;  
$init.loginUsrname:cn=root;searchBase:o=ibm,c=us}
```

Note: All initialization attributes are to be prefixed with **\$init**.

- 4.

```
tdisrvctl -h itditest -T trust.kdb -W secret -op start -c examples/ADCustomConnector.xml  
-r ADAssemblyLine -alop search -f inputFile
```

Input file format:

```
=====
```

```
Key1:value1
```

```
Key2:value2
```

5. Command to run an AssemblyLine AL1 in simulate mode:

```
tdisrvctl -h itditest -T trust.kdb -W secret -op start -c examples/ADCustomConnector.xml -r AL1 -s
```

6. Command to load multiple config instances:

```
tdisrvctl -op start -c C1.xml -m test -f PropertyStorename=TestProp.properties,  
PropStore2=propfile2 ... -r AL1,AL2
```

7. Command to run temp config instance:

```
tdisrvctl -op start -c C1.xml -t -r AL1
```

8. Command to start a config on a particular server and receive its logs:

```
tdisrvctl -h itditest -op start -c rs.xml -listen
```

9. Command to start an AssemblyLine on a particular server and receive its logs:

```
tdisrvctl -h itditest -op start -c rs.xml -r AL1 -listen
```

10. Command to execute an assembly line synchronously on a particular server:

```
tdisrvctl -h itditest -op start -c rs.xml -r AL1 -sync
```

stop The usage for the stop operation is:

```
tdisrvctl [general_options] -op stop -c [config]
-r [AL_list | all]
```

where:

-c config	Name of Config.
-r AL_list	Comma-separated list of ALs to stop or keyword "all."
-f	Force a controlled shutdown of AssemblyLines.

Note:

1. The '-c' option is mandatory.
2. While specifying the "-c" option specify the COMPLETE configuration file path on the remote server, or give a path relative to the "configs" folder. To see the relative paths use the "report" option of tdisrvctl:

```
tdisrvctl -op report -l
```
3. The keyword "all" indicates all AssemblyLines.
4. The -r option requires that -c option should also be specified. This is because the AssemblyLines mentioned in the command *must* belong to one of the Configs in the -c option.
5. The -f option is optional.
6. The argument to the -c option is case-sensitive, and must match the name of the config file exactly as known by the server instance, reported by for example "tdisrvctl -op status".

Example:

To stop assembly line AL1 and AL2, of Config C1 on remote server itditest:

```
tdisrvctl -h itditest -T trust.jks -W secret -op stop -c C1.xml -r AL1,AL2
```

tombstone

This option can be used to view tombstone details of previously run Configs, AssemblyLines and EventHandlers (historical).

The usage for the tombstone operation is:

```
tdisrvctl [general_options] -op tombstone -c [config]
-r [AL_name] ]
[-age n]
[[attribute_list] | all ]
```

where:

-age n	Tombstone record for the last 'n' days (default is 1 day).
-c config	Name of Config.
-r AL_name	Name of AssemblyLine.
all	Tombstone attributes: show all.

attribute_list:

-ct	Component type.
-cn	Component name.

-guid	Tombstone entry's guid
-et	Event type.
-ex	Exit code.
-stime	Component's start time.
-ctime	Tombstone create time.
-desc	Error description.
-um	User message.
-stat	Statistics (valid for ALs only).

Note:

1. The '-c' option is mandatory.
2. While specifying the "-c" option specify the COMPLETE configuration file path on the remote server, or give a path relative to the "configs" folder. To see the relative paths use the "report" option of `tdisrvctl`:

```
tdisrvctl -op report -l
```
3. The argument to the -c option is case-sensitive, and must match the name of the config file exactly as known by the server instance, reported by for example "`tdisrvctl -op status`".

Examples:

1. To see the last 2 days tombstone entries (all attributes) for config C1.xml

```
tdisrvctl [general_options] -op tombstone -c C1.xml -age 2 all
```
2. To see tombstone entries for config C1 for the past 3 days:

```
tdisrvctl -h itdiserver -op tombstone -c C1 -age 3 all
```
3. To see tombstone entries for config C1 for the last 24 hours (specific attributes):

```
tdisrvctl -h itdiserver -op tombstone -c C1 -ct -ctime -cn -um
```
4. To see the tombstone entry for AL1 of "rs.xml"

```
tdisrvctl -h itdiserver -op tombstone -c C1 -r AL1
```

deletetombstone

This option can be used to delete tombstone entries for previously run AssemblyLines.

The usage for the delete tombstone operation is:

tdisrvctl [general_options] **-op deletetombstone -guid** <GUID number>

where

-guid "GUID number" is the unique identifier for the tombstone to be deleted. The GUID for a tombstone can be obtained by viewing the contents of the tombstone; see the entry about the tombstone option for details as to how to obtain the GUID.

debug This option can be used to set the Debug mode values of connectors and function components of a running AssemblyLine. When you set the Debug mode of a connector with specified parser, the Debug mode of the parser is also initialized with the same value.

The usage for the debug operation is as follows:

tdisrvctl [general_options] **-op debug -c** config
-r assembly_line
[-alc al_component]
-on/off

where:

-c config	Name of Config.
-r assembly_line	Name of AssemblyLine.
-alc al_component	name of the AssemblyLine component.
-on	flag to enable debug.
-off	flag to disable debug.

Note:

1. The '-c' and '-r' options are mandatory and require exactly one Config/AssemblyLine to be specified.
2. While specifying the "-c" option specify the COMPLETE configuration file path on the remote server, or give a path relative to the "configs" folder. To see the relative paths use the "report" option of tdisrvctl:

```
tdisrvctl -op report -l
```
3. The argument to the -c option is case-sensitive, and must match the name of the config file exactly as known by the server instance, reported by for example "tdisrvctl -op status".
4. If the -alc option is not specified all components in the specified AssemblyLine will be affected.

Examples:

1. To view the Debug mode value of the components in the AssemblyLines of a specified Config:

```
tdisrvctl -op report -c C1
```
2. To enable the debug mode for all components in the running AssemblyLine al2:

```
tdisrvctl -op debug -c C1-r al2 -on C1-r al2 -on
```
3. To disable the debug mode for specified components in the running AssemblyLine al3:

```
tdisrvctl -op debug -c C1-r al3 -alc comp1,comp2 -off
```

Other points to note

- If the user specifies the **-T** option or the **-K** option, it means the command line utility must use SSL.
- If no **-h** (host) option is specified, the command line interface searches for the environment variable **TDI_RSRV**. If TDI_RSRV is not set or empty, then it uses "localhost" as default. This is also the case for the **-p** (port) option: if **-p** is not specified then it searches for **TDI_RPORT**, and if that is not specified either, it uses the default of "1099".
- The tdisrvctl command will return an exit code of zero if the operation is successful and non-zero if the operation fails. Possible reasons for operation failure are:
 - A connection cannot be established to the remote Server.
 - The remote Server reported an error (probably related to the operation that was executed).
 - An AssemblyLine, which is executed synchronously, failed (see the "-sync" option of the "start" operation).
- The tdisrvctl command line utility will use Log4J logging APIs for logging error messages. The Log4J configuration file is specified in the startup script (the .bat or .sh) file. The command uses a file called tdisrvctl-log4j.properties to set up the Log4J logging. If the solution directory is specified the command sets an environment variable for pointing to the log configuration file in the solution

directory. If the solution directory is not specified then the command uses the log configuration file present in the install directory.

- The `tdisrvctl-log4j.properties` file has the complete path of location where the logs are to be created. The log files are created in the `TDI_install_dir/logs` directory by default. The location can be customized as needed.
- All reported error and warning messages are displayed with an error code prefix. This error code can be used to search the *Messages* for an explanation of the error message and operator response.

Chapter 14. Logging and debugging

You can know more about Logging and debugging and its working using the information provided here.

IBM Security Directory Integrator uses a logging class to record messages to a number of various log channels. All IBM Security Directory Integrator components use this logging class which in turn invokes an industry standard logging tool (Log4J). While Log4J provides a variety of output channels and formats, there are other logging utilities with overlapping and additional output channels that you as an IBM Security Directory Integrator user may need. Many of these are open source libraries that are not bundled with IBM Security Directory Integrator. To enable inclusion of these 3rd party logging utilities, the IBM Security Directory Integrator logging component is modeled to act as a proxy between IBM Security Directory Integrator and the actual logging implementations, called LogInterface implementations. Refer to the section "Creating additional Loggers" in *Reference* section of for more information on how to create, configure and program your own LogInterface classes.

Note: Enable or disable logging in IBM Security Directory Integrator by configuring the `com.ibm.di.logging.enabled` property. To enable logging, use `com.ibm.di.logging.enabled=true` (default). To completely disable logging, use `com.ibm.di.logging.enabled=false`.

The remainder of this section describes how to use the logging class that is bundled with IBM Security Directory Integrator, called `com.ibm.di.log.TDILog4J`.

Logging and debugging by the system is mainly done through the Task object (the current AssemblyLine). Logging can either be done explicitly (in script) or done by the various components themselves.

The Log4J logging engine is a very flexible framework that lets you log to file, eventlog, and syslog. Logging can be tuned so it suits most needs. It can be a great help when you want to troubleshoot or debug your solution. By means of the aforementioned logging class, IBM Security Directory Integrator has additional tracing facilities (discussed in Chapter 15, "Tracing and FFDC," on page 233), though in most cases, the logging functionality described here suffice.

IBM Security Directory Integrator components may have specific troubleshooting guidelines; for more information, always check the particular component's section in the *Reference* and *Troubleshooting and Support* sections of the IBM Knowledge Center for IBM Security Directory Integrator.

The log scheme for the server (`ibmdisrv`) is described by the file `Log4J.properties` and elements of the Config file, see "Log4J default parameters" on page 229.

Note: Any of the aforementioned properties files can be located in the Solution Directory, in which case the properties listed in these files override the values in the file in the installation directory.

You can create your own appenders to be used by the Log4J logging engine by defining them in the `Log4J.properties` file. You can use drivers built-in to Log4J like the default one, which is defined with the statement:

Log4J.appender.Default=org.apache.Log4J.FileAppender

The phrase `org.apache.Log4J.FileAppender` defines this appender to use the `FileAppender` class. Additional Log4J compliant drivers are available on the Internet, for example drivers that can log using JMS or JDBC. In order to use those, they need to be installed into the IBM Security Directory Integrator installation `jars` directory after which appenders can be defined using those additional drivers in `Log4J.properties`. For more information, refer to <http://jakarta.apache.org/log4j/docs>.

In addition to the IBM Security Directory Integrator built-in logging, you can log by adding script code in your `AssemblyLine`. This is described in much more detail in the *Configuring* section of the IBM Knowledge Center for IBM Security Directory Integrator, in which you also find out how the interactive debugger works.

Script-based logging

You can issue messages to the `AssemblyLine`'s configured loggers at any time using JavaScript, at any point where scripting is possible (hooks, script components, and so on.)

The explicit `logmsg()` calls available to you (that is, **`task.logmsg()`** & **`main.logmsg()`**) can have an optional string parameter indicating the Log4J level at which the messages are to be logged. Default is INFO. If the log-level given by the user is invalid for Log4J, the message is logged at DEBUG level. Levels include DEBUG, INFO, WARN, ERROR, FATAL.

If you use

```
task.logmsg()
```

your messages will be logged along with the other messages from the `AssemblyLine`. If you are running your `AssemblyLine` from the Configuration Editor, that will be in the CE output window. If your `AssemblyLine` also uses other logging methods, the messages will be there too.

When you use

```
main.logmsg()
```

your message will be logged along with other messages from the Config Instance. This will be in the log file(s) or other loggers created by the Config Instance, which are typically not seen in the Configuration Editor.

Logging using the default Log4J class

You can know more about logging using the default Log4J class using the information provided here.

Configuring the default logging of IBM Security Directory Integrator, which uses Apache Log4J is done globally (using the file `Log4J.properties` which specifies global defaults for Server tasks) or specifically, using the Configuration Editor, for each `AssemblyLine` or `Config File` as a whole. To provide this level of flexibility and customization, the Java Log4J API is used.

Only the parameters that describe how messages are logged are described here.

All log configuration windows operate in the same way: For each one you can set up one or more log schemes. These are active at the same time, in addition to whatever defaults are set in the `Log4J.properties` file; see “Log4J default parameters” on page 229.

Many (but not all) loggers support a Character Encoding option, to control what character set the log files are written in. There are many different character sets; for an informal overview check <http://download.oracle.com/javase/6/docs/technotes/guides/intl/encoding.doc.html>.

The possible log schemes are as follows:

FileRollerAppender

Sometimes, you want to log to file but keep a limited number of files, as they can fill your disks. `FileRollerAppender` generates a new file for each run of the Server. The system saves only the specified number of previous logs. If your log is called `mylog.txt`, and you ask for 2 generations, then after 3 runs you have a `mylog.txt` (last run) as well as the files `mylog.txt.1` and `mylog.txt.2`, where `mylog.txt.2` is the oldest log. From this point, you do not get more files, only newer versions with the same name. Keep two generations of backup files.

`FileRollerAppender` has the following parameters:

File Path

The name of the file to log to. The path is relative to where you installed IBM Security Directory Integrator. The special macro `{0}` used in filenames is replaced by the name of the Server. Similarly, `{1}` used in filenames is replaced by a unique identifier generated by the system for you. The `{1}` macro has no relevance for the special case where you use `FileRollerAppender`, but is important where you want unique file names.

Number of backup files

If your File Path was `mylog.txt`, and you select 2 backup-files, the two previous runs have their files renamed to `mylog.txt.1` and `mylog.txt.2` when you run a third time.

Layout

Determines the format of the log message. Options are:

- Pattern (used if you want to customize the way the messages are logged)
- Simple (format containing just the loglevel and the message)
- HTML (creates an HTML file containing some (relative) time info, thread info, loglevel, category, and message)
- XML (similar to HTML, but generates an XML file (using namespace-prefix `Log4J`))

Pattern

Only used when **Layout** is **Pattern**. See “Creating your own log strategies” on page 230.

Log level

Severity level of the log messages. Options are (from maximum to minimum information):

- DEBUG
- INFO
- WARN

- ERROR
- FATAL

Character Encoding

Character Encoding to be used; like Cp1252, ISO-8859-1, and so on.

Log Enabled

Click to enable the use of this Appender.

ConsoleAppender

Logs to the console (standard output). This is in the window where you started the server (ibmdisrv) or the execute task-window in the Configuration Editor (ibmditk). **Console** has the following parameters:

Layout

See **FileRollerAppender**, previous.

Pattern

See **FileRollerAppender**, previous.

Log level

See **FileRollerAppender**, previous.

Log Enabled

See **FileRollerAppender**, previous.

FileAppender

Logs to a file. **File** has the following parameters:

File Path

See **FileRollerAppender**, previous.

Append to file

Click to append log information to file. If option is not enabled, the file is overwritten.

Layout

See **FileRollerAppender**, previous.

Pattern

See **FileRollerAppender**, previous.

Log level

See **FileRollerAppender**, previous.

Character Encoding

Character Encoding to be used; like Cp1252, ISO-8859-1, and so on.

Log Enabled

See **FileRollerAppender**, previous.

This is the appender set up by default; see “Log4J default parameters” on page 229.

SyslogAppender

Enables IBM Security Directory Integrator to log on UNIX Syslog. **Syslog** has the following parameters:

Host name/IP Address

Host to log on to.

Syslog Facility

Legal facilities found in the drop-down. Must be supported by the host you are logging to.

Print Facility String

If set, the printed message includes the facility name of the application.

Layout

See **FileRollerAppender**, previous.

Pattern

See **FileRollerAppender**, previous.

Log level

See **FileRollerAppender**, previous.

Log Enabled

See **FileRollerAppender**, previous.

NTEventLog

Enables applications to log to the Windows NT Event Log (on Windows platforms). **NTEventLog** has the following parameters:

Source

The "source" name appearing in the NT event log; usually the title of the application doing the logging.

Layout

See **FileRollerAppender**, previous.

Pattern

See **FileRollerAppender**, previous.

Log level

See **FileRollerAppender**, previous.

Log Enabled

See **FileRollerAppender**, previous.

DailyRollingFileAppender

The daily rolling file appender rotates the log file every day. When the output file is rolled it is given a name consisting of the base name plus a date pattern string; that is, filename.yyyy-mm-dd. It usually is used with the **Append to file** parameter set to **true**. **DailyRollingFile** has the following parameters:

File Path

See **FileRollerAppender**, previous.

Append to file

Create new file or append to existing file, depending on whether this is checked. You usually want this on when using the **DailyRollingFile**.

Date Pattern

How often the file is rotated. Use the drop-down to choose resolution from minutes to months. For example, if the **File Path** is set to example.log and the **DatePattern** set to '. 'yyyy-MM-dd, on 2003-10-31 at midnight, the logging file example.log is copied to example.log.2003-10-31. Logging for 2003-11-01 continues in example.log until it rolls over the next day.

Layout

See **FileRollerAppender**, previous.

Pattern

See **FileRollerAppender**, previous.

Log level

See **FileRollerAppender**, previous.

Character Encoding

Character Encoding to be used; like Cp1252, ISO-8859-1, and so on.

Log Enabled

See **FileRollerAppender**, previous.

Also see the example under “Logging using the default Log4J class” on page 224.

SystemLogAppender

This Appender creates log files in a catalog hierarchy under *TDI_install_dir/system_logs*. For each Config File, there is a corresponding directory with logfiles named *AL_xxx*, where *xxx* is the name of the AssemblyLine being run.

This Appender has the following parameters:

Pattern

Specifies the format of the log as defined by LOG4J. The default value is:

```
"%d{ISO8601} %-5p [%c] - %m%n"
```

Additional values available in the field are:

```
"%d{HH:mm:ss} %p [%t] - %m%n"  
"%p [%t] %c %d{HH:mm:ss,SSS} - %m%n"
```

Log level

See **FileRollerAppender**, previous.

Character Encoding

Character Encoding to be used; like Cp1252, ISO-8859-1, and so on.

Log Enabled

See **FileRollerAppender**, previous.

Log Levels and Log Level control

You can refer to the log levels listed here.

Log levels can be

- ALL
- DEBUG
- INFO
- WARN
- ERROR
- FATAL
- OFF

ALL logs everything. DEBUG, INFO, WARN, ERROR and FATAL have increasing levels of message filtration. Nothing is logged on OFF.

You can issue log messages to the system or AssemblyLine logs by using the `logmsg()` method from JavaScript, wherever IBM Security Directory Integrator allows scripting. It can take one or two parameters. See the Java API documentation for the `logmsg()` declaration (package `com.ibm.di.server`, class `AssemblyLine` or class `RS`).

The interface for the `logmsg()` method (both main and task) with additional log level parameter is **logmsg (String logLevel, String msg)**. The legal values for `logLevel` are: "FATAL", "ERROR", "WARN", "INFO", "DEBUG", corresponding to the log levels available for log Appenders. Any unrecognized value is treated as "DEBUG".

Note that the IBM Security Directory Integrator `logmsg()` JavaScript calls `log` on INFO level by default. This means that setting `loglevel` to WARN or lower silences your `logmsg` as well as all Detailed Log settings. However, with the level parameter to the `logmsg()` call you can override the log level for individual `logmsg()` calls.

Log4J default parameters

You can refer to the default configuration for changing the content.

When IBM Security Directory Integrator is installed, a *FileAppender* is used for the default logger. If you want to change the default logger you must change the content of the `log4j.properties` file situated in the *TDI_installdir/etc* folder. The default configuration is as follows:

```
# This is the default logger, you will see that it logs to ibmdi.log
log4j.appender.Default=org.apache.log4j.FileAppender
log4j.appender.Default.file=logs/ibmdi.log
log4j.appender.Default.layout=org.apache.log4j.PatternLayout
log4j.appender.Default.layout.ConversionPattern=%d{ISO8601} %-5p [%c] - %m%n
log4j.appender.Default.append=false
```

The *FileAppender* logger truncates the content of the `ibmdi.log` file (situated in *TDI_installdir/logs*) each time the IBM Security Directory Integrator server is started. If want to change that behavior you must change the `log4j.appender.Default.append` property to true.

In the `log4j.properties` file you can find also two examples for changing the default logger to *RollingFileAppender* or *DailyRollingFileAppender*. If you want to use them just uncomment the preferred one and comment the *FileAppender* logger:

```
#####ROLLING FILE SIZE APPENDER
##RollingFileAppender rolls over log files when they reach a certain size specified by the
##MaxFileSize parameter

#log4j.appender.Default=org.apache.log4j.RollingFileAppender
#log4j.appender.Default.File=logs/ibmdi.log
#log4j.appender.Default.Append=true
#log4j.appender.Default.MaxFileSize=10MB
#log4j.appender.Default.MaxBackupIndex=10
#log4j.appender.Default.layout=org.apache.log4j.PatternLayout
#log4j.appender.Default.layout.ConversionPattern=%d{ISO8601} %-5p [%c] - %m%n

#####DAILY OUTPUT LOG4J SETTINGS
## With the DailyRollingFileAppender the underlying file is rolled over at a user chosen frequency.
##The rolling schedule is specified by the DatePattern option

#log4j.appender.Default=org.apache.log4j.DailyRollingFileAppender
#log4j.appender.Default.file=logs/ibmdi.log
#log4j.appender.Default.DatePattern='.'yyyy-MM-dd
#log4j.appender.Default.layout=org.apache.log4j.PatternLayout
#log4j.appender.Default.layout.ConversionPattern=%d{ISO8601} %-5p [%c] - %m%n
```

These are some of the parameters you find in the file `Log4J.properties` (for `ibmdisrv` and `ibmditk`).

Full documentation can be found at <http://jakarta.apache.org/log4j/docs>.

Log4J.rootCategory=DEBUG, Default

DEBUG is the loglevel for the named Appender (Log4J term called

Default). If you set the loglevel to OFF or to the level above INFO, you do not get output from your script logmessages (see the following log terms):

Log4J.appender.Default

Defines what type of Appender the named appender Default is. It can be one of the following:

- FileRollerAppender (generates a new file for each run of the Server)
- ConsoleAppender (log to console)
- FileAppender (log to file)
- SyslogAppender (log to UNIX Syslog)
- NTEventLog (log to Windows NT EventLog)
- DailyRollingFileAppender (saves old files with a datestamp in their names)
- SystemLogAppender (In a folder structure under *root_directory/system_logs*)

Log4J.appender.Default.file

Default log file for FileAppender, relative to your installation directory (default *ibmdi.log*).

Log4J.logger.com.ibm.di.*

Log level of various IBM Security Directory Integrator components. Note that, for example, *ibmditk* shows the log level of the IBM Security Directory Integrator Configuration Editor itself (not the processes you are running inside it). Do not change these.

Creating your own log strategies

You can use this framework to differentiate how the different AssemblyLines log.

You can use this framework to differentiate how the different AssemblyLines log.

Note: This information is intended for users who want to continue using the *global.properties* file to customize logging output. You can customize logging output through the Configuration Editor (*ibmditk*).

The following section defines a log scheme called *CONSOLE*, which later can be used by specific AssemblyLines:

```
Log4J.appender.CONSOLE=org.apache.Log4J.ConsoleAppender
Log4J.appender.CONSOLE.layout=org.apache.Log4J.PatternLayout
Log4J.appender.CONSOLE.layout.ConversionPattern=%d [%t] %-5p - %m%n
```

Now in order to have the AssemblyLines *myAL* use this, you need the lines:

```
Log4J.logger.AssemblyLine.myAL=INFO, CONSOLE
```

Refer to the full Log4J (version 1.2) documentation for description of the *ConversionPattern* parameters. Here are some parameters:

- %d** Date/time depending on format.
- %p** Priority.
- %c** Category.

Note: this is typically in the form *Type.alName.xxx*. *Type* can be *EventHandler* or *AssemblyLine*, *alName* is the name of the *AssemblyLine* (or *EventHandler* as named by the creator), and *xxx* is a unique ID for the thread. **%c{2}** outputs *alName & unique ID*.

%m Message.
%n Newline.
%t Threadname.

Chapter 15. Tracing and FFDC

This is a logging library similar to Log4J, but which is used inside IBM Security Directory Integrator specifically for tracing and First Failure Data Capture (FFDC).

In addition to the user-configurable logging functionality described in Chapter 14, “Logging and debugging,” on page 223, IBM Security Directory Integrator is instrumented throughout its code with tracing statements, using the JLOG framework. To which extent this becomes visible to you, the end user, depends on a number of configuration options in the global configuration file `jlog.properties`, and the Server command line option `-T`.

Note: Normally, you should be able to troubleshoot, debug and support your solution using the logging options described in the section, Chapter 14, “Logging and debugging,” on page 223. However, when you contact IBM Support for whatever reason, they may ask you to change some parameters related to the tracing functionality described here to aid the support process.

Tracing Enhancements

Most Connectors and Parsers have entry and exit trace statements. A number of classes on the IBM Security Directory Integrator server have trace statements added to the listed items.

Listed items:

- Method entry and exit points.
- Interactions with third party software.
- Thread creations.

Understanding Tracing

Tracing is done in the code of IBM Security Directory Integrator using JLOG's PDLogger object. PDLogger or the Problem Determination Logger logs messages in Logxml format (a Tivoli® standard), which IBM Support understands and for which they have processing tools.

The basic level of information traced, as handled by the PDLogger APIs, is:

```
Date | Time | ClassName | methodName | MachineName | IP | {Entry/Exit/Exception} | [Parameter]
```

Basic tracing information means Time, Level (Min, Mid, Max), Location in code, that is Method name and Entry/Exit. The "|" character serves a documentation purpose only, it is not part of the actual log.

Tracing is not performed using Log4J Appenders for the following reasons:

1. Trace is always to be enabled
2. You wouldn't want multiple traces enabled in the server (could be several for each AL if Appenders were used).

The PDLogger is attached to the JLOG **SnapMemory** handler and the **JlogSnapHandler**.

The **SnapMemory** Handler logs trace messages to memory. On the trigger of a LogEvent (that is, an occurrence of a specific Log level Trace message, as defined by the `jlog.levelflt.level` filter, or an application crash or on the occurrence of a specific TMS XML messageID) the Trace memory buffer is written to a file by the **JlogSnapHandler**.

To make Tracing and Log messages in IBM Security Directory Integrator unique across all IBM products, they are prefixed with a unique prefix: **CTGDI**.

All error messages are prefixed with a unique TMSXML messageID that indicates the cause of the error and an operator response.

All info messages are also prefixed with a unique TMSXML messageID that may or may not provide the operator response.

Configuring Tracing

You can use the `jlog.logger.level` property in the `jlog.properties` file to set the desired trace level.

The trace level can be set to any of the following JLOG log level (hierarchy, from most severe to least severe):

- FATAL
- ERROR
- WARNING
- INFO
- DEBUG_MIN
- DEBUG_MID
- DEBUG_MAX

The default level is FATAL.

Default Trace level as well as whether Tracing is done to file or memory is defined in the default `jlog.properties` file. This file is placed in the `TDI_install_dir/etc` folder. If you use a solution directory, it is placed in the `TDI_Solution_dir/etc` folder.

Setting trace levels dynamically

IBM Security Directory Integrator ships a `LogCmd.bat` (for Windows) and `LogCmd.sh` (for Unixes) scripts. You can use them to set the Trace properties dynamically.

JLOG logger starts a command server on the default port (9992) to listen for log commands sent by the `logcmd` command line utility.

For the `logcmd` scripts to work, the command server needs to be started first. To start the log command server, you need to set `jlog.noLogCmd=false` in the `jlog.properties` file.

The listen port of this server can be changed by setting the `jlog.logCmdPort` property in the `jlog.properties` file to the desired value. For more information about these properties read the comments in the `jlog.properties` file.

Usage of the `logcmd` command is as follows:


```
logcmd -o port_number { [-h] | [help] |
    [list {node_name} ] |
    [config node_name] |
    [set node_name key_name=value |
    [remove node_name {key_name} ] |
    [dump handler_name] | [save {all} ] }
```

where

-o port_number

The port number to use to connect to the log command server. If not specified the default port (9992) is assumed.

-h | help

Displays syntax information for the command.

list Lists the names of all known logging objects (nodes).

list node_name

Lists the names of the children of the node name. Not all logging objects have children.

config node_name

Lists the all the configuration properties for the node.

set node_name key_name=value

Sets a property key for the node name. If the logging object, node_name, does not exist, the logging object is created and the property is added.

remove node_name

Removes the configuration object node_name. A logging object that has been instantiated from this configuration is not affected by removing the configuration node.

remove node_name key_name

Removes the configuration property key_name from the logging object node_name. If the object supports a hierarchical inheritance of properties, a subsequent logcmd config node_name command may show the key just removed. In that case, it was inherited from an ancestor.

save {all}

Saves the logging configuration to persistent store. If **all** is specified, the entire configuration is saved; otherwise, only those configuration nodes that were originally loaded from the file are saved.

Useful JLOG parameters

You can refer to the listed JLOG parameters here.

Property	Value	Description
jlog.snapmemory.queueCapacity	Default 10000	The number of logevents that can be stored in the snapmemory handlers queue.
jlog.snapmemory.dumpEvents	true	The handler immediately sends all the queued events to its output listeners when the property is set to true. The property can then be reset to false.
jlog.snapmemory.userSnapDir	CTGDI/FFDC/user/	The directory to place the trace dump file when a user triggers an FFDC action by using the logcmd scripts.
jlog.snapmemory.isSync	Default false	The log events are dumped to the snap shot file synchronously when the property is set to true. This does not spawn a new thread, and causes the logger to block until the snapshot is complete.

Property	Value	Description
jlog.snapmemory.userSnapFile	userTrace.log	
jlog.snapmemory.triggerFilter	jlog.levelflt	The level filter to be used to take JFFDC action.
jlog.snapmemory.msgIds	*E	The TMSXML message filter to be used for JFFDC action.
jlog.snapmemory.mode	PASSTHRU or BLOCK. Default is PASSTHRU.	The listed IDs are blocked when the msgIDs property is set to BLOCK. When set to PASSTHRU, the listed IDs are sent to the filter.
jlog.snapmemory.msgIDRepeatTime	10000 (in milliseconds)	The minimum time in milliseconds, after passing a logEvent with a given TMS message ID, before another logEvent with the same id can be passed.

The default value for jlog.snapmemory.triggerFilter sets up a trigger filter named jlog.levelflt. An attribute of such a filter is the message severity, which takes one of the JLOG Log values as described above. By default, the entries

```
jlog.levelflt.className=com.ibm.log.LevelFilter
jlog.levelflt.level=FATAL
```

set up the FFDC code to cause the memory buffer to be dumped to the trace log when a trace message of severity FATAL occurs. The jlog.levelflt.level property can take any of the other Log level values as well, but only values of ERROR or FATAL make much sense as otherwise the amount of FFDC dumping is very high, causing huge slowdowns of the IBM Security Directory Integrator Server.

Chapter 16. Administration and Monitoring

You can use the AMC to start, stop and manage IBM Security Directory Integrator Configs and AssemblyLines remotely. You can know more about the AMC through the information provided here.

The IBM Security Directory Integrator Administration and Monitoring Console (AMC) user interface is deployed into the Integrated Solutions Console (ISC).

IBM Security Directory Integrator also ships an Action Manager with the AMC. The Action Manager is a stand-alone Java application that interacts with the AMC database and uses the Remote Server API to manage remote AssemblyLines.

The Administration and Monitoring Console is comprised of a Java WAR file (Web Archive) and a WAB file (Web Bundle) that can be deployed on ISC SE and IBM Dashboard Application Services Hub.

The current Action Manager, bundled with IBM Security Directory Integrator AMC, supports IBM Security Directory Integrator Version 7.1.1, 7.1, and 7.0. Versions of IBM Security Directory Integrator earlier than version 7.0 are not supported.

Note: IBM Security Directory Integrator and solutions developed and deployed with it can also be monitored with IBM Tivoli Monitoring (ITM) Server and Portal or IBM Netcool[®]/OMNIBus, by virtue of IBM Security Directory Integrator's Java Management Extension (JMX) interface. You can find supported examples that show how you can accomplish this in the appendix entitled, Appendix B, "Monitoring with external tools," on page 349.

Installation and Configuration

You can use the links provided here to install the Administration and Monitoring Console.

See "Installing IBM Security Directory Integrator" on page 8 for information about installing IBM Security Directory Integrator and the Administration and Monitoring Console. Installing AMC also installs the Action Manager. If you choose a custom IBM Dashboard Application Services Hub to deploy AMC or defer deployment of AMC during installation, see "Deploying AMC to a custom ISC SE or IBM Dashboard Application Services Hub" on page 47 for information on additional deployment requirements.

Deploying AMC into the Integrated Solutions Console

You can deploy the AMC into the Integrated Solutions Console using the instructions provided here.

About this task

These instructions require that you be familiar with the IBM Security Directory Integrator Installation procedures. See "Using the platform-specific IBM Security Directory Integrator installer" on page 12 for information about IBM Security Directory Integrator Installation. Installing AMC also installs the Action Manager.

If you want to deploy IBM Security Directory Integrator Administration and Monitoring Console into the ISC automatically, select one of the following options:

- Embedded instance of ISC SE
- Existing instance of ISC

If you do not want to deploy AMC into ISC automatically at installation time, select "Do not specify. I will manually deploy AMC at a later time."

The installer automatically installs ISC and deploys AMC in it if you select "Embedded instance of ISC SE" or "Existing instance of ISC" during IBM Security Directory Integrator installation.

To install and deploy the Administration and Monitoring Console into the ISC SE:

1. Invoke the IBM Security Directory Integrator installer.
2. During installation, select the **Custom** installation. (Typical installation does not offer the AMC option.)
3. On the "Select Features" window of the installation, select **AMC: Administration Monitoring Console** and **embedded Web platform** (includes Integrated Solutions Console, Standard Edition (ISC SE)).
4. Finish installing IBM Security Directory Integrator.

Deploying AMC as a Windows service or UNIX process using the IBM Security Directory Integrator installer

You can deploy the AMC as a Windows service or UNIX process using the IBM Security Directory Integrator installer.

You can register AMC as a Windows service or UNIX process if the following conditions are satisfied:

- The person installing IBM Security Directory Integrator must have administrative permissions (Administrators group on Windows or root on UNIX).
- You have selected to install AMC into an "Embedded instance of ISC Standard Edition."
- You have selected "Register AMC as a system service" and given the service a name. The default service name is `tdiamc`

Deploying AMC on existing IBM WebSphere Application Server environment

You can deploy the AMC on existing IBM WebSphere Application Server environment.

To deploy AMC on the existing IBM WebSphere Application Server environment, modify the `tdiISCHome.bat` or `tdiISCHome.sh` script file to set the following parameters:

- Set the **TDI_ISC_RUNTIME** parameter to IBM WebSphere Application Server
- Set the **TDI_ISC_HOME** parameter to `WAS_HOME` directory

For example:

```
set TDI_ISC_RUNTIME=WAS
set TDI_ISC_HOME=C:\Program Files\IBM\WebSphere\AppServer
```

Starting and logging in the AMC and Action Manager

You can start and stop the Administration and Monitoring Console and Action Manager by running the scripts provided here.

About this task

The scripts are shipped in the *TDI_install_dir/bin/amc* folder:

- To start AMC, run the `start_tdiamc` script.
- To start AM, run the `startAM` script.

For more information about these scripts, see “AMC and AM Command line utilities” on page 283.

The above will start AMC and AM using a Derby database configured locally on the machine running in network mode on localhost at port 1528. For more information on alternate settings and configurations for both AMC and AM, see sections “Enabling AMC” and “Enabling Action Manager” on page 248.

Once the Administration and Monitoring Console is started, you can access it from the following URL: <http://localhost:13100/ibm/console>; for more information, see “Log in and logout of the console” on page 255.

Stopping AMC and AM can be accomplished by running the `stop_tdiamc` and `stopAM` scripts, respectively.

Note:

1. For information on adding users and user roles, see section “AMC in the Integrated Solutions Console” on page 242.
2. For information on AM, see section “Action Manager” on page 243.
3. For information on usage of individual panels in AMC, see the online panel help, or section “Administration and Monitoring Console User Interface” on page 255.
4. You will be registering IBM Security Directory Integrator Servers in AMC so that solutions from IBM Security Directory Integrator configurations can be administered. AMC will only be able to find Configs that each IBM Security Directory Integrator server has loaded when its started up. By default, IBM Security Directory Integrator Servers have the remote server API enabled. Ensure that your *TDI_install_dir/configs* folder has the Configs you want to administer and monitor (or put them in the config folder of your solution directory if your server is using a solution directory).
5. For an example walk through of on using AMC and Action Manager for a few simple tasks, see “Example walkthrough of creating a Solution View and Rules” on page 288.

Enabling AMC

Refer to steps listed here to configure the Administration and Monitoring Console.

The configuration file for the Administration and Monitoring Console is the `amc.properties` file that is located at the same level as the `WEB-INF` directory. This file contains the AMC's database configuration properties, LDAP properties, SSL related properties and help server details.

By default, the Administration and Monitoring Console makes use of Derby Version 10 to store data. When AMC is started for the first time, AMC creates a `tdiamcdb` folder inside the Web server directory and creates the tables needed for AMC to function. The Derby database can be accessed in either the network mode

or embedded mode. By default, AMC is shipped with Derby configured in network mode. The following properties in `amc.properties` are applicable to Derby configured for network mode:

```
com.ibm.di.amc.jdbc.database=jdbc:derby://localhost:1528/tdiamcdb;create=true
com.ibm.di.amc.jdbc.driver=org.apache.derby.jdbc.ClientDriver
com.ibm.di.amc.jdbc.urlprefix=jdbc:derby:
com.ibm.di.amc.jdbc.user=APP
com.ibm.di.amc.jdbc.password=APP
com.ibm.di.amc.jdbc.start.mode=automatic
com.ibm.di.amc.jdbc.host=localhost
com.ibm.di.amc.jdbc.port=1528
com.ibm.di.amc.jdbc.sysibm=true
```

The property `com.ibm.di.amc.jdbc.database` points to Derby in network mode, running on `localhost:1528`. The database name being accessed is `tdiamcdb`, and `create=true`, indicating that AMC should create the database if not found.

You should change the `create=true` to `create=false` once your environment is set, so that in case the database path gets modified, AMC does not re-create the database, but instead throws a "Database not found" exception. You should also ensure that the database path be set to an absolute path to avoid any confusion about the database path later.

Other databases than Derby can be configured by setting the appropriate properties; see "Console Properties" on page 261.

AMC provides a separate startup and shutdown script for Action Manager. AMC allows the Action Manager to run remotely and provides a separate Derby start or shutdown script.

The Administration and Monitoring Console can also be configured to connect to the Derby database in Embedded Mode. In this case, the Action Manager, a separate application that also talks to the AMC database, is unable to connect to AMC's database. This is because in Embedded Mode, only one JVM at a time is allowed to connect to the Derby database. The following example shows the `amc.properties` file with Derby configured for embedded mode:

```
##Location of the database (embedded mode)
configured for embedded mode:
##Location of the database (embedded mode)
com.ibm.di.amc.jdbc.database=tdiamcdb
com.ibm.di.amc.jdbc.driver=org.apache.derby.jdbc.EmbeddedDriver
com.ibm.di.amc.jdbc.urlprefix=jdbc:derby:
com.ibm.di.amc.jdbc.user=APP
com.ibm.di.amc.jdbc.password=APP
```

The `com.ibm.di.amc.jdbc.database` property points to the location of the AMC database. We suggest that this value be set to an absolute path to avoid any confusion about the database path later.

Running Action Manager remotely

You can use the steps listed here and take care of specified instructions to run the Action Manager remotely.

Beginning with IBM Security Directory Integrator 7.0, you can run Action Manager remotely without starting the AMC first. The database for AMC, Derby, must be running in Network mode in order for Action Manager to connect to it. IBM Security Directory Integrator 7.0 also provides start and shutdown scripts for the Derby data store so that a user can start Action Manager remotely without starting the AMC.

Note:

1. Before you start Action Manager for the first time, you must have run AMC at least once. This is because AMC creates the necessary database tables required for AM.
2. You can find the scripts in this section in the following folder of the Install Directory of the remote computer: *TDI_install_dir\bin\amc\ActionManager*.
3. The instructions in the startup and shutdown sections that follow are for Action Manager and Derby running on different remote computers, and with AMC not running.
4. To verify that AMC, Action Manager or Derby has stopped, check the logs.

AMC and Action Manager startup

If you want to run Action Manager and Derby with AMC running, start AMC by typing `start_tdiamc.bat(sh)` and start the Action Manager by typing `startAM.bat(sh)`. The `tidamc` script calls the `startNetworkServer.bat(sh)` script, thereby starting the Derby database in network mode.

Note: The `startAM.bat(sh)` script has the Classpath defined for all the jars required by the Action Manager. There are two variables namely `CLASSPATH` and `DB_CLASSPATH`. The `DB_CLASSPATH` has the path separated list of JAR files required for achieving JDBC Connectivity with the database. When AMC is configured to use Oracle, MS SQL Server or DB2 the corresponding JDBC JAR files of these databases should be added to the `DB_CLASSPATH` variable.

AMC and Derby shutdown

The `stop_tdiamc.bat(sh)` script calls the `stopNetworkServer.bat(sh)` script. This ensures that the Derby Network server is stopped when AMC is shutdown.

Note: If Action Manager (AM) is running, this should be shut down first.

Action Manager remote startup

This section assumes that Action Manager and Derby are running on different computers.

1. Start Derby using the script `startNetworkServer.bat(sh)`.
2. Start Action Manager using the script `startAM.bat(sh)`.

The `startNetworkServer` script is used for starting the Derby database server in Network mode. The Derby server starts in Network mode on port 1528. The port selected is different from the default port for Derby.

Action Manager shutdown

The Action Manager is stopped using the `stopAM.bat(sh)` script located in the *TDI_install_dir/bin/amc* directory. This script uses the processID of the started AM to kill it. The processID is obtained by the `startAM` script and is stored in a file, which in turn is read by the `stopAM` script.

To stop the Derby database, type `stopNetworkServer`, which stops the Derby database server in Network mode. This should be done after AM is stopped, not before.

AMC Logs

The Administration and Monitoring Console logs are stored in the ISC log in the environment in which AMC runs. You can refer to the information provided here to configure the logs.

- for ISC SE, the log file is created under `#{LWI_HOME}/logs`;
- for IBM Dashboard Application Services Hub, the logs are logged in `#{WAS_HOME}/profiles/#{profileName}/logs/#{serverInstance}/SystemOut.log`

The configuration of AMC logs can be done by modifying the `WEB-INF/classes/logging.properties` file. AMC logging follows the Java logging standard (`java.util.logging`).

You can view and delete AssemblyLine logs in the **AssemblyLine Logs** window. To reach **AssemblyLine Logs** on the **Monitor Status** window: (**Monitor Status > Solution View Details > View Logs**). In the **Solution View Details** window, select the AssemblyLine whose logs you want to view. In the **AssemblyLine Logs** window, select any logs you want to delete, and select **Delete**. You can delete one or multiple logs. To view a log, click its hyperlink.

The **Solution View Details** window also contains the **Action Manager Logs** table. You can select and delete logs from **Action Manager Logs**.

You can manage all of your logs in the **Log Management** window. You can specify criteria for displaying logs, and you can delete logs for all AssemblyLines or for a single AssemblyLine. You can select to delete all logs for AssemblyLine(s) specifying a date range, or you can delete the *n* most recent logs, where you enter the number of most recent logs.

AMC in the Integrated Solutions Console

You can view the list of changes integrated in the AMC, learn to add and remove users, and different kind of roles that can be assigned through the information provided here.

The Integrated Solutions Console (ISC) is designed to offer a common console to organize administrative console functions using industry-standard technologies. Starting with IBM Security Directory Integrator 7.0, integration of the AMC into the Integrated Solutions Console (ISC) comes with the following changes. The primary navigation links for AMC are:

- **Administration and Monitoring Console**
 - Servers
 - Solution Views
 - Monitor Status
 - Action Manager
- **Advanced**
 - Log Management
 - Console Properties
 - Preferred Solution Views
 - Property Stores

Console user authority

Using **ISC Console User Authority**, you can add and remove users to AMC. In the AMC for IBM Security Directory Integrator v7.0 and later, the following are the roles:

Table 28. AMC roles

User Role	Description
administrator	Users with this role assigned are able to configure the roles other users are assigned to.
iscadmins	Users with this role assigned have the ability to control the settings of the ISC console itself.
Security Directory Integrator AMC Admin	This role is considered by the IBM Security Directory Integrator AMC application deployed in the ISC console. Users with this role assigned are able to administer Servers, Solutions View roles, Manage Console Properties. (This was the superadmin role in the AMC prior to IBM Security Directory Integrator 7.0)
Security Directory Integrator AMC User	This role is considered by the IBM Security Directory Integrator AMC application deployed in the ISC console. Users with this role assigned are able to use the provided by the IBM Security Directory Integrator AMC Admin resources.

Within the *SDI AMC user* role, user privileges are assigned roles found in the Solution View:

- Admin
- Config Admin
- Execute
- Read

The roles control access to functions on the console. You can see only those functions for which you have roles assigned. For example, users with the Security Directory Integrator AMC Admin role automatically have administrator privileges over all Solution Views. Administrators can configure the properties required for the Web administration tool (modifying properties related to `amc.properties` file, available from Console Properties in the left navigation pane). A user with the Security Directory Integrator AMC User role in ISC is the same as the current non-admin AMC user. Security Directory Integrator AMC Users cannot access any administrative windows such as IBM Security Directory Integrator Servers and Console Properties.

ISC features the AMC Admin Group or `iscadmins`. A user in the `iscadmins` group has the same privileges as the administrator.

Administrator and the `iscadmins` group

The administrator, defined as the user who has installed the application, can manage AMC users. The administrator can add or remove users from the local OS Registry to the AMC application and assign or edit roles. IBM Security Directory Integrator v7.0 and later offers a new **SDI AMC Admin** group. The superadmin role does not exist after IBM Security Directory Integrator version 6.1.1.

Action Manager

The Action Manager is a standalone Java application that allows you to monitor multiple IBM Security Directory Integrator Servers and AssemblyLine execution using user-defined rules, triggering conditions and actions defined in AMC. You can view the list of different trigger types, actions, and threads here.

The Administration and Monitoring Console (AMC) has an Action Manager window that allows users to configure various Action Manager rules.

A rule is a combination of a trigger type and a set of associated actions. A rule specifies that when a triggering condition is detected, then the associated set of actions must be executed. The various trigger types available in AMC are described below:

Table 29. Action Manager triggers

Trigger Type	Trigger Details	User Input for trigger
No trigger	A rule with this triggering type has no triggering condition, and as a result never gets triggered by itself. The only way this rule can be executed is if some other rule executes this rule	No details required
On AssemblyLine start	A rule with this triggering type gets triggered when the Action Manager receives an AL start event for this particular AL.	"AssemblyLine name"
On AssemblyLine termination	A rule with this triggering type gets triggered when the Action Manager receives an AL termination event for this particular AL.	"AssemblyLine name"
On config load	A rule with this triggering type is triggered when a Config is loaded.	"Name of Config to load"
On config unload	A rule with this triggering type is triggered when a Config is unloaded. The Config must be loaded to be unloaded.	"Name of Config to unload"

Table 29. Action Manager triggers (continued)

Trigger Type	Trigger Details	User Input for trigger
On query AssemblyLine result	<p>Note: A rule with this triggering type should not be used with a short-running AL. This is because Action Manager stores the handle of the AL object on receiving the Start AssemblyLine event. Later on, receiving the Stop AssemblyLine event, Action Manager uses this handle to query the final work entry attributes. If the AL terminates before Action Manager can store the handle, then Action Manager is not able to query the work attributes. Usually an execution time of 10 seconds is sufficient (this can be achieved by putting a <code>system.sleep(10)</code> before the AL terminates, for example in the epilog hook).</p> <p>When running On query AL result, Action Manager polls the AL continuously for the specified polling interval. The trigger first checks for the attribute value, starting the AL after the specified polling interval. Next, the trigger checks the AL result entry.</p> <p>On query AL result is a rule that is triggered when the last "work" entry of the specified AL contains the specified "Attribute" matching the given "condition" and "value". This condition is checked only when the ActionManager receives a Stop AssemblyLine event. The user can specify a time interval. The specified AL run periodically depending upon the time interval specified A rule with this triggering type is triggered when the last "work" entry of the specified AL, contains the specified "Attribute" matching the given "condition" and "value". This condition is checked only when the Action Manager receives a Stop AssemblyLine event.</p> <p>To configure the Query on AL result trigger, enter values for the following fields:</p> <ul style="list-style-type: none"> • AssemblyLine name • Attribute • Condition • Value • Polling Interval • Polling Unit 	"AssemblyLine name", "Attribute", "Condition", "Value", "Polling Interval," and "Polling Unit."
On server API failure	A rule with this triggering type is triggered when the Action Manager is unable to connect to the remote server using the Server API. You can configure different polling time intervals for each Assembly Line depending upon AL execution time.	"Polling Interval" and "Polling Unit."
On received Event	A rule with this triggering type is triggered when the Action Manager receives an event which satisfies the criteria mentioned. Note: If any of the criteria are to be ignored, just leave it blank.	"Event type", "Event Source", "Event Data". Event Data is optional. Event type or source - one of them must be specified.

Table 29. Action Manager triggers (continued)

Trigger Type	Trigger Details	User Input for trigger
On Property	<p>A rule with this triggering type is triggered when the specified property meets the specified condition. The Action Manager periodically checks for this property. You can configure a polling interval and polling units when configuring this trigger.</p> <p>Note: This rule gets triggered only once, and gets reset back to ready state only when Action Manager detects that this property does not meet the specified criteria any longer. This is done so that the rule does not repeatedly get triggered for a single occurrence of the triggering condition.</p>	"Polling Interval," "Polling Unit"."Property name", "Condition", and "Value".
On local variable	<p>A rule with this triggering type is triggered when the specified variables meet the specified condition. The Action Manager periodically checks for this property .</p> <p>Note: This rule gets triggered only once, and gets reset back to the ready state only when Action Manager detects that this variables does not meet the specified criteria any longer. This is done so that the rule does not repeatedly get triggered for a single occurrence of the triggering condition.</p>	"Local Variable" ," Condition", "Value".
Inspect AssemblyLine exit code	<p>A rule with this triggering type is triggered when an AssemblyLine terminates with an error.</p> <p>Inspect AL Exit Code also searches for an error object string for every abnormal AL termination. Under Configure Trigger, if the trigger is Inspect AL Exit Code, you can enable Inspect Error Object. In the Value field, type the string you want for Error Object. Note that if the Value field is empty, then the rule triggers for every abnormal termination of an AL. If Inspect Error Object is not selected, the trigger waits for the AL to terminate and inspects the exit code for an attribute value (entered by the user). Type values for both the Attribute Name and Value.</p> <p>In the Inspect AL Exit code trigger, the Action Manager no longer starts the AL, and there is no polling. The trigger only checks the AL result one time after the AL runs.</p>	"AssemblyLine name"; if "Inspect Error Object" is enabled, you only need supply "Value." If "Inspect Error Object" is disabled, values for "Attribute," "Condition," and "Value" are needed.
Time since last execution	<p>A rule with this triggering type get triggered when the Action Manager detects that the specified assembly line has not run for the specified period. Note: This rule is triggered only once. After that Action Manager wait for receiving a Start AssemblyLine event before resetting the Rule back to Ready mode. This is done so that the rule does not repeatedly get triggered for a single occurrence of the triggering condition.</p>	"AssemblyLine name", "Not Run Since" and "Unit".
Timer	<p>A rule with this triggering type is triggered continuously within an interval defined by a number of units and the measure of seconds, minutes, hours or days.</p>	"Interval" and "Unit."

When a rule gets triggered, the Actions associated with the rule are executed by the Action Manager sequentially. The following are the various types of Actions that are available in AMC:

Table 30. Action Manager actions

Action	Action Details	User Input for action
Start AssemblyLine	This action starts the specified AL of the specified config file on the specified IBM Security Directory Integrator server. The Config field should mention the complete path of the configuration on the remote server. The Config Password field is optional and is required only if the remote config is password protected.	"AssemblyLine", "Of Configuration", "On Server", "Config Password".
Stop AssemblyLine	This action stops the specified AL of the specified configuration on the specified IBM Security Directory Integrator Server. The Config field should mention the complete path of the configuration on the remote server.	"AssemblyLine", "Of Configuration", "On Server".
Enable/Disable Rule	This action Enable or Disable the chosen rule.	"RuleName" "State"
Execute Rule	This action cause the execution of the specified rule, which in-turn imply execution of all the actions specified in that particular rule.	"RuleName"
Notify Event	This action cause the Action Manager to emit an event with the specified details to the Server associated with the current Solution View. See the Session.sendCustomNotification() API for details.	"Event type", "Source", "Data".
Modify Property	This action cause the Action Manager to modify the selected property based on the specified operation.	"Property", "Operation", "Value".
Copy Property Value	This action cause the Action Manager to copy the value of the Source property to the Destination property.	"From Property", "To Property".
Write to Log	This action causes a log of the specified Severity/Message/Description to be logged into the Action Manager logs and the AMC database. The same log is shown when the user goes to the Monitor Status -> Solution View Details -> Action Manager Results table. It is advised to always have at least one Log action (containing descriptive text) in every rule.	"Severity", "Message", "Description".
Send Email	This action causes an email to be sent to the recipient you specify. You supply the content of the email. Along with the content, the Action Manager provides other details before sending the mail. In the content input area as well as in the subject line, you can specify the variable %EVENT_DATA% value. Specifying %EventData% inserts the actual value of the Eventdata variable when the mail is sent. %Action_Error% can also similarly be substituted here. If Attach Action Manager Log is enabled, the Action Manager logs (as specified in the am_logging.properties file) are sent as an email attachment.	"To", "From", "Subject", "Attach Action Manager Log" (Selected/Not Selected), "Content".

Table 30. Action Manager actions (continued)

Action	Action Details	User Input for action
Modify local variables	This action causes the action manager to increment, decrement or set the value of the specified variable to the specified value.	"Variable", "Operation", "Value".
Execute command	This action causes the specified command to execute on the target computer. The command can be any generic command or an IBM Security Directory Integrator specific command.	"Target Machine", "Port", "Username", "Password", "Keystore", "Keystore Password", "Protocol", and "Command".

Rules that are configured for Solution Views in AMC, are stored in AMC's Derby Database. When the Action Manager is run, it connects to the AMC database in network mode, reads the Action Manager-related tables, and creates threads in memory for every rule specified. Each of these threads listens/polls for its respective triggering conditions. The moment any thread detects the occurrence of its respective triggering condition, it queries the database for the set of actions associated with the rule, and executes them sequentially.

The Action Manager runs the following threads in addition to the rule threads that are listening for trigger conditions:

1. HealthAssemblyLine - The Health AssemblyLine thread periodically triggers the Health ALs for querying the status of the solutions, and logging the status back into the AMC database. The health AL must store the status in the "healthAL.result" and "healthAL.status" attributes of their final work entry.
2. ServerStatusListener - The ServerStatusListener thread is created for every server registered with AMC. This thread checks for the server accessibility. If the server has become inaccessible, all rules threads created for the server are terminated (except for those with triggering type 'On Server API failure'). Similarly if the server becomes accessible, rule threads are created for any rules associated with this server.
3. ConfigLoadReloadListener - The ConfigLoadReloadListener thread is created for every running server registered with AMC. It is registered to the remote server for any config load unload events. Rule threads are appropriately terminated, created or refreshed depending on the config event.
4. ServerModificationListener - The ServerModificationListener thread checks for any updates to the set of servers registered in AMC. Depending on the type of change (added, removed, and so on.) rule threads are terminated, created or refreshed.
5. DatabaseModificationListener - This database listener thread continuously monitors addition, modification or deletion of rules. Whenever any changes in the rules are detected, the Action Manager threads are added/recreated appropriately at runtime.

The Action Manager also updates the AMC database with its run details. Whenever an Action Manager rule is triggered, Action Manager logs an entry into the AMC database, registering the rule name that got triggered, and the triggering time. Also, if any Log action is configured for the rule, then that also gets logged into the AMC database. These database entries are used to show the appropriate status in Monitor windows of AMC.

Enabling Action Manager

You can use the instructions provided here to enable the Action Manager.

The Action Manager is installed in the *TDI_install_dir/bin/amc/Action Manager* folder. It contains the following files:

- *am_logging.properties* - This file controls Action Manager logging properties. Just like AMC, it also follows the *java.util.logging* logging standard.
- *am_config.properties* - This is the configuration file for the Action Manager.
- *testadmin.jks* - This is the ActionManager's truststore and keystore file.

Note: This is a sample truststore and keystore file; for added security, you should generate your own.

The Action Manager connects to AMC's Derby database using the Network Mode driver.

The following properties (in *am_config.properties*) must point to the Administration and Monitoring Console's database:

```
com.ibm.di.amc.am.jdbc.database=jdbc:derby://localhost:1528/C:/Program Files/IBM/AppSrv
/profiles/amcprofile/tdiamcdb;create=false
com.ibm.di.amc.am.jdbc.driver=org.apache.derby.jdbc.ClientDriver
com.ibm.di.amc.am.jdbc.urlprefix=jdbc:derby:
com.ibm.di.amc.am.jdbc.user=APP
{protect}-com.ibm.di.amc.am.jdbc.password=APP
com.ibm.di.amc.am.jdbc.start.mode=automatic
com.ibm.di.amc.am.jdbc.sysibm=true
com.ibm.di.amc.am.jdbc.networkserver.host=localhost
com.ibm.di.amc.am.jdbc.networkserver.port=1528
```

Note: Both AMC and AM support alternative databases, like MS SQL, Oracle and so forth. In order for AMC and AM to connect to one of those alternative databases, the configuration statements in *amc.properties* and *am_config.properties* will look very different.

When the Action Manager is started, it attempts to connect to AMC's database. If it fails in performing any initial setup tasks, it exit with an exception message. Check the *am_config.properties* file to ensure it points to the correct database. If the database settings appear to be correct, then ensure that the database that Action Manager is attempting to connect to is running in network mode and that AMC can connect to the same database. You may use the *startNetworkServer.bat* (sh) to start the Derby DB in network mode.

SSL settings and encryption properties for AM are configured in the following set of properties:

```
# Action Manager SSL properties
javax.net.ssl.trustStore=TDI_Install_dir/serverapi/testadmin.jks
{protect}-javax.net.ssl.trustStorePassword=administrator
javax.net.ssl.trustStoreType=jks
javax.net.ssl.keyStore=TDI_Install_dir/serverapi/testadmin.jks
{protect}-javax.net.ssl.keyStorePassword=administrator
javax.net.ssl.keyStoreType=jks
# Action Manager encryption properties
com.ibm.di.amc.am.encryption.keystore = TDI_Install_dir/testserver.jks
com.ibm.di.amc.am.encryption.key.alias = server
com.ibm.di.amc.am.encryption.keystoretype = jks
com.ibm.di.amc.am.encryption.transformation = RSA
com.ibm.di.amc.am.stash.file = TDI_Install_dir/idisrv.ssh
```

These properties are similar to the encryption properties used by the server. For convenience the location of the stash file has been added as a property: *com.ibm.di.amc.am.stash.file*. By default the AM will reuse the server's keystore and stash file for encryption/decryption of AM protected properties.

Further configuration of run-time rules, triggers and actions is described under "Action Manager" on page 272.

Action Manager status in real time

You can view the status of Action Manager in a window by following the instructions provided here. Further you can also view the contents of the window.

When you login to AMC, a one line Action Manager status displays in the **Welcome to AMC** panel. The Welcome to AMC panel displays the Action Manager status in a link, for example, "Action Manager is running" or "Action Manager is not running." To launch the **Action Manager Status** window, click the hyperlink. The Action Manager Status window displays Action Manager Status in real time, as well as thread details and trigger details. This window displays status information in real time. This window shows:

- Action Manager Status, for example the Boot Time
- Action Manager Thread Details
- Action Manager Trigger Details

The AMC directly queries the Action Manager using the APIs exposed by Action Manager. If AMC cannot establish a session with the Action Manager, the AMC concludes that the Action Manager is not available because the Action Manager has stopped. In addition to Action Manager status, AMC displays details of thread information and trigger details.

Action Manager creates a number of threads. Some Action Manager threads monitor the essential functionality of the Action Manager such as the Database Modification Listener and the ServerStatusListenerThread. Moreover, from these threads the Action Manager creates threads for each of the trigger rules that is configured in AMC. With the Remote Method Invocation (RMI) Layer, AMC can query the status of the various trigger-related threads. Using the RMI based query, AMC knows the state of these threads, thread priority, and so on. AMC can also query the triggers that have been executed over a period of time.

Two new properties belong to the **Display real time Action Manager Status** requirement. The properties that allow AMC to display the Action Manager status in real time are `am.api.host` and `am.api.port`. Action Manager status used an RMI layer around the Action Manager that exposes an API to be used by AMC for querying the Action Manager for its status.

AMC force trigger for a given rule

You can use the **Force Trigger** to execute the actions configured for the selected rule.

AMC allows users to force a trigger for a specific rule. Forcing a trigger gives the user an idea of what the Action Manager does when the rule is triggered. When you select **Disable Rule**, the selected rule is disabled.

Action Manager can execute a set of actions configured for a particular trigger (rule) explicitly. The AMC user does not have to wait for the triggering condition of a rule to be satisfied before the configured actions are carried out. Users can define actions that are to be executed so that they can test those actions. Users can execute all of the supported actions using **Force Trigger**. However, the **Revert** action is effective for only some (a subset of) the supported actions.

- Modify Property

- Copy Property
- Write to Log
- Enable Disable Rule

AMC and Action Manager security

The Administration and Monitoring Console (AMC) is a web-based application for monitoring and managing remote IBM Security Directory Integrator solutions. You can learn about its features and different security combinations through the information provided here.

Introduction

The following features of AMC have been improved:

- Encryption or concealment of passwords that are stored in the `amc.properties` file
- Use of stash files to store keystore passwords
- Enablement of the BUILTIN authentication scheme in the Derby database

AMC uses the Remote Server API to communicate with IBM Security Directory Integrator. For this reason, all the security restrictions and configuration settings that are applicable to IBM Security Directory Integrator Remote Server API clients (as mentioned in previous sections) are valid for AMC too.

Action Manager is installed with AMC. Action Manager configures itself and behaves based on rules set in the AMC database by AMC users. To monitor remote AssemblyLines and to take action based on configured rules, Action Manager, just like AMC, uses the IBM Security Directory Integrator Remote Server API to communicate with IBM Security Directory Integrator servers.

Note: Communication between AMC and AM using RMI is not protected in any way.

AMC and SSL

You can run the Administration and Monitoring console in SSL mode by following the instructions provided here.

Multiple IBM Security Directory Integrator servers can be registered with AMC. Each IBM Security Directory Integrator server may be configured differently; one IBM Security Directory Integrator server could be running with SSL off, one with SSL on, one with Custom Authentication on and SSL on - and various other combinations. AMC can be used to connect and administer any of these servers simultaneously. As mentioned earlier, to configure IBM Security Directory Integrator to run in SSL mode the `api.remote.ssl.on` property should be set to **true** in `global.properties` (or `solution.properties`).

As AMC is a web application running inside a Web Container it automatically inherits some properties and security restrictions from the Web Container. For instance, if the Web Container has an SSL keystore or SSL truststore configured, then that would be automatically applicable to AMC. But AMC can also override that - and specify its own keystore and truststore.

For being able to communicate with IBM Security Directory Integrator Remote Server API running on SSL, AMC must have a keystore configured which contains

the certificate that is trusted by the IBM Security Directory Integrator remote Server API (that is, it must be present in IBM Security Directory Integrator's truststore's trusted certificates section) and AMC must have a truststore configured which contains the certificate that is sent by the IBM Security Directory Integrator remote Server API. In other words - the certificate that is present in IBM Security Directory Integrator server's keystore must be present in AMC's truststore and the certificate that is present in IBM Security Directory Integrator truststore must be present in AMC's keystore.

For example, the default installation of IBM Security Directory Integrator is shipped with certain stores (.jks files). When you run IBM Security Directory Integrator in SSL mode, then to connect to AMC its keystore and truststore must both be set to the same value: *TDI_install_dir/serverapi/testadmin.jks* and the password being "administrator". Since testadmin.jks contains both trusted certificates and signer certificates, a connection gets established. It is recommended that you set up your own SSL keystores and truststores.

In AMC, the path of the truststore and keystore can be set by logging into AMC as "SDI AMC Admin" (Console Administrator) and navigating to the following window: **Advanced -> Console Properties -> SSL settings**. The settings for truststore and keystore are written to **amc.properties** file inside the tdiadc folder in Web Container. You can alternatively choose to edit the **amc.properties** file directly. With IBM Security Directory Integrator 7.0, AMC can be deployed in ISC Standard Edition (SE) or in ISC Advanced Edition (AE). Depending on the ISC runtime, the location of the testadmin.jks file varies. For example, if AMC is deployed in ISC SE, then the location will be *ISC_RUNTIME_INSTALL_DIR/runtime/isc/eclipse/plug-ins/AMC_7.0.0*. On the other hand, if AMC is deployed in IBM Security Directory Integrator, then the location is *ISC_RUNTIME_INSTALL_DIR/systemApps/isc-lite.ear/tdiadc.war*. The keystore and truststore password are set to "administrator" by default. To establish an SSL based connection with a remote IBM Security Directory Integrator server, you must start the server in "SSL enabled" mode, and for a Non SSL based connection, start the server in "SSL disabled" mode.

Attention: Default SSL settings are provided. However, using the default certificates does not increase the security more than just using a plain connection, so after installation, you should replace the default SSL certificates and update the keystores and truststores accordingly in order to increase security.

For each IBM Security Directory Integrator server running over SSL that you wish to register with AMC, you must import the necessary certificate into AMC's truststore and the necessary AMC's key certificate into IBM Security Directory Integrator's truststore. The idea here is that AMC must trust IBM Security Directory Integrator and IBM Security Directory Integrator must trust AMC to be able to make a secured two-way SSL connection.

Since AMC runs inside a Web Container, the URL for AMC is <http://hostname:port/ibm/console>.

Action Manager monitors running Configs and AssemblyLines on remote IBM Security Directory Integrator Servers based on rules configured in AMC. Action Manager ships with the keystore and truststore required to connect to a remote IBM Security Directory Integrator server. The SSL properties are defined in the *am_config.properties*. See details on how to configure AMC for SSL in previous sections - the same is applicable for Action Manager.

AMC and remote IBM Security Directory Integrator server

AMC can connect to multiple IBM Security Directory Integrator Servers remotely. You can learn to configure each Server in many ways.

Configured ways:

- Non SSL
- SSL
- Custom Authentication with Non-SSL
- Custom Authentication with SSL

This section looks at each of these cases in detail.

When a remote IBM Security Directory Integrator server is configured for non SSL (that is, `api.remote.ssl.on=false`) then the keystore or truststores of AMC do not come into play, even if correctly configured - since no SSL connection is being attempted. In this case the AMC Server's computer IP address must be registered with the IBM Security Directory Integrator server. This is done by editing the `global.properties` (or `solution.properties`) file. The property to update is: **api.remote.nonssl.hosts**. Once the AMC computer's IP address is entered in the `global.properties` file of the remote IBM Security Directory Integrator server, AMC is able to connect to that particular server. It is a way of saying - I trust remote server connections (AMC connections) from only those computers whose IP addresses I have mentioned in my **api.remote.nonssl.hosts** property.

Note: If the IBM Security Directory Integrator server is running on the same computer as AMC, then editing this property is not required.

When a remote IBM Security Directory Integrator server is configured for SSL (that is, **api.remote.ssl.on=true**), then the SSL keystore and truststore for AMC must be setup appropriately.

For details on this, see the previous section on AMC and SSL. In addition to being configured for SSL or Non-SSL, a remote IBM Security Directory Integrator server may also require Custom Authentication - in which a username and password must be passed while making a connection to the remote IBM Security Directory Integrator server. The remote IBM Security Directory Integrator server validates this user name and password against some third party repository like LDAP, file, database, script, and so on and then make a decision on whether to allow the Server API client to make a connection or not. In such cases, while registering a server with AMC (**Servers -> Modify Server**) in the Authentication mode window - select **LDAP or Custom Authentication** and enter the Username and Password that AMC must pass every time it attempts to connect to the specified remote IBM Security Directory Integrator Server.

Note: If the Username or Password (in case of custom authentication) or SSL keystores or truststores (in case of SSL) are not set up correctly, then AMC is unable to connect to the remote IBM Security Directory Integrator Server and show that server as "Stopped" or "Not running."

AMC and role management

Every user (or group) in AMC can be assigned a role in AMC for a particular Solution View. You can learn about the available roles and their meaning through the information provided here.

This role assignment can be done in the **Solution Views** window by selecting a particular Solution View and by clicking **Configure Access Control Lists (ACLs)**. The **Configure ACLs** window displays. Select the Name of the user you want to configure and click **Configure Users** on the toolbar. The **Configure Users** window displays. Select the **User ID** and select one of the available roles:

- Read
- Execute
- Admin
- Config Admin

Note: You must reload Solution Views created using the Auto Update option. Use the **Refresh Solution View** in the **Solution Views** window. For Solution Views marked for auto update, you must reload the config file and refresh the Solution View by clicking the **Refresh Solution View**. If a user fails to refresh a Solution View created using the **Simple** option and flagged for auto update, the Solution View may cause inconsistencies in the AMC database. Inconsistencies in Solution Views that are not updated could result in incorrect behavior by the Action Manager.

These roles are in increasing order of privilege - indicating that Config Admin is the highest privilege and Read is the lowest. Any functionality that is available to a user with "Read" role for a Solution View, definitely is available to a user with "Execute" privilege on that Solution View. Any functionality that is available to a user with "Execute" privilege on a Solution View, is available to a user with "Admin" privilege, and so on.

The following is the meaning of these roles

Read This means that this user can only read the "details" of this Solution View - such as what are the ALs inside this view, what are properties inside this view, what is the status of these ALs, and so on. This user cannot modify, start, stop, or change any detail of this Solution View.

Execute

This is essentially a Read user with one extra privilege - the ability to Start and Stop AssemblyLines.

Admin

This user can administer the Solution View, without being able to modify the Solution View itself. This user can do everything that the "Execute" privilege user can do, and additionally he can modify properties, delete logs, configure rules, and so on, for this Solution View.

Config_Admin

This user can virtually do anything to the Solution View - including modifying the view itself, modifying the permissions of other users on this view, and so on. This is the highest privilege that can be given to a user for a particular Solution View.

The above roles can be assigned to any Group too. Therefore, if a user "test" and "tdi" are part of the "DBAdmin" group, and the "DBAdmin" group is given "ConfigAdmin" privilege over a Solution View "SynchDatabase", then both "test" and "tdi" automatically get ConfigAdmin privilege over the "SynchDatabase" Solution View.

Note:

1. If the "test" user is explicitly given "read" privilege for the same Solution View, then "Read" get precedence over the privilege he gets from being part of the "DBAdmin" group. This is done so that "specific" role assignment gets priority over role assignment from groups. This allows people to restrict or give higher access to individuals - without worrying about inherited access from being part of some groups.
2. If the "test" user is part of two Groups - where Group1 has "read" access and Group2 has "admin" access over the same Solution View - then in this case the test user get the higher of the two privilege - in this case being "admin", unless a specific role is already assigned to "test" for the same Solution View - in which case the specific role assigned to "test" is given precedence [point 1 above].

AMC and passwords

You can know about storing passwords through the information provided here.

Any password field that is stored in the `amc.properties` file, such as LDAP Bind password, keystore password, and so on, are all encrypted before being written to the `amc.properties` file. Also, AMC never displays any Password fields or protected fields on console. All such fields are masked out.

AMC and encrypted configs

You can learn to use password protected configs through the information provided here.

AMC allows users to load and connect to password protected configs. On the Load Reload window of AMC, a password text box has been provided - where the users must enter the password of the config they are attempting to start before clicking **Start**. Similarly, in the Action Manager Screen - for the Start AssemblyLine action, a password field has been provided where the user can enter the password of the config. Action Manager passes this password while attempting to start the Config.

Note: AMC cannot detect that the remote config being started is a password protected config. For this reason, if the password is not specified or incorrectly specified, then the user just see an error message saying - "Unable to start the config". The user can see the IBM Security Directory Integrator Server logs where an exact message is provided.

Administration and Monitoring Console User Interface

You can learn in detail about the AMC user interface through the information provided here.

Log in and logout of the console

You can use the instructions provided here to login and logout of the console.

Open a Web browser and type the following address:

```
http://hostname:port/ibm/console
```

Where *port* is the port where your Web server is running. When deployed on the bundled web container the ports by default are 13100 for HTTP and 13101 for HTTPS communication.

The login page can also be launched by using the `launchAMC.html` file which is placed in the `TDI_install_dir/bin/amc` folder.

The IBM Security Directory Integrator Administration and Monitoring Console login page window is displayed.

Logging on to the console as the console administrator

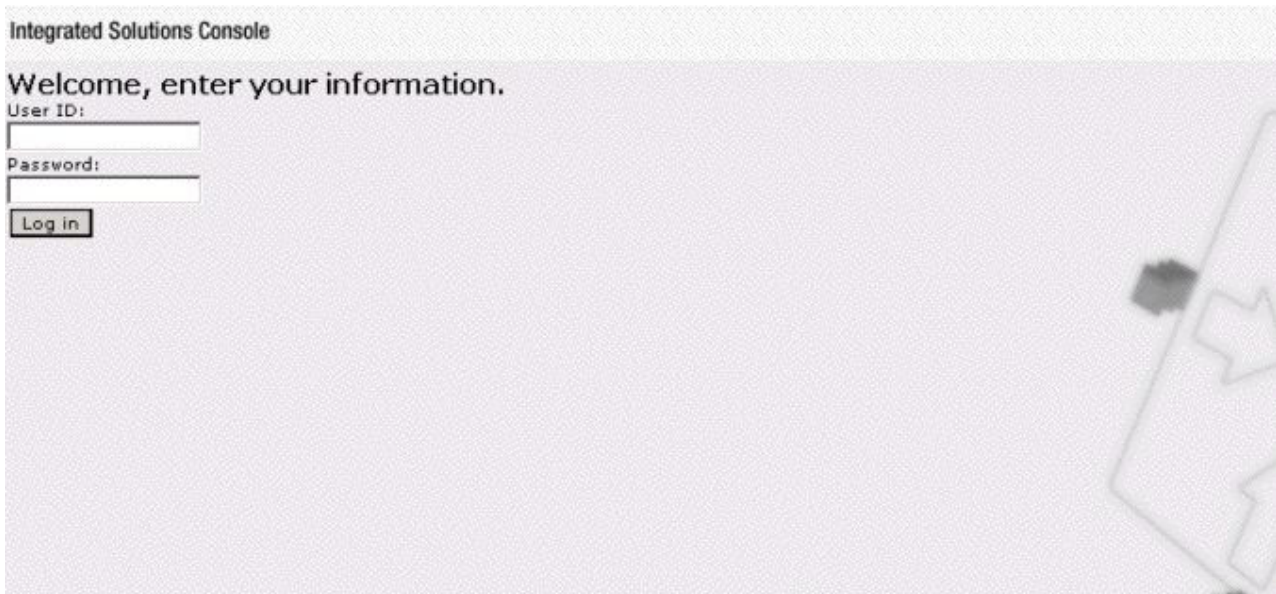
The console administrator is a user who can:

- Configure the properties required for the AMC
- Set the authentication mechanism used for AMC logins
- Add new users and configure users' roles

When logging in for the first time use the system username and password you have installed IBM Security Directory Integrator, AMC and the embedded web platform with. If you have deployed AMC in IBM WebSphere Application Server/IBM Dashboard Application Services Hub then you would need to log in with a user that has been assigned the *iscadmins* and *administrator* roles.

Note: The embedded web platform uses the PAM authentication mechanism on UNIX and Linux boxes to validate the system username and password provided on log in. This is why on AIX machines you must have the `auth_type` parameter set to `PAM_AUTH` in the `/etc/security/login.cfg` file.

To log in to the Integrated Solutions Console, type your user name and password in the boxes provided in the login window and click the **Log in** button.



The **Logout** button is in the upper right hand corner of the console, next to **Help**. When you click **Logout**, you are returned to the Log in page.

AMC Console Layout

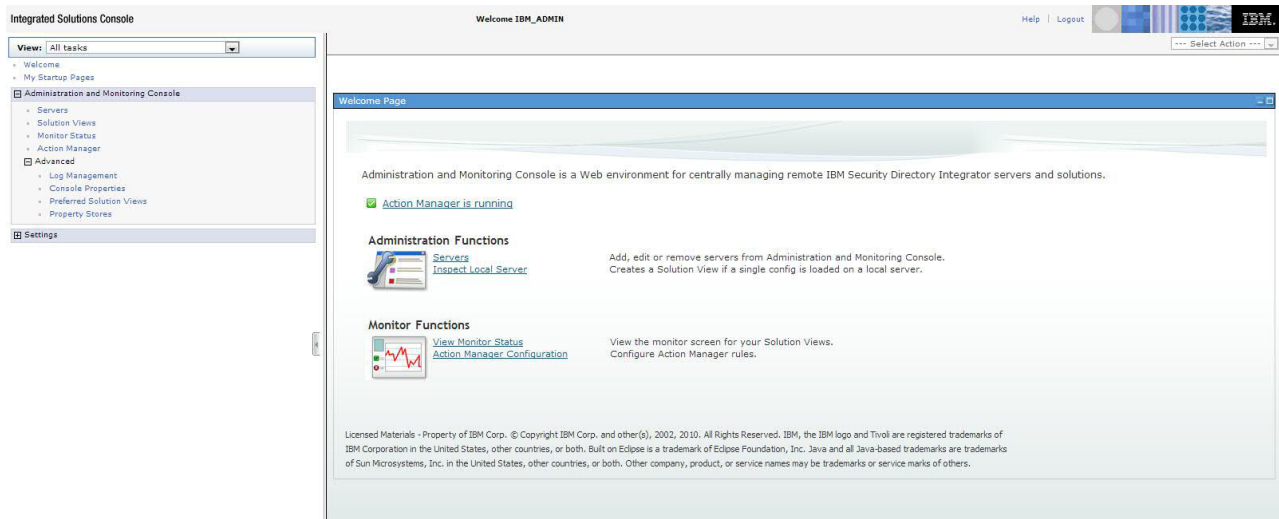
You can refer to the listed components of the IBM Security Directory Integrator Administration and Monitoring Console.

Navigation Area

The Navigation area provides a tree view that allows users to navigate through the tasks available to the user in the console. You can open and close folders in the navigation area and select tasks (non-folders) to launch in the Work Area of the console framework.

Work Area

The Work Area contains the necessary information and input fields to complete the task you are currently working on.



Logging off the console

To log off of the console, click **Logout** in the navigation area.

Using AMC tables

You can use the information in AMC tables to search for, organize and perform actions on these table items.

The IBM Security Directory Integrator Administration and Monitoring Console displays certain information, such as lists of attributes and entries, in tables.

IBM Security Directory Integrator Administration and Monitoring Console tables provide icons to help you organize and find information in the table. Some icons appear on some tables and not on others, depending on the current task. The following is a comprehensive list of the icons you might encounter:

- Click the **Show Filter Row** icon to display filter rows for every column in the table. See “Filtering” on page 259 for more information about filtering.
- Click the **Hide Filter Row** icon to hide filter rows for every column in the table. See “Filtering” on page 259 for more information.
- Click the **Clear all filters** icon clear all filters set for the table. See “Filtering” on page 259 for more information.
- Click the **Edit sort** icon to sort the information in the table. See “Sorting” on page 258 for more information.
- Click the **Clear all sorts** icon to clear all sorts set for the table. See “Sorting” on page 258 for more information.
- Click the **Collapse table** icon to hide the table data.

- Click the **Expand table** icon to display the table data.
- Click the **Select all** icon to select all items in the table.
- Click the **Deselect all** icon to deselect all selected items in the table.
- Click the **Export** icon to export the table data.

Select action drop-down menu

You can use the **Select action** drop-down menu to view a comprehensive list of all available actions for a selected table and to perform operations on the table contents.

About this task

For example, instead of using the icons to display and hide sorts and filters, you can use the **Select action** drop-down menu. You can also use the **Select action** drop-down menu to perform operations on the table contents; for example, on the **Manage attributes** window, actions such as **View**, **Add**, **Edit**, **Copy** and **Delete** appear not only as buttons on the toolbar, but also in the **Select action** drop-down menu. If the table supports it, you can also display or hide the **Show find** toolbar using the **Select action** drop-down menu. See Finding for more information on finding table items.

To perform an action using the Select action menu:

1. If necessary, select an item from the table.
2. Click the **Select action** drop-down menu.
3. Select the action you want to perform; for example **Shutdown server**.
4. Click **Go**.

Paging

You can use the navigation controls at the bottom of the table to view different table pages.

You can enter a specific page number into the navigation field and click **Go** to display a certain page. You can also use the **Next** and **Previous** arrows to move from page to page.

Sorting

You can change the way items in a table are sorted.

About this task

1. Do one of the following:
 - Click the **Edit sort** icon on the table.
 - Click the **Select action** drop-down menu, select **Edit sort** and click **Go**.

A sorting drop-down menu appears for every column in the table.

2. From the first sort drop-down menu, select the column on which you'd like to sort. Do the same for any of the other sortable columns on which you'd like to sort.
3. Select whether to sort in ascending or descending order by selecting **Ascending/ Descending** from the drop-down menu. Ascending is the default sort order. You can also sort using column headers. On every column is a small arrow. An arrow pointing up means that column is sorted in ascending order. An arrow pointing down means that column is sorted in descending order. To change the sort order, simply click on the column header.

4. When you are ready to sort, click **Sort**.

To clear all the sorts, click the **Clear all sorts** icon.

Finding

You can find a specific item or items in a table.

About this task

Note: The Show find toolbar option is available on some tables and not on others, depending on the current task.

1. Select **Show find toolbar** from the **Select action** drop-down menu and click **Go**.
2. Enter your search criteria in the **Search for** field.
3. If desired, select a condition upon which to search from the **Conditions** drop-down menu. The options for this menu are:
 - **Contains**
 - **Starts with**
 - **Ends with**
 - **Exact match**
4. Select the column upon which you want to base the search from the **Column** drop-down menu.
5. Select whether to display results in descending or ascending order from the **Direction** drop-down menu. Select **Down** to display results in descending order. Select **Up** to display results in ascending order.
6. Select **Match case** if you want search results to match the upper and lower case criteria in the **Search for** field.
7. When you have entered the desired criteria, click **Find** to search for the attributes.

Filtering

You can filter items in a table.

About this task

1. Do one of the following:
 - Click the **Show filter** row icon. Click the **Select action** drop-down menu, select **Show filter** row and click **Go**.
2. Filter buttons appear above each column. Click **Filter** above the column on which you want to filter.
3. Select one of the following conditions from the **Conditions** drop-down menu:
 - **Contains**
 - **Starts with**
 - **Ends with**
4. Enter the text you want to filter on in the field; for example, if you selected **Starts with**, you might enter **C**.
5. If you want to match case (upper case text or lower case text) select **Match case**.
6. When you are ready to filter the attributes, click **OK**.
7. Repeat the above steps 2-6 for every column on which you want to filter.

To clear all the filters, click the **Clear all filters** icon.

To hide the filter rows, click the **Show filter** icon again.

Servers

You can view the registered server through this window. Additionally, the console administrator can add, edit, delete and shut down IBM Security Directory Integrator servers from this window, as well as launch the Config Files window.

When AMC is started, it automatically has a local IBM Security Directory Integrator server, registered on port 1099. Therefore, in the **Servers** window, one entry is under LOCAL SERVER is already present with its state depicted as running or unavailable depending on its status.

To load or reload a config, select **Servers** and click **Config Files** in the toolbar of the Servers window. The **Config Files** window appears.

You can choose the operations you want to perform from the tool bar at the top of the table or using the Select action drop-down menu, such as:

Add Click Add on the toolbar.

Delete Select the radio button next to the server you want to delete and click **Delete** on the toolbar.

Modify Select the server for which you want to modify information and click **Modify** on the toolbar.

Config Files Select the server for which you want to list configuration files. When you click the **View Config Files** link in the Solution Views window, it launches the Config Files window. Each configuration file is labelled as loaded or not loaded. The toolbar provides a variety of load, unload, and reload options.

Shutdown server Select the server you want to shut down and click the Shutdown Server on the toolbar.

Shutdown gracefully Shuts down a running server gracefully (create new Threads that wait for the AssemblyLines to stop).

Note: Graceful shutdown is not supported for IBM Security Directory Integrator servers earlier than v7.1.

Add a server

You can add an IBM Security Directory Integrator server to the Administration and Monitoring Console (AMC).

About this task

Once you have added an IBM Security Directory Integrator server to the AMC, you can then use features on other AMC windows to add Solution Views to the IBM Security Directory Integrator server and to create and define views for the Solution Views associated with the IBM Security Directory Integrator server.

To add a new IBM Security Directory Integrator server:

1. Enter a name for the IBM Security Directory Integrator server in the **Name** field.
2. Enter the host name or IP address of the computer on which the IBM Security Directory Integrator is running in the **Hostname** field.
3. Enter the port number on which the IBM Security Directory Integrator server is configured to run.
4. Select the desired authentication mode. If you selected the LDAP or Custom authentication method, enter the username and password to be used for authentication.
5. Click **OK**.

Modify a server

You can edit the information for an existing IBM Security Directory Integrator server.

About this task

To edit an existing server:

1. Look at the displayed Server ID. If you want to change the Server ID, click **Change Server ID**.
2. Type the Name for the server.
3. Enter the host name or IP address of the computer on which the IBM Security Directory Integrator server is running in the **Hostname** field.
4. Enter the port number on which the IBM Security Directory Integrator server is configured to run.
5. Select the desired authentication mode. If you selected the LDAP or Custom authentication method, enter the username and password to be used for authentication.
6. Click **Cancel** to exit the window without making any changes, or click **OK** to save the changes.
7. Click **Test Connection** to see whether the connection to the server succeeds or not based on the current settings.

Console Properties

Use the Console Properties window of AMC to manage configuration information such as database configuration of AMC, SSL settings, Action Manager log rotation frequency, and so on.

General

Use the General window of AMC to set log rotation frequency of Action Manager in days.

SSL

Use the SSL window to configure SSL settings for AMC. The SSL settings apply to AMC's SSL connection to the remote IBM Security Directory Integrator server. The SSL properties that are exposed are only the AMC's keystore and the trust store properties. If SSL is turned on in the remote server, an administrator needs to make sure that the required certificate is imported in his store for the connection to work. An administrator should import each remote server's certificate that he wishes to connect to in his store.

JDBC Properties

JDBC properties are used to define the connections settings to the Derby database, or to other databases compatible with the Administration and Monitoring Console, such as Oracle and MS-SQL Server. The AMC database stores AMC configuration information, connection details, and Action Manager rules and results.

The IBM Security Directory Integrator AMC supports alternative databases in addition to Derby. AMC bundles the Derby database. AMC communicates with its database using the Java Database Connectivity (JDBC) protocol. JDBC is a generic protocol and can be easily extended to other databases. AMC support for alternate databases enables you to have AMC installed and communicating to an existing database. The database stores Action Manager logs, results, and so forth. The Integrated Solution Console **Advanced** -> **Console Properties** section groups the **JDBC properties** to Derby or to another database. In the case of Derby, you can configure the database to run in both embedded as well as network mode. The default database is Derby and the default mode is network mode.

From this window you can:

- Select a database from the **Database Type** field, options are Derby, MS SQL Server, Oracle and DB2.
- Type the a value for the JDBC URL in the **JDBC URL** field.
- Type the user name for the database in the **Username** field.
- Type the password for the database in the **Password** field.
- Type the JDBC driver name in the **JDBC Driver** field.

As for the JDBC URL and JDBC Driver parameters, the following table provides some guidance:

Table 31. Driver parameters

Database	JDBC URL	JDBC Driver	Driver .jar file
Derby	jdbc:derby://host:port/database [;create=true create=false]	org.apache.derby.jdbc.ClientDriver	derby.jar
MS SQL Server (2005)	jdbc:sqlserver://host:port; databasename=database	com.microsoft.sqlserver.jdbc.SQLServerDriver	sqljdbc.jar
Oracle	jdbc:oracle:thin:@host:port:database	oracle.jdbc.driver.OracleDriver	ojdbc14.jar
DB2	jdbc:db2://host:port/database	com.ibm.db2.jcc.DB2Driver	db2jcc.jar

Note:

1. Depending on the database selected the corresponding driver .jar file must be copied to *TDI_install_dir\lwi\libs*.
2. Configuration of the Action Manager is also needed in order to specify the new database from where it will work. The same .jar file must be added to *TDI_install_dir/bin/amc/ActionManager/jars* and adjustments must be made to the *am_config.properties* file.
3. If you decide not to use Derby, but one of the alternatives, keep in mind that the database specified in the JDBC URL must already exist before you start AMC (otherwise AMC won't be able to create one and populate it). This is not needed if Derby is used because it supports the "create=true" option in the JDBC URL, thus causing AMC to automatically create the database (if it does not exist) when started.

Solution Views

Use the Solution Views window to view, Add, Modify, and Delete Solution Views.

- To add a Solution View, click the **Add** button on the toolbar.
- To modify an existing Solution View, select the Solution View and click **Modify**. Follow the steps in the **Modify Wizard**. Under **Modify Solution View**, click **Next** to go to the next step, and click **Finish** when you have completed the steps.
- To configure Access Control Lists for a Solution View, select the Solution View for which you want to configure ACLs and select **Configure ACLs...** on the toolbar.
- To delete an existing Solution View, select the Solution View you want to delete and click the **Delete** button on the toolbar.
- To launch a separate panel to Add / Edit / Modify local AM variables for that Solution View, click **Local Variables...**

Note: You must reload Solution Views created using the Auto Update option.

When you **Modify a Solution View** AMC checks to see if the Solution View was created using **Auto Update**. If the Solution View selected for modification was created using Auto Update, a message appears, saying:

The selected Solution View is marked for auto update.
Ensure that auto update is disabled to modify the Solution View.

Solution Views are listed in the Solution Views table. If a specific Solution View was created using Auto Update, a >> short menu appears when you click on the arrows up and to the right of the Solution View name. You can select **Refresh Solution View** or **Disable Auto Update**. For Solution Views marked for auto update, you must reload the config file and refresh the Solution View by clicking the **Refresh Solution View**. If a user fails to refresh a Solution View created using the **Simple** option and flagged for auto update, the Solution View may cause inconsistencies in the AMC database. Inconsistencies in Solution Views that are not updated could result in incorrect behavior by the Action Manager.

Configure ACLs

You can set the Access Control Lists (ACLs) for a user and associate that user with a specific Solution View.

About this task

- To configure a user or users, select the user or users you want to configure and click **Configure Users** on the toolbar.
 1. Select the user you want to assign a role to from the **User ID** drop-down menu.
 2. Select the radio button next to the role or roles you want to assign the selected user:
 - **Read** - Allows the user to read Solution View details like ALs, Tombstones, logs, properties belonging to the Config, and so on.
 - **Execute** - Allows the user to read and start/stop AssemblyLines
 - **Admin** - Grants the user Reader and Execute roles. This role also allows users to delete logs and tombstones.
 - **Config Admin** - Grants the user the ability to start and stop a Config, modify the Solution View, and assign and modify ACLs for other users.
 3. Click **Apply**.

- To remove an existing user, select the user from the table and click **Remove**.

When you are finished making changes, click **Apply**.

Local variables

You can learn to work with local variables with the information provided here.

Select Solution Views from the AMC left hand navigation pane. The **Solution Views** window appears. Select **Local Variables** from the toolbar. In the **Local Variables** window, you can select and **Add**, **Modify**, or **Delete** local variables for a Solution View.

The Action Manager triggers and actions must provide support for local variables that you can set or increment using rules and actions. Local variables can be used as triggering conditions for other rules. For example, a local variable can be set to a value of 1 and then can be incremented for every occurrence of the event and the local variable (in this example, the number 1 set to increment for every occurrence of the event) - the local variable can trigger the rule "Terminate AssemblyLine". When the variable reaches a value of 10, you can configure a new rule to be triggered. The new rule could start a new AssemblyLine on a different server. Set these "local", AM-specific variables to a "Solution View". That means that the one variable created in a rule belonging to one Solution View can only be used in that Solution View's rules and is not accessible to rules of another Solution View.

Add a Solution View

You can give users access to information in the configuration file without granting them the ability to edit the configuration file directly.

About this task

Administrators can use a Solution View to filter a configuration file for specific information so that only certain information within the configuration file is displayed. You can create multiple Solution Views for each Config, with each view exposing different information contained in the configuration file.

To Add a Solution View, select **Solution View** and select **Add** on the toolbar of the **Solution Views** window.

1. Enter view details:
 - a. Enter a name for the Solution View in the **Solution View Name** field.
 - b. Enter a description of the Solution View in the **Description** field.
2. Select the **Server** and **Configs** (configuration file) you want to use to create a Solution View:
 - From the **Server** menu, select the IBM Security Directory Integrator server containing the configuration file you want to use to create a Solution View. This menu is empty if no IBM Security Directory Integrator servers have been added to the Administration and Monitoring Console.
 - Select the configuration file you want to use to create a Solution View from the **Configs** list. The menu contains all currently loaded Configs.

Note: Click the **View config files** button to go to the Config files window. You can perform load or unload operations for the configs in this window.

3. Click **Add** on the **Solution Views** toolbar.
 - a. Type the name of the solution view you want to create in the **Solution View Name** field.

- b. Type an optional **Description** for the Solution View you are creating.
- c. Select the **Server** that contains the configuration file and AssemblyLines you want to use for creating a Solution View.
- d. Select the configuration file you want to use from the **Configs** list.
- e. Enable or disable **Auto Update**.

When the AssemblyLines or properties for a configuration change, **Auto Update** automatically changes the Solution View.

Note: When Auto Update is selected, you cannot edit the Solution View you created with Auto Update on, nor can you create Rules and Triggers for Solution Views made while Auto Update is on. If you want to edit the Solution View or add Rules and Triggers, you must disable Auto Update. The users would have to disable the auto update functionality in order to be able to create a Rules and triggers for Solution Views marked for auto updation. Review any changes to the config in Solution View by using the Refresh button on the Solution View window. This button is only be visible to configs with auto-update set to true. Any config created manually using the Create Solution View wizard has the auto-update flag set to false.

Note: You must reload Solution Views created using the Auto Update option. Use the **Refresh Solution View** in the **Solution Views** window. For Solution Views marked for auto update, you must reload the config file and refresh the Solution View by clicking the **Refresh Solution View**. If a user fails to refresh a Solution View created using the **Simple** option and flagged for auto update, the Solution View may cause inconsistencies in the AMC database. Inconsistencies in Solution Views that are not updated could result in incorrect behavior by the Action Manager.

- 4. Use the following options in creating a solution view:

Simple

Create a Solution View with common default options.

Auto Update

For Solution Views marked for auto update, you must reload the config file and refresh the Solution View.

Create Solution View from published solution.

Creates the Solution View from the published solution as specified in the IBM Security Directory Integrator Configuration Editor (CE). This option requires that your active configuration instance have a published solution associated with it, and also requires an IBM Security Directory Integrator 7.0 server.

Create Solution View with all AssemblyLines exposed.

Creates a Solution View with all AssemblyLines from the config instance exposed, and no properties and no Health AL defined. Use this option for a quick start (useful for development purposes). Available for IBM Security Directory Integrator 6.0 and later servers.

Create Solution View with all AssemblyLines exposed and all properties exposed.

Creates a Solution View with all AssemblyLines from the config instance exposed, and all properties and no Health AL defined. This option does not expose the Java properties. Available for IBM Security Directory Integrator 6.1 and later servers. Use this option for a quick start (useful for development purposes).

Create Solution View with all AssemblyLines exposed and all User properties exposed.

Creates a Solution View with all AssemblyLines from the config instance exposed, and all properties and no Health AL defined. This is similar to a quick start type of option. This option is disabled for IBM Security Directory Integrator 6.0 servers (because IBM Security Directory Integrator properties are not available in IBM Security Directory Integrator 6.0 Servers)

Note: In order to be able to see user defined properties in the Property Stores panel you should do either of these:

- Place the .properties file in the folder containing the configuration file
- Specify an absolute path to the properties file when creating the property store in the CE (**New Property Store > Connector tab > Configuration tab > Collection path/URL** parameter)

5. Click **OK** to finish creating the Solution View.

Config files (allows loading/reloading of configurations)

You can know more about Config files and learn to work on those with the information provided here.

To reach the Config Files window, and to access options for loading, reloading, unloading, and refreshing of config files, select **Solution Views** in the left navigation area. Select a server and a config file, then click the **View Config Files** button. This launches the Config Files window. This window displays loaded Configs and the Configs in the configs folder of the remote IBM Security Directory Integrator server. When AMC is connected to an IBM Security Directory Integrator server, the Config Files window shows a listing of all files in the remote config folder (whether the files are valid IBM Security Directory Integrator config files or not). You should perform Load operations on valid IBM Security Directory Integrator Config files only, otherwise an error message displays in AMC. The status Loaded or Unloaded displays with green (Loaded) and red (Unloaded) icons in the Status column. You can select one or more configs from the **Select** column of the Config files table. Once you have selected a config, you can **Load**, **Load As...**, **Unload**, **Reload**, or **Refresh** using the buttons at the top of the table. If you want to load a password protected Config, select the Config and type the password in the Password field.

Whether an action is successful or unsuccessful, a message displays after the action (Load, Load with Run name, Reload, Unload, and Refresh) executes, describing the outcome. For Load, Reload, and Unload, the new status for the configs that you selected displays in the Status column.

Note: You must have superadmin or config admin privileges to perform these actions.

- To load Configs, select the configurations you want to load and click **Load**.
- To load multiple instances of one Config, select the config you want to load and click **Load As...** The **Custom Load** window opens, allowing you to specify **Config File**, **Config Run name**, **Config Password**, and **Property Store Value**.
- To unload Configs, select the configurations you want to unload and click **Unload**.

Note: Loading a server does not automatically start the AssemblyLines associated with the selected Config. Only those AssemblyLines designated as AutoStart starts upon loading.

- To reload Config, select the loaded configurations you want to reload and click **Reload**. You can only reload a configuration that has the status of Loaded.
- To refresh Configs, click **Refresh**. Information for all of the configs in the table is redisplayed.
- Click **Close** when you are finished making changes.

Note: Users must maintain data integrity.

- For example, if a Solution View and rules have been created for a config named config1.xml, and with a run name of ABC, do not load a different config, for example, config2.xml, with the name ABC either as a solution name or a run name.
- If you want to reuse Solution Views that you created using a specific run name and set of property files, you must unload this config using the same run name and property files.

Custom load:

You can use AMC to load multiple config instances by performing the listed steps.

About this task

The IBM Security Directory Integrator server supports loading multiple instances of the same config with different run names. If you load config instances using **Load As...**, you can use these configs to create Solution Views and Rules.

1. From the **Welcome** page, select **Servers -> Config Files**.
2. Click **Load As...**
The **Custom Load** window appears.
 - a. Select the **Config File** from which you want to create multiple instances and click **Go**.
 - b. Type the **Config Run name**.
 - c. Type the **Config Password**.
 - d. Type the **Property Store Value** for each Property Store Name.
3. Click **OK** to use the values you have entered to create an instance of the config with the Run name you have specified. After an instance of the config is created, you are returned to the **Load Reload** window.
4. Click **Cancel** if you do not want to create the config with the values you have specified in the **Custom Load** window.

Monitor Status and Action Manager

You can know what all actions you can perform with Monitor Status through the information provided here.

If you have not done so already, expand the **Monitor Status** category in the main navigation area of the Administration and Monitoring Console.

Do one of the following:

- To view information about each Solution View, see the Monitor Status table. Information regarding the Solution Views, such as Action Manager Status,

Health Check Result and Health Check Status, display. You can also display **Solution View Details**, **Server Information**, and **Show Preferred Views**.

- To add, edit or delete Action Manager rules, click "**Action Manager**" on page 272.

Monitor Status

You can get displayed high level information about each preferred Solution View.

This window displays the views selected on the Preferred Views window accessed from **Advanced** -> **Preferred Solution Views**. High level information , such as:

Action Manager Status

Displays the status of the Action Manager rules for the selected Solution View: A blue exclamation mark indicates that no Action Manager rules have been triggered recently. An yellow triangle containing an exclamation mark indicates that an Action Manager rule has been triggered recently.

Health Check Result

Displays the health check result obtained from the healthAL.result final work entry attribute in the Solution View's Health AssemblyLine. This value is displayed as text.

Health Check Status

Displays the health check status obtained from the healthAL.status attribute in the Solution View's Health AssemblyLine.

Additionally, if you have designated a .gif file with the same name as the returned status value in the Administration and Monitoring Console's resources/amc_images/healthAL directory, the .gif image is also displayed in this column. For example, if the healthAL.result is returned as "Error", and you have created an "Error.gif" in the above mentioned directory, the Error.gif image displays in the table column.

From this window you can:

- View Solution View details - To view the details of a specific Solution View, select the desired Solution View and click **Solution View Details**
- View IBM Security Directory Integrator Server Information - To view the details of the server to which the Solution View belongs, click **Server Information**.
- Show Preferred Solution Views - Click **Show Preferred Views** to view preferred Solution Views. This button is visible only if Preferred Solution Views are defined. You can define preferred Solution Views on the "Preferred Solution Views" window under **User Preferences**.

Solution View Details:

You can view the specific details of Solution view.

The Solution View details panel in turn provides deeper view of the details specific to a Solution View which an administrator can take a look at and take action upon.

This window contains two tables. The top table displays the AssemblyLines associated with the selected Solution View and the status of each Solution View. The bottom table displays log information about recently triggered Action Manager rules.

When you are through making changes, click **Close**.

Solution View Details Table:

The **Solution View Details** table contains the listed columns.

Columns

Select Select the radio button next to the AssemblyLine on which you want to perform an action.

AssemblyLines

Displays the name of the AssemblyLine.

Status Displays the AssemblyLine's status; for example, **Running** or **Stopped**.

Start Time

AssemblyLine is running

Start Time is when the running AL started. Start Time is based on the running AL.

AssemblyLine is stopped

The time when the last run of the AL started. Start Time is based on the most recent tombstone entry for the AL. (Available only with IBM Security Directory Integrator servers).

Last Stop Time

The time when the last run of the AL terminated. Stop Time is based on the most recent tombstone entry for the AL. (Available only with IBM Security Directory Integrator servers).

Statistics

Displays the current statistics of the running AssemblyLine.

Actions

You can choose the operations you want to perform from the tool bar at the top of the table or using the **Select action** drop-down menu, such as:

- View Tombstones - Select the AssemblyLine you want to view and click the **View Tombstones** button
- View Logs - Select the AssemblyLine you want to view and do one of the following:
 - Click the **View Logs** button on the toolbar.
 - Select **View Logs** from the **Select action** drop-down menu and click **Go**.
- Manage Properties - Select the radio button next to the AssemblyLine with properties you want to manage and click the **Manage Properties** button on the toolbar.
- Start AssemblyLine -
 1. Select the AssemblyLine you want to start
 2. Click the **View pop-up** button
 3. Click **Start AssemblyLine**.
- Stop AssemblyLine - Select the AssemblyLine you want to stop and do one of the following:
 1. Select the AssemblyLine you want to stop
 2. Click the **View pop-up** button
 3. Click **Stop AssemblyLine**.

Note: From IBM Security Directory Integrator v7.1 a new option is available - "Stop AssemblyLine gracefully". When selected the AssemblyLine will be stopped in a new Thread. Stopping AssemblyLine gracefully is not available for IBM Security Directory Integrator servers earlier than v7.1.

- Solution View Details - Click the **Solution View Details** button. Select the component you would like to view, for example, AssemblyLines.

Start AssemblyLine

Run the selected AssemblyLine.

Start AssemblyLine synchronously

AMC waits for the AL to terminate and shows the status of the run AL periodically. The output schema attributes of the AssemblyLine after its termination are viewable for synchronous AL runs.

Start AssemblyLine in simulate mode

The Assembly Line executes all components except for the connectors in the add, update, and delete modes. In essence, putEntry, modEntry and deleteEntry methods of connectors are not invoked in simulate mode. As a result, an Assembly Line running in simulate mode does not perform any additions, modifications, or deletions on third party repositories. For more information on simulate mode, see the corresponding section in *Configuring Directory Integrator*.

View Tombstones

If you have tombstones enabled on the remote IBM Security Directory Integrator server, the Administration and Monitoring Console can display the tombstone entries for terminated AssemblyLines. This window displays useful information about tombstone entries, such as when the entry was changed to the tombstone state.

Delete Tombstones

On the **Monitor Status** window, select an AssemblyLine. Select the arrow to the right of the AssemblyLine and select **Delete Tombstones** from the menu. This launches the **Delete Tombstones** window. The component details section of this window identifies the Solution View and AssemblyLine that are being worked on. In the choose delete criteria section, select one of the options to specify which tombstones you want to delete:

- Select **All Tombstones** to delete all of the tombstones for the selected AssemblyLine.
- Use **Start Date** and **End Date** to specify the date range from which the tombstones are to be deleted. AMC calculates the number of days from the selected date to the current date. AMC then deletes the tombstones generated for the calculated number of days.
- Use **Number of entries to return** to indicate a whole number indicating the number of recent tombstones to delete. When you click **Delete**, a confirmation message appears. When you confirm, AMC executes the delete command.

View Logs

Logs for a given AssemblyLine are displayed on the View Logs window. **Monitor Status** -> **Solution View Details** -> **View Logs** to view

the list of log files for the selected AssemblyLine, click the radio button next to the log you want to view and click **View Logs**.

Note: In order to view an AssemblyLine log in the Administration and Monitoring Console, the AssemblyLine must log using the SystemLog logger.

Action Manager results table:

You can know about the **Action Manager Results** table columns and how to perform operations on Action Manager Results.

When a rule set in the Action Manager is triggered, information about the violation is logged, such as the source of the violation, a description of the error and the time at which the violation occurred. These details are displayed in the **Action Manager Results** table.

Columns

The **Action Manager Results** table contains the following columns:

Select Select the radio button next to the message on which you want to perform an action.

Source
Displays the name of the Action Manager rule that was triggered.

Severity
Displays the severity of the message.

Message
Displays the message associated with the Action Manager action.

Description
Displays additional information about the message.

Timestamp
Displays the time at which the Action Manager rule was triggered and the message was generated.

Actions

Select the result or results you want to delete and click **Delete**.

View Components:

The View Components operation allows you to view the different connectors, function components and so forth configured in the selected AssemblyLine. N

Note: Branching components (IF, SWITCH, etc.) and Script components are not displayed. This is intentional design - attention is focused on Connectors/Function Components which are the key items.

Show Preferred Solution Views:

Preferred Solution Views are the default Solution Views that are displayed on **Monitor Status** window.

Refreshing Solution View Details in AMC

You can use the instructions provided here to change the refresh interval.

The Solution View Details window is refreshed after a set interval of time to view the current AssemblyLine status. By default, the refresh rate is set to 600 seconds. The Integrated Solutions Console administrator has the privilege to change the refresh interval.

To change the refresh interval:

1. Go to the login page of AMC.
2. Type your user name and password, and click **Log in**. The Welcome page of Integrated Solutions Console appears.
3. In the left navigation tree, click **Settings** -> **Manage Global Refresh**.
4. In the Manage Global Refresh window, click the **Monitor Status** link.
5. Change the refresh configuration settings and click **OK**.

Action Manager

You can add, delete or modify rules, triggers and actions to be performed as a result of rules execution and triggering conditions.

Add/Edit configuration rules:

You can add or edit configuration rules through this window.

Using the settings on this window you can create an "Action Manager" on page 243 (or modify an existing one) for the current Solution View.

A rule consists of two parts:

- The condition under which the rule is to be invoked, called a "trigger."
Some examples of triggers are Server API failure, AssemblyLine failure, or failure of the AssemblyLine to run at the specified intervals.
- The set of alternate actions to be performed when the trigger is encountered.

Configuration rules settings:

This window is concerned with the first part of the rule: defining triggers. You can select a name, description, and trigger type.

Name

Enter a name for the rule. If you are adding a rule, this field is required.

Description

Enter an optional description of the rule.

Trigger type

The trigger type defines the conditions under which a rule is invoked. From the drop-down menu, select a trigger type:

No trigger

Rule has no triggering condition.

On AssemblyLine termination

Rule is triggered when the specified AssemblyLine is terminated.

On Config Load

Rule is triggered when the Action Manager receives a Config load event for this particular config.

On Config Unload

Rule is triggered when the Action Manager receives a Config Unload event for this particular config.

On Query AssemblyLine result

Rule is triggered when the last "work" entry of the specified AssemblyLine contains an attribute matching a given condition and value.

On server API failure

Rule is triggered when the Action Manager is unable to connect to the remote server using the Server API. This rule is triggered only once. The rule resets when it detects that it can reconnect to the server using the Server API.

On received Event

Rule is triggered when the Action Manager receives an event that meets the criteria specified in the Event type, Event Source and Event Data fields.

On Property Trigger

Rule is triggered when the specified property meets the determined Property name, Condition and Value specifications.

On Local Variable

Rule is triggered when the specified variables meet the specified condition. The Action Manager periodically checks for this property.

Note: This rule gets triggered only once, and gets reset back to ready state only when Action Manager detects that this variables does not meet the specified criteria any longer. The rechecking ensures that the rule is not repeatedly triggered for a single occurrence of the triggering condition.

Inspect AssemblyLine Exit Code

Rule is triggered when an AssemblyLine terminates abnormally. You can define an error object that Action Manager searches for in the AssemblyLine Exit Code.

Time since last execution

Rule is triggered when the specified AssemblyLine has not run for the determined period of time.

Timer Trigger

Rule is triggered continuously within the given interval.

Configure trigger:

Each trigger type has a different selection of settings. Check if you do not see some of the fields listed below on your window, it is because the trigger type you currently have selected does not support them.

Source

Enter the source you want to monitor.

Data Enter the data you want to monitor.

Property name

From the drop-down menu, select the property name you want to monitor.

Condition

Select the condition you want to use to compare the property and value.

Possible options are:

- equals

- not equals
- greater than
- less than

Value Enter the value you want to monitor.

Configured actions:

You can add, delete, and modify actions through this table. You can also move actions up and down in the table.

For every action in the configured actions table that you can select, there is a column where you can enable the special trigger **Execute on Error**. **Execute on error** performs the action you have selected when an error condition occurs.

- To select an action to manage, enable the radio button that precedes each action that is listed.
- To add an action, click **Add**.
- To delete an action, select the action you want to delete and click **Delete**
- To modify an action, select the action you want to modify and click **Modify** .
- To move an action up one position in the table, select the action you want to move and click **Move Up**.
- To move an action down one position in the table, select the action you want to move and click **Move Down**.

Selecting **Execute on Error** carries out actions only if an error has occurred during the execution of any of the previous actions. You can use such actions to take corrective measures for handling any error that might have occurred during the execution of any previous actions. Action Error variables: AMC and Action Manager allow you to make the action error available in the various actions. At any point of time, if an error occurs during the execution of any configured actions, this error becomes available to you in the form of special reserved variables. You can then use these reserved variables in other actions you have configured. When the following actions are executed, Action Manager replaces the string %Action_Error% by the actual error that occurred during the execution of the previous actions. If no error occurs, the variable %Action_Error% is not replaced and stays as it is.

- Send Email
- Execute command
- Send event action
- Write Log action

Add/Modify Action:

When a rule is triggered, the Action Manager executes the actions associated with the rule. You can specify or modify the actions you want Action Manager to take when the rule is triggered.

From the drop-down menu, select an action type, and configure it. Click **OK** when you are finished.

Start AssemblyLine

This action starts an AssemblyLine. If you select this action, you must specify the name of the AssemblyLine you want to start and its associated Config (and possibly the Config's password).

Server This is a drop-down list of configured Servers. LocalServer means the Server on the computer Action Manager is executing.

Select from remote config folder

Check box; if enabled, queries the remote Server for available Config files. The Config files displayed are those present in the folder whose path is specified for the `api.config.folder` property in the `global.properties` file.

Config name

Enter the Config to which the AssemblyLine in the AssemblyLine field belongs. If **Select from remote config folder** is checked, you are presented with a list of available Config files on the remote Server, if unchecked, you must fill in the name of a locally-available Config file.

This field is required.

Config password

If required, enter the Config password for the selected Config file. This field is applicable only if the config is password protected.

AssemblyLine

Enter the name of the AssemblyLine to start.

Configure AssemblyLine Operation

This hyperlink launches the 'Select Operation' dialog. If the AssemblyLine has been defined with one or more custom Operations, this dialog enables you to select such an Operation. Subsequently, you are prompted for the AssemblyLine's Initialization attributes and Operation attributes for this Operation. This label is shown only for IBM Security Directory Integrator 6.1.X and IBM Security Directory Integrator servers if configured and is not applicable for IBM Security Directory Integrator 6.0.

Stop AssemblyLine

This action stops an AssemblyLine. If you select this action, you must specify the name of AssemblyLine you want to stop and its associated Config.

Server This is a drop-down list of configured Servers. LocalServer means the Server on the computer Action Manager is executing.

Select from remote config folder

Check box; if enabled, queries the remote Server for available Config files.

Config name

Enter the Config to which the AssemblyLine in the AssemblyLine field belongs. If **Select from remote config folder** is checked, you are presented with a list of available Config files on the remote Server, if unchecked, you must fill in the name of a locally-available Config file.

This field is required.

AssemblyLine

Enter the name of the AssemblyLine to stop.

Enable/Disable Rule

Select the Enable/Disable Rule to enable or disable an Action Manager rule.

Rule name

Select the name of the rule-Solution View pair that you want the action "Enable/Disable Rule" to execute. In previous versions of IBM Security Directory Integrator, you selected the rule name instead of a rule-Solution View pair, which is a feature available in the current version. This option belongs to the action "Enable/Disable Rule."

State Select the desired state from the drop-down menu. If you want to enable the rule in the **Rule name** field, select **Enabled**. If you want to disable the rule, the select **Disable**.

Execute Rule

This action causes the Action Manager to execute the specified rule. Action Manager then executes the actions associated with the specified rule. The trigger condition associated with the specified rule is not required to be satisfied.

Rule name

Select the name of the rule-Solution View pair that you want the action "Execute Rule" to execute. In previous versions, you selected the rule name instead of a rule-Solution View pair, which is a feature available in the current version. This option belongs to the action "Execute Rule."

Execute Command

The Execute Command action can execute the command entered in the Command field on the target computer specified under **Target Computer Name**. The command can be any generic command or an IBM Security Directory Integrator specific command. The Execute Command can be used when a user configures a rule to execute commands that are specific to the target computer or to execute IBM Security Directory Integrator commands that are not exposed by AMC. For example, in AMC we do not have actions that can restart a server or load a config. The user has to perform the restart or reload commands using either the IBM Security Directory Integrator Server or Config Files windows. If any error occurs while executing the command, it is captured in the %ACTION_ERROR% variable, which can be further used by the Action Manager,

Target Computer Name

Name or IP address of the target computer. Action Manager connects to the computer specified in this field. If neither a computer hostname nor an IP address is specified, the command executes on the computer where the Action Manager is running.

Port Port specifies the channel over which the Action Manager can connect to the target computer where the command is to be executed.

Username

The user name is verified for authentication and authorization when establishing a connection with the target computer.

Password

The password is verified for authentication and authorization when establishing a connection with the target computer.

Keystore

Keystore path is entered and used in case certificate authentication is required when connecting to the target computer.

Keystore Password

Keystore password is required when certificate authentication is mandatory for connection to the target computer.

Protocol

The protocol that is to be used for establishing a connection with the remote machine. Protocol can have the following values, WINDOWS, RSH, SSH OR REXEC (Windows, remote shell, secure shell, or remote execution protocols).

Command

Command that is to be executed.

Notify Event

This action causes the Action Manager to send an event with the specified details to the IBM Security Directory Integrator server associated with the current Solution View. To add this action to the rule, select **Notify event**. If you select this action, you must specify an event type.

Event type

Enter an event type. This field is required.

Source

Enter a source for the event type.

Data Enter data for the event type.

Modify property

This action causes the Action Manager to modify a property based on a specific operation and value. If you select this action, you must also select a value.

Property name

Select the property you want to modify from the drop-down menu.

Operation

From the drop-down menu, select the operation you want to use to modify the property. Possible options are:

- Set
- Increment
- Decrement

Value Enter the desired value. This is a required field.

Copy property value

This action causes the Action Manager to copy the value of the source property to the destination property.

From property

From the drop-down menu, select the property you want to copy from.

To property

From the drop-down menu, select the property you want to copy to.

Write to log

This action creates a log of the Action Manager rules that have been invoked, according to the specified severity, message and description. This log can be viewed under **Monitor Status**, on the "Solution View Details"

window in the **AM results** table. Having at least one log action for every rule is recommended. If you select this action, you must enter a message in the **Message** field.

Severity

Select the desired severity from the drop-down menu. Possible options are:

- Severe
- Warning
- Info
- Fine

Message

Enter the desired message.

Description

Optionally, enter a description.

Send Email

This action causes an email to be sent to the recipient you specify. You supply the content of the email. Along with the content, the Action Manager provides other details before sending the mail. In the content input area as well as in the subject line, you can specify the variable `%EVENT_DATA%` value. Specifying `%EventData%` inserts the actual value of the Eventdata variable when the mail is sent. `%Action_Error%` can also similarly be substituted here. If Attach Action Manager Log is enabled, the Action Manager logs (as specified in the `am_logging.properties` file) are sent as an email attachment. In the content input area, you can specify the variable `%EVENT_DATA%` value. Specifying `%EventData%` in the content puts the actual value of the Eventdata variable when the mail is sent. `%Action_Error%` is also similarly be substituted here. If **Attach Action Manager Log** is enabled, the Action Manager logs (as specified in the `am_logging.properties` file) are sent as an email attachment.

Substitute variable for event data:

You can select data that is output from certain Action Manager triggers, and use that data in certain actions that are triggered by a rule.

Select **Action Manager** from the left navigation pane or select **Action Manager** from the **Welcome** screen. Under Action Manager, you can Add a rule to a Solution View. You can name a new rule, and edit or delete an existing rule. You can make Event Data available when configuring or sending that data to other actions.

Use Action Manager to make event data available when configuring actions for a trigger. In Action Manager, you can **Add**, **Modify**, or **Delete** a rule. When you add a rule, you name the rule and select the **Trigger type**. AMC and Action Manager make the data available to the triggered actions in the form of a reserved variable. The action then uses the data that is stored in the variable. You can use this reserved variable in any of the actions you have configured for this trigger.

The following trigger types can produce event data that can be consumed by actions:

- On Start AssemblyLine: Event data is available as `%Event_Data%`.
- On AssemblyLine Terminate: Event data from On AssemblyLine Terminate is available as `%Event_Data%`.

- On Received Event: Event data from the received event is mapped as %Event_Data%.
- On Local Variable: Event data from the Local variable event is mapped as %Event_Data%.
- On Config Load: Event data from the On Config Load event is available as %Event_Data%.
- On Config Unload: Event data from the trigger would be available as %Event_Data%.
- On query AssemblyLine result: Event data is available as %attribute_name %. The %attribute_name% variable is replaced with the details about the actual attribute from the last work entry.
- Inspect AssemblyLine Exit Code: Event data is available as %attribute_name % and %Event_Data%.
 - Inspect Error Object set to enabled: While configuring the Inspect AssemblyLine exit code trigger, if the user enables Inspect Error Object (sets the option to true), the %Event_Data% variable is replaced with actual error data. The %attribute_name% variable is not available for actions.
 - Inspect Error Object set to disabled: While configuring the Inspect AssemblyLine exit code trigger, if the user sets the Inspect Error object to disabled (sets the option to false), the %attribute_name% variable is replaced with the details about the actual attribute from the last work entry. The %Event_Data% variable is not be available for actions.

Triggers that can produce event data:

You can use the listed trigger types to produce event data that can be consumed by actions.

- On Start AssemblyLine: Event data is available as %Event_Data%.
- On AssemblyLine Terminate: Event data from On AssemblyLine Terminate is available as %Event_Data%.
- On Received Event: Event data from the received event is mapped as %Event_Data%.
- On Local Variable: Event data from the Local variable event is mapped as %Event_Data%.
- On Config Load: Event data from the On Config Load event is available as %Event_Data%.
- On Config Unload: Event data from the trigger would be available as %Event_Data%.
- On query AssemblyLine result: Event data is available as %attribute_name %. The %attribute_name% variable is replaced with the details about the actual attribute from the last work entry.
- Inspect AssemblyLine Exit Code: Event data is available as %attribute_name % and %Event_Data%.
 - Inspect Error Object set to enabled: While configuring the Inspect AssemblyLine exit code trigger, if the user enables Inspect Error Object (sets the option to true), the %Event_Data% variable is replaced with actual error data. The %attribute_name% variable is not available for actions.
 - Inspect Error Object set to disabled: While configuring the Inspect AssemblyLine exit code trigger, if the user sets the Inspect Error object to disabled (sets the option to false), the %attribute_name% variable is replaced with the details about the actual attribute from the last work entry. The %Event_Data% variable is not be available for actions.

Actions that can access event data:

You can know more about Actions that can access event data through the information provided here.

The actions executed for each of the above triggers can access the event data produced by the triggers using the %Event_Data% variable. Every occurrence of %Event_Data% is replaced with the actual event data for that trigger. The following action types can use event data available from their respective triggers:

- **Notify Event:** Users can specify the %Event_Data% variable in the Data text field only.
- **Write to log:** Users can see a log message that is logged to a database. If the log message, after substitution for the %Event_Data% variable, exceeds 500 characters, the log message is truncated to the first 500 characters. This is because the database has a limit of 500 characters only.
- **Send E-mail:** Any event data specified by %Event_Data% or error data specified by %Action_Error% is substituted in the subject line of the email. Action Manager appends other data about execution before sending the mail. You can specify the variable %EVENT_DATA% value in the content textbox. Specifying %EventData% in the content substitutes the actual value of the Eventdata variable when the mail is sent. You can also similarly substitute %Action_Error% here. If **Attach Action Manager Log** is enabled, the Action Manager logs (as specified in the am_logging.properties file) are sent as an email attachment.

For any action being executed, such as the Send E-mail action, the Execute command action, the Log action, the Start AssemblyLine action, and so on, executes in response to the same trigger, the string of %Event_Data% automatically gets replaced by the event data generated by that trigger.

View Rules Summary:

You can view the current Action Manager for the selected AssemblyLine.

Click **View Rules Summary**. The table lists all the defined rules, triggers and actions associated with the Solution View. When you are done viewing, click **Close**. Only those rules which are in Enabled state are listed here.

Property Stores

You can know more about Property Stores and how to use them using the information provided here.

If you have not done so already, expand the **Property Stores** category in the navigation area of the Administration and Monitoring Console. To add or edit Java, Solutions, Global, System, User Property and Password Store properties, click **Advanced > Property Stores**.

When you are done entering the desired the property values, click **OK** to save your changes.

The order in which these Property Stores are listed is significant. The Property Stores are evaluated from top to bottom, but the last definition of a given Property is the one that is used. By default, the system is set up such that properties defined in a solution-specific properties file called solution.properties (residing in the Solution Directory) override corresponding ones in the system-wide global.properties file.

Note: Certain System Properties and Java Properties are read-only. These read-only properties are shown in the respective Property Stores. Trying to modify these properties has no effect.

Select Solution View

This window allows you select a Solution View. The menu contains only those Solution Views for which you have any type of access rights amongst *Read*, *Execute*, *Config_Admin* or *Admin*. Nothing is displayed if you do not have access to any of the Solution Views being created. Once you have selected a view, click **Set**.

After you select a Solution View, you can manage properties by clicking on the other property tabs, such as **Solution Properties** and **Global Properties**.

Solution Properties

This window allows you to add, edit and delete properties in the Solution Properties list.

Global Properties

This window allows you to add, edit and delete Global Properties.

Java Properties

This window allows you to add, edit and delete Java properties.

System Properties

This window allows you to add, edit and delete System properties.

Password Store

This window allows you to add, edit and delete properties in the Password Store.

User Property Store

This window allows you to add, edit and delete properties in the User Property Stores list.

The Property Stores drop-down menu contains a list of property stores configured by the user. Global, Solution, Java and Password Stores properties are not included. Select the property store whose associated properties you wish to view, add, edit or delete.

Log Management

You can know more about Log Management and how to use them using the information provided here.

If you have not done so already, expand the **Advanced** category in the navigation area of the Administration and Monitoring Console. To delete log files for all AssemblyLines, for a particular AssemblyLine, or to delete by date, click **Log Management**. When you select a new Solution View, you can click **Refresh**. Clicking **Refresh** lists all of the AssemblyLines that belong to the Solution View you just selected.

This window allows you to select the name of a Solution View. The AssemblyLines listed for deletion are taken from the Solution View you choose. You can choose to delete log files for all AssemblyLines, or for a particular AssemblyLine. You can also specify logs to delete by date. To manage display and deletion of logs:

1. Select the Solution View with the AssemblyLines whose logs you want to clean up from the **Solution View** menu.
2. In the **Choose Component** section, do one of the following:
 - Select the **All AssemblyLines** radio button to delete the logs of all AssemblyLines within the selected Solution View.
 - Select the **Specific assembly line** radio button to delete only those logs associated with a specific AssemblyLine.
3. If you selected **Specific assembly line**, select the AssemblyLine with logs you want to delete from the menu.
4. In the **Display Log Files** section, do one of the following:
 - To delete all logs belonging to the selected AssemblyLine(s), select **All**.
 - To delete logs within a date range, use the **Start Date** and **End Date** options. Logs created within the two dates specified will be deleted. Enter the desired dates in the date field; its format is locale-dependent. You can also use the Calendar button, which lets you specify a date by choosing from a calendar.
 - For previous versions of IBM Security Directory Integrator servers, to delete logs older than a certain date, select the **End Date** option. All logs older than the date specified are deleted.
 - To preserve your most recent logs, select the **Display first** radio button. Enter the number of recent logs you want to save. Used to specify that keep those log files that are recent and list the others. You draw the line on *recent* using the edit box. If you type 20, you are telling AMC to keep the most recent 20 log files and list the rest of the files in the table so that they are available for deletion. If you type the number 10, the 10 most recent logs are saved.
5. In the **Logfiles** table, a list of log files displays. In the Select column, select any logs you want to delete and click **Delete**. In the **Select Action** menu, you can choose any of the following options:
 - Export data
 - Change all selected
 - Collapse table
 - RestoreWhen you have selected one of these options, click **Go**.
6. From the display that results from the criteria you have selected, choose the logs you want to delete and click **Delete** to remove the specified logs. When you are finished deleting logs, click **Close** to exit this window.

Preferred Solution Views

You can select the Solution Views that you want to be loaded by default in the monitor window, using the **Preferred Solution Views** panel.

The "preferred" Configs are shown by default in the monitor status page when it is opened. If there are no views defined then this panel will simply display a message saying that there are no views. Once a set of views are defined then a user can set what he would like to see as the default views. This panel can be viewed by any user who has a set of views assigned to him by the superadmin.

You can make a Solution View preferred by selecting its checkbox in the **Select** column, and click **Enable As Preferred**.

Conversely, you can disable the Preferred status for a Solution View by selecting its checkbox in the **Select** column, and click **Disable As Preferred**.

AMC and AM Command line utilities

You can know more about AMC and Action Manager command line utilities using the information provided here.

A number of command line utilities are included with AMC and its associate product, the Action Manager (AM). These command line utilities help in installing, uninstalling or re-installing the AMC war file. There are also scripts for backup and restore, as well as a migration script. The migration script is for migrating to future versions of AMC and AM, and not for migrating from previous version to the current version. All these scripts get installed in the *TDI_install_dir/bin/amc* directory.

install The `install.bat` (.sh) script is used to deploy the AMC console module on ISC SE or IBM Dashboard Application Services Hub. The script relies on the `setupCmdLine` script for setting up the necessary environment variables and the `tdiISCHome` script for determining the location of the ISC runtime and also the type of runtime being used, that is, whether it is the embedded Web platform or IBM WebSphere Application Server. This script is called by the installer.

Usage: `install`

This script does not take any parameters.

uninstall

The `uninstall.bat` (.sh) script uninstalls the AMC console module from ISC SE or IBM Dashboard Application Services Hub. The script relies on the `setupCmdLine` script for setting up the necessary environment variables and the `tdiISCHome` script for determining the location of the ISC runtime and also the type of runtime being used, that is, whether it is the embedded Web platform or IBM WebSphere Application Server.

Usage: `uninstall`

This script does not take any parameters.

backupamc

The `backupamc.bat` (.sh) script backups all the configuration related information of AMC (configuration files, logs, and so forth.) A `backup_tdiadc` folder will be created inside the backup directory.

Usage: `backupamc [-d folder_to_create_backup_in]`

If the `-d` option is not specified, the files are copied to the *TDI_install_dir/bin/amc/ActionManager/backup_tdiadc* directory.

The following files are backed up:

1. `amc.properties`
2. `logging.properties`
3. `amcdbschema.xml`

4. amcdbhandler.properties

restoreamc

The restoreamc.bat (.sh)script will restore the backed up files to a fresh AMC deployment. The backed up files need to be first obtained by using the backupamc script for this to work.

Usage: restoreamc

This script does not take any parameters.

migrateamc

This provides a single backup, restore, uninstall and install command. This will backup the old AMC data, uninstall the old AMC plug-in archive, installs the new AMC and restores the old AMC configuration data.

This script requires the new AMC plug-in archive to be copied into the *TDI_install_dir/amc* directory.

Usage: migrateamc.bat [-d *backup_directory*]

start_tdiamc

This script is a convenient wrapper utility to start AMC. This script internally starts the ISC runtime. If the runtime is the embedded Web platform then it calls the lwi start command, else if the runtime is IBM WebSphere Application Server then it calls the startServer server1 command. Before starting the ISC runtime the script calls the startNetworkServer command, which is used for starting the Derby database in secured network mode. If the database type is anything other than Derby, then this script only starts the ISC Runtime.

On Windows platforms:

Usage: start_tdiamc [*Service name*]

If a service name is passed, the service will be started instead of calling lwiStart.

On Unix Platforms:

Usage: start_tdiamc

stop_tdiamc

This script is a convenient wrapper utility to stop AMC. This script internally stops the ISC runtime. If the runtime is the embedded Web platform then it calls the lwi stop command, else if the runtime is IBM WebSphere Application Server then it calls the stopServer server1 command. After executing the command the script makes a call to the stopNetworkServer script to stop the Derby database. If the database type is anything other than Derby this scripts only stops the ISC Runtime.

On Windows platforms:

Usage: stop_tdiamc [*Service name*]

If a service name is passed, the service will be stopped instead of calling lwiStop.

On Unix Platforms:

Usage: stop_tdiamc

startAM

The Action Manager is started using the startAM.bat(.sh) script located in the *TDI_install_dir/bin/amc* directory.

Note: The script has the Classpath defined for all the jars required by the Action Manager. There are two variables, CLASSPATH and DB_CLASSPATH. The DB_CLASSPATH has the path separated list of .jar files required for achieving JDBC Connectivity with the database. When AMC is configured to use Oracle, MS SQL Server or DB2 the corresponding JDBC .jar files of these databases should be added to the DB_CLASSPATH variable.

On Windows, the script accepts an optional service name parameter that can be used to start an already registered service:

```
startAM.bat [service name]
```

stopAM

The Action Manager is stopped using the stopAM.bat(.sh) located in the *TDI_install_dir/bin/amc* directory. This script uses the processID of the started AM to kill it. The processID is obtained by the startAM script and is stored in a file, which in turn is read by the stopAM script.

On Windows, the script accepts an optional service name parameter that can be used to stop an already registered service:

```
stopAM.bat [service name]
```

startNetworkServer

This script is used for starting the Derby database server in network mode, on port 1528. The port selected is different from the default port of Derby.

Usage: startNetworkServer

stopNetworkServer

This script is used for stopping the Derby database server in network mode.

Usage: stopNetworkServer

setDBType

This script is used for setting the type of database that you are using. The script sets the property namely DB_TYPE. If the DB_TYPE is set to Derby, then on executing the startNetworkServer script the Derby database will be started on the host and port that you specified in the startNetworkServer script file. The setDBType also sets the database user name and password. The database user name and password are required by startNetworkServer to enable the BUILTIN security mechanism and to add the user to the list of authorized users.

The setDBType script is called internally by the startNetworkServer and stopNetworkServer scripts for setting the DB_TYPE and the DB_USER and DB_PASSWORD properties.

backupamcdb

This script is used during the migration of an AMC database. The script backs up the AMC database and has the data exported in an IBM Security Directory Integrator defined XML format. This script is called by the installer when you choose the migration path.

Usage:

```
backupamcdb -d folder_which_contains_AMC_backup  
-p location_of_the_amc.properties_file
```

restoreamcdb

This script is used to restore the AMC database during migration. The scripts are called by the installer when you choose the migration path.

Usage:
restreamcdb -d *folder_which_contains_AMC_backup*
-p *location_of_the_amc.properties_file*

backupam

This script is used for backing up the Action Manager properties files. The script backs up the `am_config.properties` and `am_logging.properties` file.

Usage: backupam [-d backup_directory]

The archived info is created in the backup folder. If the `-d` option is not specified the files are copied to the `TDI_install_dir/bin/amc/ActionManager/backup_tdiamc` directory.

restream

This script is used for restoring the properties files of Action Manager which were backed up using the backupam script. The restore script restores the `am_config.properties` and `am_logging.properties` files.

Usage: restream [-d backup_directory]

If the `-d` option is not specified, the files are copied from the `TDI_install_dir/bin/amc/ActionManager/backup_tdiamc` directory.

setAMCRoles

This script is used for mapping the user who is installing IBM Security Directory Integrator AMC to the ISC admin and SDI AMC Admin roles. This script is introduced in IBM Security Directory Integrator 7.0.

Once these roles are granted to the install user, that user has the authorization to add new users and grant them with the necessary roles. The install user becomes the administrator for the AMC console module.

Usage: setAMCRole *username* [OS Group]

The OS Group is an optional parameter while deploying AMC on ISC SE.

tdimigam

This script is used for migration of the `am_config.properties` file.

The usage for this command is:

```
tdimigam -f propfile [-b backfile] [-n newfile] [-v] [-?]
```

where:

- f propfile - The name of the file to migrate
- b backfile - Backup the original file with the specified name
- n newfile - Name to give the file that is migrated
- v - Enable verbose mode
- ? - Prints the usage statement

Logging for this command is controlled by the `tdimigam-Log4J.properties` file.

tdimigamc

This script is used for migration of the `amc.properties` file. The options of this script are similar to those of `tdimigam` and `tdimigbl` that are used for migration of the `am_config.properties` and `global.properties` files respectively.

The usage for this command is:

```
tdimigamc -f propfile [-b backfile] [-n newfile] [-v] [-?]
```

where:

```
-f propfile - The name of the file to migrate
-b backfile - Backup the original file with the specified name
-n newfile - Name to give the file that is migrated
-v           - Enable verbose mode
-?           - Prints the usage statement
```

Logging for this command is controlled by the `tdimigamc-Log4J.properties` file.

addAMCService

This script is used for adding AMC as a service on a system.

Usage: `addAMCService Service_Name`

On Windows the script registers the Generic Windows Service executable (`TDI_install_dir/bin/amc/amcwin-service.exe`) from the IBM Platform Integration Toolkit. The Generic Windows Service uses the configuration file `TDI_install_dir/bin/amc/amcwin-service.ini`. That file specifies the name of the service and the start/stop commands. The file is automatically populated by the installer or the "addAMCService" script.

By default the file looks like this:

```
[Service]
ServiceName=$service_name$
WorkingDirectory="$install_dir$bin\amc"
StartCommand="$install_dir$bin\amc\amc-service.bat" start amc am"
StopCommand="$install_dir$bin\amc\amc-service.bat" stop amc am"
```

This means that by default the AMC service runs both the AMC and the Action Manager.

After you call "addAMCService", you can edit the `.ini` file to customize which components are run by the service (both AMC and AM, just AMC or just AM).

For example to run only the AMC, specify start and stop commands like the following:

```
StartCommand="$install_dir$bin\amc\amc-service.bat" start amc"
StopCommand="$install_dir$bin\amc\amc-service.bat" stop amc"
```

To start/stop the service use the GUI "Services" utility (**Control Panel -> Administrative Tools -> Services**) or the Service Controller command line tool:

```
sc start <service name>
sc stop <service name>
```

Beware that in the "Services" utility the display name of the registered service looks like this:

```
IBM Tivoli Directory Integrator Administration and Monitoring Console - myamc
```

where "myamc" is the service name that you specified as an argument to "addAMCService.bat".

On UNIX the script appends a line like the following at the end of the `/etc/inittab` system file:

```
<service name>::once:<install dir>/bin/amc/amc-service.sh" start amc am
```

To stop the service use the "amcservice.sh" script from the *TDI_install_dir/bin/amc* folder:

```
amcservice.sh stop amc am
```

or just

```
amcservice.sh stop amc
```

if the service runs only AMC.

deleteAMCService

The script is used for removing AMC as a service from a system.

Usage: `deleteAMCService Service_Name`

setDerbyProps

The script sets the necessary Derby database properties that are used by the `startNetworkServer` and `stopNetworkServer` scripts.

Usage: `setDerbyProps`

amcservice

This script starts/stops the whole Administration and Monitoring configuration. The following configurations are supported:

- both AMC and AM
- only AMC
- only AM

Internally the script calls "start_tdiamc"/"stop_tdiamc" and "startAM"/"stopAM". It is intended to be used when registering an Operating System service.

Usage: `amcservice [start|stop] [amc] [am]`

Examples:

```
amcservice start amc
amcservice stop amc am
```

Example walkthrough of creating a Solution View and Rules

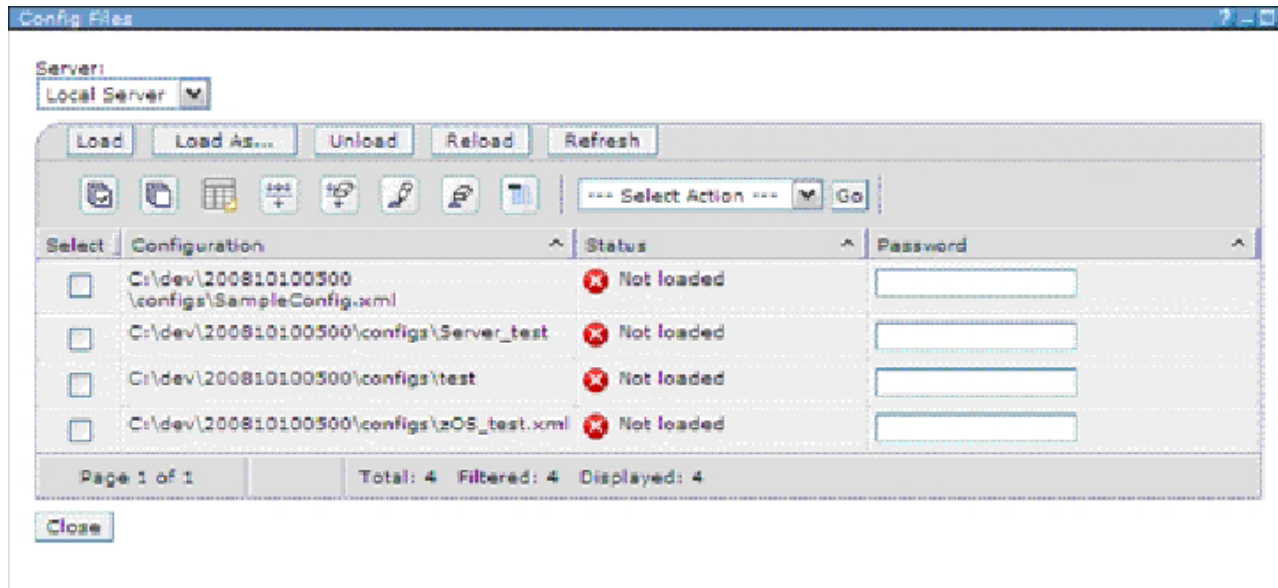
You can view the steps to create a Solution View, configure a rule and trigger it in Action Manager.

It is assumed that IBM Security Directory Integrator is installed along with AMC. The configuration `SampleConfig.xml` used in this example is the one that you are supposed to create following the tutorial in the *Getting Started*, "Introducing IBM Security Directory Integrator" > "Creating your first AssemblyLine". It should be copied into the *TDI_install_dir/configs* folder, where *TDI_install_dir* is the installation directory of IBM Security Directory Integrator. This solution reads data from the `examples/Tutorial/People.csv` file and writes it to `examples/Tutorial/Output.xml`.

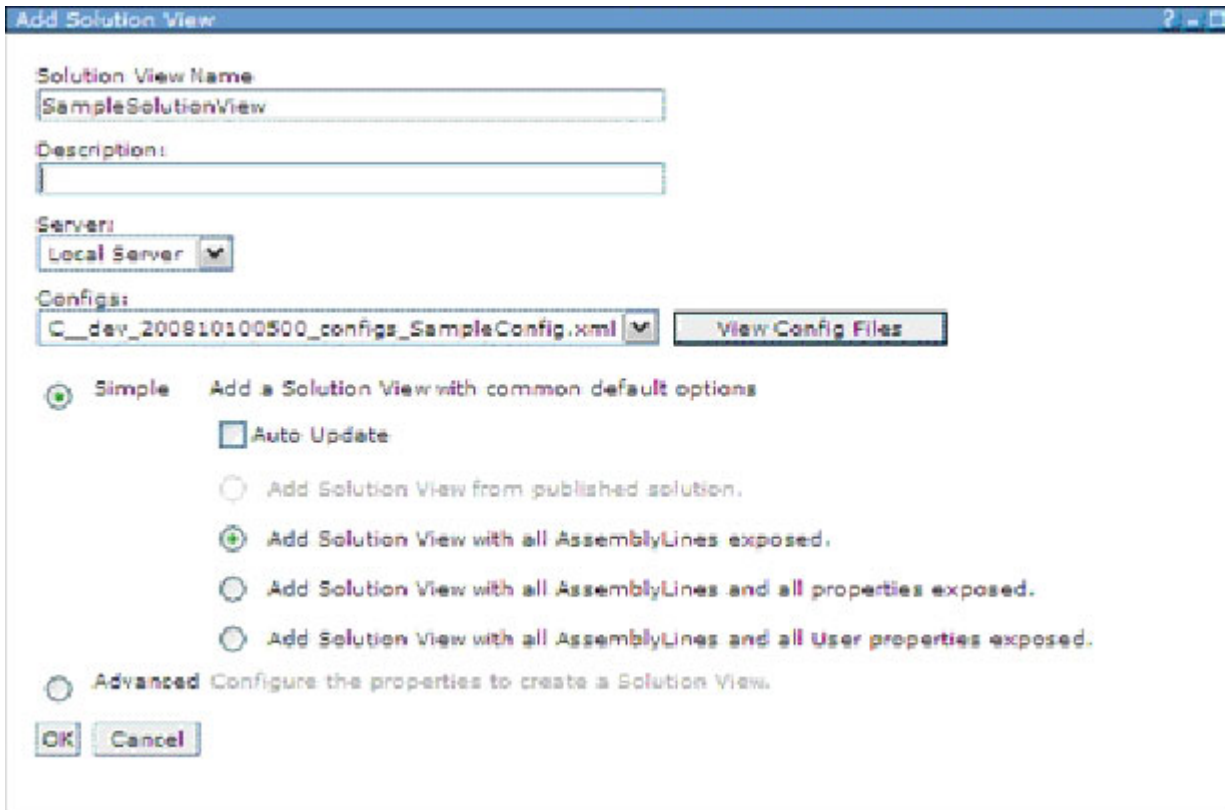
This section illustrates all the steps to create a config view, configure rules, and trigger this rule in Action Manager. It is assumed that IBM Security Directory Integrator is installed along with AMC. The sample config (`SampleConfig.xml`) and associated files are available in the downloads section and should be copied into the *TDI_install_dir/configs* folder. This solution reads data from the `sample.csv` file and writes to `sample.xml`.

Steps

1. Start the IBM Security Directory Integrator server in daemon mode.
2. Start AMC using the following command - `TDI_install_dir\bin\amc\start_tdiamc.bat`
3. Logon to the AMC console using the URL with the following syntax - `http://hostname:port/ibm/console` using the default username and password.
4. After logging on to the AMC console, select the Servers on the navigation panel and then choose the Server you wish to use. Press the **Config files** button and the following panel is shown.

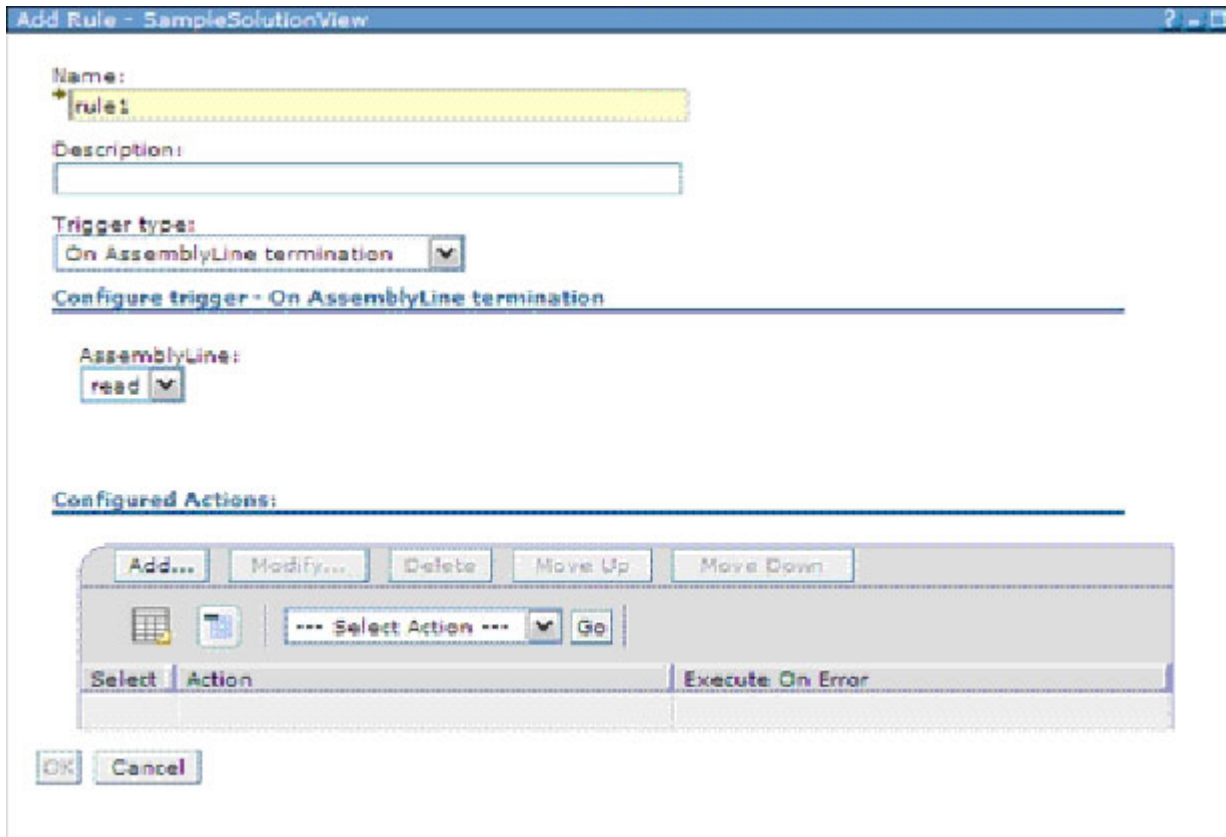


- Select the `SampleConfig.xml` file and click the **Load** button.
5. Select the Solution View link of the navigation panel to create a solution view for the loaded config. Select the Add button and the following panel will be shown.

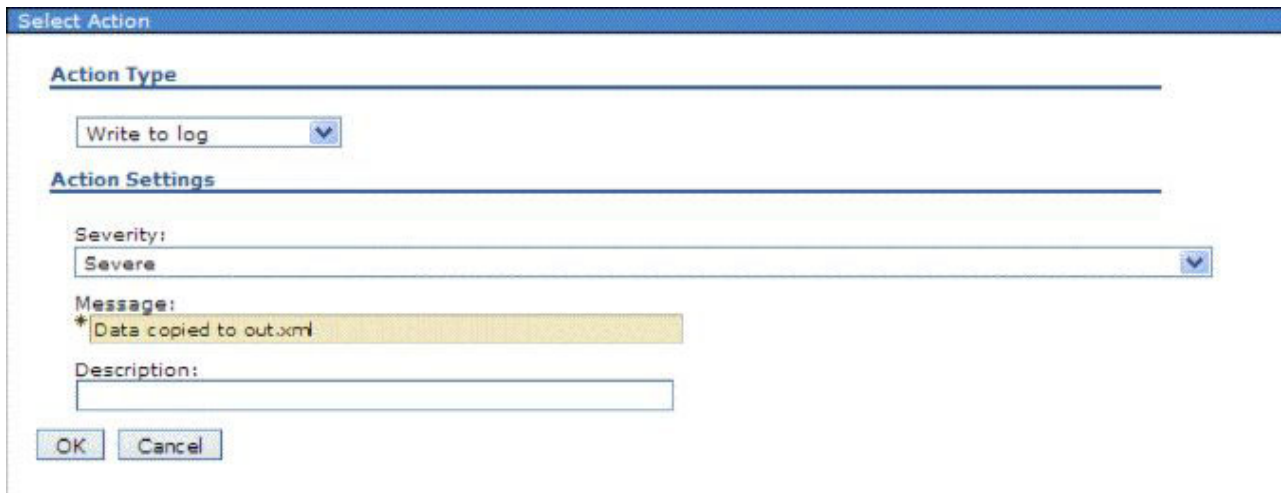


Add a suitable name for the config, for example, SampleSolutionView. On selecting OK, a message indicating that the Solution View SampleSolutionView was created successfully will be shown.

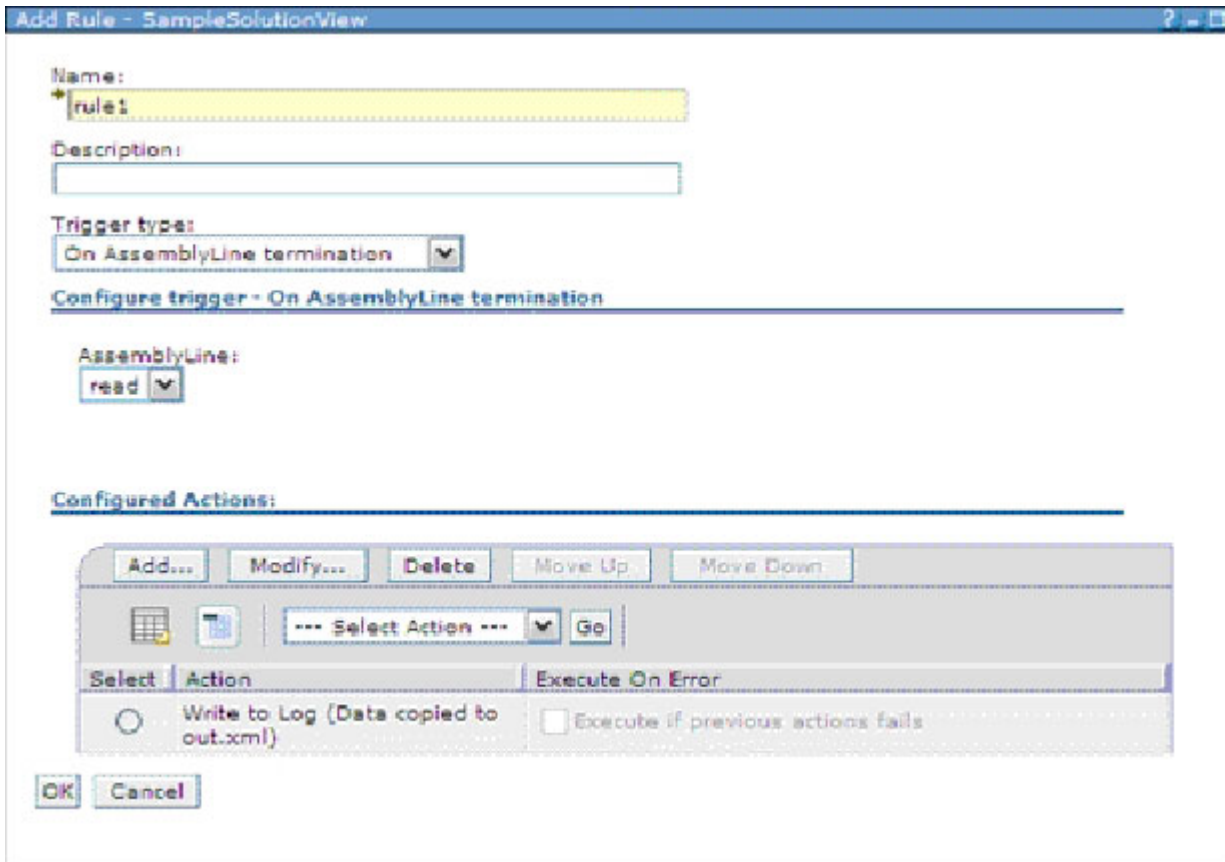
6. Select the Action Manager link on the navigation panel to reach the Action Manager rules configuration screen. Select "SampleConfigView" from the Select solution views dropdown box. When clicking the **Add** button in the Configured Rules section, the following panel will be shown.



Enter a name, for example "rule 1". Select trigger type "On AssemblyLine termination" and click the **Add** button in the Configured Actions section.

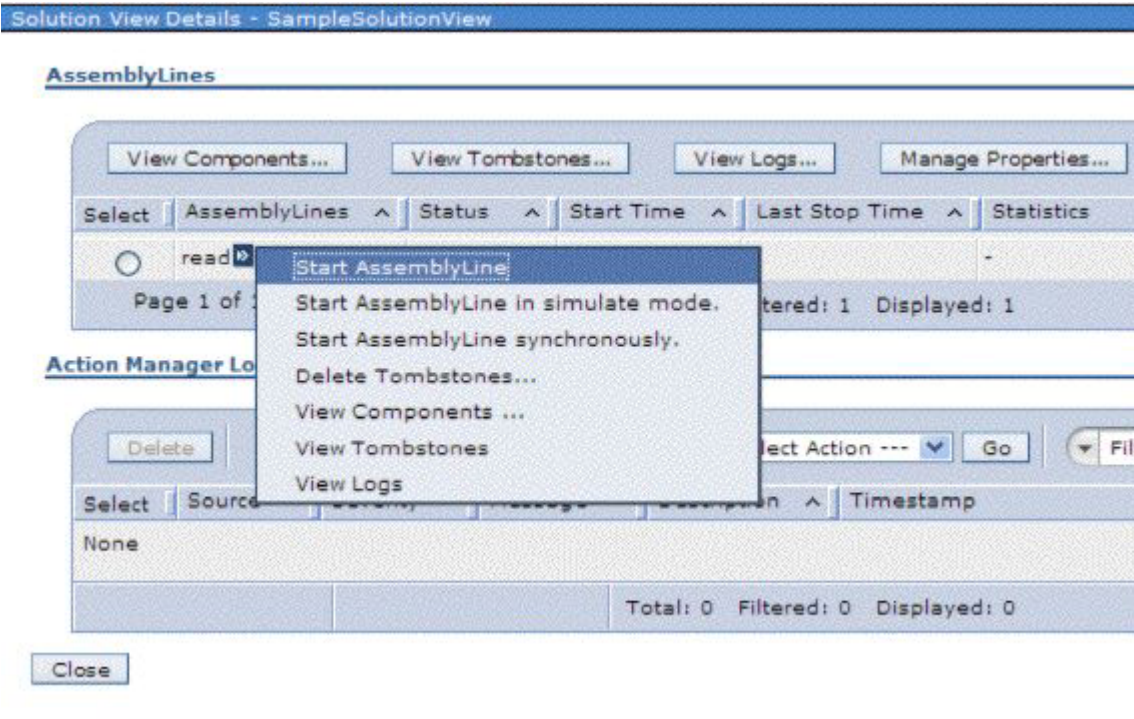


In the Select Action panel, select the "Write to log" option in the Action Type combo box. Add the text "data copied to out.xml" in the Message text box and click **OK**. The following rule panel will be shown.



Clicking **OK** completes the creation of this rule and the AMC configuration required for this example.

7. Start Action Manager using `TDI_install_dir\bin\amc\startAM.bat`. A thread is created for the rule "rule1", which should be waiting for the termination of the specified AL.
8. To trigger this rule, the "read" AssemblyLine of SampleConfig.xml needs to be executed. Select **Monitor Status** on the navigation panel. On the Monitor Status panel, select the SampleSolutionView and click on the **Solution View Details** button. The following panel will be shown.



- Start the AL using the option shown above. The rule will be triggered and the following status will be displayed on the Action Manager console.

```

C:\WINDOWS\system32\cmd.exe - startAM.bat
2008-10-16 17:05:01 con.ibm.di.amc.actionmanager.ServerConfigHandler startAMCServerModificationListe
nerThread
INFO: CTGDJB659I Started AMCServerModificationListener thread.
2008-10-16 17:05:01 con.ibm.di.amc.actionmanager.api.AMService initialize
INFO: AM.RMI.REGISTRY.SYSTEM.SECURITYMANAGER.NULL
2008-10-16 17:05:01 con.ibm.di.amc.actionmanager.api.AMService initialize
INFO: AM.RMI.REGISTRY.RESETTING.SYSTEM.SECURITYMANAGER.NULL
2008-10-16 17:05:01 con.ibm.di.amc.actionmanager.api.AMService initialize
INFO: CTGDJB720I The Action Manager RMI registry is started on port 13104.
2008-10-16 17:05:01 con.ibm.di.amc.actionmanager.AMHandler startHealthALManagerThread
INFO: CTGDJB512I Started Health AssemblyLine Manager.
2008-10-16 17:05:01 con.ibm.di.amc.actionmanager.AMHandler main
INFO: CTGDJB511I Action Manager initialization completed.
2008-10-16 17:05:01 con.ibm.di.amc.actionmanager.AMHandler startDatabaseModificationListener
INFO: CTGDJB532I Database modification listener started.
2008-10-16 17:06:23 con.ibm.di.amc.actionmanager.AssemblyLineTerminateListener triggerRule
INFO: CTGDJB549I Rule rule1 triggered.
2008-10-16 17:06:24 con.ibm.di.amc.actionmanager.RuleExecutionManager writeToLog
SEVERE: CTGDJB622I [Message]: Data copied to out.xml
[Description]: None
2008-10-16 17:06:24 con.ibm.di.amc.actionmanager.RuleExecutionManager execute
INFO: CTGDJB615I Action successfully executed.
[Solution View]: SampleSolutionView
[Rule]: rule1
[Action Order]: 2
[Action Type]: WRITE_LOG

```

On AMC, the Action Manager logs table on the Solution View Details panel will be shown as below.

Solution View Details - SampleSolutionView

AssemblyLines

View Components... View Tombstones... View Logs... Manage Properties...

--- Select Action --- Go

Select	AssemblyLines	Status	Start Time	Last Stop Ti...	Statistics
<input checked="" type="checkbox"/>	read	Stopped	-	-	-

Page 1 of 1 Total: 1 Filtered: 1 Displayed: 1

Action Manager Logs

Delete

--- Select Action --- Go

Select	Source	Severity	Message	Description	Timestamp
<input type="checkbox"/>	rule1	SEVERE	Data copied to out.xml	None	16.10.2008 17:06:24

Page 1 of 1 Total: 1 Filtered: 1 Displayed: 1

Close

Chapter 17. Touchpoint Server

You can use Touchpoint server to provide access to IBM Security Directory Integrator components. Refer to the information provided here to know more about its implementation.

The Touchpoint Server provides access to IBM Security Directory Integrator components (Connectors and AssemblyLines) through a ReSTful communication protocol. Clients send HTTP requests to the Touchpoint Server in order to request from an IBM Security Directory Integrator Server to create a Touchpoint Instance which the client can then "talk" to.

A Touchpoint Instance is implemented using standard IBM Security Directory Integrator Components and allows HTTP based clients to access third party systems that IBM Security Directory Integrator can "speak" to. In this context the Touchpoint Instance can be thought of as a "proxy" between a client application and a remote service, as it allows clients to use a unified protocol for communication to a variety of systems that don't have an HTTP based interface.

Touchpoint concepts

You can refer to information provided here to learn more about different touchpoint concepts.

In essence, the Touchpoint protocol is a provisioning protocol which gives access to IBM Security Directory Integrator Connectors and AssemblyLines through HTTP. When you create a Touchpoint, you get a "proxy" through which to work with a remote system. Once the Touchpoint is created and configured, you only send HTTP requests and you are completely isolated from the specifics of that system.

The sections below provide details about the different concepts associated with Touchpoints.

Touchpoint Server

You can use the Touchpoint server to perform various tasks like storing information, controlling instances. Refer the information provided here to know more usages of Touchpoint server.

The Touchpoint Server is the application that stores information about the defined Touchpoints in the particular domain. The Touchpoint Server is used to control the remote Touchpoint instances. It is responsible for their configuration, starting and stopping. The Touchpoint Server is provided as a service running inside the IBM Security Directory Integrator Server.

Clients of the Touchpoint Server are using the Atom Publishing Protocol to access details for the Touchpoint Providers, Touchpoint Types and Touchpoint Instances. For more details on the defined schema see section "Touchpoint Schema" on page 306.

Touchpoint Provider

You can use the Touchpoint provider to create the instances. Refer the information provided here to know more usages of Touchpoint provider and differences between a Touchpoint server and provider.

A Touchpoint Provider is a server on which the Touchpoint Server is creating the Touchpoint Instances. In our case a Touchpoint Provider can be any IBM Security Directory Integrator Server version 7.1 or later. The Touchpoint Server is only able to work with IBM Security Directory Integrator Servers, as any other Touchpoint Provider is not supported.

The Touchpoint Server is shipped as an add-on to the IBM Security Directory Integrator Server. It runs in the server's JVM which enables it to communicate with the server through the Local Server API. This is why the IBM Security Directory Integrator Server has been registered as a local Touchpoint Provider by default. In order to register a Touchpoint Provider, representing a remote IBM Security Directory Integrator Server, you must use the RMI settings of the remote server. The registration for both a Local and a Remote Server is performed in the standard Touchpoint Server configuration file.

Note: No user interface panels to configure the Touchpoint Server or Touchpoint Providers currently exist in the Configuration Editor or the Webadmin tool, AMC. All configuration must be done by means of XML configuration files.

For more details see section "Touchpoint Configuration" on page 310.

Once registered the Touchpoint Provider cannot be changed through the Atom interfaces. Additionally the Atom interface hides some of the details specifying the connection the remote IBM Security Directory Integrator Server. Those details are only used by the Touchpoint Server in order to communicate with the Touchpoint Provider and are not meant to be seen by the clients of the protocol.

Touchpoint Type

You can know about the categories of touchpoint types through the information provided here.

A Touchpoint Type is an abstract notation that provides meta-information for each Touchpoint Instance and determines its behavior. Every Touchpoint Instance has exactly one Touchpoint Type, while there is no limit to how many Touchpoint Instances can be created for a particular Type.

There are three categories of Touchpoint Types:

standard

This category corresponds to the IBM Security Directory Integrator Connectors supported by the chosen Touchpoint Provider and start with the prefix `system`. Every Touchpoint Provider has its own set of standard Touchpoint Types, as it provides different Connectors to you. By choosing any of these types, you specify that they will rely on the base Touchpoint Template for the structure of their Touchpoint Instance (for details about Templates see section "Touchpoint Template" on page 301). Furthermore, the chosen Type determines the inheritance of Template's Service Connector (the Connector working with a third-party system). For example, if you need to read data from a RDBMS, the JDBC Connector can be used. Hence, you will create a Touchpoint Instance with Type

system:/Connectors/ibmdi.JDBC and configure it appropriately. These Touchpoint Types support only the Provider and Initiator Touchpoint roles.

custom

This category represents custom Touchpoint Templates provided by you and are distinguished by the prefix `file`. Instead of relying on the base template that comes with IBM Security Directory Integrator, you can create your own ones, tweaking the Touchpoint behavior according to your needs. In exchange, though, they lose some of the flexibility offered by the base Template. Once a custom Template has been created, the type of its Connectors cannot be changed, limiting the created Touchpoint Instances to always work with the same type of remote system. An example use of the custom Touchpoint Type is if you need to work with several data sources, for example, read data from a RDBMS and add information from an LDAP server. This cannot be achieved by a single Touchpoint Instance relying on the base Touchpoint Template. To solve this, you can create a custom Touchpoint Template and use its corresponding Touchpoint Type (for example, `file:/template_file_name.xml`) for creating a Touchpoint Instance. The only limitation is that all subsequent Touchpoints created from this Type will also work with a RDBMS and an LDAP server (since the type of the Service Connector cannot be changed). These Types support all Touchpoint roles.

virtual

This category consists of only one Touchpoint Type with name `virtual://Intermediary`. Unlike the above Types, which are connected to some actual resource (an IBM Security Directory Integrator Connector for standard Types and a Template file for custom Types), this Touchpoint Type is used for providing a way to create an Intermediary Touchpoint Instance out-of-the-box. For this purpose it relies on the base Touchpoint Template provided with IBM Security Directory Integrator. This Touchpoint Type supports only the Intermediary Touchpoint role.

When creating a Touchpoint Instance, you need to provide the configuration of the Connector that will communicate to the third-party system. If a standard Touchpoint Type is used, this means providing the configuration of the corresponding IBM Security Directory Integrator Connector. For custom Types you must provide configuration for the Service Connector of the custom Template. Finally, for virtual Types no configuration is needed as Intermediary Touchpoint Instances rely only on HTTP Components for performing their task.

How do you find out what parameters are needed to create a Touchpoint Instance? For this purpose the Touchpoint Server supports Property Sheet Definitions - XML documents providing information for the schema of an IBM Security Directory Integrator Component. To obtain the URL of the Property Sheet Definition, send a HTTP GET request to the URL of the particular Touchpoint Type. It defines the required connector parameters, their default values, as well as other useful information needed when configuring a Touchpoint Instance.

In general, the information provided by Property Sheet Definitions is similar to the one available when configuring a Connector in IBM Security Directory Integrator's Config Editor (except for parameter descriptions). For more details on Property Sheet Definitions and their usage see section "Property sheet definitions" on page 314.

Touchpoint Instance

You can use a Touchpoint Instance to manage access over the HTTP protocol. Refer the information provided here to know more about this.

By nature, a Touchpoint Instance represents a proxy that enables access to a remote service over the common HTTP protocol. However, the communication flow varies significantly, depending on the role of the Touchpoint Instance. Here are more details for the supported Touchpoint roles:

Provider

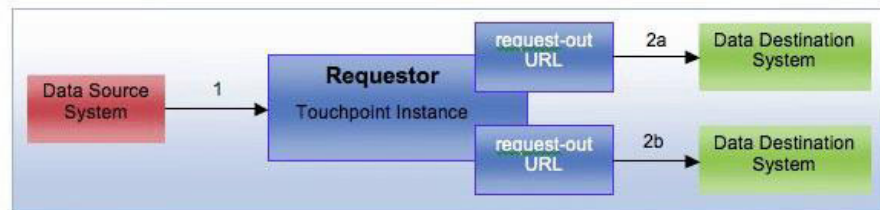
This mode provides access of clients to third party services. A client sends an HTTP request to the Touchpoint Instance (1), which in turn translates the request to the native language of the remote system and sends that to it (2). The remote system responds with the proper result to the Touchpoint Instance (3), which then transmits it back to the client (4) through HTTP.



As can be seen from this diagram, the Provider has a **single input interface**, represented by a URL on which you can send your requests (request-in URL). This URL is intrinsic, meaning that the Touchpoint Instance creates it and provides it to you.

Initiator

This mode provides transport of information between two ends. The Touchpoint Instance is the one that requests a piece of data from one system using the system dependent language (1) and then "pushes" this data through HTTP to several Data Destination System (2a, 2b) - HTTP servers capable of receiving data (for example, a Provider Touchpoint).



To achieve this, the Initiator is allowed to have **multiple output interfaces** - request-out URLs where the available data is sent. These destination points (also referred to as Destinations) can be added or removed from the Initiator at runtime, which gives a lot of flexibility for distributing data between systems. The Initiator starts working when its first request-out URL is added and stops when all of them are removed. By default the Initiator Touchpoint sends data to all of its Destinations, disregarding their responses. This behavior can be modified (for example, to stop the Initiator, if any of the Destinations fails to receive the data) by editing the base Template or providing a custom one.

Intermediary

This mode provides a forwarding service. The Intermediary Touchpoint Instance accepts requests (1) and then sends them to several Destinations

through HTTP (2a, 2b). The Destinations respond to the Touchpoint Instance (3a, 3b), it merges their responses and transmits the result back to the caller (4).

For a Client Application, the Intermediary looks like a Provider giving access to some third party system, while for the Data Destination Systems it is an Initiator sending data.



The Intermediary Touchpoint has a **single input interface** - request-in URL and **multiple output ones** - request-out URLs. You configure the output interfaces the same way as for the Initiator role, and the input interface is similar to the Provider's one (meaning its URL is set by the Touchpoint Server). In its simplest form, the Intermediary Touchpoint does not modify the forwarded data but you may provide such logic by editing the base Touchpoint Template or by providing a custom one.

Another specific characteristic of the Intermediary Touchpoint can be noticed when it acts as a proxy for accessing several Providers. As mentioned above, you can interact with this complex system as if it was a simple Provider. However, there is an explicit need to merge the responses returned from the end Providers. The default available logic is:

- if one of the Destinations returns a successful response, it is returned to the caller;
- if several Destinations return a successful response, the data in the response bodies is merged and returned with an HTTP code 200;
- if all Destinations return failure responses, an error response with code 500 is returned to the caller. In its HTTP body, it will contain the following information for each Destination:
 - Request to URL 'the URL of the Destination' failed.
 - HTTP status: the returned HTTP error code
 - HTTP body: the returned HTTP body

To change the merging behavior you need to edit the “Touchpoint Template” on page 301 used.

For more details on the communication protocol between the Client and the Touchpoint, see section “Touchpoint Instance communication protocol” on page 312.

Besides the above mentioned roles you must specify two more configuration items when setting up a Touchpoint Instance. These are:

- **Property Sheet** of the Touchpoint. Each Touchpoint Instance is representing a specific Touchpoint Type, which means that every Touchpoint Instance has a specific configuration. It complies with the one defined by the Touchpoint Type (the Property Sheet Definition) and reflects the schema of the IBM Security Directory Integrator Connector working with the remote system for this Touchpoint (known as the Service Connector).

Note: The Intermediary Touchpoint does not use a Service Connector for its working, so no such configuration information is needed for it. When setting up such Touchpoints, users should send an empty property sheet, as any provided parameters will just be ignored.

For more information about the Property Sheets format see section “Instance Configuration” on page 310.

- **Admin state** of the Touchpoint. This item is used for micro managing the state of the Instance. For example, you may want to disable a running Provider Touchpoint for a given period of time. Instead of deleting it, you can simply set its admin state to disabled. This way, when it is needed again, you only have to update its state to enabled, and it will be running again.

Another possible use case is for the Intermediary and Initiator roles. As soon as you add the first Destination to such Touchpoints, they start working (and presumably sending data). Additional Destinations can be added later on, but this may lead to data lost as some of the data will already be sent. To solve this, you can create such Touchpoints with disabled admin state (preventing them from starting) and add as many Destinations as needed. When the configuration is done, you can change the state to enabled, and the Touchpoint will start sending data to all the Destinations.

After creating the Touchpoint Instance you can access its current Operational state. There are two different states a Touchpoint can be in and their meaning varies with the role of the Instance.

A *Provider Touchpoint Instance* has the following states:

- **Unavailable** - when the Touchpoint Instance is intentionally disabled through its Admin state.
- **Available** - when the Touchpoint is configured and its admin state is enabled.

The status of a Provider Touchpoint has one additional parameter. Besides the operational state, you can get the request-in of the Touchpoint - the URL address where you send your request so that the Provider Touchpoint can communicate them to the remote system.

An *Initiator Touchpoint Instance* has the following states:

- **Unavailable** - the Touchpoint Instance is in this state if:
 - it is not fully configured, meaning that it has no Destinations (request-out URLs);
 - it is intentionally disabled, by setting its Admin state;
 - its Service Connector has finished reading from the data source.

This specific case is due to the behavior of the Service Connector of the Initiator Touchpoint Instance. If this Connector is a standard Iterator, it will read its configured data source and when done will stop, thus stopping the whole Touchpoint. However, in the case of Change Detection or JMS Connectors, it will continue to wait for new data and the Touchpoint Instance will keep running indefinitely (and staying Available). In this case it should be stopped explicitly by deleting it or setting its Admin state to disabled.

- **Available** - when in this state, the Touchpoint Instance is actually reading data from the data source.

As mentioned above, the *Intermediary Touchpoint Instance* in essence is a combination of a Provider and an Initiator Touchpoint Instance. This is reflected on its status as well.

- **Unavailable** - the Touchpoint Instance is in this state if:

- it is not fully configured, meaning that it has no Destinations (request-out URLs);
- it is intentionally disabled, by setting its Admin state;
- **Available** - when in this state, the Touchpoint admin state is enabled and it is actively forwarding its incoming requests to the Destinations.

Similarly to the Provider Touchpoint, the Intermediary has a request-in URL, which can be obtained from its Status Entry.

Touchpoint Template

You can use the Touchpoint Template to start the IBM Security Directory Integrator configuration.

When a Touchpoint Instance is started, the Touchpoint Server creates an IBM Security Directory Integrator configuration and starts it as a temporary IBM Security Directory Integrator ConfigInstance on its associated Touchpoint Provider (IBM Security Directory Integrator Server). The IBM Security Directory Integrator Configuration is based on a base Template provided with the Touchpoint Server. The path to this template file is specified in the Touchpoint Server “Configuration” on page 317, the default being *TDI_install_dir/etc/TouchpointTemplate.xml*. It is placed on the Touchpoint Server side and is a general template not associated with a particular IBM Security Directory Integrator Server. The configuration of each Touchpoint Instance is the one that is filled into the base template before it is started as a ConfigInstance.

The default base template contains the following structure:

AssemblyLines:

- **ProviderServer** - this is the AssemblyLine responsible for receiving HTTP requests and starting the appropriate handler ALs based on those requests. It acts as a common entry point for accessing all of the Provider and Intermediary Touchpoint Instances managed by the Touchpoint Server.

Using this technique, only one HTTP Server Connector is needed for communicating with all of these Touchpoints which mean only one TCP port should be opened (by default this is 1097). This avoids possible firewall problems that can occur if a large amount of random ports needs to be opened.

The ProviderServer AssemblyLine uses the following IBM Security Directory Integrator Components:

- *HttpServer* - the Connector receiving the requests from client applications. By default it listens on port 1097, though this can be changed from the “Configuration” on page 317 of the Touchpoint Server.
- *HandleRequest* - an IBM Security Directory Integrator Script Component that determines which AssemblyLine should be started depending on the request.
- **ProviderHandler** - this AssemblyLine is started by the ProviderServer when a request for the Provider Touchpoint Instance is received. It performs the actual communication with the remote system. The Components used are:
 - *ServiceConnector* - this is the Connector that communicates with the remote system. It is set to Passive state, which means that it is controlled from a script instead of the AssemblyLine flow. An

important characteristic of this Connector is that it inherits from the `GenericServiceConnector` in the library of the base template (Inherit from: `/Connectors/GenericServiceConnector`). The role of this parent Connector is described below.

Note: A Touchpoint AssemblyLine should have only one Service Connector (or none, as is the case with Intermediary). If several are provided, they all will have the same configuration.

- *HandleRequest* - this Script uses a servlet-like structure to handle the incoming requests. It initializes the ServiceConnector and depending on the received request performs a different operation over the data source. For more details on the supported Provider operations see section "Touchpoint Instance communication protocol" on page 312.
- **IntermediaryHandler** - when the ProviderServer AL receives a request for an Intermediary Touchpoint Instance it redirects it to this AssemblyLine. Since by default the Intermediary just forwards data to several destinations, it needs only one Component:
 - *SendToDestinations* - this Script Component forwards every request (passed to the AL in the form of a work Entry) to the configured Touchpoint Destinations using the call:

```
sendToDestinations(work, mergeResponsesCallback)
```

This method is provided by the Sender Script Component located in the library of the base Template. Besides the entry that will be sent, it requires a callback function that is used to merge the responses from each Destination. See section "Touchpoint Instance" on page 298 for details on the default merging behavior of the Intermediary Touchpoint.

- **Initiator** - This AssemblyLine is used to represent an Initiator Touchpoint Instance - the only Touchpoint role that is not accessed through the ProviderServer entry point. This AssemblyLine consists of the following Components:
 - *ServiceConnector* - this is the Connector used to feed the AL with data from the remote system. As with the ProviderHandler AssemblyLine, this Connector inherits from the `GenericServiceConnector` in the library of the base Template.

Note: Touchpoint AssemblyLines should have only one Service Connector (or none, as is the case with Intermediary). If several are provided, they all will have the same configuration.

- *ConvertToHTTPContent* - a Script Component that transforms the data read from the Service Connector to an HTTP entry.
- *SendToDestinations* - this Script Component forwards the received request to the Touchpoint Destinations using the call:

```
sendToDestinations(work, null)
```

The same call is used by the Intermediary Touchpoint Instance. The only difference is that here no callback function for merging the responses from the different Destinations is provided. See section "Touchpoint Instance" on page 298 for details on the default merging behavior of the Initiator.

Resources (library of the base Template):

- **Connectors** - several Connectors performing specific tasks for the various AssemblyLines:

- *GenericServiceConnector* - this Connector is the parent of all Service Connectors. When a Touchpoint Instance is being created, the Touchpoint Server configures this Connector to work with the specific remote system. Since the Service Connector in each AssemblyLine inherits from it, they all get the same configuration.

Note:

1. The name of the Service Connector in the AssemblyLine is not important provided that it inherits from the *GenericServiceConnector*.
2. Only one Service Connector per AssemblyLine should be provided.

The Touchpoint Server handles the *GenericServiceConnector* differently, depending on the Touchpoint Type:

- When using a **standard** Touchpoint Type the *GenericServiceConnector* is used for setting both the inheritance and parameters of the Service Connectors. For example, consider creating a Provider Touchpoint Instance working with a RDBMS. You create this Instance from Type `system:/Connectors/ibmdi.JDBC`, configure it in Provider mode and pass the parameters needed by the JDBC Connector. This means that the inheritance of the *GenericServiceConnector* will be overridden to `system://Connectors/ibmdi.JDBC` and the provided JDBC parameters will be set to it. This way, both ServiceConnector-s in the ProviderHandler and Initiator AssemblyLines will get the same configuration as well (since they inherit from the *GenericServiceConnector*). The ProviderHandler AssemblyLine will be started and you get a Provider Touchpoint Instance for working with a RDBMS.
- When using a **custom** Touchpoint Type the *GenericServiceConnector* is used for figuring out the type of the Service Connectors and setting their parameters. This time the inheritance of the *GenericServiceConnector* will not be changed. Instead, it will be used by the Touchpoint Server to determine the type of the Service Connectors so that you can know their schema. Next, when the Touchpoint is being configured, the parameters will again be set to the *GenericServerConnector*, thus propagating them to its children.
- When using a **virtual** Touchpoint Type the *GenericServiceConnector*, at least for now, is not used, because so far only the Intermediary Touchpoint Type is part of this scheme, and it does not connect to third-party systems.
- *HTTPClientConnector* - the HTTP Client used for sending data to Destinations by the Initiator and Intermediary Touchpoints. It is used by the Sender script and made available to the Touchpoints through the `sendToDestinations()` method.
- *MemoryPropertiesConnector* - a Script Connector used by the MemoryProperties Store for holding the request-out URLs of the Destinations. It offers the ability to communicate with a running Touchpoint Instance and add/remove Destinations to it (for more details see the description of the MemoryProperties Store). This Connector mimics the behavior of the Properties Connector with the major difference that does not store the provided data in a file.

- **Properties:**

- *MemoryProperties* - a Properties Store used for communicating to a running Touchpoint AssemblyLine. It relies on the MemoryProperties Connector for storing the passed data in memory. A separate ConfigInstance is started for each Touchpoint, so each one has its own MemoryProperties Store and there is no risk of interweaving communication messages.

The Communication procedure is as follows:

1. The Touchpoint Server uses the Remote Server API to store a specific property - `com.ibm.di.tp.destinations`, in the MemoryProperties store. Its value is a `java.util.List` of `java.util.Map`s holding the provided destination parameters (for example, request-out and request-error URLs) configured for the Touchpoint.
2. Each time a Destination is added or removed from an Initiator or Intermediary Touchpoint Instance the Server updates this property's value, providing the current list of URLs.
3. On each iteration the Initiator or Intermediary Touchpoint AssemblyLine gets the current value of this property and send its data to the stored URLs (this is done by the `sendToDestinations()` routine).

The base Touchpoint Template provides a properly configured MemoryProperties Store. Every time a Touchpoint is created the Touchpoint Server checks if the MemoryProperties is missing and if so, adds it to the configuration. Thus, if you do not configure this Store in your custom Templates the default one will still be available. On the other hand, if you modify the MemoryProperties to alter the communication behavior (for example, use a Properties Connector, so that the Destination URLs are persisted to a file), the Touchpoint Server will not overwrite their new configuration.

- **Scripts:**

- *Sender* - a script providing the `sendToDestinations()` method. This routine reads the content of the `com.ibm.di.tp.destinations` property in the MemoryProperties store to get the Destination request-out/request-error URLs and sends the provided work entry using the `HttpClientConnector`.

Note: The request-error URL is not used by default; it is only provided to the TouchpointTemplate for you, so you can enhance the default error recovery mechanism or implement a custom one.

- *Utils* - this script provides a set of utility functions used by the Touchpoint AssemblyLines.

Custom templates let you provide complex/customized behavior in an AssemblyLine and expose it as a new Touchpoint Type. The default base template is used for direct provisioning of IBM Security Directory Integrator Components and an Intermediary Touchpoint without requiring you to know anything about what is happening under the hood of IBM Security Directory Integrator. However, you may want to add more logic and/or Connectors to a Touchpoint Instance than a single Component. For this purpose the Touchpoint Server accepts custom templates to facilitate new custom Touchpoint Types and behavior.

A custom template's structure *should* be like the one of the base Template. However, if you do not need your custom Touchpoint Type to support all Touchpoint roles you may provide only a subset of the AssemblyLines. The minimum requirements for each role are:

- **Provider role.** To support it, the custom Template should contain the *ProviderHandler* AssemblyLine and the following Library Components: the *Utils* Script Component and the *GenericServiceConnector*.

Note: The base Template's ProviderServer AL is used to delegate request to all Touchpoint Instances. Thus, even if a ProviderServer AL exists in a custom template, it won't be used by the Touchpoint Server. The one of the base template will be used instead.

- **Initiator role.** This role requires the Initiator AssemblyLine and the following Library Components: the *Sender* and *Utils* Script Components, *MemoryPropertiesConnector*, *GenericServiceConnector* and *HttpClientConnector*.

Note: The MemoryProperties Store is not listed, because if missing, it will be added by the Touchpoint Server.

- **Intermediary role.** This role requires the *IntermediaryHandler* AssemblyLine and the following Library Components: the *Sender* and *Utils* Script Components, *MemoryPropertiesConnector* and *HttpClientConnector*.

If you try to create a Touchpoint Instance in a role which requirements are not fulfilled by the custom Type's Template an Exception will be thrown.

You must keep in mind these important points when editing the base Touchpoint Template or creating a custom one.

1. Their Service Connectors (the ones communicating with a third-party system) should inherit from the *GenericServiceConnector* located in the Library.
2. Only one Service Connector should be used in a Touchpoint AssemblyLine. If several are provided, they all will get identical configurations which will render them useless.
3. If you want to change the Store used for communicating Destination URLs, it still should be called *MemoryProperties*.

Resource Persistence

You can learn about persistence and its uses through the information provided here. Further, you can also take care of the conditions while restarting the Touchpoint Instance.

The Touchpoint Server supports persistence of the Atom documents, which could be either automatically generated by it or provided by a remote client. The persistence that is used by the server is stored in a configured folder, in a tree-like structure. The default persistence directory is `solution_directory/tp_state`.

The persisted resources are read back again when the Touchpoint Server is starting up. At this point the Touchpoint Server restores the complete resource tree. The Touchpoint Server persist the Touchpoint Instance configurations in order for a Touchpoint Instance to survive a restart of the server. A Touchpoint Instance is restarted when all of the following conditions are met:

- All of the required configurations for that Touchpoint Instance are available in the persistence storage.
- The remote Touchpoint Provider is up and running.

- There is no other Touchpoint Instance with that configuration running on the Touchpoint Provider.

In the case when the remote Touchpoint Provider is not running the Touchpoint Instance will not be automatically started right after the Touchpoint Provider comes back up. You must either make sure that the Touchpoint Provider is running prior to starting the Touchpoint Server or send a GET request to the Touchpoint Types Feed resource to force the Touchpoint Server to refresh the connection to the remote Touchpoint Provider.

Editing of the files in the persistence directory is not advisable. Currently, those files should be edited by the Touchpoint Server only.

Touchpoint Schema

Touchpoint Schema comprises of many components. You can know more about the schema through the information provided here. Further, you can also know about resource tree and operations allowed.

This section specifies the communication protocol between the Touchpoint Server and a client that is using it for provisioning of Touchpoint Instances. This section does not describe the communication between a client and the Touchpoint Instance itself.

The Touchpoint Server is providing access to the various resources that are involved in the definition of a Touchpoint Instance. The representation of these resources is in a tree-like form. The access to each resource is done using Atom documents over the HTTP/HTTPS protocol.

This is the schema used by the Touchpoint Server:

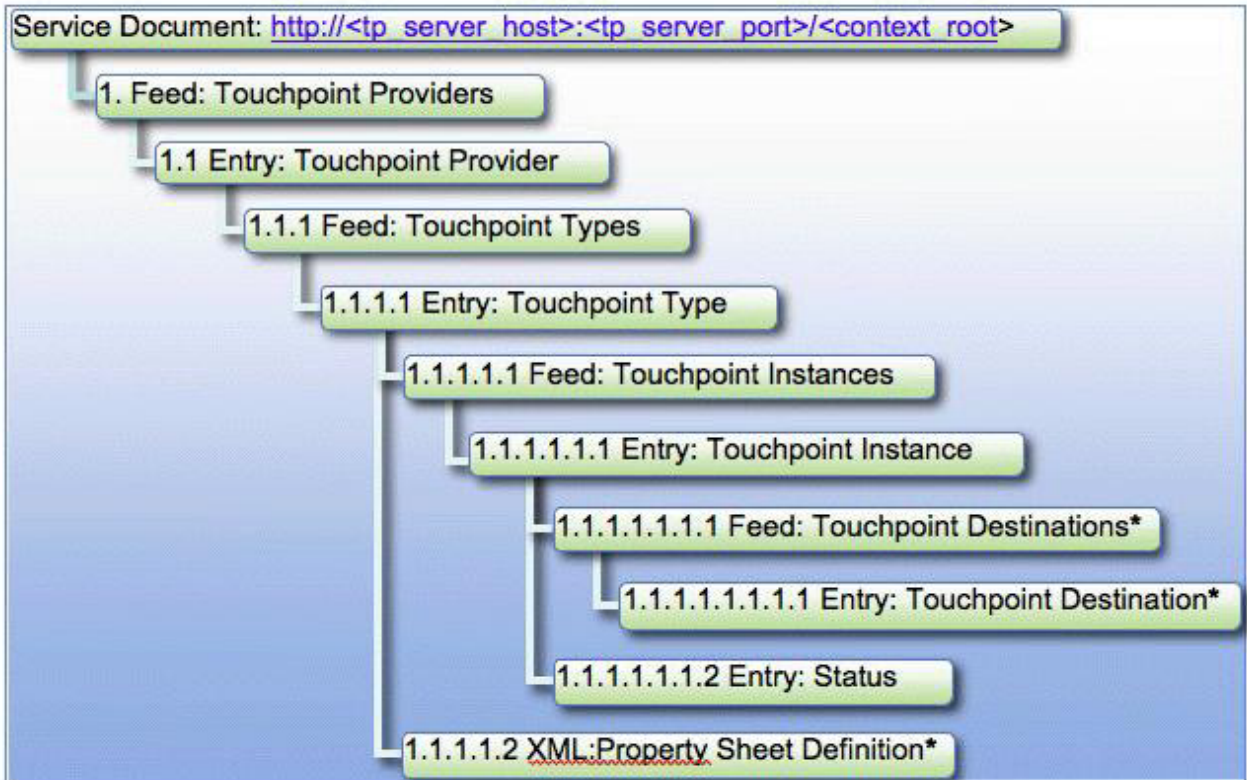


Figure 1. Touchpoint server schema tree

* These tree nodes are available in certain cases. For details see the table below.

In the above schema the following variables are used:

- **tp_server_host** - this is the host address the Touchpoint Server is listening on
- **tp_server_port** - this is the port the Touchpoint Server is listening on (defaults to 1098)
- **context_root** - this is the context root under which the Touchpoint Server application is available (defaults to "tp")

The navigation of the tree is done in a ReSTful way, meaning that client applications should only know the entry point (that is, the URL of the Service Document) and the type of references (Atom links) the Touchpoint Server defines for accessing each of the nodes in the resource tree. These references (URLs) are automatically generated by the Touchpoint Server. Once obtained, the URLs will stay the same until the Touchpoint Server is updated to a newer version. This implies that client applications can "remember" the obtained URLs between updates of the Touchpoint Server, but should work their way back to the particular resource URL if the Touchpoint Server is updated.

According to the protocol specified in the table below, the following steps should be performed in order for a client application to navigate to the Touchpoint Instance Feed starting from the Service Document.

1. Send an HTTP GET request to the Service Document URL. This will return the Service Document from which the Touchpoint Providers Feed URL could be obtained.

2. Send an HTTP GET request to the Touchpoint Providers Feed URL. This will return the Feed Document from which the Touchpoint Provider Entry Reference URL could be obtained.
3. Send an HTTP GET Touchpoint Provider Entry Reference URL. This will return the Entry Document representing the particular Touchpoint Provider. This Entry contains the URL to the Touchpoint Type Feed.
4. Send an HTTP GET request to the Touchpoint Type Feed URL. This will return the Feed Document containing complete copies of all the Touchpoint Type Entries valid for this context. From a Touchpoint Type Entry the client application can obtain the Touchpoint Instance Feed URL.

The table below describes the allowed operations for each resource. Additionally the following points are applicable for the whole resource tree:

- Each entry has a link with relation "self" that points to the standalone Entry Document.
- Resources that accept HTTP methods PUT and DELETE have a link with relation "edit". All such request should be sent to that link (URL).
- Every request to the Touchpoint Server is annotated with the HTTP ETag response-header as defined by the HTTP specification. The ETag value can be used in conjunction with the request-headers If-Match, If-None-Match and If-Range to let the Touchpoint Server know what the client application's preconditions are before servicing the request.

Table 32. Allowed operations per resource

Resource	GET	POST	PUT	DELETE
Service Document	Retrieves the Service Document which contains a list of the available services. The URL to the Touchpoint Providers Feed is set as "href" attribute to a collection that belongs to the category "connectivity-provider".	N/A	N/A	N/A
1. Feed: Touchpoint Providers Categories (term: scheme): connectivity-provider: http://www.ibm.com/xmlns/prod/scmp#resource	Retrieves the list of Touchpoint Provider Entries. All entries are references to the actual Entry Documents representing the available Touchpoint Providers.	N/A	N/A	N/A
1.1 Entry: Touchpoint Provider	Retrieves a Touchpoint Provider Entry the way it was configured in the Touchpoint Server configuration file. This Entry provides some additional details in the <[http://www.ibm.com/xmlns/prod/scmp]:data/> element. The link to the Touchpoint Types Feed is provided using a link with relation " http://www.ibm.com/xmlns/prod/scmp#touchpoint "	N/A	N/A	N/A
1.1.1 Feed: Touchpoint Types Categories (term: scheme): touchpoint: http://www.ibm.com/xmlns/prod/scmp#resource	Retrieves the list of Touchpoint Type Entries. These entries are complete copies of the actual Entry Documents representing the Touchpoint Types.	N/A	N/A	N/A

Table 32. Allowed operations per resource (continued)

Resource	GET	POST	PUT	DELETE
<p>1.1.1.1 Entry: Touchpoint Type</p> <p>Categories (term: scheme):</p> <p>touchpoint: http://www.ibm.com/xmlns/prod/scmp#resource</p> <p>resource-type: http://www.ibm.com/xmlns/prod/scmp#aspect</p> <p>A term uniquely identifying a Touchpoint Type: http://www.ibm.com/xmlns/prod/scmp#touchpoint-type</p>	<p>Retrieves a Touchpoint Type Entry. The URL to the Touchpoint Instance Feed is provided as a link with relation "http://www.ibm.com/xmlns/prod/scmp#instance-feed"</p> <p>The URL to the Property Sheet Definition XML is provided as a link with relation "http://www.ibm.com/xmlns/prod/scmp#property-sheet-definition".</p> <p>Note: The virtual Touchpoint Type does not have a Property Sheet Definition, because no Connectors need to be configured for Intermediary Touchpoints.</p>	N/A	N/A	N/A
<p>1.1.1.1.1 Feed: Touchpoint Instances</p> <p>Categories (term: scheme):</p> <p>touchpoint: http://www.ibm.com/xmlns/prod/scmp#resource</p>	<p>Retrieves the list of Touchpoint Instance Entries. All entries are references to the actual Entry Documents representing the available Touchpoint Instances.</p>	<p>Creates a new Touchpoint Instance Entry. The entry MUST contain a "Touchpoint Configuration" on page 310 that contains the "Touchpoint Configuration" on page 310. The entry must contain a category from the</p> <p>"http://www.ibm.com/xmlns/prod/scmp#touchpoint-role" scheme.</p>	N/A	N/A
<p>1.1.1.1.1.1 Entry: Touchpoint Instance</p> <p>Categories (term: scheme):</p> <p>touchpoint: http://www.ibm.com/xmlns/prod/scmp#resource</p> <p>provider-tp OR initiator-tp OR intermediary-tp: http://www.ibm.com/xmlns/prod/scmp#aspect</p> <p>A term uniquely identifying a Touchpoint Type: http://www.ibm.com/xmlns/prod/scmp#touchpoint-type</p>	<p>Retrieves the Touchpoint Instance Entry. This Entry contains three links to the resources that describe the Touchpoint Instance:</p> <ul style="list-style-type: none"> An URL used for updating/deleting this Touchpoint Instance is provided using a link with relation "edit"; An URL to the Destination Feed is provided using a link with relation http://www.ibm.com/xmlns/prod/scmp#tp-destination <p>Note: The Provider Touchpoints do not support Destinations, so for them this link is missing.</p> <ul style="list-style-type: none"> An URL to the Status Entry is provided using a link with relation http://www.ibm.com/xmlns/prod/scmp#status. An URL to the Touchpoint Type Entry to which this Touchpoint Instance belongs. The link has relation http://www.ibm.com/xmlns/prod/scmp#resource-type 	N/A	<p>Updates the Touchpoint Instance entry. The provided entry must contain the complete copy of the Entry Document and not just the changes. This operation cannot change the role of the Touchpoint Instance. This operation restarts a running Touchpoint Instance in order to reconfigure it.</p>	<p>Deletes the Touchpoint Instance Entry.</p>
<p>1.1.1.1.1.1.1 Feed: Touchpoint Destinations</p>	<p>Retrieves the Touchpoint Instance Destinations Feed. This Feed could contain multiple Touchpoint Instance Destination Entries.</p>	<p>Creates a new Touchpoint Instance Destination Entry. It MUST contain a data element configuring the request-out URL to the Destination.</p>	N/A	N/A

Table 32. Allowed operations per resource (continued)

Resource	GET	POST	PUT	DELETE
1.1.1.1.1.1.1.1 Entry: Touchpoint Destination Categories (term: scheme): tp-destination: http://www.ibm.com/xmlns/prod/scmp#resource	Retrieves the Touchpoint Destination Entry that contains the request-out URL to the remote HTTP Service in the <[http://www.ibm.com/xmlns/prod/scmp]:data/> element. This entry provides a link with relation "edit" to enable updates/deletes on this resource.	N/A	Updates the Touchpoint Instance entry. The provided entry must contain the complete copy of the Entry Document and not just the changes. This operation cannot change the role of the Touchpoint Instance. This operation does NOT restart a running Touchpoint Instance in order to alter the request-out URL.	Deletes the Touchpoint Destination Entry.
1.1.1.1.1.1.2 Entry: Status Categories (term: scheme): touchpoint: http://www.ibm.com/xmlns/prod/scmp#resource status: http://www.ibm.com/xmlns/prod/scmp#aspect	Retrieves the Touchpoint Instance Status Entry which describes the operational state of the particular Touchpoint Instance. It is contained inside the <[http://www.ibm.com/xmlns/prod/scmp]:data/> element. Note: For Provider and Intermediary Touchpoints the status also contains one request-in URL, which should be used by clients when querying the Touchpoint.	N/A	N/A	N/A
1.1.1.1.2 XML: Property Sheet Definition	Retrieves the "Property sheet definitions" on page 314 for the selected Touchpoint Type. It holds the schema of an IBM Security Directory Integrator Connector in the form of an XML document.	N/A	N/A	N/A

Touchpoint Configuration

You can provide the schema of the configuration data in order to configure and start a Touchpoint Instance.

Every configuration data element is placed inside a **<data>** element within the Atom document. The namespace this data element must belongs to is:
<http://www.ibm.com/xmlns/prod/scmp>

Instance Configuration

You can refer to the details provided here to know more about configuration data.

This configuration data specifies:

- Role that the Touchpoint Instance will be running in. The Touchpoint Instance relies on this data to decide which AL from the template to use. In the atom document below, it is denoted by the "{role}" token in the category element. Its supported values are provider-tp, initiator-tp and intermediary-tp.
- Admin state of the created Touchpoint Instance. It is marked by the "{admin_state}" token in the Atom document. The supported values are enabled and disabled.
- Property Sheet XML containing the configuration parameters for Touchpoint Instance's Service Connector. The {param_name} is determined by the Property Sheet Definition of the chosen Touchpoint Type for standard Types or the Property Sheet Definition of the Service Connector for custom Types. The "{param_value}" is determined by the user depending on the wanted configuration. Besides configuration parameters, you can also set the mode of the Service Connector by setting a parameter with {param_name} equal to

"\$initMode" and a string value with the mode name (for example, Iterator, AddOnly). For details about this parameter see section "Property sheet definitions" on page 314.

- The two parameters **{TouchpointID}** and **{version}** are provided in order to allow you to specify additional information for the particular Touchpoint Instance, which has meaning for the creator only. The creator is responsible for ensuring these values are valid in the context of the client application. These values are not interpreted by the Touchpoint Server in any way, it only persists them.

The POSTed Atom Document when creating a Touchpoint Instance Entry Resource is:

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>{id}</id>
  <title>Touchpoint Instance Title</title>
  <author><name>Author Name</name></author>
  <content/>
  <category term="{role}" scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-role" />
  <scmp:data
xmlns:scmp="http://www.ibm.com/xmlns/prod/scmp"
  xsi:schemaLocation="http://www.ibm.com/xmlns/prod/scmp
  http://localhost:1098/tp/schema/touchpoint.xsd" >
    <scmp:touchpoint>
      <scmp:admin-state>{admin_state}</scmp:admin-state>
      <touchpointID>{touchpoint_id}</touchpointID>
      <version>{version}</version>
      <scmp:propertySheet>
        <scmp:property propertyName="{param_name}">
          <scmp:value>{param_value}</scmp:value>
        </scmp:property>
        ...
      </scmp:propertySheet>
    </scmp:touchpoint>
  </scmp:data>
</entry>
```

Please note that an ID for the created Touchpoint Instance can also be provided in place of the "{id}" token. However, no matter what the passed value is, the Touchpoint Server will overwrite it with an automatically generated one. This way it guarantees the uniqueness of the Touchpoint Instance ID.

Destination Configuration

You can learn to add a destination in the Touchpoints by working on a Atom document.

Both the Intermediary and Initiator Touchpoints require a Destination to be configured, before they become operational. Furthermore, they support multiple such Destinations and the ability to add and remove them at runtime.

This is done by POSTing an Atom Document on the Touchpoint Destination Feed URL of the created Touchpoint Instance. Keep in mind that no such URL is available for Provider Touchpoints. Here is how it should look like:

```
<entry xmlns="http://www.w3.org/2005/Atom">
  <scmp:data
xmlns:scmp="http://www.ibm.com/xmlns/prod/scmp"
  xsi:schemaLocation="http://www.ibm.com/xmlns/prod/scmp
  http://localhost:1098/tp/schema/touchpoint.xsd" >
    <scmp:destination>
      <scmp:request-out>{request-out_URL}</scmp:request-out>
      <scmp:request-error>{request-error_URL}</scmp:request-error>
    </scmp:destination>
  </scmp:data>
</entry>
```


As can be seen from this snippet, only the "{request-out_URL}" is needed for configuring a Destination; the "{request-error_URL}" is optional.

Touchpoint Instance communication protocol

You can refer the protocol used to communicate with a Touchpoint Instance. In most cases a Touchpoint Instance will be derived from the Touchpoint Template.

Provider Touchpoint

You can use the Provider Touchpoint which handles HTTP methods described here.

Table 33. Provider Touchpoint HTTP methods

HTTP Method	URL Query Parameters	Connector mode	HTTP Request Content	HTTP Response Content	HTTP Response Code
GET	-	Iterator	-	all Entries found	"200 OK" if at least one Entry was found "404 Not Found" if no Entries are found
GET	link criteria	Lookup	-	all Entries found	"200 OK" if at least one Entry was found "404 Not Found" if no Entries are found
POST	-	AddOnly	Entry to be added	-	"201 Created" if the operation was successful
PUT*	link criteria	Update	Entry with updated attributes	-	"201 Created" if the Entry did not exist and was added "204 No Content" if a single Entry matches the link criteria and the Entry was updated successfully
DELETE	link criteria	Delete	-	-	"204 No Content" if the operation was successful (the operation will fail if multiple Entries match the link criteria)

(*) Note that there is a difference in the handling of the PUT method between our implementation and the HTTP 1.1 specification (<http://tools.ietf.org/html/rfc2616#section-9.6>). According to the HTTP specification, a PUT request is supposed to replace the whole resource. In our implementation if the entry exists we do not replace it as a whole, but only replace the specified attributes.

The Link Criteria for the Connector operations is derived from the query parameters of the requested URL. For example, a GET request with URL "http://localhost/mytp?username=jsmith" will result in a Lookup operation with link criteria "username=jsmith". Each query parameter corresponds to an EXACT match criterion. The criteria derived from multiple query parameters are combined using the 'AND' logic operator. For example: "?firstname=john&age=50" corresponds to ((firstname equals "john") AND (age equals "50")).

Query parameters are required for PUT and DELETE requests. POST requests are not expected to contain query parameters. If a GET request contains query parameters, it is translated to a Lookup mode operation. Otherwise it is translated to an Iterator mode operation.

On GET requests you can use an optional HTTP header named "X-TDI-TP-SizeLimit" to limit the number of returned Entries. The value of the header must be an integer larger than zero.

All HTTP Methods interpreted by the default Touchpoint Template are compliant with the HTTP Specification according their safety and idempotence characteristics.

Initiator Touchpoint

Refer to the information provided here to know more about the Initiator Touchpoint.

The Initiator Touchpoint Instance acts as an HTTP client. It has an Iterator Connector which produces Entry objects that the AssemblyLine sends to a configured Destination URLs. For each Entry it sends a single POST request, whose content is an "Representation of Entry objects as HTTP content."

Intermediary Touchpoint

You can use the Intermediary Touchpoint as a moderator between several Touchpoint Instances.

The Intermediary Touchpoint Instance is similar to both the Provider and Initiator roles. It accepts requests on a particular request-in URL as a Provider and sends the received data to multiple Destinations as an Initiator. Due to this forwarding functionality it can be used as moderator between several Touchpoint Instances from the other roles.

Representation of Entry objects as HTTP content

You can refer to the example shown here to view entry objects as HTTP content.

Example:

```
<tp:data xmlns:tp="http://www.ibm.com/xmlns/prod/tdi/72/tp">
  <tp:entry>
    <tp:attribute name="username">
      <tp:value><![CDATA[jsmith]]</tp:value>
    </tp:attribute>

    <tp:attribute name="mail">
      <tp:value>jsmith@ibm.us.com</tp:value>
      <tp:value>john.smith@gmail.com</tp:value>
    </tp:attribute>
  </tp:entry>
</tp:data>
```

The HTTP content must be encoded in "UTF-8". It must contain a single data element which can contain zero or more entry elements.

XML Schema description on the touchpoint data format:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
  targetNamespace="http://www.ibm.com/xmlns/prod/tdi/72/tp"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  xmlns=http://www.w3.org/2001/XMLSchema xmlns:tns=http://www.ibm.com/xmlns/prod/tdi/72/tp >

  <element name="data" type="tns:TouchpointDataType" />

  <element name="entry" type="tns:EntryType" />

  <element name="attribute" type="tns:AttributeType" />

  <element name="property" type="tns:PropertyType" />

  <element name="value" type="string" />
```

```

<complexType name="TouchpointDataType">
  <choice minOccurs="0" maxOccurs="unbounded">
    <element ref="tns:entry" />
  </choice>
</complexType>

<complexType name="EntryType">
  <choice minOccurs="0" maxOccurs="unbounded">
    <element ref="tns:property" />
    <element ref="tns:attribute" />
  </choice>
</complexType>

<complexType name="AttributeType">
  <choice minOccurs="0" maxOccurs="unbounded">
    <element ref="tns:property" />
    <element ref="tns:attribute" />
    <element ref="tns:value" />
  </choice>
  <attribute name="name" type="string" use="required" />
  <attribute name="namespaceURI" type="anyURI" />
</complexType>

<complexType name="PropertyType">
  <simpleContent>
    <extension base="string">
      <attribute name="name" type="string" use="required" />
      <attribute name="namespaceURI" type="anyURI" />
    </extension>
  </simpleContent>
</complexType>

</schema>

```

Touchpoint Status Entry schema

You can retrieve the status of the Touchpoint Instance by sending an HTTP GET request to the URL of the Status Entry.

“Touchpoint Instance communication protocol” on page 312.

```

<entry xmlns="http://www.w3.org/2005/Atom">
  <id>Status ID</id>
  <link href="{touchpoint_instance_status_URL}" type="application/atom+xml;type=entry"
    rel="self"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#resource" term="touchpoint"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#aspect" term="status"/>
  <scmp:data
    xmlns:scmp="http://www.ibm.com/xmlns/prod/scmp"
    xsi:schemaLocation="http://www.ibm.com/xmlns/prod/scmp
    http://localhost:1098/tp/schema/touchpoint.xsd" >
    <scmp:touchpoint-status>
      <scmp:request-in>{request-in_URL}</scmp:request-in>
      <scmp:op-state>{op-state}</scmp:op-state>
    </scmp:touchpoint-status>
  </scmp:data>
</entry>

```

The returned Atom Entry Document contains a **<data>** element that describes the status of the Touchpoint Instance. The data element belongs to the `http://www.ibm.com/xmlns/prod/scmp` namespace. It has the following syntax:

- **{op-state}** - is the string keyword describing the current state of the Touchpoint Instance as defined in section “Touchpoint Instance” on page 298.
- **{request-in_URL}** - is the URL for accessing Provider and Intermediary Touchpoint Instances.

Property sheet definitions

The Property Sheet Definition is an XML document that determines the schema of an IBM Security Directory Integrator Connector. You can learn more about it through the information provided here.

It can be obtained from a link inside the Touchpoint Type Entry (as described in section "Touchpoint Schema" on page 306).

Depending on the Touchpoint Type the Property Sheet Definition varies:

- For standard Types, it contains the schema of the IBM Security Directory Integrator Connector corresponding to the Touchpoint Type.
- For custom Types, it contains the schema of the Service Connector in the custom Touchpoint Template.
- For the virtual Type (virtual://Intermediary) there is no Property Sheet Definition, as there is no third-party system to communicate to (this role relies solely on HTTP).

Besides the schema parameters of an IBM Security Directory Integrator Connector the Property Sheet Definition contains the modes, supported by the Connector. They are stored as optional values for the propertyDefinition with name `$initMode`. Their values directly match the Connector mode names - "Iterator", "AddOnly", "CallReply", and so forth.

Here is an example Property Sheet Definition for the File Connector (Touchpoint Type system:/Connectors/ibmdi.LDAP):

```
<?xml version="1.0" encoding="UTF-8"?>
<propertySheetDefinition xmlns="http://www.ibm.com/xmlns/prod/scmp">
  <propertyDefinition required="true" hidden="false" readOnly="false"
    propertyType="string" multiple="false"
    propertyName="ldapUrl">
    <label label="LDAP URL" lang="en"/>
    <!--one label for the different languages supported by TDI -->
  </propertyDefinition>
  <!--the rest of LDAP Connector's parameters -->
  <propertyDefinition required="false" readOnly="false" propertyType="string"
    multiple="false" propertyName="$initMode">
    <label label="$initMode" lang="en"/>
    <!--one label for the different languages supported by TDI -->
    <option>
      <value>AddOnly</value>
      <label label="AddOnly" lang="en"/>
      <!--one label for the different languages supported by TDI -->
    </option>
    <option>
      <value>Iterator</value>
      <label label="Iterator" lang="en"/>
      <!--one label for the different languages supported by TDI -->
    </option>
    <!--the rest of modes supported by the LDAP Connector -->
  </propertyDefinition>
</propertySheetDefinition>
```

To shorten the listing, only one of its configuration parameters and the `$initMode` property definition are included. As can be seen this Connector has a parameter named `ldapUrl` which is required and its value is a string. Also, its English label as can be seen in the CE is **LDAP URL** (the labels for the rest of the languages supported by IBM Security Directory Integrator are skipped). The `$initMode` parameter is also present, and as can be seen from its optional values this Connector supports both Iterator and AddOnly modes (and several others which are skipped).

Property Sheet Definitions significantly assist you when creating Touchpoint Instances, as you do not need to know the schema of the Connector in advance. Relying on their information, you can perform this task, much like you configure Connectors inside IBM Security Directory Integrator's Configuration Editor - by seeing the needed parameters, checking their expected values (a string, a number or a set of predefined values) and providing them in the configuration.

XML Schema locations

You can define the location of XML Schema through the information provided here.

An XML Schema document is provided for each `scmp:data` element. The document defines all elements which appear inside the `scmp:data` element.

The location of the schema document is specified in an `xsi:schemaLocation` (XML Schema Instance location) attribute defined on the `scmp:data` element. For example:

```
<scmp:data
  xmlns:scmp="http://www.ibm.com/xmlns/prod/scmp"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/xmlns/prod/scmp
http://localhost:1098/tp/schema/tdi-connectivity-provider.xsd">

  <scmp:connectivity-provider>
```

The location of the schema is a valid URL which can be de-referenced by web clients: Any client of the Touchpoint Server can resolve the URL which appears inside the `xsi:schemaLocation` attribute and access the actual schema document.

Moreover if the schema contains a reference to another schema document (for example, via "import", "include" or "redefine" XML Schema element), the URL in the reference can be resolved by web clients to obtain the referred schema.

Error flows

Appropriate error messages are issued to the standard log if an error is thrown. Refer to the information provided here to know more about possible error situations and XML document syntax.

The possible situations for which an error might be produced are:

- Incorrect configuration of Touchpoint Server.
- Exceptions thrown as result of communication problems with the Persistence Store (File System errors).
- Error in communication with the clients of the Touchpoint Server.
- Error in communication with of the Touchpoint Server with IBM Security Directory Integrator.

When the error is due to an invalid information/request sent by the user, and which violates the Touchpoint Server protocol, it will produce an XML document with the following syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns2:error xmlns:ns2="http://www.ibm.com/xmlns/prod/scmp">
  <creation-time>2010-02-23T14:49:06.384+02:00</creation-time>
  <code>100005</code>
  <details>
    <detail>
      <name>schema</name>
      <value>http://www.ibm.com/xmlns/prod/scmp#touchpoint-role</value>
    </detail>
  </details>
  <native-msgid>ABCD1234E</native-msgid>
  <summary>Missing role category</summary>
</ns2:error>
```

Where:

- **creation-time** element denotes the time the error has occurred in a format specified by `http://www.w3.org/TR/1998/NOTE-datetime-19980827`.
- **code** element is one of the following codes:

- 100000 – unknown error occurred (could not narrow it further)
 - 100001 – Missing required atom:link. Details element will provide: **rel** - relationship name of expected link
 - 100002 – Missing required scmp:data element –Details element will provide: **qname** - missing element qname
 - 100003 – Invalid atom:entry in POST/PUT operation (for example, parse error)
 - 100004 – Invalid value for scmp:data element – Details element will provide: **qname** - invalid element qname; **value** - invalid supplied value
 - 100005 – Missing required atom:category. Details element will provide: **scheme** - expected scheme name
 - 100006 – Invalid atom:category value. Details element will provide: **scheme** - expected scheme name; **term** - invalid supplied term
 - 100007 – Too many atom:link for given relationship. Details element will provide: **rel** - relationship name of extra link
 - 100008 – Too many atom:category values for scheme. Details element will provide: **scheme** - overpopulated scheme name
 - 330000 – Default connectivity-provider-specific error (unable to narrow down further)
- **details** element is containing more details for the specific error. See the particular error code to find out what kind of detailed information is expected for each error.
 - **native-msgid** element denotes the message short id.
 - **summary** element contains the human readable error message.

When the error is not caused by protocol violation but is from a different source, a human readable representation is returned in a plain text format. The error also contains the exception stack trace so you can report it to the Touchpoint Server administrator in order for it to be resolved.

Configuration

You can configure the Touchpoint server with the help of information provided here.

The Touchpoint Server runs inside a web container. The default web container shipped with IBM Security Directory Integrator is configured by the following properties in `global.properties` or `solution.properties`:

- `tp.server.on` - specifies whether the bundled web container and the Touchpoint Server should be started. Default value: *false*.
- `tp.server.port` - specifies the port the web container will be listening on. Default value: 1098.
- `tp.server.auth` - specifies whether the Touchpoint Server will use HTTP Basic authentication. Default value: *false*.
- `tp.server.auth.realm` - specifies the realm HTTP basic authentication. Default value: "IBM Security Directory Integrator Touchpoint Server".

The Touchpoint Server first considers the value of the `api.remote.bind.address` property and if that is not set, the value of the `com.ibm.di.default.bind.address` property. In this way it is able to effectively filter the access to "multi-homed" hosts.

The web container is able to use SSL for securing the transportation layer. It reuses the Remote API's settings and is enabled by setting the `api.remote.ssl` property. SSL client authentication is enabled by the `api.remote.ssl.client.auth.on` property. The server SSL keys are configured using the well known Remote API properties:

- `api.keystore`
- `api.client.keystore.pass`
- `api.client.key.pass`
- `api.client.keystore.type`

HTTP basic authentication (<http://tools.ietf.org/html/rfc2617>) can be configured using the `tp.server.auth` and `tp.server.auth.realm` properties. It is disabled by default. See section "Authentication" on page 319 for more information on authentication.

The configuration of the Touchpoint Server is specified using an XML file. The path to this file is specified in the `global.properties` or `solution.properties` file using the property `tp.server.config`. An example Touchpoint Server configuration file is shipped in the `etc` directory of the IBM Security Directory Integrator installation.

The following syntax is used by the Touchpoint Server configuration file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tp:tpServerConfig xmlns:tp="http://www.ibm.com/xmlns/prod/tdi/72/tp" tp:version="1.0">

  <!-- specifies the encryption settings used when encrypting passwords -->
  <tp:encryptionConfig stash="idisrv.sth">
    <tp:keyStore>testserver.jks</tp:keyStore>
    <tp:keyStoreType>jks</tp:keyStoreType>
    <tp:keyAlias>server</tp:keyAlias>
    <tp:transformation>RSA</tp:transformation>
  </tp:encryptionConfig>

  <tp:templateConfig>
    <tp:baseTemplate>etc/TouchpointTemplate.xml</tp:baseTemplate>

    <!-- Specify the path to the directory that holds the Touchpoint templates. -->
    <!--
    <tp:customTemplatesDir>templates</tp:customTemplatesDir>
    -->
  </tp:templateConfig>

  <!-- specifies the persistence settings that configure the place to persist the state -->
  <tp:persistenceConfig>
    <tp:enabled>true</tp:enabled>
    <tp:location>tp_state</tp:location>
  </tp:persistenceConfig>

  <!-- configures the touchpoint providers (nodes) -->
  <tp:nodeConfigs>
    <!-- Default connection to the local server -->
    <tp:tdiNodeConfig tp:local="true" tp:id="default">
      <!-- The host of the remote node which all
      Provider Touchpoint Instances will receive requests on -->
      <tp:providerHost>localhost</tp:providerHost>
      <!-- The port of the remote node which all
      Provider Touchpoint Instances will receive requests on -->
      <tp:providerPort>1097</tp:providerPort>

      <tp:title>Example Touchpoint Provider</tp:title>
      <tp:author>John Doe</tp:author>
      <tp:email>jdoe@example.org</tp:email>
      <tp:summary>Example Touchpoint Provider Atom Entry</tp:summary>

      <tp:contact>Local Administrator</tp:contact>
      <tp:location>Main building, 5th fl.</tp:location>
      <tp:organization>Example Organization</tp:organization>
    </tp:tdiNodeConfig>
  </tp:nodeConfigs>
</tp:tpServerConfig>
```



```

<!-- Here is an example of a remote server connection -->
<!--
  <tp:tdiNodeConfig id="remote" local="false">
    <tp:title>Example Touchpoint Provider</tp:title>
    <tp:author>John Doe</tp:author>
    <tp:email>jdoe@example.org</tp:email>
    <tp:summary>Example Touchpoint Provider</tp:summary>

    <tp:host>localhost</tp:host>
    <tp:port>1099</tp:port>
    <tp:user>username</tp:user>
    <tp:password protect="true" encrypted="false">password</tp:password>

    <tp:contact>Jack Smith</tp:contact>
    <tp:location>5th fl.</tp:location>
    <tp:organization>Example Organization</tp:organization>

    <tp:providerHost>localhost</tp:providerHost>
    <tp:providerPort>1097</tp:providerPort>
  </tp:tdiNodeConfig>
-->
</tp:nodeConfigs>
</tp:tpServerConfig>

```

Authentication

You can learn more about the authentication aspects of the Touchpoint Server through the information provided here.

There are two aspects of the Touchpoint Server related to authentication:

- being an HTTP server
- being a client of the remote RMI Server API of the IBM Security Directory Integrator Servers, which are configured as connectivity providers.

As an HTTP server, the Touchpoint Server supports HTTP basic authentication of HTTP clients. It does not use a separate user registry. Instead the Touchpoint Server delegates authentication requests to the Server API of the local IBM Security Directory Integrator Server (the one that hosts the Touchpoint Server).

As a remote Server API client, the Touchpoint Server needs to authenticate against the remote IBM Security Directory Integrator Server like any other Server API client would do. Note that authentication is not required when the connectivity provider is the local IBM Security Directory Integrator Server.

For more on Server API authentication, see the appendix called *Server API in Reference*.

Examples

You can refer to the links provided here to use the shipped examples and use the other examples for creating a Touchpoint Instance using JDBC connector.

Shipped example

You can use the steps provided here to refer the shipped example of creating a Touchpoint Instance.

The main example demonstrating the use of the Touchpoint Server functionality in IBM Security Directory Integrator, demonstrating example steps for creating both Provider and Initiator Touchpoint Instances, is shipped with the installation. The steps are spelled out in the document found at *TDI_Install_dir/examples/TouchpointClient/Touchpoint_Example.pdf*. In addition to that, a sample

implementation of those steps is provided to demonstrate how they might look when written in the Java programming language. The Java code is structured in two packages, as follows:

- **com.ibm.di.tp.client.api** – this package contains the code demonstrating how the communication with the Touchpoint Server is performed. It contains nothing but methods used for querying the Touchpoint Server.

Note: This code depends on Apache HttpClient v3.x.

- **com.ibm.di.tp.client.gui** – this package contains a sample UI client that uses the com.ibm.di.tp.client.api package to interact with the Touchpoint Server in order to provide Touchpoint Instances. You can use this UI for testing purposes when provisioning Touchpoint Instances. You may start this utility using the provided scripts: startClient.bat and startClient.sh.

Example steps for creating a Touchpoint Instance using a JDBC Connector

Refer to the instructions provided here to provision a Touchpoint Instance.

In order to provision a Touchpoint Instance, you need to send an HTTP POST request to the Touchpoint Server. The URLs for the requests are obtainable according to the application “Touchpoint Schema” on page 306. For the purpose of this example we are using pseudo URLs as: <Resource Name URL>. You can obtain the appropriate URL at runtime.

Provider Touchpoint Instance

You can use the example provided here to create a Touchpoint Instance Entry resource.

POST <Touchpoint Instance Feed URL>

```
Body:<entry xmlns="http://www.w3.org/2005/Atom">
  <id>some ID</id>
  <title>Provider Touchpoint Instance</title>
  <author><name>author_name</name></author>
  <content/>
  <category term="provider-tp" scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-role" />
  <scmp:data xmlns:scmp="http://www.ibm.com/xmlns/prod/scmp" >
    <scmp:touchpoint>
      <scmp:propertySheet>
        <scmp:property propertyName="jdbcSource">
          <scmp:value>some source value</scmp:value>
        </scmp:property>
        <scmp:property propertyName="jdbcDriver">
          <scmp:value>the driver class</scmp:value>
        </scmp:property>
        <scmp:property propertyName="jdbcTable">
          <scmp:value>the table name</scmp:value>
        </scmp:property>
        <!--The rest of the parameters required by a JDBC Connector-->
      </scmp:propertySheet>
      <scmp:admin-state>enabled</scmp:admin-state>
    </scmp:touchpoint>
  </scmp:data>
</entry>
```

The Touchpoint Server will return a response similar to:

```
201 Created
Location: <Touchpoint Instance Entry URL>
Body:
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/"
xmlns:ns3="http://www.w3.org/1999/xhtml">
  <id>generated_id</id>
  <updated>2010-02-17T18:33:55.302+02:00</updated>
  <title>Provider Touchpoint Instance</title>
  <link href="<Touchpoint Instance Entry URL>" type="application/atom+xml;type=entry" rel="self"/>
  <link href="<Touchpoint Instance Entry URL>" type="application/atom+xml;type=entry" rel="edit"/>
  <link href="<Touchpoint Type Entry URL>"
```

```

type="application/atom+xml;type=entry" rel="http://www.ibm.com/xmlns/prod/scmp#resource-type"/>
<link href="<Touchpoint Instance Status Entry URL>" type="application/atom+xml;type=entry"
rel="http://www.ibm.com/xmlns/prod/scmp#status"/>
<author><name>author_name</name></author>
<category scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-type" term="system:/Connectors/ibmdi.JDBC"/>
<category scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-role" term="provider-tp"/>
<category scheme="http://www.ibm.com/xmlns/prod/scmp#resource" term="touchpoint"/>
<ns2:data xmlns:ns2="http://www.ibm.com/xmlns/prod/scmp">
  <ns2:touchpoint>
    <ns2:admin-state>enabled</ns2:admin-state>
    <ns2:propertySheet>
      <ns2:property propertyName="jdbcSource" xmlns="" xmlns:ns5="http://www.w3.org/2005/Atom">
        <ns2:value>some source value</ns2:value>
      </ns2:property>
      <ns2:property propertyName="jdbcDriver" xmlns="" xmlns:ns5="http://www.w3.org/2005/Atom">
        <ns2:value>the driver class</ns2:value>
      </ns2:property>
    <!--The rest of the parameters required by a JDBC Connector-->
  </ns2:propertySheet>
</ns2:touchpoint>
</ns2:data>

```

Note that the Touchpoint Server changes the ID of the entry to guarantee its uniqueness.

The URL used for accessing the created Touchpoint Instance can be retrieved through the Status Entry URL. To do this, send an HTTP GET request to URL <Touchpoint Instance Status Entry URL>. The received response looks like this:

```

200 OK
Location: <Touchpoint Instance Status Entry URL>
Body:
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/"
xmlns:ns3="http://www.w3.org/1999/xhtml">
  <id>generated_id</id>
  <link href="<Touchpoint Instance Status Entry URL>" type="application/atom+xml;type=entry" rel="self"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#resource" term="touchpoint"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#aspect" term="status"/>
  <ns2:data xmlns:ns2="http://www.ibm.com/xmlns/prod/scmp">
    <ns2:touchpoint-status>
      <ns2:request-in>Touchpoint Provider Request-in URL</ns2:request-in>
      <ns2:op-state>available</ns2:op-state>
    </ns2:touchpoint-status>
  </ns2:data>
</entry>

```

Initiator Touchpoint Instance

You can use the example provided here to create a Initiator Touchpoint Instance.

POST <Touchpoint Instance Feed URL>

```

Body:
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>some ID</id>
  <title>Initiator Touchpoint Instance</title>
  <author><name>author_name</name></author>
  <content/>
  <category term="initiator-tp" scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-role" />
  <scmp:data xmlns:scmp="http://www.ibm.com/xmlns/prod/scmp" >
    <scmp:touchpoint>
      <scmp:propertySheet>
        <scmp:property propertyName="jdbcSource">
          <scmp:value>some source value</scmp:value>
        </scmp:property>
        <scmp:property propertyName="jdbcDriver">
          <scmp:value>the driver class</scmp:value>
        </scmp:property>
        <scmp:property propertyName="jdbcTable">
          <scmp:value>the table name</scmp:value>
        </scmp:property>
        <!--The rest of the parameters required by a JDBC Connector-->
      </scmp:propertySheet>
      <scmp:admin-state>enabled</scmp:admin-state>
    </scmp:touchpoint>
  </scmp:data>
</entry>

```

The Touchpoint Server returns a response similar to:

```

201 Created
Location: <Touchpoint Instance Entry URL>
Body:
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/"
xmlns:ns3="http://www.w3.org/1999/xhtml">

```

```

<id>generated_ID</id>
<updated>2010-02-17T18:33:55.302+02:00</updated>
<title>Initiator Touchpoint Instance</title>
<link href="<Touchpoint Instance Entry URL>" type="application/atom+xml;type=entry" rel="self"/>
<link href="<Touchpoint Instance Entry URL>" type="application/atom+xml;type=entry" rel="edit"/>
<link href="<Touchpoint Type Entry URL>"
type="application/atom+xml;type=entry" rel="http://www.ibm.com/xmlns/prod/scmp#resource-type"/>
<link href="<Touchpoint Instance Status Entry URL>"
type="application/atom+xml;type=entry" rel="http://www.ibm.com/xmlns/prod/scmp#status"/>
<link href="<Touchpoint Instance Destination Feed URL>"
type="application/atom+xml;type=feed" rel="http://www.ibm.com/xmlns/prod/scmp#tp-destination"/>
<author><name>author_name</name></author>
<category scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-type" term="system:Connectors/ibmdi.JDBC"/>
<category scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-role" term="initiator-tp"/>
<category scheme="http://www.ibm.com/xmlns/prod/scmp#resource" term="touchpoint"/>
<ns2:data xmlns:ns2="http://www.ibm.com/xmlns/prod/scmp">
  <ns2:touchpoint>
    <ns2:admin-state>enabled</ns2:admin-state>
    <ns2:propertySheet>
      <ns2:property propertyName="jdbcSource" xmlns="" xmlns:ns5="http://www.w3.org/2005/Atom">
        <ns2:value>some source value</ns2:value>
      </ns2:property>
      <ns2:property propertyName="jdbcDriver" xmlns="" xmlns:ns5="http://www.w3.org/2005/Atom">
        <ns2:value>the driver class</ns2:value>
      </ns2:property>
    </ns2:propertySheet>
    <!--The rest of the parameters required by a JDBC Connector-->
  </ns2:touchpoint>
</ns2:data>
</content/>
</entry>

```

Note that the Touchpoint Server changes the ID of the entry to guarantee its uniqueness.

Furthermore, this time the Touchpoint Server response contains a Touchpoint Instance Destination Feed URL, which is needed for configuring the Touchpoint Destinations.

Next, we will add one Destination to the Initiator Touchpoint:

```

POST <Touchpoint Instance Destination Feed>
Body:
<entry xmlns="http://www.w3.org/2005/Atom">
  <scmp:data xmlns:scmp="http://www.ibm.com/xmlns/prod/scmp" >
    <scmp:destination>
      <scmp:request-out>Request-out URL</scmp:request-out>
    </scmp:destination>
  </scmp:data>
</entry>

```

The Touchpoint Server returns a response similar to:

```

201 Created
Location: <Touchpoint Instance Destination Entry URL>
Body:
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/"
xmlns:ns3="http://www.w3.org/1999/xhtml">
  <id>generated_id</id>
  <updated>2010-02-18T10:52:35.108+02:00</updated>
  <link href="<Touchpoint Instance Destination Entry URL>" type="application/atom+xml;type=entry" rel="self"/>
  <link href="<Touchpoint Instance Destination Entry URL>" rel="edit"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#resource" term="tp-destination"/>
  <ns2:data xmlns:ns2="http://www.ibm.com/xmlns/prod/scmp">
    <ns2:destination>
      <ns2:request-out>Request-out URL</ns2:request-out>
    </ns2:destination>
  </ns2:data>
</entry>

```

At this point, the Initiator Touchpoint Instance starts executing.

Intermediary Touchpoint Instance

The steps needed to create a Touchpoint Instance in this Role are a combination of the ones for the Provider and Initiator roles.

First, we create a Touchpoint Instance Entry resource. This time the Touchpoint Instance Feed URL is concrete - `http://<tp_server_host>:<tp_server_port>/<context_root>/tp-node/default/tp-type/virtual__Intermediary/tp-inst` .

POST <Touchpoint Instance Feed URL>

```
Body:
<entry xmlns="http://www.w3.org/2005/Atom">
  <id>some ID</id>
  <title>Intermediary Touchpoint Instance</title>
  <author><name>author_name</name></author>
  <content/>
  <category term="intermediary-tp"
  scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-role" />
  <scmp:data xmlns:scmp="http://www.ibm.com/xmlns/prod/scmp" >
    <scmp:touchpoint>
      <scmp:propertySheet>
        <!--No parameters are required-->
      </scmp:propertySheet>
      <scmp:admin-state>enabled</scmp:admin-state>
    </scmp:touchpoint>
  </scmp:data>
</entry>
```

The Touchpoint Server returns a response similar to:

```
201 Created
Location: <Touchpoint Instance Entry URL>
Body:
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/"
xmlns:ns3="http://www.w3.org/1999/xhtml">
  <id>generated_ID</id>
  <updated>2010-02-18T11:20:00.546+02:00</updated>
  <title>Intermediary Touchpoint Instance</title>
  <link href="<Touchpoint Instance Entry URL>" type="application/atom+xml;type=entry" rel="self"/>
  <link href="<Touchpoint Instance Entry URL>" type="application/atom+xml;type=entry" rel="edit"/>
  <link href="<Touchpoint Type Entry URL>" type="application/atom+xml;type=entry"
  rel="http://www.ibm.com/xmlns/prod/scmp#resource-type"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-type" term="virtual://Intermediary"/>
  <link href="<Touchpoint Status Entry URL>" type="application/atom+xml;type=entry"
  rel="http://www.ibm.com/xmlns/prod/scmp#status"/>
  <link href="<Touchpoint Destinations Feed URL>" type="application/atom+xml;type=feed"
  rel="http://www.ibm.com/xmlns/prod/scmp#tp-destination"/>
  <author><name>author_name</name></author>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#touchpoint-role" term="intermediary-tp"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#resource" term="touchpoint"/>
  <ns2:data xmlns:ns2="http://www.ibm.com/xmlns/prod/scmp">
    <ns2:touchpoint>
      <ns2:admin-state>enabled</ns2:admin-state>
      <ns2:propertySheet/>
    </ns2:touchpoint>
  </ns2:data>
  <content/>
</entry>
```

Note that the Touchpoint Server changes the ID of the entry to guarantee its uniqueness.

Next, we will add one Destination to the Intermediary Touchpoint:

POST <Touchpoint Instance Destinations Feed>

```
Body:
<entry xmlns="http://www.w3.org/2005/Atom">
  <scmp:data xmlns:scmp="http://www.ibm.com/xmlns/prod/scmp" >
    <scmp:destination>
      <scmp:request-out>Request-out URL</scmp:request-out>
    </scmp:destination>
  </scmp:data>
</entry>
```

The Touchpoint Server should return a response similar to:

```
201 Created
Location: <Touchpoint Instance Destination Entry URL>
Body:
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/"
xmlns:ns3="http://www.w3.org/1999/xhtml">
  <id>generated_id</id>
  <updated>2010-02-18T10:52:35.108+02:00</updated>
  <link href="<Touchpoint Instance Destination Entry URL>" type="application/atom+xml;type=entry" rel="self"/>
  <link href="<Touchpoint Instance Destination Entry URL>" rel="edit"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#resource" term="tp-destination"/>
  <ns2:data xmlns:ns2="http://www.ibm.com/xmlns/prod/scmp">
```

```

    <ns2:destination>
      <ns2:request-out>Request-out URL</ns2:request-out>
    </ns2:destination>
  </ns2:data>

```

Finally, we will get the URL used for sending request to the Intermediary Touchpoint Instance. As for the Provider Touchpoint, this is done through the Status Entry.

Send an HTTP GET request to the <Touchpoint Instance Status Entry URL>, available in the Touchpoint Instance Entry.

The Touchpoint Server returns a response similar to:

```

200 OK
Location: <Touchpoint Instance Status Entry URL>
Body:
<?xml version="1.0" encoding="UTF-8"?>
<entry xmlns="http://www.w3.org/2005/Atom" xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/"
xmlns:ns3="http://www.w3.org/1999/xhtml">
  <id>generated_id</id>
  <link href="Touchpoint Instance Status Entry URL" type="application/atom+xml;type=entry" rel="self"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#resource" term="touchpoint"/>
  <category scheme="http://www.ibm.com/xmlns/prod/scmp#aspect" term="status"/>
  <ns2:data xmlns:ns2="http://www.ibm.com/xmlns/prod/scmp">
    <ns2:touchpoint-status>
      <ns2:request-in>Touchpoint Intermediary Request-in URL</ns2:request-in>
      <ns2:op-state>available</ns2:op-state>
    </ns2:touchpoint-status>
  </ns2:data>
</entry>

```

Chapter 18. Tombstone Manager

You can retrieve exit status and other information through tombstones. Read the information provided here to know more about its features.

IBM Security Directory Integrator can keep track of configurations or AssemblyLines that have terminated. This way, you can tell when your AssemblyLines last ran, without going into the log of each one.

This is accomplished by the *Tombstone Manager* of IBM Security Directory Integrator that creates *tombstones* for each AssemblyLine and configuration as they terminate. Tombstones contain exit status and other information that you can request through the Server API. Tombstone Manager also:

- Displays the status of an entire IBM Security Directory Integrator configuration in an AMC status window.
- Ensures repeated runs of AssemblyLines within Action Manager, for example, every 24 hours.
- Provides status information to Server API clients about AssemblyLines that run asynchronously.

The Tombstone Manager API is documented in the Java API documentation; look for class `com.ibm.di.api.Tombstone`.

Configuring Tombstones

Select the required options to configure the tombstone creation. You can also refer to the list of switches in your Config.

The creation of Tombstones for AssemblyLines and Config Instances is configured by means of check boxes in a number of screens in the Configuration Editor (CE), as well as a number of options in the `global.properties` or `solution.properties` files.

Once configured, your Configs contain the following switches:

At the configuration level:

- Config switch: specifies whether tombstones are created or not for the Config Instance itself.
- All AssemblyLines switch: specifies whether tombstones are created for all AssemblyLines from this configuration.

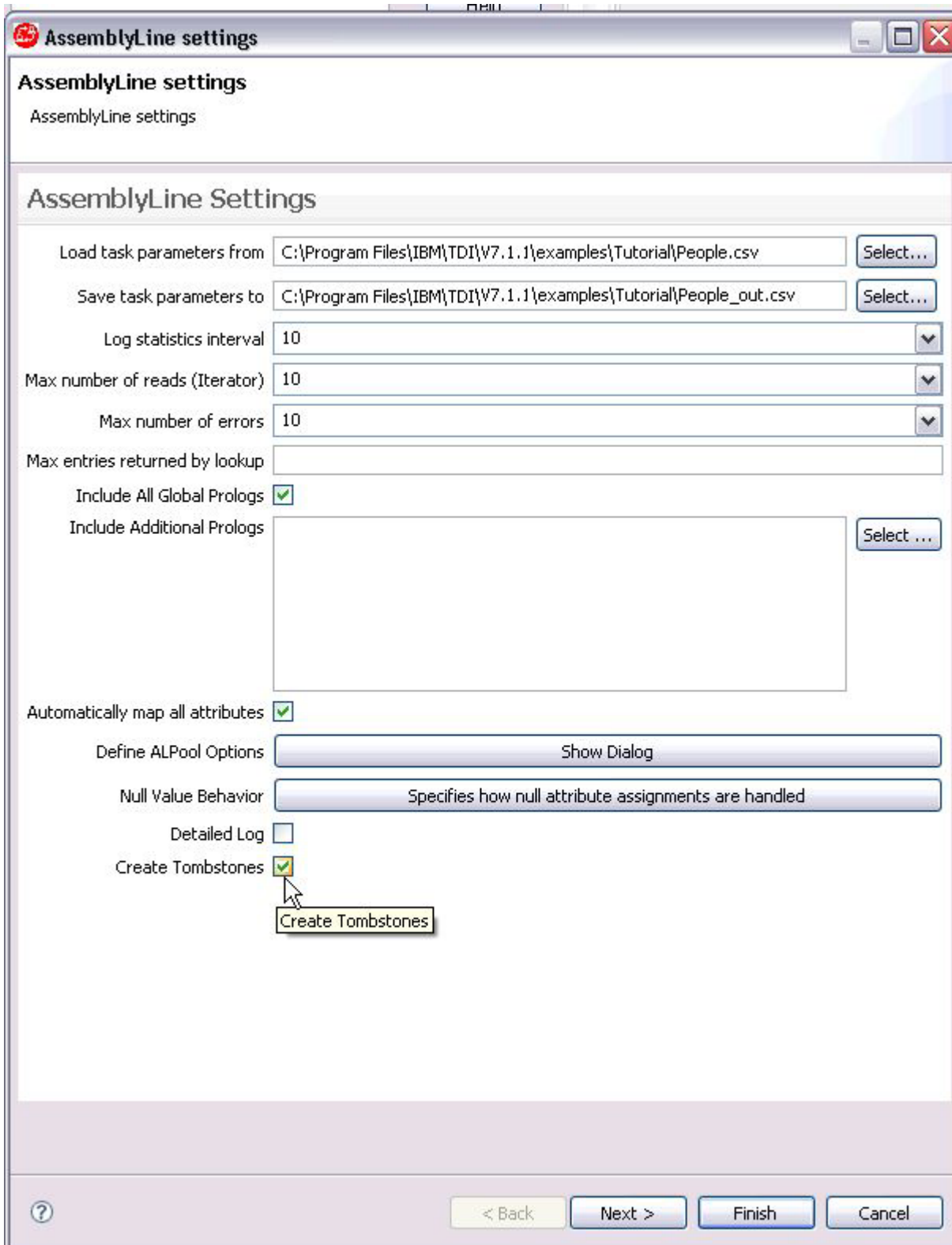
At the AssemblyLine level:

A switch that specifies whether tombstones are created for that particular AssemblyLine. This switch is only taken into account when the "All AssemblyLines switch" at the configuration level is switched off.

Configuration Editor Configuration screen

You can configure tombstones creation for an AssemblyLine using the displayed AssemblyLine Configuration window.

The Create Tombstone option is near the bottom of the window.



Checking **Create Tombstones** cause tombstones to be generated for this AssemblyLine when runs, even when the master switch for AssemblyLines is disabled.

AssemblyLine Configuration screen

Refer to the tombstone record and list of attributes to enable the tombstone.

When AssemblyLine Tombstones are disabled using the Configuration option shown above, tombstone generation can still be enabled individually per AssemblyLine by using the appropriate option in the AssemblyLine configuration screen, **Create Tombstones**.

A sample tombstone record could look like this:

Table 34. Tombstone record

Field Name	Value
Component Type ID	1
Event Type ID	0
StartTime	11.11.2005 11:11:54
TombstoneCreateTime	11.11.2005 17:22:45
Component Name	"ActiveDirectoryChangeLogSynchronizer"
Configuration	"C:\TDI_SOL_DIR\rs.xml"
Exit Code	0
Error Description	""
GUID	"432640786324026346432"
Statistics	[get:571] [add:571] [err:0]

The statistics returned can be one or more of the following attributes:

Table 35. Statistics returned in a Tombstone record

Attribute	Description
add	Total number of entries the AssemblyLine has added (performed by Connectors in AddOnly mode)
mod	Total number of entries the AssemblyLine has modified (performed by Connectors in Update mode)
del	Total number of entries the AssemblyLine has deleted. (performed by Connectors in Delete mode)
get	Total number of entries the AssemblyLine has retrieved (performed by Connectors in Iterator mode)
request	Total number of requests accepted when there is a Server mode Connector in the AssemblyLine
callReply	Total number of Call/Reply operations the AssemblyLine has executed (performed by Connectors in CallReply mode)
err	Total number of errors encountered
skip	Total number of entries the AssemblyLine has skipped entries
lookup	Total number of Lookup operations the AssemblyLine has executed (performed by Connectors in Update/Delete/Lookup mode)
ignore	Total number of entries the AssemblyLine has ignored (performed by Connectors in Update/Delta mode)
reconnect	Total number of times the AssemblyLine has attempted to reconnect to another client

Table 35. Statistics returned in a Tombstone record (continued)

Attribute	Description
exception	The exception text if the AssemblyLine terminated with an exception
getTries	Total number of times the AssemblyLine has attempted to retrieve an entry (performed by Connectors in Iterator mode)
getclientTries	Total number of times the AssemblyLine has attempted to get the next connected client (performed by Connectors in Server mode)
nochange	Total number of entries the AssemblyLine processed but left unchanged
branchtrue	Total number of Branch components executed by the AssemblyLine because their expression evaluated to true
branchfalse	Total number of Branch components skipped by the AssemblyLine because their expression evaluated to false
loopstart	Total number of Loop components executed by the AssemblyLine
loopcycles	Total number of cycles executed for all Loop components that had more than one cycle in an AssemblyLine
reconnectTime	Time in ms after last reconnect was attempted by the AssemblyLine

The Tombstone Manager

You can perform a number of tasks with Tombstone Manager. It refers to values of configuration properties. For more details read the information provided here.

The Tombstone Manager monitor the number of tombstone records at runtime and delete old records as per the values of the `com.ibm.di.tm.autodel.age`, `com.ibm.di.tm.autodel.records.trigger.on`, `com.ibm.di.tm.autodel.records.max` configuration properties.

- The Tombstone Manager tracks Config Instances and AssemblyLines stop events.
- The Tombstone Manager uses the local Server API calls for registering for event notifications and receiving stop events for Config Instances and AssemblyLines.
- The Tombstone Manager uses the IBM Security Directory Integrator System Store for data persistence.
- The Server API (documented in the Java API documentation) interfaces contain calls for querying the Tombstone Manager for various data, like AssemblyLine and Config Instance Tombstones.
- The Tombstone Manager provides options for deleting old tombstone records.

A possible AssemblyLine tombstone lifecycle could look like:

- The Tombstone Manager receives a Server API event that an AssemblyLine has terminated (this assumes the Server API and the Tombstone Manager are turned on and the configuration file specifies that tombstones are created for this AssemblyLine).
- The Tombstone Manager extracts the required data from the Server API event and creates a corresponding database tombstone record in the System Store.
- While the tombstone record is available in the System Store, queries can be executed through the Server API calls that provide all the information contained in the tombstone record.
- The tombstone record is deleted from the System Store either when an explicit cleanup Server API call is executed that deletes it, or when the logic for automatic deletion of old tombstone records collects it. Neither of these events is required, so theoretically, the tombstone record may live forever.

Tombstone Manager

The Tombstone Manager task is configured by means of properties in the `global.properties` or `solution.properties` file for your Config instance.

Note: In order for the Tombstone Manager to function, the Server API must be switched on; that is, the property `api.on` must be set to **true**.

The relevant properties are:

com.ibm.di.tm.on

Master switch for the Tombstone Manager. Values are **on** and **off** - if set to **off**, no Tombstones are generated even if specified in the Config file; neither are they managed (nor can they be queried using the Server API, or AMC).

The default value for this property is **false**.

com.ibm.di.tm.autodel.age

The number of days a Tombstone live. When this property is present and contains an integer value greater than 0 the Tombstone Manager automatically delete all tombstone records that are older than the specified number of days.

The logic for tombstone record deletion is triggered on IBM Security Directory Integrator Server startup and once a day on a long running IBM Security Directory Integrator Server.

The default value for this property is **0**.

com.ibm.di.tm.autodel.records.trigger.on

Specifies the total number of tombstone records that trigger the logic for trimming the number of tombstone records to a certain number.

The default value for this property is **10000**.

com.ibm.di.tm.autodel.records.max

The number of Tombstones to be retained once the trigger specified by the previous parameter, `com.ibm.di.tm.autodel.records.trigger.on` is exceeded.

The default value for this property is **5000**.

com.ibm.di.tm.create.all

This property acts as an override switch for the values specified in the Config files. When this property is set to **true**, Tombstone Manager create Tombstones for every AssemblyLine and Config Instance regardless of the values specified in the configurations. This is useful to turn on Tombstone creation for pre-6.1 configurations that do not have tombstone values without modifying the configurations.

The automatic cleanup logic determined by the `com.ibm.di.tm.autodel.age` property is independent of the automatic cleanup logic determined by the `com.ibm.di.tm.autodel.records.trigger.on` and `com.ibm.di.tm.autodel.records.max` properties.

The Tombstone Manager uses the IBM Security Directory Integrator logging framework and logs its messages in the IBM Security Directory Integrator Server main log.

An example section in the `global.properties` or `solution.properties` file could look like:

```
com.ibm.di.tm.on=true  
com.ibm.di.tm.autodel.age=90  
com.ibm.di.tm.autodel.records.trigger.on=50000  
com.ibm.di.tm.autodel.records.max=25000  
com.ibm.di.tm.create.all=false
```

This set of configuration properties specifies that: The Tombstone Manager is turned on. Tombstones older than 90 days are automatically deleted. Also when the total number of tombstone records reaches 50000, the oldest 25000 tombstone records is automatically deleted.

Chapter 19. Multiple IBM Security Directory Integrator services

IBM Security Directory Integrator services can be registered as different services. Learn more about these in the information provided here.

IBM Security Directory Integrator as Windows Service

You can perform various tasks with Windows Service. Learn in detail about these through the information provided here.

In IBM Security Directory Integrator there is a mechanism that allows multiple IBM Security Directory Integrator server instances to be registered as Windows services. Each instance requires a separate solution directory. After creating a solution directory, a utility program should be copied in it. The name of the program is `ibmdiservice.exe`. The configuration of the utility program and the Windows service is made with a properties file named `ibmdiservice.props`. Each solution directory should contain a configuration properties file.

Each Windows service must have a different name. A property called "servicename" in the property file specifies a name that is used in creation of the Windows service name and the Windows service display name. The Windows service name is formed by prefixing the value of the "servicename" property with the "ibmdisrv-" prefix. The Windows service display name is formed by inserting the value of the "servicename" property between the brackets of "IBM Security Directory Integrator ()". For example if the "servicename" property is set to "test" the Windows service name is "ibmdisrv-test" and the Windows service display name is "IBM Security Directory Integrator (test)". If the "servicename" property is not present or has no value default names are used. The default names for the Windows service name and the Windows service display name are "ibmdisrv" and "IBM Security Directory Integrator".

A property exists so it can be configured whether the Windows service is started automatically on Windows startup or has to be started manually. The name of the property is "autostart" and the valid values for it are "true" and "false".

Note: This property is used during installation and uninstallation as well as while the service is running. That is why the property value must not be changed after the Windows service has been installed.

For more information about the IBM Security Directory Integrator Windows service configuration properties file see the "Configuring the service" on page 333" section.

Installing and uninstalling the service

Perform the steps listed here to install and uninstall the service.

Installing the service

You can install the service using the steps listed here.

About this task

Do the following to install the IBM Security Directory Integrator service:

Procedure

1. Make sure the IBM Security Directory Integrator is installed. The installation folder of the IBM Security Directory Integrator is referred to as *root_directory*. See installer for Windows platforms.
2. Choose a solution folder that is used by IBM Security Directory Integrator when it is started as a Windows service - this can be any folder of your choice. Once IBM Security Directory Integrator is installed as a service the solution folder used by the service cannot be changed until it is uninstalled as a service. Note that choosing the solution folder for the Windows service does not prevent from running IBM Security Directory Integrator with any other solution folder.
3. Once the solution folder is chosen copy into that folder all files from the *root_directory/win32_service* folder: these are "ibmdiservice.exe", "ibmdiservice.props" and "Log4J.properties".
4. Execute the following command from the solution folder chosen for the Windows Service: `ibmdiservice.exe -i`

Uninstalling the service

You can uninstall the service using the steps listed here.

About this task

Note: In order to use the IBM Security Directory Integrator 7.2 version of the "ibmdiservice.exe" utility program any registered pre-IBM Security Directory Integrator 7.2 Windows service must be uninstalled and then the IBM Security Directory Integrator 7.2 windows service must be installed. This is necessary because the IBM Security Directory Integrator 7.2 windows service uses a different default name for the Windows service name – "ibmdisrv" as opposed to the pre-IBM Security Directory Integrator 7.2 default name of "IBM Security Directory Integrator".

Do the following to uninstall the IBM Security Directory Integrator service:

1. Make sure the IBM Security Directory Integrator service is stopped.
2. Execute the following command from the solution folder chosen when you installed the service:

```
ibmdiservice.exe -u
```

Note:

1. Uninstalling the IBM Security Directory Integrator service does not uninstall the IBM Security Directory Integrator itself. You can still use the IBM Security Directory Integrator but it is not registered and run as a Windows service. You can install IBM Security Directory Integrator service again later.
2. If the IBM Security Directory Integrator service is installed and you wish to completely uninstall the IBM Security Directory Integrator (not just the service), do the following:
 - a. Uninstall the Windows service.
 - b. Uninstall the IBM Security Directory Integrator (see uninstaller for Windows platforms).

Starting and stopping the service

Use the listed options to start and stop the service.

The IBM Security Directory Integrator service automatically starts the IBM Security Directory Integrator at system boot. The IBM Security Directory Integrator is not, however, automatically started when the service is installed. After installing the service you have three options to start the service:

- Restart the computer.
- Start the IBM Security Directory Integrator service from the Windows Services window.
- Use the command line. See “Command line support” on page 337

Manual start and stop

You can manually start and stop the IBM Security Directory Integrator service from the Windows Services window.

In the **Services** window you must select the service IBM Security Directory Integrator and, depending on the Windows version, either click the **Start/Stop** button, or right-click on the service name and select **Start/Stop**.

You can also use the command line; see “Command line support” on page 337.

Changing service startup type

By default, the IBM Security Directory Integrator service is configured to start automatically on system boot.

You can manually change the service startup mode from the Windows Services window to **Manual** or **Disabled**.

Logging

The IBM Security Directory Integrator service logs all messages (**error**, **info** and **debug**) in the Application Windows system log. You can view these messages with the Windows Event Viewer.

Configuring the service

You can specify the properties in the `ibmdiservice.props` file to configure the IBM Security Directory Integrator service.

The IBM Security Directory Integrator service is configured through the `ibmdiservice.props` file placed in the solution folder chosen during installation of the log service.

Note: Before running the service, make sure this file is properly configured as described in this section. The service could fail if the file contains incorrect values. The following properties are specified in the `ibmdiservice.props` file:

path Specifies the PATH environment variable used for running the IBM Security Directory Integrator process (this property is usually the same as the PATH variable from `ibmdisrv.bat`, but you can change it). This is an optional property.

ibmdiroot

Specifies the root folder of the IBM Security Directory Integrator (for example, C:\Program Files\IBM\TDI\V7.2). This is a required property.

configfile

Specifies the file path to the IBM Security Directory Integrator configuration file. This is an optional property.

assemblylines

Specifies in a comma-delimited format the AssemblyLines that are started automatically when the IBM Security Directory Integrator service is started. This is an optional property.

cmdoptions

Specifies other options that are directly passed to the IBM Security Directory Integrator on service startup (see Chapter 13, "Command-line options," on page 205 for the full list of IBM Security Directory Integrator options).

One such option could be the **-c** option; here you could specify multiple config files (separated by commas), something which is not allowed by the **configfile** parameter.

The requirements when you use this configuration are:

- The full path is required in either \\ for Windows or / for UNIX syntax for each assembly line
- The names of the configuration files must be contained in one set of quotes separated by a comma.
- The **-d** option is required.
- The file names must not contain any spaces.

For example:

```
cmdoptions=-c"C:/TDI7.1-Solutions/myConfig/Config1.xml",  
C:/TDI7.1-Solutions/AnotherConfig/TechNotes.xml" -d
```

This is an optional property.

servicename

Specifies a name that is used to form the Windows service name and the Windows service display name. The windows service name is set to the value of the **servicename** property prefixed with the "ibmdisrv-" prefix. The windows service display name is created by inserting the value of the **servicename** property between the brackets of the "IBM Security Directory Integrator ()" expression.

For example, if the property value is "test" the Windows service name will be "ibmdisrv-test" and the Windows service display name will be "IBM Security Directory Integrator (test)". If the **servicename** property is not present or has no value, default names are used. The default Windows service name is "ibmdisrv" and the default Windows service display name is "IBM Security Directory Integrator".

Note: This property is used during installation and uninstallation as well as while the service is running. That is why the property value must not be changed after the Windows service has been installed.

autostart

Specifies whether the Windows service starts automatically on Windows start-up or whether it has to be started manually. The valid values for this property are **true** and **false**. A value of **true** specifies that the Windows

service is started on Windows start-up and a value of **false** specifies that the service has to be started manually. If this property is not present or has no value, then the default value of **true** is used.

This property is used during Windows service installation and changing it after the Windows service has been installed has no effect.

controlledshutdown

Specifies whether the Windows service will terminate the server gracefully or will hard kill the server process. The valid values for this property are "true" and "false". A value of "true" specifies that the Windows service will stop the IBM Security Directory Integrator server gracefully and a value of "false" indicates that the server process will be hard killed. If this property is not present or has no value, then the default value of "false" is used.

debug Specifies **true** or **false** to correspondingly turn debug information on or off. When debug information is turned on, detailed trace messages are dumped in the Application Windows system log. This is an optional property.

Note: When specifying properties in the configuration file, specify each property on a single line and use the following format:

```
<property_name>=<property_value>
```

There must be no spaces around the equals (=) sign.

An example of a completed `ibmdiservice.props` file looks like the following:

```
path=C:\Program Files\IBM\TDI\V7.2\jvm\jre\bin;  
C:\Program Files\IBM\TDI\V7.2\libs;  
ibmdiroot=C:\Program Files\IBM\TDI\V7.2  
configfile=rs.xml  
assemblylines=AssemblyLine1,AssemblyLine2  
cmdoptions=  
debug=false  
controlledshutdown=false
```

Note: If you change any of the properties in `ibmdiservice.props`, you must restart the service for the changes to take effect.

IBM Security Directory Integrator as Linux/UNIX Service

You can perform various tasks with Linux/UNIX Service. Learn in detail about these through the information provided here.

Deployment methods

On Linux and UNIX platforms, there are two different ways of ensuring that certain system jobs or 'daemons' start and stop at respectively system initiation and system termination:

1. Using a script in `/etc/init.d` containing the logic to start and stop the daemons you are interested in. This script you then (hard)link to scripts in `/etc/rc3.d`: their names beginning with `SXX...` and `KXX...` - the `XX` being a numeral which causing the files to show up in the right sequence in the `/etc/rc3.d` directory. The scripts starting with `S` are called when the system reaches run phase 3 at system startup, and the scripts starting with `K` are called when the system terminates.
2. By editing the `/etc/inittab` file.

The latter process is what we describe here. Some of the information presented could be used to construct scripts using the first deployment method.

Tailoring /etc/inittab

In order to start up IBM Security Directory Integrator daemon processes when the UNIX/Linux OS starts appropriate entries must be added to the */etc/inittab* file. The registering of IBM Security Directory Integrator as a windows service on Windows translates to adding a line of text to the */etc/inittab* file on UNIX/Linux. The un-installation of the IBM Security Directory Integrator windows service on Windows translates to removing the corresponding entries from the */etc/inittab* file. For each IBM Security Directory Integrator daemon process that needs to be started on system startup one line of text must be added to the */etc/inittab* file. The format and meaning of the entries in this file is described below. Each entry in the */etc/inittab* file has the following format:

Identifier:RunLevel:Action:Command

A description of each of these fields is as follows:

- The **Identifier** field is a string (at least a single character in length) that uniquely identifies an object. This string is used to uniquely identify the corresponding command.
- The **RunLevel** field is the run-level in which this entry can be processed. Run-levels effectively correspond to a configuration of processes in the system. Each process started by the init command is assigned one or more run-levels in which it can exist. A run-level is represented by the numbers 0 through N, where N is a positive integer different for the different UNIX/Linux operating Systems (for example on some AIX computers N is 9, on RedHat Linux N is 6, and so on.). If the OS is running in run-level 3, for example, then only processes specified for run-level 3 are started.

The **RunLevel** field can define multiple run-levels for a process by selecting more than one run-level in any combination from 0 through N. For example, if IBM Security Directory Integrator needs to run in run-level 3 and 6, then the run-level must be specified as "36". If no run-level is specified, the process is assumed to be valid at all run-levels.

It is recommended that no run-level numbers are specified, unless the specific IBM Security Directory Integrator solution specifically needs to.

- The **Action** field is a value from a set of predefined actions which tells the **init** command how to treat the process specified in the **Command** field. There are many actions recognized by the init command, but for running the IBM Security Directory Integrator server as a daemon process it is recommended that the **once** action be used. The semantics of the **once** action are:

When the init command enters a run-level that matches the entry's run level, start the process, and do not wait for its termination. When it dies, do not restart the process. When the system enters a new run level, and the process is still running from a previous run level change, the program not be restarted. All subsequent reads of the */etc/inittab* file while the init command is in the same run level cause the init command to ignore this entry.

- The **Command** field specifies the shell command to run.

Here are three example IBM Security Directory Integrator-related entries in */etc/inittab*:

```
tdi1::once:/opt/IBM/TDI711_1/ibmdisrv -c "/opt/IBM/TDI711_1/myconfigs/rs1.xml" -r "testAL1"  
tdi2::once:/opt/IBM/TDI711_2/ibmdisrv -c "/opt/IBM/TDI711_2/myconfigs/rs2.xml" -r "testAL2"  
tdi3::once:/opt/IBM/TDI711_3/ibmdisrv -c "/opt/IBM/TDI711_3/myconfigs/rs3.xml" -r "testAL3"
```

This example starts three IBM Security Directory Integrator server instances which are installed in different folders.

Note: There are some differences in the different UNIX/Linux operating systems for system startup. That is why the information provided here covers the main issues of starting IBM Security Directory Integrator on a UNIX/Linux system and does not refer to any specific UNIX/Linux system.

As an example of an `/etc/inittab` file, detailed information about the `/etc/inittab` configuration file for an AIX system can be found at <http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.files/doc/aixfiles/inittab.htm>

Graceful shutdown

On UNIX systems we always perform a graceful shutdown of the service. This is achieved by a shutdown hook added to the Java Runtime. That way, when the server is stopped with either SIGINT or SIGTERM signals, this hook is executed and the server is gracefully terminated.

Another advantage of this approach is that this hook will be invoked on all platforms when the server is stopped by pressing CTRL+C in its console window.

Note: You can also specify an external program to be started from within the JVM shutdown Hook. This external program is configured using an optional property in the `global.properties` or `solution.properties` file: `jvm.shutdown.hook`. If configured the external program will be started right after the server has shutdown gracefully.

Command line support

You can use the script for starting and stopping the service with the help of information provided here.

IBM Security Directory Integrator provides a script for starting and stopping the IBM Security Directory Integrator service on Windows and UNIX systems. The script is located in the `TDI_install_dir/bin` directory, and is named `servicemgr.bat(sh)`.

The usage of the script is as follows:

```
servicemgr service_name start|stop
```

where:

service_name

is the name of the service. For Windows systems this is `ibmdisrv` by default or the value of `servicename` property in the `ibmdiservice.props` file. For UNIX systems this is the identifier field from the `/etc/inittab` file.

start | stop

the desired command to perform on the service.

Appendix A. Example Property files

You can customize the property files for installing the IBM SDI. Learn more about the various available text files and the Solution Directory through the information provided here.

An installation of IBM Security Directory Integrator is to a large extent customized by means of a set of text files containing one of more **properties**, usually in the form of a keyword or identifier followed by a value. The following global property text files can be found at the root/etc level of the IBM Security Directory Integrator installation directory:

- “Log4J.properties” on page 340
- “jlog.properties” on page 341
- “derby.properties” on page 342
- “global.properties” on page 342

Properties set in any of those files form a baseline for the entire IBM Security Directory Integrator installation for all users on that computer. However, if your Solution Directory is different from the installation directory, you can have a set of text files in your Solutions Directory that mirror their counterparts in the installation directory. A property listed in any of those files overrides anything set in any of the global installation property files mentioned above. Furthermore, a Java property set inside a Config file takes the highest precedence, and overrides anything in a global property file or the property files in the Solution Directory.

You can specify the Solution Directory in multiple ways:

- By setting the environment variable *TDI_SOLDIR* before starting the Configuration Editor or the Server
- By specifying the *-s* parameter to the *ibmditk* script to start the Server. This takes precedence over setting *TDI_SOLDIR*.

If *TDI_SOLDIR* equals the installation directory, the behavior is like in older versions of IBM Security Directory Integrator: all property files are read from there, and the remarks about property files in the Solutions Directory do not apply.

In any other case, the first time you run the IBM Security Directory Integrator Server, it makes a copy of all the property files into your Solutions Directory (it does not overwrite these files if they already exist). You can now tailor these files to your particular needs, without affecting the property files in the installation directory. The files remaining in the installation directory continue to form a baseline configuration for other instances of IBM Security Directory Integrator.

Note: The file *global.properties* is copied to a file called *solutions.properties* in your Solutions Directory. Other files, like *Log4J.properties* and the files in the *amc* and *serverapi* folders are copied under their own name.

In addition, if your Solution Directory was setup during product installation using the IBM Security Directory Integrator installer, the setup will contain a working System Queue setup. If the Solution Directory is created by any other means (manually, or by the Server by using the *-s* option) then you will either have to

disable the System Queue in your solution.properties file, or setup a System Queue yourself - see "System Queue Configuration" on page 157.

Log4J.properties

This file sets a baseline for the log-strategy for the server (ibmdisrv).

Log options configured in the Logging tab in the Configuration Editor are written into the Config file, and are supplementary to or supersede the following:

```
# This file controls the logging strategy for the server (ibmdisrv) when started
# from the command line.
# Look at executetask.properties for the logging strategy of the server when started
# from the Configuration Editor (ibmditk).
# Look at ce-log4j.properties for the logging behavior of the Configuration Editor (ibmditk).
#
# You will normally configure the logging strategy of the server by adding appenders
# using the Configuration Editor (ibmditk). This file only defines the baseline
# that is independent of the configuration files you are using.
#
# See the IDI documentation for more information on the contents of this file.
#

log4j.rootCategory=INFO, Default

# This is the default logger, you will see that it logs to ibmdi.log
log4j.appender.Default=org.apache.log4j.FileAppender
log4j.appender.Default.file=logs/ibmdi.log
log4j.appender.Default.layout=org.apache.log4j.PatternLayout
log4j.appender.Default.layout.ConversionPattern=%d{ISO8601} %-5p [%c] - %m%n
log4j.appender.Default.append=false

#Example settings for changing the default logger

#####ROLLING FILE SIZE APPENDER
##RollingFileAppender rolls over log files when they reach a certain size specified by the
##MaxFileSize parameter

#log4j.appender.Default=org.apache.log4j.RollingFileAppender
#log4j.appender.Default.File=logs/ibmdi.log
#log4j.appender.Default.Append=true
#log4j.appender.Default.MaxFileSize=10MB
#log4j.appender.Default.MaxBackupIndex=10
#log4j.appender.Default.layout=org.apache.log4j.PatternLayout
#log4j.appender.Default.layout.ConversionPattern=%d{ISO8601} %-5p [%c] - %m%n

#####DAILY OUTPUT LOG4J SETTINGS
## With the DailyRollingFileAppender the underlying file is rolled over at a user chosen frequency.
##The rolling schedule is specified by the DatePattern option

#log4j.appender.Default=org.apache.log4j.DailyRollingFileAppender
#log4j.appender.Default.file=logs/ibmdi.log
#log4j.appender.Default.DatePattern='.'yyyy-MM-dd
#log4j.appender.Default.layout=org.apache.log4j.PatternLayout
#log4j.appender.Default.layout.ConversionPattern=%d{ISO8601} %-5p [%c] - %m%n

# You may change the logging category of these subsystems to DEBUG
# if you want to investigate particular problems. This may
# generate a lot of output.
# ...com.ibm.di.config describes the loading of the configuration file (.xml),
# and how the internal configuration structure is built.
# ...com.ibm.di.loader gives information about jar files, and where classes are found.
# It also loads idi.inf files, which provides Connectors/Parsers/EH information
# for the Configuration Editor.
log4j.logger.com.ibm.di.config=WARN
log4j.logger.com.ibm.di.loader=WARN

# Uncomment the lines below to activate them

# Here is an example on how to make a logger that logs to the console
#log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
#log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
#log4j.appender.CONSOLE.layout.ConversionPattern=%d [%t] %-5p - %m%n0

# Here is an example that logs to myFile.log
#log4j.appender.fileLOG=org.apache.log4j.FileAppender
#log4j.appender.fileLOG.file=myFILE.log
#log4j.appender.fileLOG.layout=org.apache.log4j.PatternLayout
#log4j.appender.fileLOG.layout.ConversionPattern=%d{ISO8601} %-5p [%c] - %m%n
#log4j.appender.fileLOG.append=false

# Finally, make use of the loggers defined above:
# Tell AssemblyLines myAL to log using CONSOLE logger defined above.
```

```
# log4j.logger.AssemblyLine.AssemblyLines/myAL=INFO, CONSOLE
# Or you could log to myFile.log
# log4j.logger.AssemblyLine.AssemblyLines/myAL=INFO, fileLOG
```

jlog.properties

You can configure and modify the JLOG-based values of the IBM SDI server.

This file configures the JLOG-based Chapter 15, “Tracing and FFDC,” on page 233 of the IBM Security Directory Integrator server. These values can be modified dynamically (during Server execution) using the LogCmd script if the property `jlog.noLogCmd` was set to `false` when the Server started.

Note: You would normally use Log4J to trace execution flow in your solution; the JLOG-based tracing and FFDC is meant to aid IBM Support should you have problems with IBM Security Directory Integrator.

```
#####
# This file controls the tracing and First Failure Data Capture (FFDC) strategy for ITDI 7.2
# See the IDI documentation for more information on the contents of this file.
#####
#-----
# Enable the JLOG's command server
#
# If the jlog.noLogCmd is set to false, then the JLOG LogManager will listen on the
# default port (9992) for JLOG log commands.
# Setting this property to false will enable you to modify the JLOG properties dynamically using the
# logcmd scripts. The logcmd scripts are placed under ITDI_HOME directory.
# The default value is set to true.
#-----
jlog.noLogCmd=true

#-----
# Set listen port for JLOG's command server
#
# If you want LogManager to listen on different port than the default one (9992) you should
# uncomment the property jlog.logCmdPort and set it to the desired port. If not uncommented
# the LogManager will listen on the default port - 9992.
#-----
#jlog.logCmdPort=9992

#-----
# Configure Jlog FileHandler for tracing into a file.
#
# By default the FileHandler is not attached to the Jlog Logger.
# Uncomment the properties with the prefix jlog.filehandler below to configure a FileHandler.
# After uncommenting this you need to add the filehandler to the logger's listeners names as shown
# below
# e.g: jlog.logger.listenerNames=jlog.snapmemory jlog.snaphandler jlog.filehandler
#-----
jlog.filehandler.className=com.ibm.log.FileHandler
#jlog.filehandler.description=JLOG File Handler for Logging and Tracing
#jlog.filehandler.encoding=UTF8
#jlog.filehandler.maxFiles=10
#jlog.filehandler.maxFileSize=2048
#jlog.filehandler.appending=true
#jlog.filehandler.fileDir=logs/
#jlog.filehandler.trace.fileName=trace.log
#-----

#-----
# create a level filter.
# The level filter is used to define the level at which JFFDC action will be triggered.
# For JFFDC to be meaningful this should be set to either FATAL or ERROR (case-insensitive).
# NOTE: Setting the trigger level to other levels such as DEBUG_MIN will trigger unwanted JFFDC
# action causing a performance drop.
#-----
jlog.levelflt.className=com.ibm.log.LevelFilter
jlog.levelflt.level=FATAL

#-----
# Configure the SnapMemoryHandler for tracing into a memory buffer.
# The SnapMemoryHandler traces into a memory buffer and dumps the contents of the memory to a file on
# trigger of a event (as defined by the level filter above) and writes the content to the specified
# file
# Properties:
# jlog.snapmemory.queueCapacity : Sets the nnumber of LogEvents that can be buffered in the memory
# jlog.snapmemory.snapFile : name of the file to which the contents of the memory will be dumped
```

```

# jlog.snapmemory.baseDir : The directory where the snapFile is placed.
#   daily subdirectories will be created under this base directory, as:
#   [baseDir]/[YYYY-MM-DD]/
#   Note: MS-DOS style path names need to be escaped with backslashes
#   eg: c:\\CTGI\\FFDC
# jlog.snapmemory.userSnapFile : The name of the file to which the user initiated (from logcmd) dumps
#   will be written to.
# jlog.snapmemory.userSnapDir : The directory where the userSnapfile is placed.
# jlog.snapmemory.msgIds : The list of TMS IDs
# jlog.snapmemory.msgIDRepeatTime : The minimum time, in milliseconds, after passing a log event with a
#   given TMS message id, before another log event with the same id can
#   be passed.
#-----
jlog.snapmemory.className=com.tivoli.log.SnapMemoryHandler
jlog.snapmemory.description=Memory handler used to trace to memory
jlog.snapmemory.queueCapacity=10000
jlog.snapmemory.dumpEvents=true
jlog.snapmemory.snapFile=trace.log
jlog.snapmemory.baseDir=CTGDI/FFDC/
jlog.snapmemory.userSnapFile=userTrace.log
jlog.snapmemory.userSnapDir=CTGDI/FFDC/user/
jlog.snapmemory.triggerFilter=jlog.levelflt
jlog.snapmemory.msgIds=*E
jlog.snapmemory.msgIDRepeatTime=10000

#-----
# Configure the JLogSnapHandler taking a snapshot of the SnapMemoryHandlers buffer
# The JLogSnapHandler takes a snapshot of the associated SnapMemoryBuffer.
#-----
jlog.snaphandler.className=com.tivoli.log.JLogSnapHandler
jlog.snaphandler.description=snaphandler to dump the memory trace
jlog.snaphandler.baseDir=CTGDI/FFDC/
jlog.snaphandler.snapMemoryHandler=jlog.snapmemory
jlog.snaphandler.triggerFilter=jlog.levelflt

#-----
# Configure the PDLogger (Problem Determination) Object and attach the Listeners to it.
# jlog.logger.level can be FATAL | ERROR | WARNING | INFO | DEBUG_MIN | DEBUG_MID | DEBUG_MAX
# The heirarchy of the log levels is from the most severe (FATAL) to the least severe (DEBUG_MAX)
# The value for this property is case-insensitive
#-----
jlog.logger.level=FATAL
#jlog.logger.listenerNames=jlog.snapmemory jlog.snaphandler
jlog.logger.listenerNames=jlog.filehandler.trace
jlog.logger.className=com.ibm.log.PDLogger

#-----
# Configure the PDLogger for the Config Editor and attach the Listeners to it.
# By default, no listeners are attached
#-----
jlog.logger.config-editor.level=FATAL
jlog.logger.config-editor.listenerNames=

```

derby.properties

This file contains some defaults for Derby in networked mode.

Most IBM Security Directory Integrator-related Derby parameters are not maintained here but in `global.properties` and `solution.properties`. More information about these parameters can be obtained from the Derby documentation.

```

# This is a sample properties file provided to show the proper format.
# We're also setting one property which make sure that
# Derby adds to the error log instead of overwriting it.
# This mode is useful for development.
derby.drda.logConnections=true
derby.drda.maxThreads=0
derby.drda.portNumber=1527
derby.drda.traceAll=true
derby.drda.timeSlice=0
derby.drda.traceDirectory=/trace

```

global.properties

This file is read by `ibmditk` (the CE) and `ibmdisrv` (the server) on startup.

This file is read and applied before a file called `solution.properties` from your Solution Directory is read and applied.

Note:

The rendition here, due to extremely long line lengths, may not be complete. Refer to an actual `global.properties` file instead.

```
##
## This file is read by ibmditk/ibmdisrv on startup
##
## Enter <name>=<value> to set system properties.
## Enter !include <file | url> to include other files
##

com.ibm.di.securityTransformation=DES/ECB/NoPadding

##
## Modify the line below to add your own jar/zip files.
## The property may specify several directories or jar files, separated by the Java Property "path.separator",
## which is ":" on Linux and ";" on Windows
## Directories will be searched recursively by the TDI Loader for jar files containing classes and resources.
## Only files with a ".zip" or ".jar" extension are searched.
## com.ibm.di.loader.userjars=c:\myjars

##
## Modify the line below to enable the config autoload feature.
## When this property is defined, the "ibmdisrv -d" command
## line will look for *.xml files in the directory specified by this property and start each one.
##
# com.ibm.di.server.autoload=autoload.tdi

##
## SYSTEM STORE
##

## Location of the database (embedded mode) - Cloudscape 10
#com.ibm.di.store.database=TDISysStore
#com.ibm.di.store.jdbc.driver=org.apache.derby.jdbc.EmbeddedDriver
#com.ibm.di.store.jdbc.urlprefix=jdbc:derby:
#com.ibm.di.store.jdbc.user=APP
#{protect}-com.ibm.di.store.jdbc.password=APP

## Location of the database to connect (networked mode) - Cloudscape 10 - DerbyClient driver
## The macro $soldir$ will be replaced by the value of the actual Solution Directory
com.ibm.di.store.database=jdbc:derby://localhost:1527/$soldir$/TDISysStore;create=true
com.ibm.di.store.jdbc.driver=org.apache.derby.jdbc.ClientDriver
com.ibm.di.store.jdbc.urlprefix=jdbc:derby://localhost:1527/
com.ibm.di.store.jdbc.user=APP
[protect]-com.ibm.di.store.jdbc.password={encr}n+Vum7tNOZU0KNp7AGy7pkAZqiJMGgPnqwg
/dhBLEL5pDBj5FY/Qp/20mOkfWdezdSvYUGkag3UkzV+NuSSBVpJ36s3QCKFDz72VOTzJla1REHpj/j/u9
/3E11ZPIAH1B1gKP77200FPJIB6mbDUUgFwIZ+FmKFH5CW6NytP+M=

#
## Derby (Cloudscape) properties required for enabling authentication
#
derby.drda.startNetworkServer=true
derby.connection.requireAuthentication=true
derby.authentication.provider=BUILTIN
derby.database.defaultConnectionMode=fullAccess

#
## Details for starting Cloudscape in network mode.
## Note: If the com.ibm.di.store.hostname is set to localhost then remote connections will not be allowed.
## If it is set to the IP address of the local machine - then remote clients can access this Cloudscape
## instance by mentioning the IP address. The network server can only be started for the local machine.
#
#com.ibm.di.store.start.mode=automatic
com.ibm.di.store.hostname=localhost
com.ibm.di.store.port=1527
com.ibm.di.store.sysibm=true

# the varchar(length) for the ID columns used in system store and pes connector tables
com.ibm.di.store.varchar.length=512

## create statements for system store tables (Cloudscape 5.1)
#com.ibm.di.store.create.delta.systable=CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, SEQUENCEID int, VERSION int)
#com.ibm.di.store.create.delta.store=CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, SEQUENCEID int, ENTRY long varbinary )
#com.ibm.di.store.create.property.store=CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, ENTRY long varbinary )
#com.ibm.di.store.create.sandbox.store=CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, ENTRY long varbinary )

## create statements for system store tables (Cloudscape 10)
com.ibm.di.store.create.delta.systable=CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, SEQUENCEID int, VERSION int);
ALTER TABLE {0} ADD CONSTRAINT IDI_CS_{UNIQUE} PRIMARY KEY (ID)
com.ibm.di.store.create.delta.store=CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, SEQUENCEID int, ENTRY BLOB );
ALTER TABLE {0} ADD CONSTRAINT IDI_DS_{UNIQUE} Primary Key (ID)
com.ibm.di.store.create.property.store=CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, ENTRY BLOB );
ALTER TABLE {0} ADD CONSTRAINT IDI_PS_{UNIQUE} Primary Key (ID)
com.ibm.di.store.create.sandbox.store=CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, ENTRY BLOB )
com.ibm.di.store.create.recal.conops=CREATE TABLE {0} (METHOD varchar(VARCHAR_LENGTH), RESULT BLOB, ERROR BLOB)

## create statements for system store tables DB2 on z/OS
#com.ibm.di.store.create.delta.systable=CREATE TABLESPACE TS1DSYS LOCKSIZE ROW BUFFERPOOL BP32K;CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, SEQUENCEID int, VERSION int) IN TS1DSYS;CREATE UNIQUE INDEX DSTX1 ON {0}
(ID ASC);ALTER TABLE {0} ADD CONSTRAINT IDI_DT_{UNIQUE} PRIMARY KEY (ID)
#com.ibm.di.store.create.delta.store=CREATE TABLESPACE TS1DST LOCKSIZE ROW BUFFERPOOL BP32K;CREATE TABLE {0}
```

```

(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, SEQUENCEID int, ENTRY BLOB) IN TS1DST; CREATE UNIQUE INDEX DSIX1 ON {0}
(ID ASC); ALTER TABLE {0} ADD CONSTRAINT IDI_DS_{UNIQUE} Primary Key (ID);CREATE LOB TABLESPACE DSENT11 BUFFERPOOL
BP32K LOCKSIZE LOB;CREATE AUX TABLE TBDSEN1 IN DSENT11 STORES {0} COLUMN ENTRY;CREATE INDEX IXEN1 ON TBDSEN1
#com.ibm.di.store.create.property.store=CREATE TABLESPACE PS3DST LOCKSIZE ROW BUFFERPOOL BP32K;CREATE TABLE {0}
(ID VARCHAR(VARCHAR_LENGTH) NOT NULL, ENTRY BLOB) IN PS3DST;CREATE UNIQUE INDEX PSIX3 ON {0} (ID ASC);
ALTER TABLE {0} ADD CONSTRAINT IDI_PS_{UNIQUE} Primary Key (ID);CREATE LOB TABLESPACE PSENT31 BUFFERPOOL BP32K
LOCKSIZE LOB;CREATE AUX TABLE TBPSEN3 IN PSENT31 STORES {0} COLUMN ENTRY;CREATE INDEX PSIXEN3 ON TBPSEN3
#com.ibm.di.store.create.sandbox.store=CREATE TABLE {0} (ID VARCHAR(VARCHAR_LENGTH) NOT NULL, ENTRY BLOB)
#com.ibm.di.store.create.recal.conops=CREATE TABLESPACE IM{UNIQUE} LOCKSIZE ROW BUFFERPOOL BP32K;CREATE TABLE {0}
(METHOD VARCHAR(VARCHAR_LENGTH), RESULT BLOB, ERROR BLOB) IN IM{UNIQUE};CREATE LOB TABLESPACE LB{UNIQUE}
BUFFERPOOL BP32K LOCKSIZE LOB;CREATE AUX TABLE AT{UNIQUE} IN LB{UNIQUE} STORES {0} COLUMN RESULT;CREATE INDEX IX
{UNIQUE} ON AT{UNIQUE};CREATE LOB TABLESPACE LS{UNIQUE} BUFFERPOOL BP32K LOCKSIZE LOB;CREATE AUX TABLE
AE{UNIQUE} IN LS{UNIQUE} STORES {0} COLUMN ERROR;CREATE INDEX IN{UNIQUE} ON AE{UNIQUE}

# Set a customized SQL statement for creation of the Tombstone Manager table.
# Keep the same table and field names. This is the default Derby statement.
#com.ibm.di.store.create.tombstones=CREATE TABLE IDI_TOMBSTONE ( ID INT GENERATED ALWAYS AS IDENTITY,
COMPONENT_TYPE_ID INT, EVENT_TYPE_ID INT, START_TIME TIMESTAMP, CREATED_ON TIMESTAMP, COMPONENT_NAME
VARCHAR(1024), CONFIGURATION VARCHAR(1024), EXIT_CODE INT, ERROR_DESCR VARCHAR(1024),
STATS LONG VARCHAR FOR BIT DATA, GUID VARCHAR(1024) NOT NULL, USER_MESSAGE VARCHAR(1024), UNIQUE (ID, GUID))

# The following two SQL statements could be used when SolidDB is used as System Store
#com.ibm.di.store.create.tombstones=CREATE TABLE IDI_TOMBSTONE (ID INT PRIMARY KEY, COMPONENT_TYPE_ID INT,
EVENT_TYPE_ID INT, START_TIME TIMESTAMP, CREATED_ON TIMESTAMP, COMPONENT_NAME VARCHAR(1024),
CONFIGURATION VARCHAR(1024), EXIT_CODE INT, ERROR_DESCR VARCHAR(1024), STATS LONG VARBINARY,
GUID VARCHAR(1024) NOT NULL, USER_MESSAGE VARCHAR(1024), UNIQUE (ID, GUID));CREATE SEQUENCE IDI_TOMBSTONE_SEQ
#com.ibm.di.store.update.tombstones=INSERT INTO IDI_TOMBSTONE (ID, COMPONENT_TYPE_ID, EVENT_TYPE_ID, START_TIME,
CREATED_ON, COMPONENT_NAME, CONFIGURATION, EXIT_CODE, ERROR_DESCR, STATS, GUID, USER_MESSAGE)
VALUES (IDI_TOMBSTONE_SEQ.NEXT, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

# the ibmsnap_commitseq column name used by the RDBMS changelog connector
com.ibm.di.conn.rdbmschlog.cdcolname=ibmsnap_commitseq

## server authentication
javax.net.ssl.trustStore=serverapi/testadmin.jks
(protect)-javax.net.ssl.trustStorePassword={encr}rI2mxgg5vvrbrnooxTCYUFeMSkCKHb14cGpZuQUW20GeCAYJjqGhiSJLuQJdNqPnNqDNI+
isW0mJkzwl0qud8l179x1JtwqLh7zEVfGvuqEwx43ACLoSb9gkG8Je07Jf0Fgp09thj6zCczqB4NCsmUv1lagBHvGtEE8Q73Xh8=
javax.net.ssl.trustStoreType=jks

## client authentication
javax.net.ssl.keyStore=serverapi/testadmin.jks
(protect)-javax.net.ssl.keyStorePassword={encr}QnVT+6gn66nE3zUnvHQE9PTM8k52BQxJsae85mmwQkechSeScysd0MrWltxCOMF2URJoZL
cqlVp3WpGp0QX+n9XQVBXG0XmIjhwIs7iDr73DpkM1oJzRruYhKPQK0xgfC+801IAfSoMxNgp761YbqIRqJHKV3sQT9VB0mM=
javax.net.ssl.keyStoreType=jks

##PKCS11 options
##Set the value of following properties to use PKCS11 enabled devices to store SDI servers private key / certificate.
com.ibm.di.pkcs11cfg=etc/pkcs11.cfg
com.ibm.di.server.pkcs11=false
com.ibm.di.server.pkcs11.library=
com.ibm.di.server.pkcs11.slot=
(protect)-com.ibm.di.server.pkcs11.password={encr}iYbgH/Y/pw4YSUXVqKQnWz1ZHka52CuCRmHnnBen3/yIt0J1K+nrepEW8jN2KBDHM
82+zIPs0YlIx9y20X/nvp/QqevgPsAvzvxzn07QtwjQAYSTKJt+i1BVCbnkhJ0iRXXrIQkwsXtyl/oUcdskk5+mNSYLttvSuR03J38=

## Turns on java debug
# javax.net.debug=true

## java interpreter override
# com.ibm.di.javacmd=
# com.ibm.di.installDir=

## Limits the number of threads IDI uses
## Must be set higher than 3 to have any effect

# com.ibm.di.server.maxThreadsRunning=500

com.ibm.di.server.securemode=false

## Following properties modified in SDI 7.1 Added property for
## keystore password and keypassword
## com.ibm.di.server.keystore
## com.ibm.di.server.key.alias

api.keystore=testserver.jks
api.keystore.type=jks
api.key.alias=server
(protect)-api.keystore.password={encr}S8BAYY1dW1y0FHuslpXHN6F4Xc76gCUvm39ZKZHSjRMmZQmY7S1p+7gh9YHE3AIquLgGf4n0MzFyBU
S/C6xy2RIjVbqHGz2YU+4SNnGjt5o53KAXIegLC2ml+JCUu4UY/P/ASjCXfHFL/iJsRI4hLHfG266p13eIJeVxf1c=
(protect)-api.key.password={encr}

## Encryption properties added in SDI 7.1
com.ibm.di.server.encryption.keystore = testserver.jks
com.ibm.di.server.encryption.key.alias = server
com.ibm.di.server.encryption.keystoretype = jks
com.ibm.di.server.encryption.transformation = RSA

## Web container
web.server.port=1098
web.server.ssl.on=true
web.server.ssl.client.auth.on=false
# web.server.session.timeout=300

## Touchpoint Server properties
tp.server.on=false
tp.server.config=etc/tp.xml
tp.server.auth=false
tp.server.auth.realm=Security Directory Integrator Touchpoint Server

## Dashboard properties
##
dashboard.on=true
dashboard.templates.folder=dashboard/templates

## Dashboard authentication properties
##

```

```

## The values for localhost and remotehost can be:
## none: No authentication is required
## deny: All connections denied
## ldap: Authentication is done by logging into an LDAP server and optionally validating group membership
## properties: Authentication is done using dashboard.auth.user.[username]=[password] properties
##
## dashboard.ldap.url
## Specify the LDAP host port and optionally a search base (ldap://<host>:<port>/<search-base>)
##
## dashboard.ldap.url.group
## Specify the LDAP host port and optionally a search base (ldap://<host>:<port>/<search-base>)
##
dashboard.auth=true
dashboard.auth.localhost=properties
dashboard.auth.remote=deny
# dashboard.auth.ldap.url=ldap://localhost:389/ou=users,ou=system
# dashboard.auth.ldap.url.group=ldap://localhost:389/cn=group1,ou=groups,ou=system
#
# Default FDS username/password
(protect)-dashboard.auth.user.admin={encr}SzFT+3+aSNWrwtySrBcCbh1Vp4bB4kKqSJuJGuRSSwtN69b1f/U1PbRQbWmQhFidmGpxEULTS9
S+x4nX0J7rDY2DPVmDfK0u4xqAMT8euS9NvIEp4MfB/whoipQhTWT3PSVVT+uCc+OnhKun0QuE551KwAQkDHPtZ+cJkKeNM=

## Server API properties
## -----

api.on=true
api.audit.on=false
api.user.registry=serverapi/registry.txt
api.user.registry.encryption.on=false

api.remote.on=true
api.remote.ssl.on=true
api.remote.ssl.client.auth.on=true
api.remote.naming.port=1099
# api.remote.server.ports=8700-8900
api.truststore=testserver.jks
api.truststore.type=jks
(protect)-api.truststore.pass={encr}DzTm01+sUaose3wpkbHk9vzZ4JzXHL8aMC2ePub4tWmuS+D70VcLI5aS8sayg0/ktc0cH6ozy6+qx1an
PpYtu1Dh7ZHDsAGDL+Temard/gJUT1xuG4FkA1r5YsDxhZ3n1d5fLa8h8YMTVDLd8qx6XZ16f/Ag0a0Yzn882wwF1=

## REST API
## -----
api.rest.on=true
api.rest.auth=false
api.rest.auth.realm=Security Directory Integrator REST API

api.rest.jmsdriver.name=com.ibm.di.systemqueue.driver.ActiveMQ
api.rest.jmsdriver.queue.sender.persistence=false
api.rest.jmsdriver.queue.sender.timeToLive=60000
api.rest.jmsdriver.param.jms.broker=vm://localhost?brokerConfig=xbean:etc/activemq.xml
# api.rest.jmsdriver.auth.username
# api.rest.jmsdriver.auth.password

## The properties determine the default bind address and the remote bind address for the Server API.
## * means bind to all network interfaces. The Remote Bind Address overrides the Default one.
## Only one IP address should be set. No hostnames are accepted.
## Mind that the java.rmi.server.hostname property is set implicitly to equal the Remote Bind Address property when used.
## This will cause the client stubs to create sockets on the specified Remote Bind Address.
# com.ibm.di.default.bind.address=*
# api.remote.bind.address=*

## Specifies a list of IP addresses to accept non SSL connections from (host names are not accepted).
## Use space, comma or semicolon as delimiter between IP addresses. This property is only taken into account
## when api.remote.ssl.on is set to false.
## api.remote.nonssl.hosts=

api.jmx.on=false
api.jmx.remote.on=false

## The configuration files placed in this folder can be edited through the Server API.
## Configuration files placed in other folders cannot be edited through the Server API.
api.config.folder=configs

## Timeout in minutes for configuration locks. A value of 0 means no timeout.
api.config.lock.timeout=0

## Timeout in minutes for loading a configuration.
api.config.load.timeout=2

## Specifies if the Server API methods for custom method invocation (Session.invokeCustom(...)) are allowed to be used.
## When api.custom.method.invoke.on is set to false and the Server API methods for custom method invocation are used,
## then an exception will be thrown.
## Only classes listed in api.custom.method.invoke.allowed.classes are allowed to be directly invoked.
## The default value is false.
api.custom.method.invoke.on=false

## Specifies the list of classes which can be directly invoked by the Server API methods for custom
## method invocation (Session.invokeCustom(...)).
## This property is only taken into account if api.custom.method.invoke.on is set to true.
## The classes in this list must be separated by a space, a comma or a semicolon.
## Example:
## api.custom.method.invoke.allowed.classes=com.ibm.MyClass,com.ibm.MyOtherClass
## In the above example only methods from the com.ibm.MyClass and com.ibm.MyOtherClass classes are
## allowed to be directly invoked.
api.custom.method.invoke.allowed.classes=

## Specifies a list of Server notification types, which will be suppressed.
## Notifications of suppressed types will not be propagated by the notifications framework.
## The notification types in the list are separated by spaces. Wildcards may be included.
## Example:
## api.notification.suppress=di.al.* di.ci.start
## The above example will suppress all Assembly Line related notifications as well as
## notifications for starting a configuration instance.
## If the property is missing or is empty, no notifications will be suppressed.
api.notification.suppress=di.server.api.authenticate di.server.api.authorize.*

```

```

## api.custom.authentication points to a JavaScript text file that contains custom authentication code.
## For example: api.custom.authentication=ldap_auth.js.
## To enable the built-in LDAP Authentication mechanism, set this property to "[ldap]".
## To enable the built-in JAAS Authentication mechanism, set this property to "[jaas]".
## For example: api.custom.authentication=[ldap]

##api.custom.authentication=[ldap]

## LDAP Authentication properties
## -----

## If this parameter is set to "true" and the LDAP Authentication initialization fails,
## the whole Server API will not be started.
## If this parameter is missing or is set to "false" any LDAP Authentication initialization errors will be logged
## and the Server API will be started.
api.custom.authentication.ldap.critical=false

## LDAP Server hostname.
api.custom.authentication.ldap.hostname=

## LDAP server port number. For example, 389 for non-SSL or 636 for SSL.
api.custom.authentication.ldap.port=

## Specifies whether SSL is used to communicate with the LDAP Server.
## When set to "true" SSL will be used, otherwise SSL will not be used.
api.custom.authentication.ldap.ssl=

## Specifies the LDAP directory location where user searches will be preformed.
## When this property is not specified user searches will not be performed.
api.custom.authentication.ldap.searchbase=

## Specifies the user id attribute to be used in searches.
## When this property is not specified user searches will not be performed.
api.custom.authentication.ldap.userattribute=

## Specifies an LDAP Server administrator distinguished name that will be used for user searches.
## When this property is not specified anonymous bind will be used for user searches.
api.custom.authentication.ldap.admindn=

## Password for the LDAP Server administrator distinguished name.
(protect)-api.custom.authentication.ldap.adminpassword={encr}

## This property specifies whether LDAP Group authentication is turned on.
## If it is set to 'true', the group membership of the authenticating user will be resolved
## and will be taken into account during authorization.
## If it is missing, the default value 'false' is used.
api.custom.authentication.ldap.groupsupport=false

## Specifies the name of the attribute of a user in LDAP that contains a list of the groups
## of which the user is a member.
## It is taken into account only if 'api.custom.authentication.ldap.groupsupport' is set to true.
api.custom.authentication.ldap.usermembershipattribute=

## Specifies how groups are named in the membership attribute of a user.
## For example, if the user's membership attribute contains values, which correspond to the 'objectSID' attributes
## of groups, set this property to 'objectSID'.
## If the user's membership attribute contains distinguished names of groups, then set this property to 'dn'.
## The property is required in case 'api.custom.authentication.ldap.groupsupport' is set to true.
api.custom.authentication.ldap.usermembershipattributecontent=

## Specifies the name of a group's attribute in LDAP, which corresponds to the way the
## group is named in the SDI User Registry.
## For example, if LDAP groups are addressed in the SDI registry by their common name, then set this property to 'cn'.
## If the User Registry contains the distinguished names of the groups, then set this property to 'dn'.
api.custom.authentication.ldap.groupnameattribute=

## Represents the LDAP directory context, where groups will be searched.
## It is required only when LDAP group support is enabled
api.custom.authentication.ldap.groupsearchbase=

## Optional property, which represents a list of space-separated attribute names.
## Specifies attributes which have non-string syntax.
## api.custom.authentication.ldap.binaryattributes=

## JAAS Authentication properties
## -----
java.security.auth.login.config=

## Enabling/Disabling FIPS Mode in SDI
## -----
## If the below property is set to true then SDI will be enforced to run in FIPS Compliant Mode.
## The default value is false, i.e. SDI will not run in FIPS Mode by default.
com.ibm.di.server.fipsmode.on=false

## Specify the unique ID for the SDI Server
## -----
## This property helps a client connecting to the SDI server to identify different servers
## running on the same IP and the same port in different time. (Default is DEFAULT_ID)
com.ibm.di.server.id=DEFAULT_ID

## Tombstone Manager properties
## -----
com.ibm.di.tm.on=false
com.ibm.di.tm.autodel.age=0
com.ibm.di.tm.autodel.records.trigger.on=10000
com.ibm.di.tm.autodel.records.max=5000
com.ibm.di.tm.create.all=false

## -----
## Help system properties
## -----

## Name of help server. The Tivoli library is at the following URL:

```



```

## http://www-01.ibm.com/support/knowledgecenter/SSCQGF/we1come

## Port for help system
com.ibm.di.helpPort=80

## -----
## AssemblyLinePool: Connector pooling defaults
## -----
##
## Note! These settings are only used when an AssemblyLine uses
## an AssemblyLinePool in combination with a Server mode connector.

## The number of seconds before a pooled connector times (e.g. is closed and no longer reused)
## Less than zero means disable connector pooling
## Zero means never timeout
## Greater than zero sets the number of seconds before a connector is closed
com.ibm.di.server.connectorpooltimeout=42

## Comma separated list of connector interfaces that we never pool
com.ibm.di.server.connectorpoolexclude=com.ibm.di.connector.FileConnector,com.ibm.di.connector.ScriptConnector

## Properties for Windows IPv6 communications.
## Uncomment these properties for Windows IPv6 communication only.
## These properties will not affect IPv4 communication or IPv6 communication on Unices.
#java.net.preferIPv4Stack=false
#java.net.preferIPv6Addresses=true

## -----
## Performance settings
## -----
##
## Enable/Disable performance logging
com.ibm.di.server.perfStats=false

### -----
### Used by Config Report
###-----
### set this is you want to override the local language for Config Reports
# com.ibm.di.admin.configreport.translation=en

##-----
## System Queue settings
##-----
## If set to "true" the System Queue is initialized on startup and can be used;
## otherwise the System Queue is not initialized and cannot be used.
systemqueue.on=true

## Specifies the fully qualified name of the class that will be used as a JMS Driver.
# systemqueue.jmsdriver.name=com.ibm.di.systemqueue.driver.IBMMQ
# systemqueue.jmsdriver.name=com.ibm.di.systemqueue.driver.JMSScriptDriver
systemqueue.jmsdriver.name=com.ibm.di.systemqueue.driver.ActiveMQ

### MQ JMS driver initialization properties
# systemqueue.jmsdriver.param.jms.broker=<host:port>
# systemqueue.jmsdriver.param.jms.serverChannel=<channel_name>
# systemqueue.jmsdriver.param.jms.qManager=<queuemanger_name>
# systemqueue.jmsdriver.param.jms.sslCipher=<cipherSuite_name>
# systemqueue.jmsdriver.param.jms.sslUseFlag=false

### JMS Javascript driver initialization properties
## Specifies the location of the script file
# systemqueue.jmsdriver.param.js.jsfile=driver.js

### ActiveMQ driver initialization properties
## Specifies the location of the ActiveMQ initialization file.
## This file is used to initialize ActiveMQ on SDI server startup.
systemqueue.jmsdriver.param.jms.broker=vm://localhost?brokerConfig=xbean:etc/activemq.xml

## This is the place to put any JMS provider specific properties needed by a JMS Driver,
## which connects to a 3rd party JMS system.
## All JMS Driver properties should begin with the 'systemqueue.jmsdriver.param.' prefix.
## All properties having this prefix are passes to the JMS Driver on initialization after
## removing the 'systemqueue.jmsdriver.param.' prefix from the property name.
# systemqueue.jmsdriver.param.user.param1=value1
# systemqueue.jmsdriver.param.user.param2=value2
# ...

## Credentials used for authenticating to the target JMS system
# (protect)-systemqueue.auth.username=<username>
# (protect)-systemqueue.auth.password=<password>

## -----
## Logging settings
## -----

## When false, all log calls made through the SDI Log class will be discarded.
com.ibm.di.logging.enabled=true

## -----
## IBM JavaScript Engine settings
## -----

## Set the type of platform - required by the IBM JS Engine when caching is used.
com.ibm.common.platform=com.ibm.common.platform.GenericPlatform

##
## Set this property to a directory to enable auto dumps of assemblylines that fails
##
# com.ibm.tdi.autodump.directory=<dump-directory>

```

```

##
## Server API client properties
##
api.client.ssl.custom.properties.on=true
api.client.keystore=serverapi/testadmin.jks
(protect)-api.client.keystore.pass={encr}E/1TC2+UF4B9BACaVcdKoa1Yj38LFi26UncEFTsWP+qj68fuJH8EwCs392IoLxvIAKH1PaWJJo7h
ySkeBxESorVACAF8oNbHXIBQOS2nvnPKx1kQZK32CaR1g5Wq8TNamQxHFr++UewBuDEAU1ki5z7zXiD2g8g0sN6cK1IHUo=
api.client.keystore.type=jks
(protect)-api.client.key.pass={encr}EU1+JvyysxVts7Yn236FysDdxV7IP77mjCOy/si0m6x8H6Hcy1epHjMgyunQcqIQeN/KpL7M0a3uqzkwk
chMURm0WR+08xyoN+hMpoAu1EvmXjubtd7jd6JVincesL5BYiSwcxGeTsnQJ/MN84RiCfGc1FzWYxvL53npGeyGxk=
api.client.truststore=serverapi/testadmin.jks
(protect)-api.client.truststore.pass={encr}Y8Np19khRasEUfwYSaAM5RkJK+NODOKezRkXgbLyZtU1V0sFiJfCkeLmw8+MndHvtVkpM/3N1n
g/y+zv9NKFABVqBVYTJxK5RZx3YV/IgQJMpJK/YhT2hSR8w5XSM7meJ01JK3NjC+Cy9my42ioyT+svLUgpQfs74DxYL8482ww=
api.client.truststore.type=jks

## -----
## Mail properties
## -----

## This property needs to be set to a valid SMTP host to be able
## to send mail using the system methods.
## mail.smtp.host=

## -----
## Enabling/Disabling NIST Mode in SDI
##-----
## If the below property is set to true then SDI will be enforced to run in NIST Compliant Mode.
## The default value is false, i.e. SDI will not run in NIST Mode by default.
com.ibm.di.server.NIST.on=false

```

Appendix B. Monitoring with external tools

You can monitor the IBM SDI with external tools like Tivoli Monitoring and Tivoli Netcool/OMNIBus. Read more about its working and corresponding components in the information provided here.

This is a "first step" into the integration between IBM Security Directory Integrator , IBM Tivoli Monitoring and IBM Tivoli Netcool/OMNIBus. It started as a proof-of-concept of this integration scenario. The IBM Security Directory Integrator to IBM Tivoli Monitoring and IBM Security Directory Integrator to OMNIBus integration capabilities shown in this document are fully supported solutions shipped with IBM Security Directory Integrator. The solutions are shipped in the examples directory but they are fully supported.

JMX was chosen for communication between IBM Security Directory Integrator and ITM, because IBM Security Directory Integrator provides a ready-to-use JMX interface and thus no development was necessary on the IBM Security Directory Integrator side.

For monitoring IBM Security Directory Integrator via Tivoli Netcool/OMNIBus an AssemblyLine was developed in order to detect IBM Security Directory Integrator events and send them to OMNIBus. The purpose of this section is to present how IBM Security Directory Integrator can be monitored by:

- Tivoli Monitoring using the IBM Security Directory Integrator JMX interface
- Tivoli Netcool/OMNIBus

Both integration scenarios are bundled as official IBM Security Directory Integrator examples and can be found in *TDI_install_dir/examples/Tivoli_Monitoring* directory.

ITM 6.2.0 and Tivoli Netcool/OMNIBus 7.2.1 were used for these examples.

Several software components were necessary in order to realize the experiments described here. Here is the list of these components, and the reference documentation used to realize their installation:

- ITM Agent Builder 6.2 - ITM Agent Builder 6.2 User's Guide
- ITM Tivoli Enterprise Portal - ITM Tivoli Enterprise Portal online documentation
- Tivoli Netcool/OMNIBus 7.2.1 - Tivoli Netcool/OMNIBus online documentation.

JMX is used for communication between IBM Security Directory Integrator and Tivoli Monitoring. On the IBM Security Directory Integrator side it is the JMX layer of the Server API that Tivoli Monitoring connects to.

For communication between IBM Security Directory Integrator and Tivoli Netcool/OMNIBus two connectors are used. A Server Notifications Connector is used to receive a set of IBM Security Directory Integrator Server Notifications, and an EIF Connector to send events to OMNIBus.

Monitoring IBM Security Directory Integrator with ITM

You can refer to the information provided here to know more about the architecture, importing existing configuration, creating agent, and many more aspects of ITM.

Short presentation of the ITM architecture

You can use a ITM to collect data and monitor the system. Learn more about the type of agents through the information provided here.

At its core, the browser provided by ITM presents data that is gathered by agents.

ITM agents are characterized by the following definition:

"The agents (referred to as managed systems) are installed on the system or subsystem requiring data collection and monitoring. The agents are responsible for data gathering and distribution of attributes to the monitoring servers, including initiating the heartbeat status." (Extract from the ITM documentation)

There can be various kinds of agents: agents to monitor operating systems or specific applications, or specifically tuned agents (that is, using the Universal Agent interface). The following diagram, taken from the ITM documentation, describes both the architecture and the deployment process of agents:

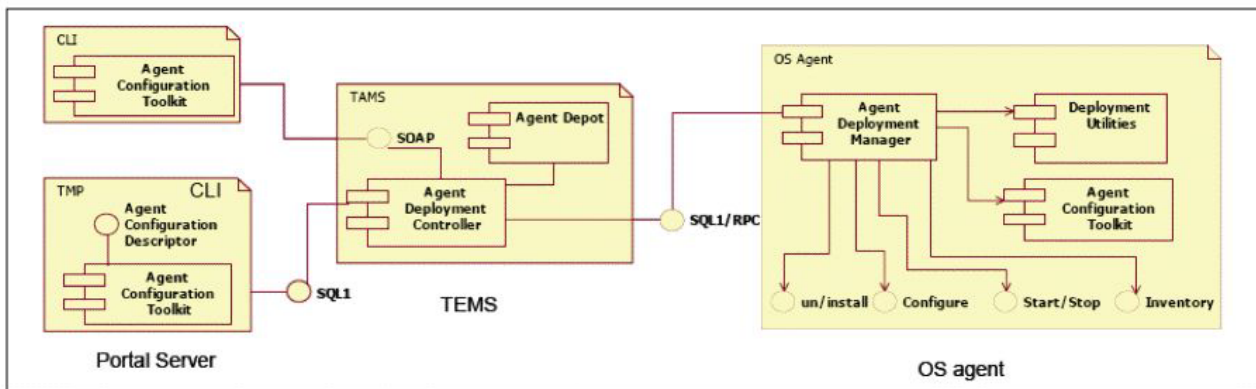


Figure 2. ITM Agents diagram

TEMS = Tivoli Enterprise Monitoring Services

TEP = Tivoli Enterprise portal

Importing an existing Agent configuration in ITM Agent Builder 6.2

Follow the steps described here to import an existing agent configuration file.

If you have an ITM Agent configuration XML file, you can import it in the ITM Agent Builder 6.2 and it will create ITM Agent project automatically. To import such a file, right-click in the ITM Agent Builder workspace, select **Import...** and select **IBM Tivoli Monitoring Agent for import**. Point to the configuration XML file (default name: itm_toolkit_agent.xml) and click **Finish**. This will create an ITM Agent Builder project with an appropriate name.

Note: If you wish to create the agent yourself, go to section “Creating an IBM SDI agent for ITM using ITM Agent Builder 6.2”; otherwise go to section “Generating the ITM Agent” on page 357.

Creating an IBM SDI agent for ITM using ITM Agent Builder 6.2

You can use the steps listed in the example shown here to create an IBM SDI agent for ITM using ITM Agent Builder 6.2.

The ITM Agent Builder is an Eclipse based platform for creating ITM Agents. The Agent that we will create for this example uses the JMX interface. From the ITM Agent Builder choose **File -> New -> IBM Tivoli Monitoring Agent**.

The ITM Agent Wizard will show up. The first step is an introduction - click **Next**. On the second step you will be asked to enter a project name. In this example we will use "SDI" as project name. Clicking **Next** brings us to the following step:

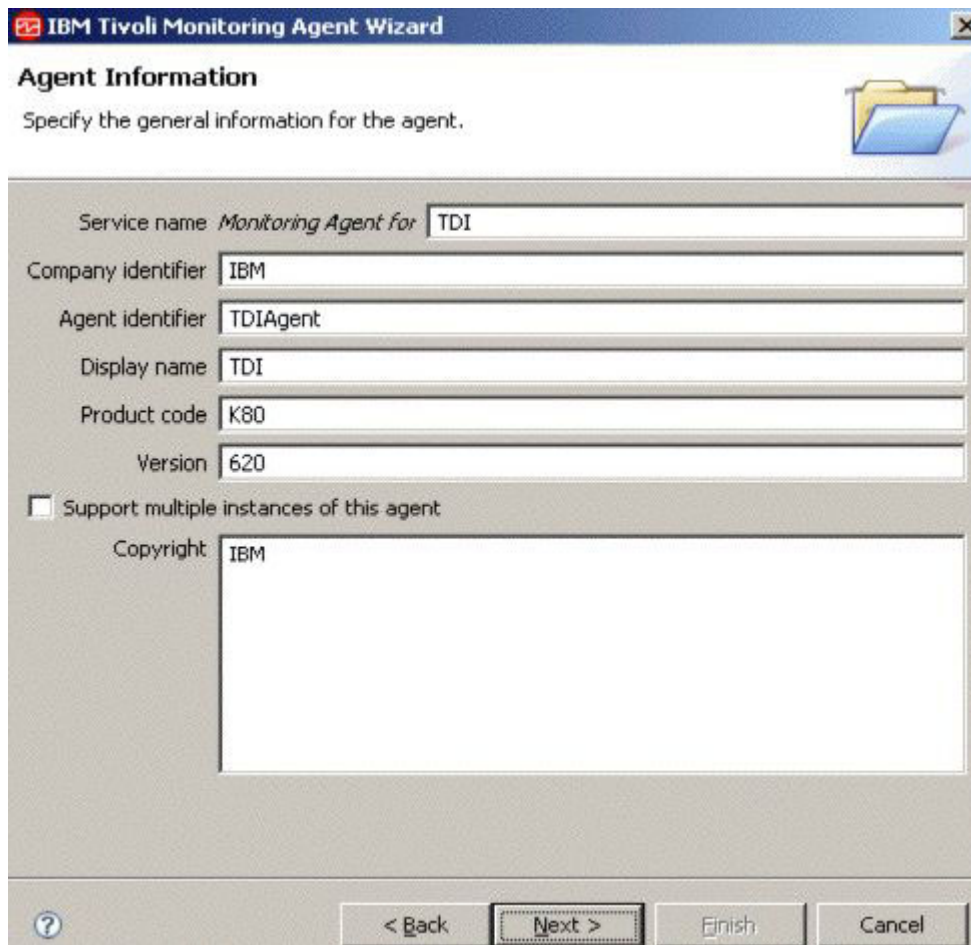


Figure 3. ITM Agent wizard Agent information

Fill all the fields with appropriate data. The Product code should be between K80 and K99 for JMX agents. click **Next**. On the next step check the **This agent will gather data from an external data source.** option and click **Next**. On this step the data source definition window is displayed:

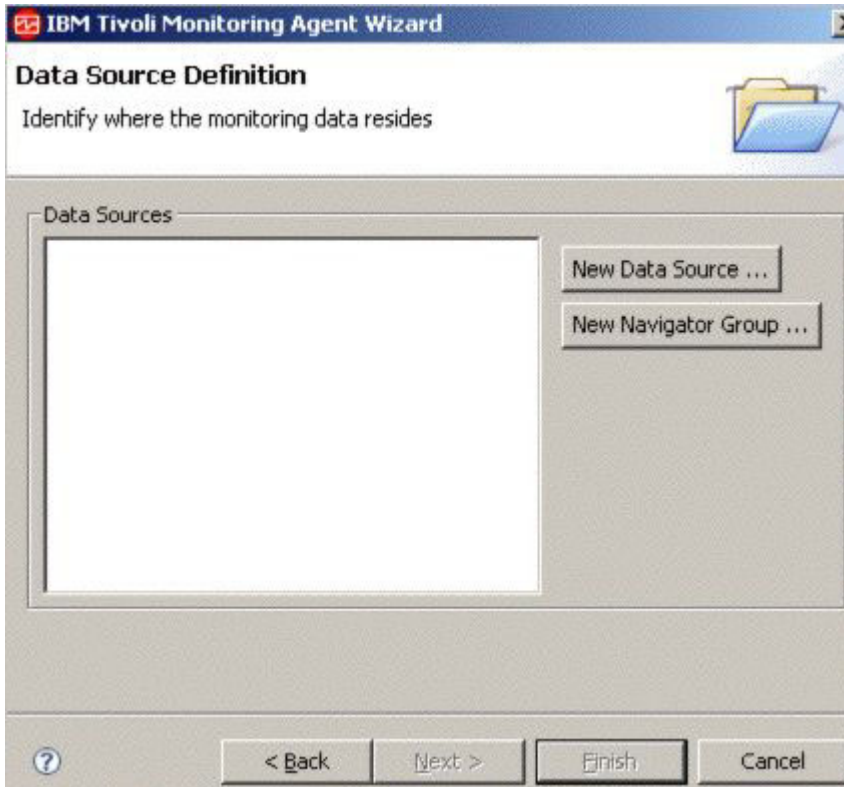


Figure 4. ITM Agent wizard, Data source definition

In order to make this step easier to configure start an IBM Security Directory Integrator Server in daemon mode and run an AL that never ends (for example an AL with an HTTP Server Connector listening for connections). Make sure the JMX API is enabled in IBM Security Directory Integrator (there is a description on how to do this later in the example).

Click the **New Data Source ...** button and then choose the **Collect data from Java Management Extensions (JMX) MBeans** option. Click **Next**. On the next window click **Browse** which should display the JMX Browser:

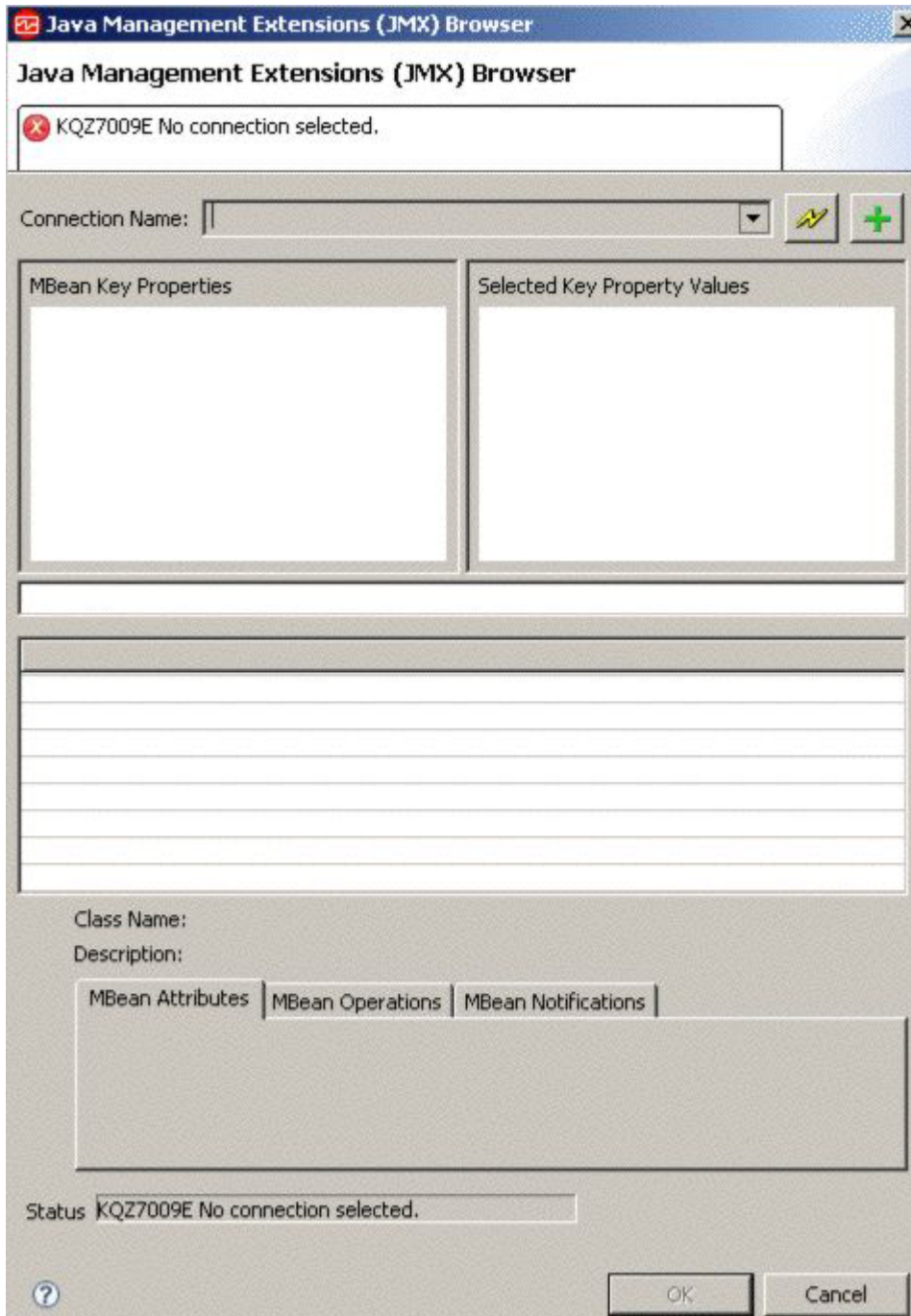


Figure 5. JMX Browser

Click the **Edit Connection Definitions** button (the green plus button). On the next step select **Standard JMX Connections (JSR-160)** and click **Next**. The new wizard window will display the available templates. Select **JSR-160 -Compliant Server** and again click **Next** to see the Connection properties of the JMX Server.

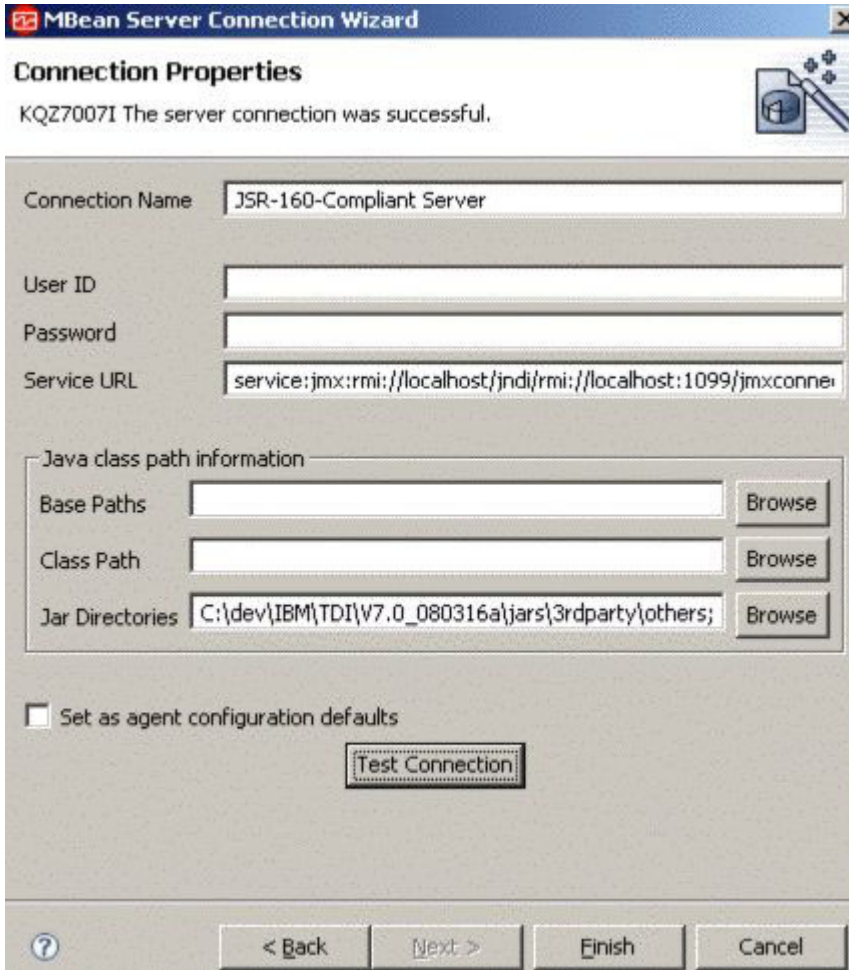


Figure 6. Server Connection wizard

In order to establish a successful connection with the IBM Security Directory Integrator JMX Service you will need to enter a valid JMX Service URL (the default IBM Security Directory Integrator JMX Service URL is `service:jmx:rmi://localhost/jndi/rmi://localhost:1099/jmxconnector`) and to configure the jar dependencies that are required for successful JMX MBeans creation (for the IBM Security Directory Integrator JMX MBeans you will need the jar files in `TDI_install_dir\jars\3rdparty\IBM`; `TDI_install_dir\jars\3rdparty\others`; `TDI_install_dir\jars\common` directories). You can test these settings by clicking the **Test Connection** button. If the whole configuration is correct a message like this will be displayed: "The server connection was successful."

After this setup click **Finish**. The wizard should bring us the previous configuration step, but this time connected to the IBM Security Directory Integrator JMX Server and will display additional information:

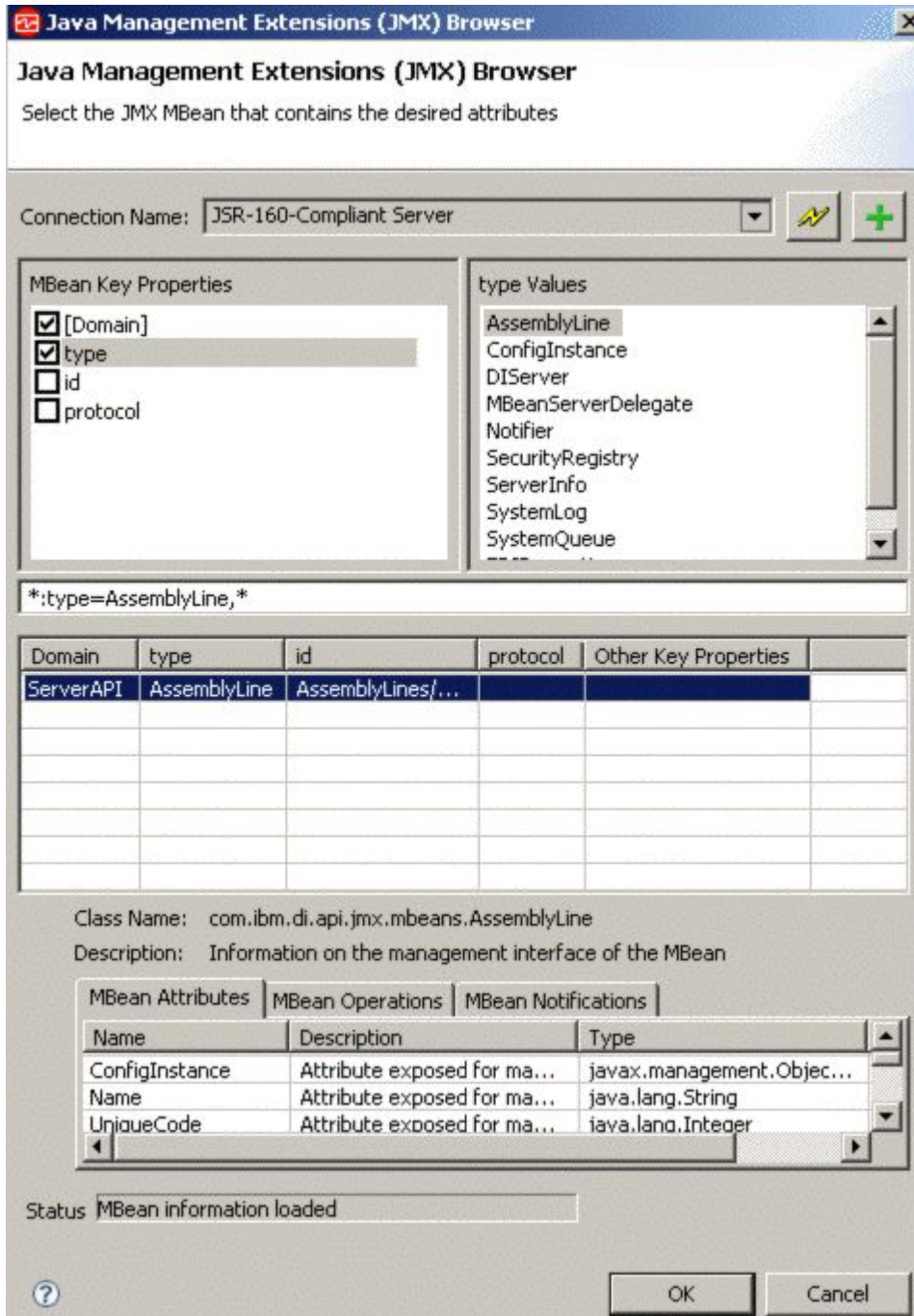


Figure 7. Browsing IBM Security Directory Integrator in JMX Browser

Select the **type** MBean Key Property and **AssemblyLine** from the type values. To see the MBean Attributes you need to select a row in the table above them. In our case there is only one row. Click **OK** and then **Finish** to complete the setup of this data source.

Create one more data source with type value **ConfigInstance** in the same way we created the **AssemblyLine** data source. These two data sources will gather information from the JMX Server for running **AssemblyLines** and started **Configuration Instances**.

The third data source is a little different from the other two. It is a kind of listener which listens for notifications (events) sent by the IBM Security Directory Integrator JMX Server. To create one like that, after clicking the **New Data Source...** button, you do not need to browse the JMX Server but simply enter `*:type=Notifier,*` for MBean pattern and click **Finish**. Two data sources will be created - one for the notification part and one for the static MBean part. Since we do not need the static part for this data source we need to remove it; right-click and select **Remove Data Source(s)**.

After completing these steps we should have three data sources created:

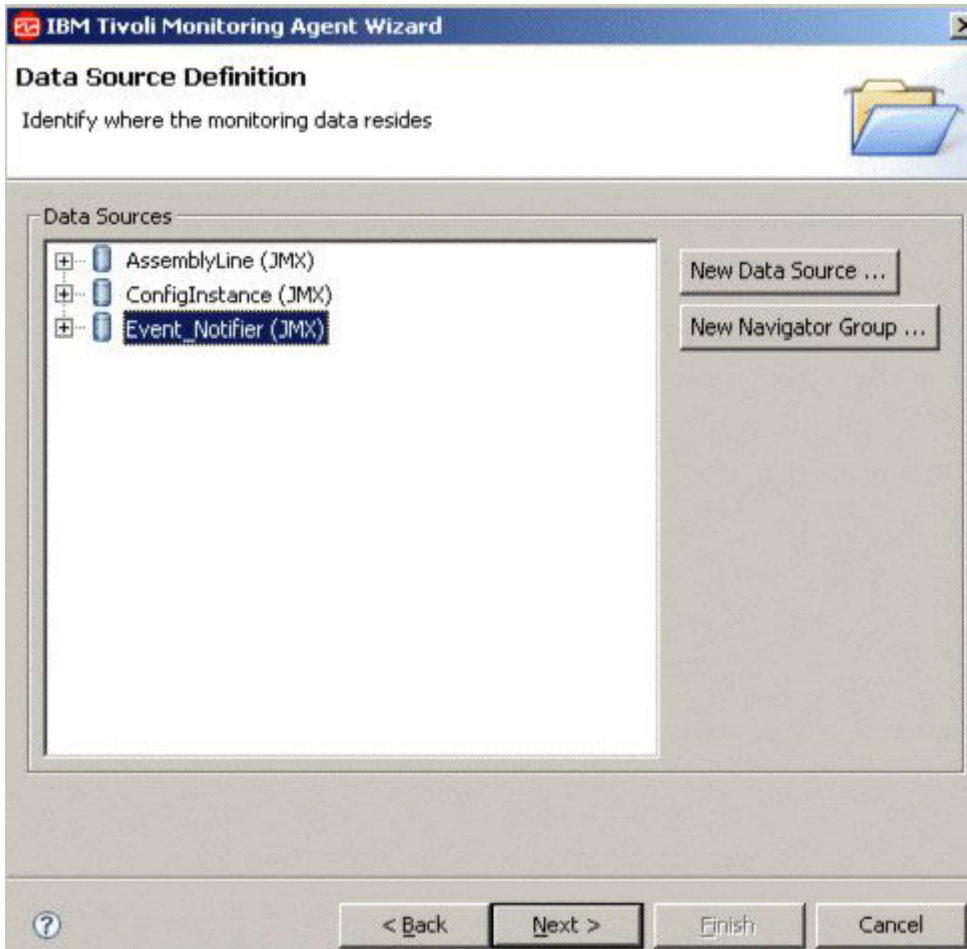


Figure 8. ITM Wizard, completed Data Source Definition

Expand the AssemblyLine data source and double-click the **ConfigInstance** attribute. In the ConfigInstance attribute configuration check the **key attribute** checkbox.

Expand the ConfigInstance data source and double-click the **ConfigId** attribute. In the ConfigId attribute configuration check the **key attribute** checkbox.

Click **Next** in order to configure the JMX Agent - Wide Options. Uncheck the JMX monitor attribute groups checkbox and select **JSR-160-Compliant Server** from the Server configuration choices.

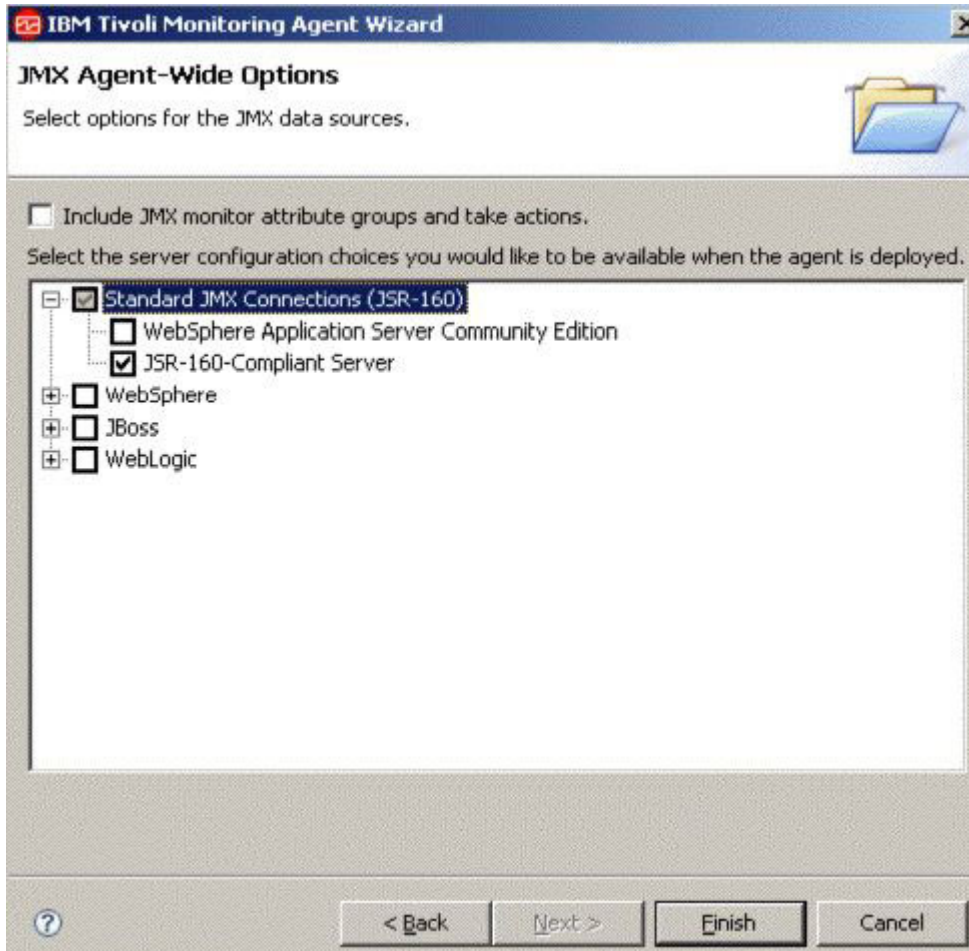


Figure 9. JMX Agent-wide options

Click **Finish** to complete the ITM Agent creation steps and save the Agent.

Generating the ITM Agent

After successfully creating an ITM agent, use the steps provided here to generate it.

After the successful creation of the ITM Agent configuration, we need to generate it in order to deploy it in ITM. From the IBM Tivoli Monitoring Agent Editor menu in the ITM Agent Builder choose **Generate Agent**. The Generate Agent Wizard will appear. This wizard has several options of Agent generation. If you use ITM and ITM Agent Builder on a single machine then the **Generate the agent files in an ITM installation on this machine** option is suitable for you. The only field which needs to be configured is the ITM installation directory. Click **Finish** to generate and deploy the ITM Agent in ITM. This may take several minutes to complete.

Note: If you want to have the agent on another machine then you can use another agent generation option – **Create a compressed file so that the agent can be installed on another system**. This will generate an archive that contains the ITM Agent installation. To install such an archived agent, you first have to copy the file to the machine where ITM is installed. Extract the files from the archive and from the command prompt start the InstallIRA.bat file with parameter the ITM install folder.

For example if ITM is installed in C:\IBM\ITM the command will look like:

```
<AgentDirectory>:\>InstallIRA.bat C:\IBM\ITM
```

Configuring the ITM Agent

Once the ITM agent has been deployed, you need to configure it. Use the steps and example provided here to perform this task.

After the successful deployment of the agent (either using an archive file or the ITM Agent Builder option to deploy it on the same machine) we have to configure it in ITM. To do so, start **Manage Tivoli Monitoring Enterprise Services** where you can manage all ITM Agents:

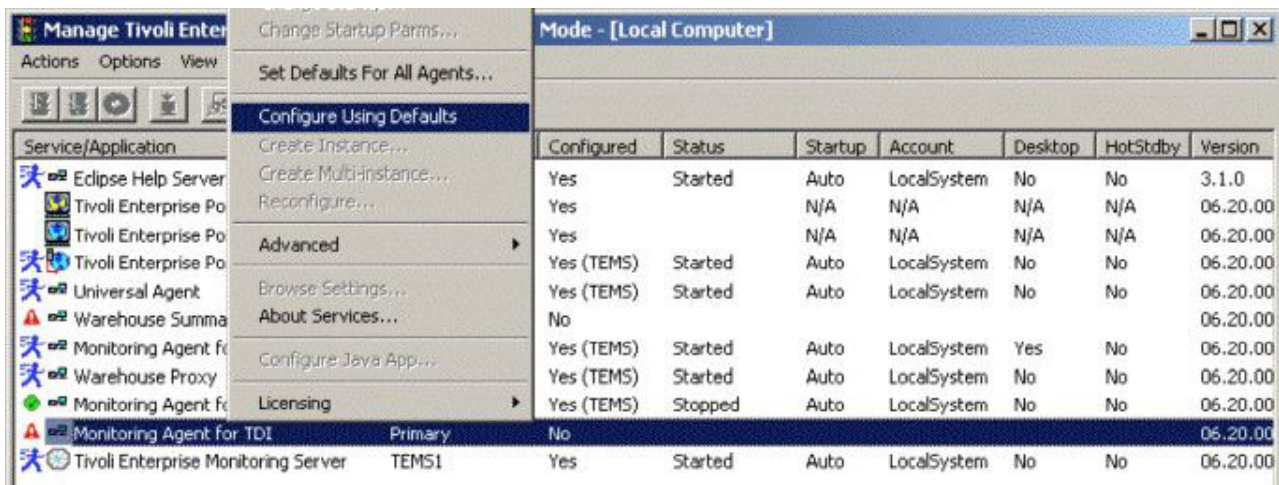


Figure 10. Manage Tivoli Monitoring Enterprise Services

Right-click the IBM Security Directory Integrator Agent and select **Configure Using Defaults**.

On the next configuration window we need to configure the JVM properties for the Agent. Browse to the Java Home that you wish to use. The log trace level is set to "Error" by default. It can be changed to higher level in order to log additional information. After finishing the Java configuration click the **Next** button.

On the next configuration step you are asked to configure the JSR-160 Compliant Server properties, that is, enter username, password, Service URL and Class Path dependencies. For our example we need to enter Service URL and Jar directories like we did in order to create the agent - Service URL `service:jmx:rmi:///localhost/jndi/rmi:///localhost:1099/jmxconnector` and Jar directories `TDI_install_dir\jars\3rdparty\IBM; TDI_install_dir\jars\3rdparty\others; TDI_install_dir\jars\common`.

Click **OK** to complete the Agent Configuration.

The IBM Security Directory Integrator Agent should be ready for use. The next step is to start the Agent. We can start it from the Manage TEMS window by right-clicking the IBM Security Directory Integrator Agent and choosing **Start**. If all steps are successful the IBM Security Directory Integrator Agent will be running now.

Monitoring IBM Security Directory Integrator data

Use the Tivoli Enterprise Portal (TEP) to monitor the data. For more details refer the listed steps.

To monitor data, we need to start the Tivoli Enterprise Portal (TEP), which is available in the ITM installation. In the navigator TEP window we can see the running Agents. The IBM Security Directory Integrator Agent is also there and we have to expand it to see the specific monitoring data sources:

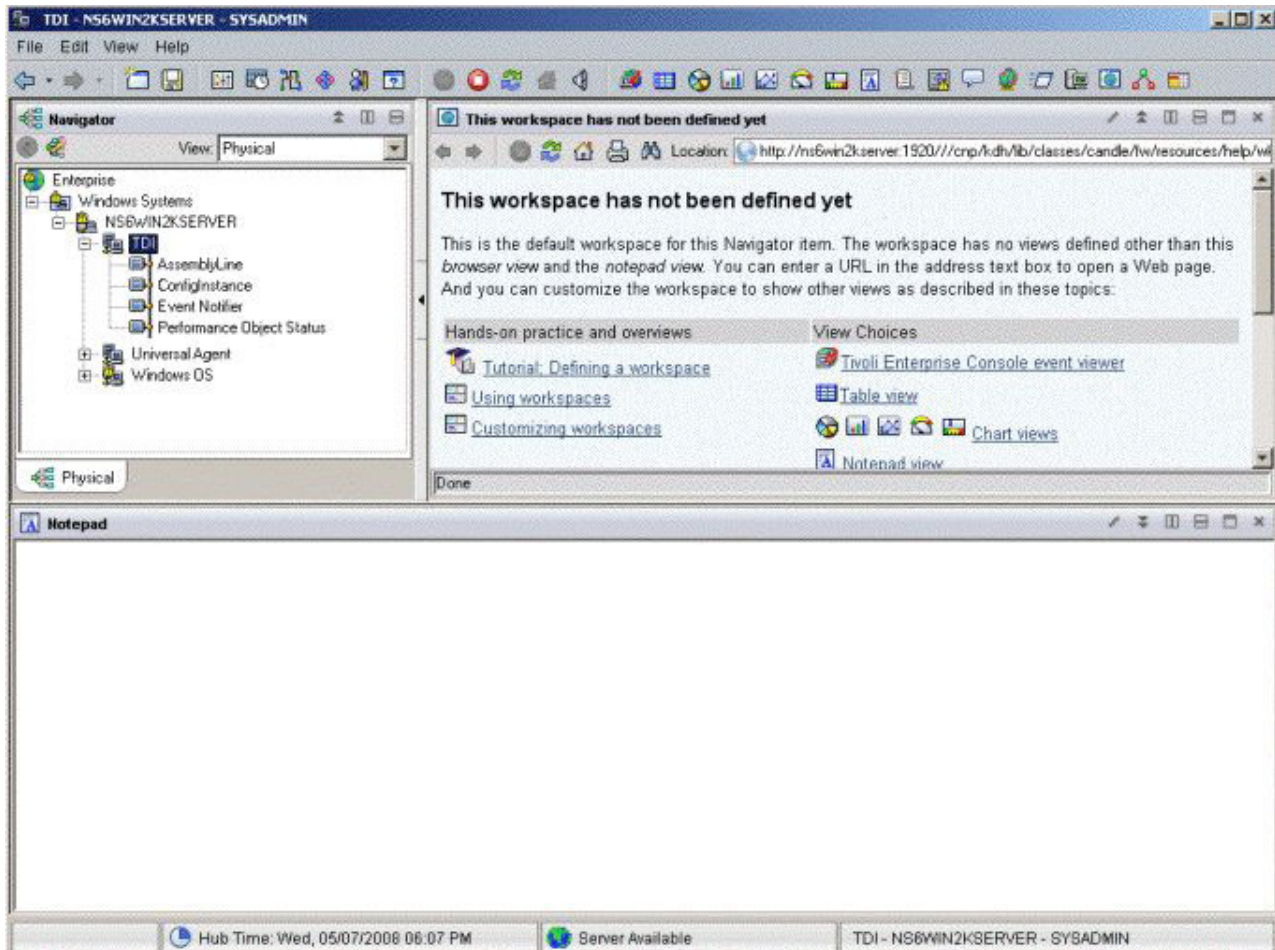


Figure 11. Tivoli Enterprise Portal (TEP) - Wizards

We can find our custom made data sources there - AssemblyLine, ConfigInstance, Event Notifier. If we have a running IBM Security Directory Integrator Server and a running AL in it, we can see it in the report table of the AssemblyLine data source.

Note: In order to display data in the Notifier report table, the IBM Security Directory Integrator Agent has to be running before an IBM Security Directory Integrator notification is triggered.

This browser can be tuned in several ways: for numeric data it is possible to have a more human readable presentation (like diagrams). It is also possible to change the layout of the tables.

This functionality is not very complex, and is well described in the ITM documentation.

As the aim of this document is not to present the whole ITM product in intricate detail, but to focus on the usage that can be done in correlation with IBM Security Directory Integrator, we will present here only the two trickiest concepts: defining thresholds, and links between tables.

For other functionality, please refer to the ITM documentation.

Defining thresholds

You can learn to work on threshold mechanism by using the example and screen captures here.

To show how the threshold mechanism works, we will create the following simple example: display a warning when more than one AssemblyLine is currently running.

This threshold will depend on data provided by the AssemblyLine table. First we need to create a situation by right-clicking on the table of the agent, and by selecting **Situations** in the contextual menu:

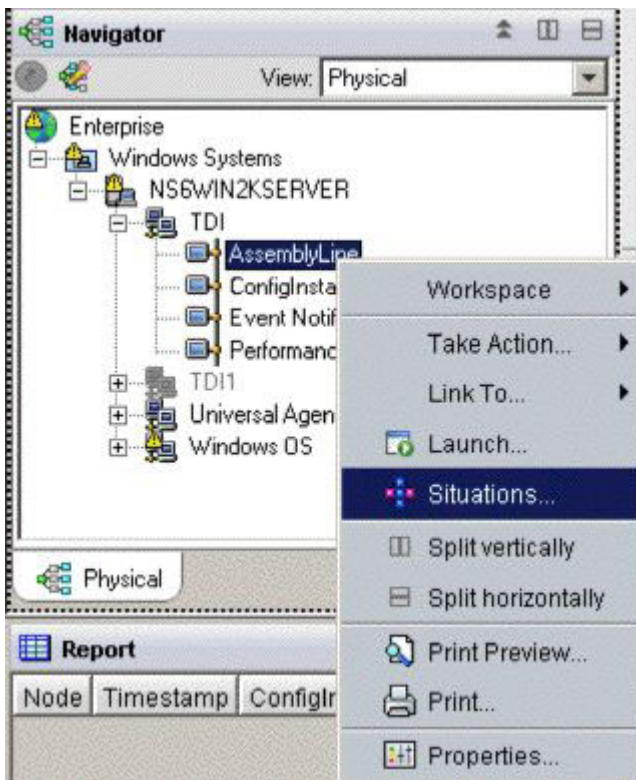


Figure 12. Situations context menu

Click on the **Create Situation** button in the upper-left corner and fill in the displayed form:

Create Situation

Name: AssemblyLines

Description: AL warning

Monitored Application: TDI

Correlate Situations across Managed Systems

Situation name:

- 1) Must be 31 characters or less,
- 2) Must start with an alphabetic character (a-z, A-Z),
- 3) May contain any alphabetic, numeric (0-9) or underscore (_) character,
- 4) Must end with an alphabetic or numeric character.

OK Cancel Help

Figure 13. Situation form

This will be the name associated to the warning. Here we describe a case study, but in real situations you would give it meaningful names.

Then we choose with which table attribute our situation will deal:

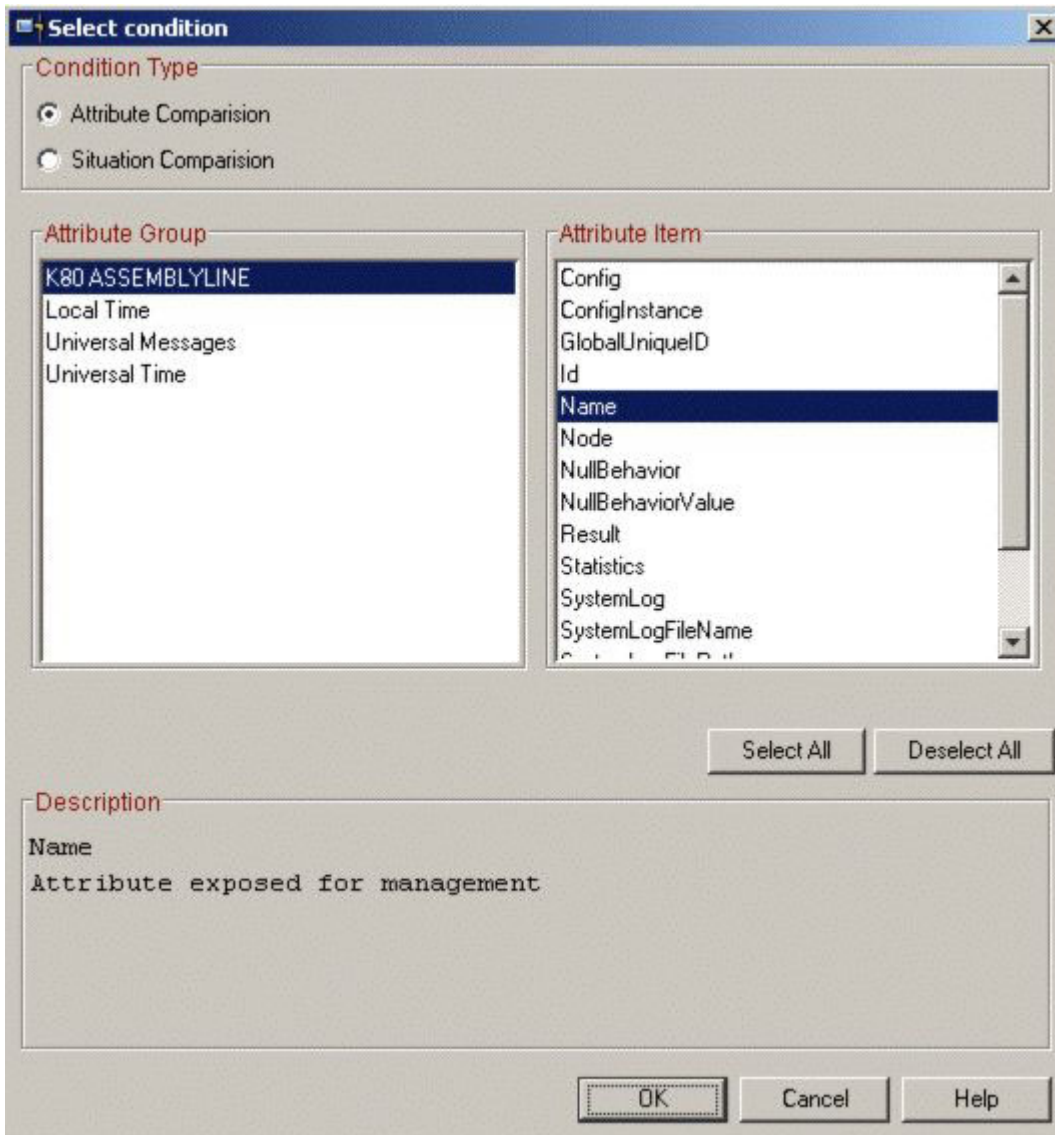
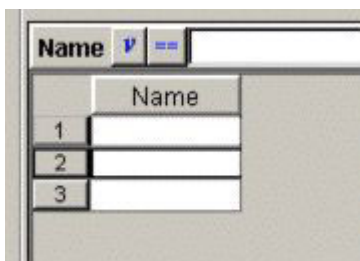


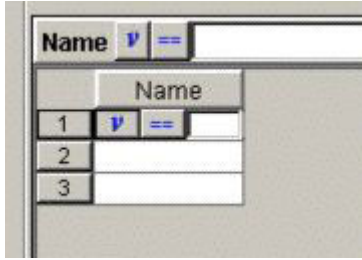
Figure 14. Situation: select condition

Indeed, we only have to consider the name of the AL to identify it.

Click in one of the cells, for example the one in line number 1:



The display changes:

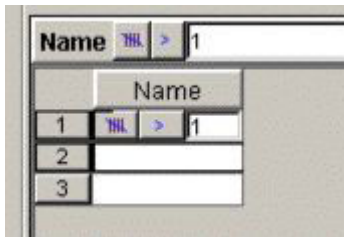


Click on the *v*, and change it to "Count of group members".

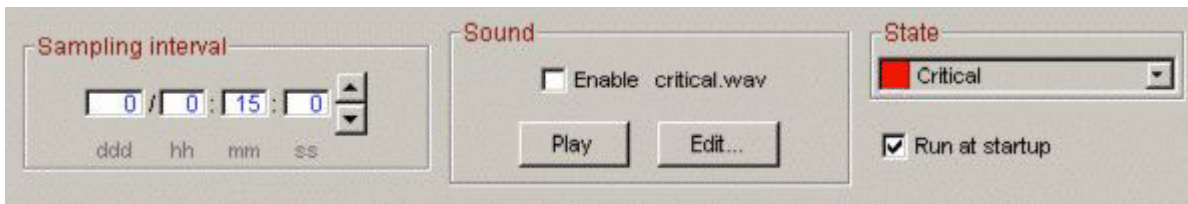
Click on the ==, and change it to >.

Set the cell space remaining on the right to 1.

We have configured a condition to the **Name** column, which will be true if we have more than one AssemblyLine running:



And change the following default settings:



To this:



The situation is set, so **Apply** and validate this window.

Start IBM Security Directory Integrator Server and start at least two AssemblyLines at the same time; for example two HTTP Server Connectors that are listening on different ports.

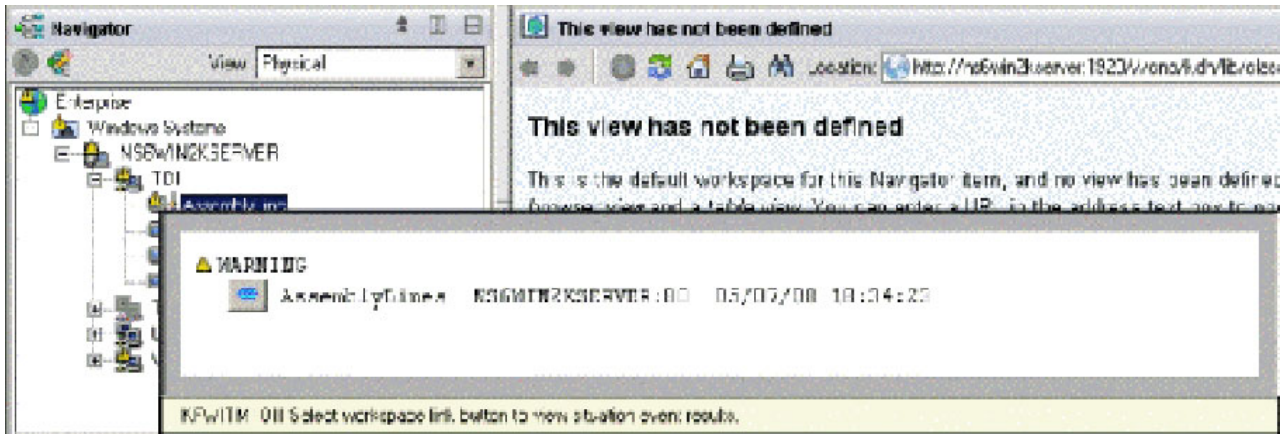


Figure 15. ITM displayed warning

The **Warning** window is opened while highlighting the warning icon.

Creating links between tables

You can create links between tables in ITM. Learn more about the purpose of creating links and how to create links with the information provided here.

Purpose of links:

You can filter the data and directly see a subset of a table by following the link.

It is possible to specify links between different tables in ITM. These links can be based upon some criteria of our choice, as we will see in this example. When a link is created you can automatically see a subset of a table by following the specified link. In this example we will create a link from the Event Notifier table that will show the currently running AssemblyLine in the AssemblyLine table. The link will be available in the Event Notifier table only for records that have the type "di.al.start". This type indicates that an AssemblyLine has been started. If the link is pressed and the AssemblyLine is still running, the AssemblyLine table will be automatically selected and only the corresponding AssemblyLine will be displayed in the table. If the AssemblyLine has already finished its execution then the displayed table will be empty.


This is an example Event Notifier table with defined link to the AssemblyLine table:

Node	Timestamp	Type	Source	Sequence Number	Time Stamp	Message
NS6WIN2KSERVER-80	05/07/08 18:33:59	di.al.stop	ServerAPI.type=Notifier,id=Notifier	6	1210174439618	AssemblyLine 'AssemblyLines/Serv
NS6WIN2KSERVER-80	05/07/08 18:33:59	di.al.start	ServerAPI.type=Notifier,id=Notifier	5	1210174439558	AssemblyLine 'AssemblyLines/Serv
NS6WIN2KSERVER-80	05/07/08 18:33:59	di.ci.start	ServerAPI.type=Notifier,id=Notifier	4	1210174439558	ConfigInstance 'tunname' started.
NS6WIN2KSERVER-80	05/07/08 18:32:49	di.ci.start	ServerAPI.type=Notifier,id=Notifier	3	1210174369672	ConfigInstance 'sss' started.
NS6WIN2KSERVER-80	05/07/08 18:32:49	di.ci.start	ServerAPI.type=Notifier,id=Notifier	2	1210174369622	ConfigInstance 'C_dev IBM_TDI \
NS6WIN2KSERVER-80	05/07/08 18:32:48	di.al.start	ServerAPI.type=Notifier,id=Notifier	1	1210174368120	AssemblyLine 'AssemblyLines/Serv
NS6WIN2KSERVER-80	05/07/08 18:32:36	di.al.start	ServerAPI.type=Notifier,id=Notifier	0	1210174355842	AssemblyLine 'AssemblyLines/Serv

Figure 16. Example Event Notifier table

The table has three loaded configurations, three started AssemblyLines (one of which has been stopped). When the **ToTheRunningAssemblyLine** link is selected,

the AssemblyLine is displayed in the AssemblyLine table (no other AssemblyLines are shown in the table):



Name	Config	ConfigInstance	GlobalUniqueID	Id	Node
AssemblyLines/Server1	Server1	ServerAPI.type=ConfigInstance.id=C_dev_IBM_TDI_V7.0_0...	11210174355752	AssemblyLines/Server1.1	NS6WIN2KSERVER.8

Figure 17. Example Event

Construction of links:

You can follow the steps listed here to construct a link.

First we need to create a key in the AssemblyLine table which will be accessible from the Event Notifier table and will correspond to the AssemblyLine ID. Right-click in the AssemblyLine table and select properties.

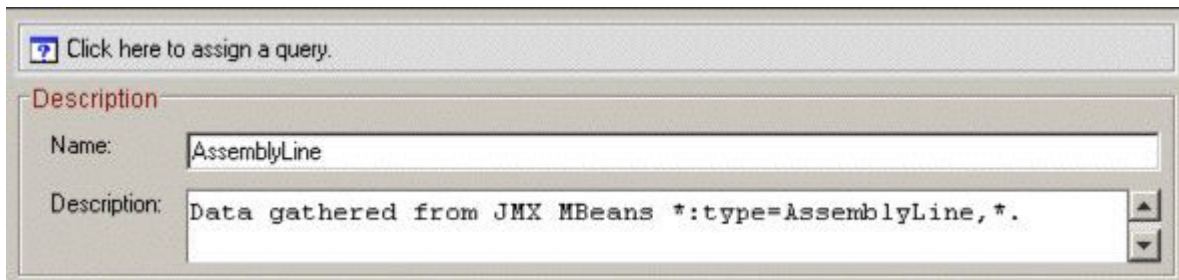


Figure 18. AssemblyLine properties

In the opened window, assign a new query by clicking the **Click here to assign a query** button.

The Query Editor will be opened where we must define another query because the existing one is static and cannot be modified. You will be asked to enter a name for it (for example "AssemblyLine2").



Figure 19. Create query selection

Go to the **Id** column of the table and enter \$keyid\$ as its value.

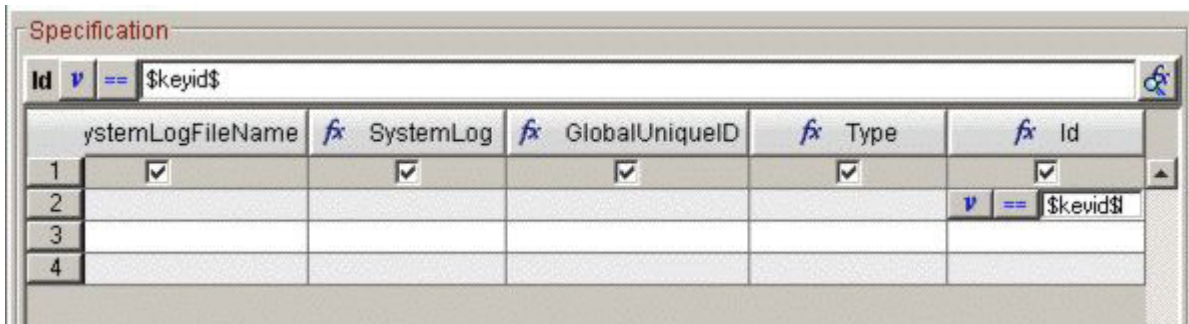


Figure 20. Query Editor

Click **OK** and apply the changes in the properties window and then click "**OK**" button.

This is all that needs to be done in the AssemblyLine table.

Go to the Event Notifier table (you will be asked to save the changed in the AssemblyLine table - click **Yes**).

Right-click on the selected row in the Event Notifier table and choose **Link To...** -> **Link Wizard...** (make sure that the type column of the selected row equals "di.al.start").

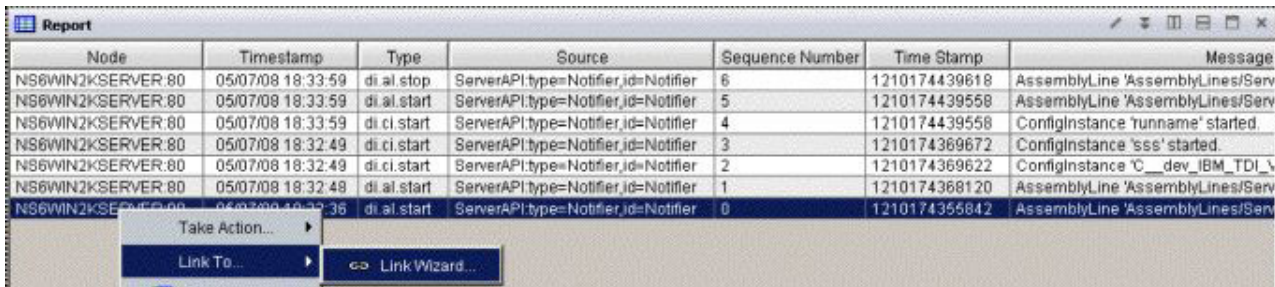


Figure 21. Link Wizard selection

The Link Wizard will be displayed and will ask whether a new link will be created, modify existing link or delete existing link. Select **Create a new link** and click **Next**. On the next screen the name and the description of the link must be entered. In this example we will use **ToTheRunningAssemblyLine** as name and "Display the corresponding AssemblyLine in the AssemblyLine table." as description. The next step will ask to specify the link type. In the example we will use **Absolute** as link type because we are linking to a specified non-dynamic workspace in the navigator view. Proceed to the next step where we must specify the workspace to which the link will direct (the AssemblyLine table).

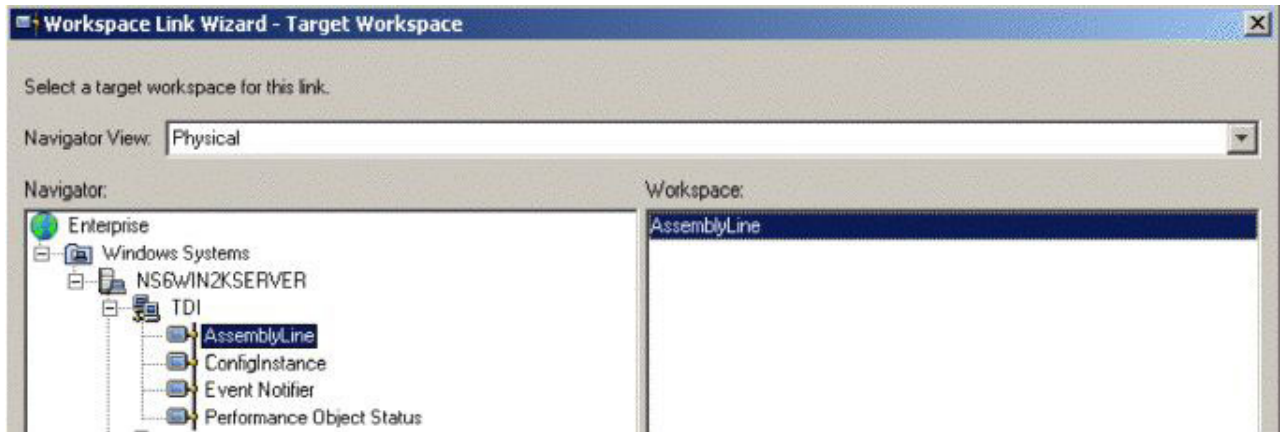


Figure 22. Link Wizard - target workspace

After selecting the AssemblyLine workspace click **Next**. This is the final step where we have to create the link conditions. We will modify two parameters in order to create the link properly - *contextIsAvailable* and *keyid*.

Select the **contextIsAvailable** parameter and click the **Modify Expression...** button (or double-click the parameter). The Expression Editor window will be displayed. Delete the current contents and click the **Symbol...** button. From the Symbols select the **Type** attribute:

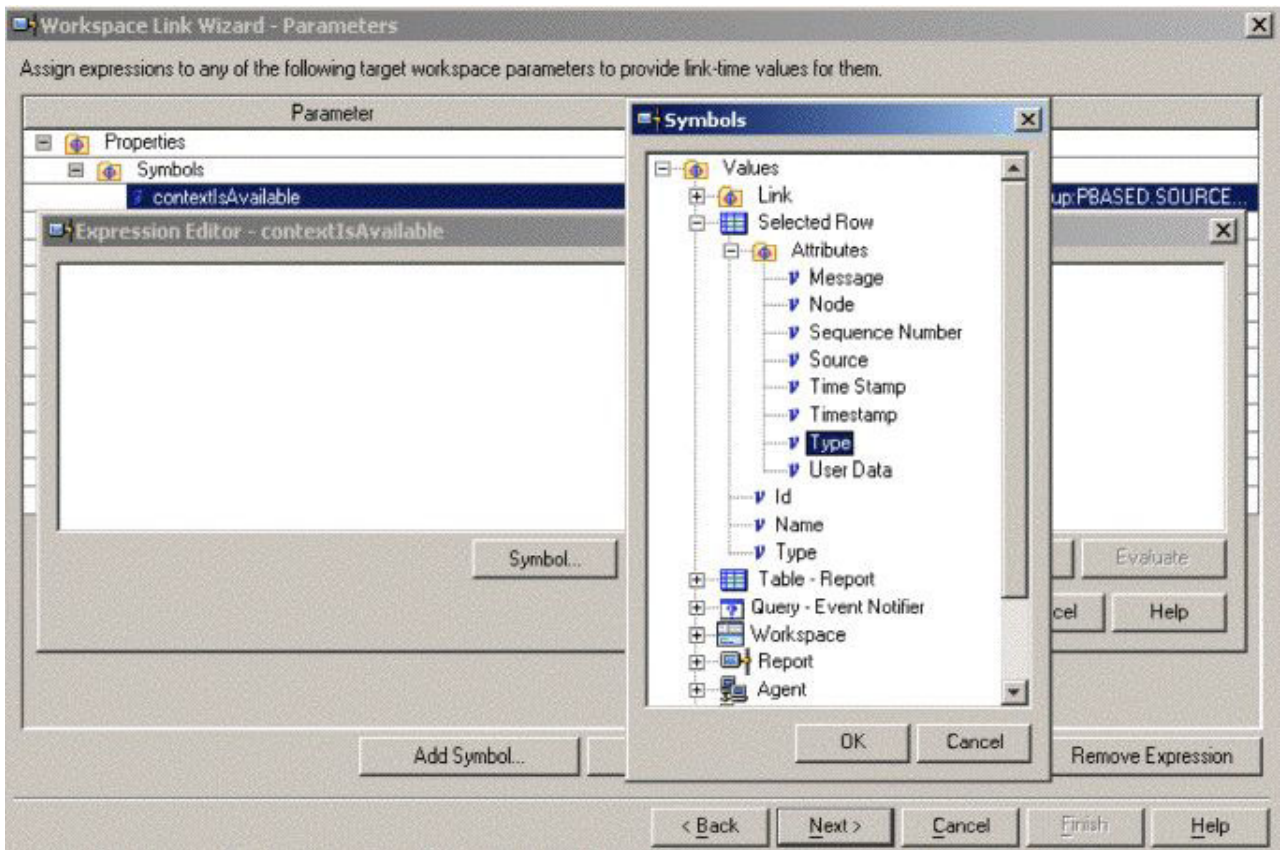


Figure 23. Link Wizard - Type attribute

Click **OK** to return to the Expression Editor window; add == "di.al.start" to create a conditional expression.

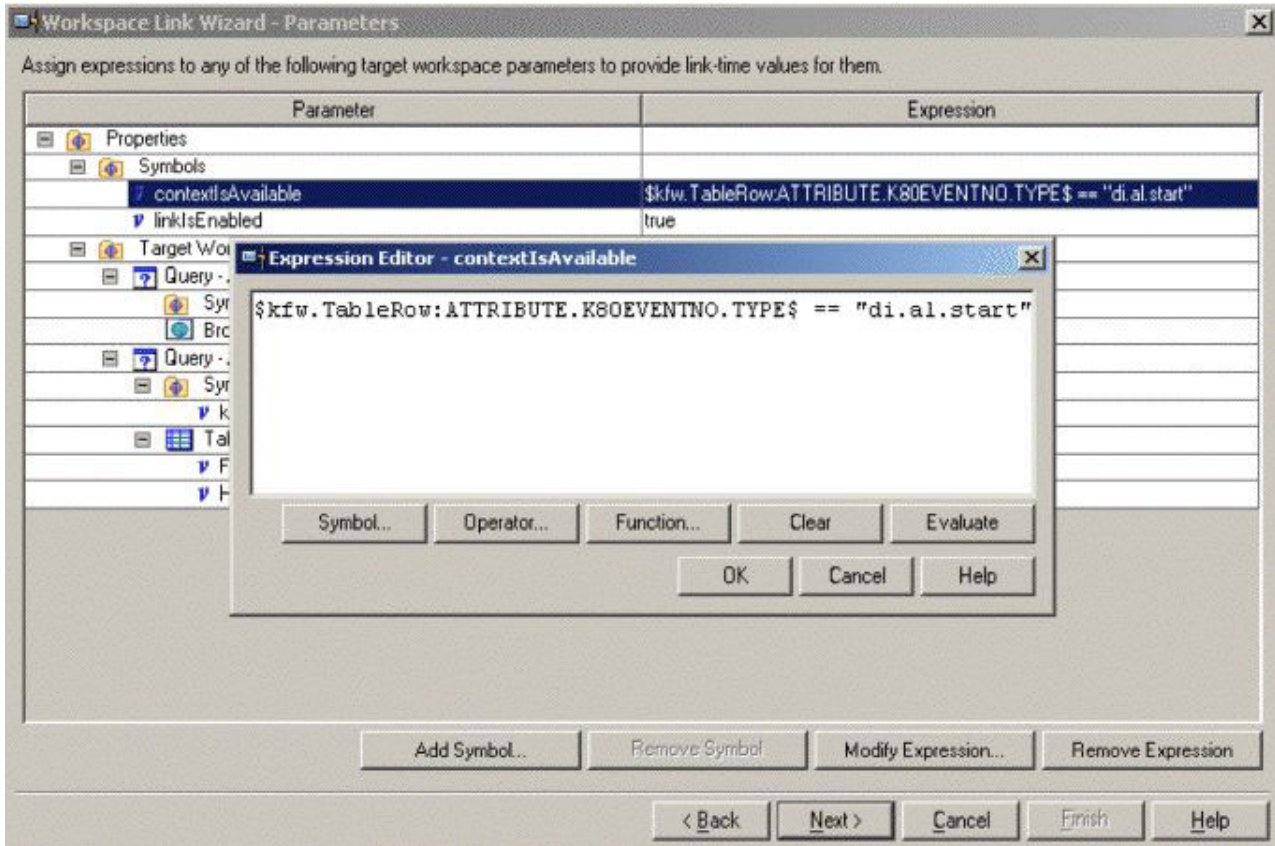


Figure 24. Link Wizard - Expression editor

Click **OK** to confirm the expression value.

Open the Expression Editor of the keyid symbol in **Query - AssemblyLine2** and add the User Data symbol.

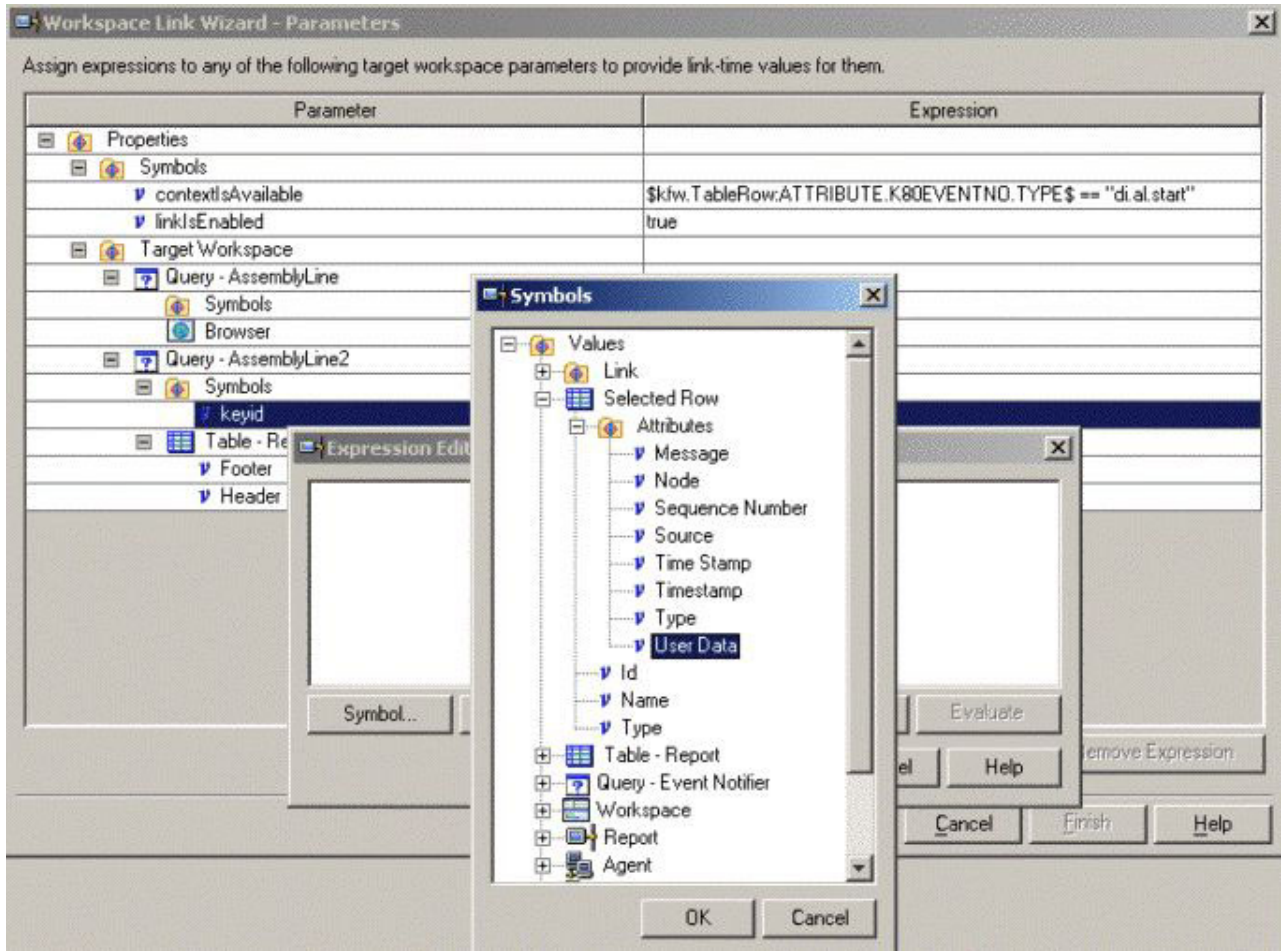


Figure 25. Link Wizard - User attribute

Click **OK** in the Expression Editor and click **Next** in the Link Wizard, which will show you a summary of the created link.

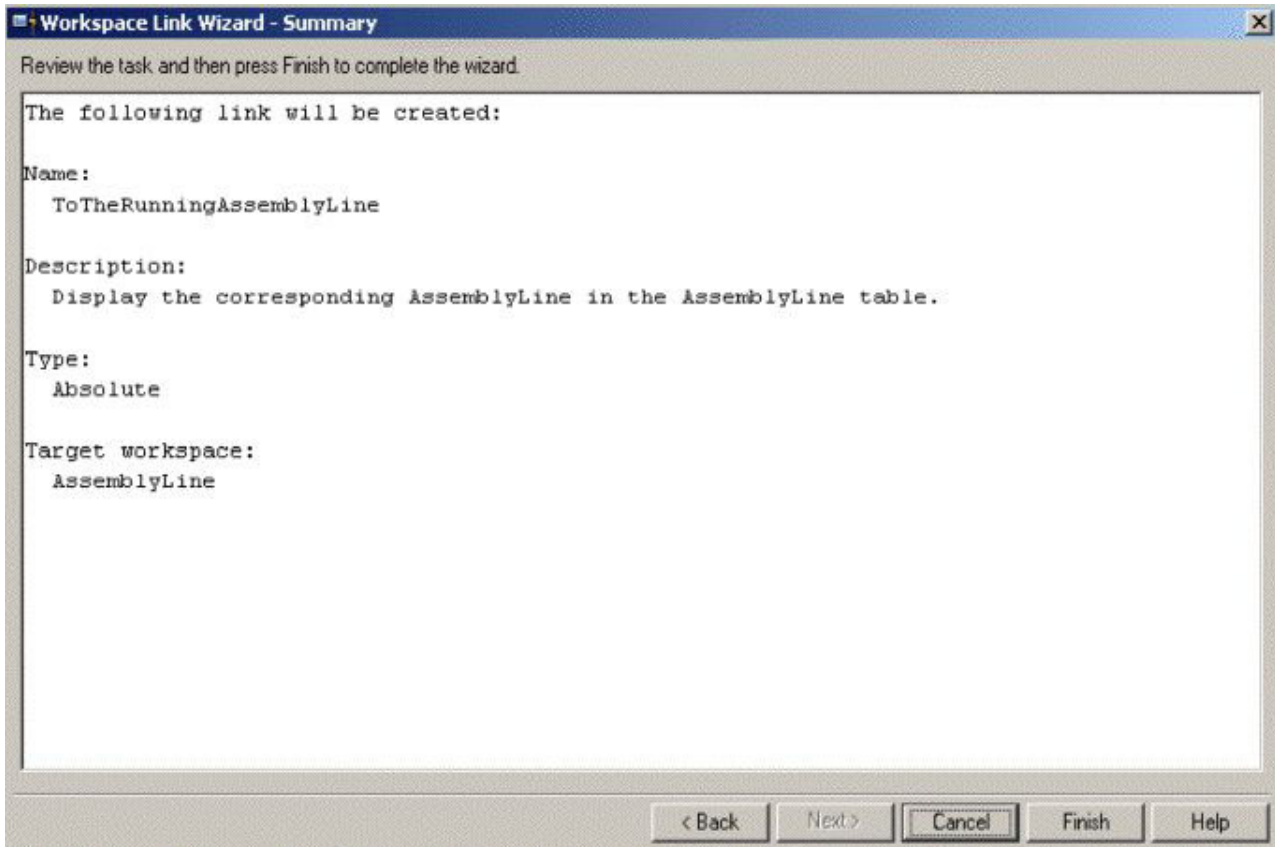


Figure 26. Link Wizard - summary

Click **Finish** to close the Link Wizard. Now you are ready to perform the steps in section "Purpose of links" on page 364.

Send custom notifications to ITM

Use the code provided here to write script for sending custom notifications.

A configuration file that demonstrates sending custom notifications is shipped with the example. The file is located at `TDI_install_dir/examples/Tivoli_Monitoring/TDI_Monitored_by_ITM/ custom_notifications.xml`.

To send custom notifications you need to write your own script that does it. The following code demonstrates it:

```
session.sendCustomNotification(aType, aId, aData);
```

This piece of code sends a custom, user defined notification to all registered listeners. The **aType** parameter is the notification type. **aId** is the notifications ID. **aData** is custom user data. Note that the aType is automatically prefixed with "user.". This means that if you send a notification of type **myType** it will be received as **user.myType**.

Limitations

You cannot use the created Agent for managing the IBM Security Directory Integrator Server.

For example it cannot start/stop AssemblyLines. It can be used for monitoring purposes only, even though the IBM Security Directory Integrator Server JMX layer exposes such methods.

Monitoring IBM SDI using OMNIBus

Refer the link provided here to learn about monitoring IBM Security Directory Integrator using OMNIBus.

Introduction

You can read more about OMNIBus in the section about the EIF Connector, in the *Reference*.

Configuring the EIF probe props file

You can set the port on which the EIF probe listens. Use the information provided here to perform this task.

In order to make sure that the port on which the EIF probe listens is the one that you expect you can set it manually.

To do this, refer to `$OMNIHOME/probes/<arch>/tivoli_eif.props` and set the value of property **PortNumber** to the number of the port on which the EIF probe will listen on.

By default if the EIF probe stays inactive (doesn't receive events) for more than 600 seconds, the service stops. You can set the timeout to infinity by setting the value of the **Inactivity** property to 0.

Your EIF prop file should look like this:

```
# BufferEvents           : "YES"
# HandleMalformedAlarms : "true"
# EIFCacheFile          : '$OMNIHOME/var/tivoli_eif.cache' (Unix)
# EIFCacheFile          : '%OMNIHOME%\var\tivoli_eif.cache' (Windows)
# EventCopies           : 1
# Inactivity            : 0
# MaxEventQueueSize     : 10000
# PortMapper            : "false"
# PortMapperNumber      : 100033057
# PortNumber            : 9998
# Retry                 : "false"
# StreamCapture         : "false"
# StreamCaptureFile     : '$OMNIHOME/var/tivoli_eif.stream' (Unix)
# StreamCaptureFile     : '%OMNIHOME%\var\tivoli_eif.stream' (Windows)
```

However, if you decide not to modify the EIF probe props file, be aware that the default port that the EIF probe listens to events is 9999 (according to the OMNIBus documentation).

Determine the severity for the events

Modify the EIF rules file to determine the severity for the events.

To determine the severity for the events you will need to do a few modifications to the EIF rules file. Here is a brief description of how you can manage the severity. For this purpose we will define any **start** events as low severity and **stop** events as high severity. Then you can type the following code in the rules file:

```

if( regmatch($ClassName, "^.*\.\start$") )
{
  @Severity = 0
}
if( regmatch($ClassName, "^.*\.\stop$") )
{
  @Severity = 4
}

```

Note that custom notifications will have the default severity which is 1. This will result in the following:

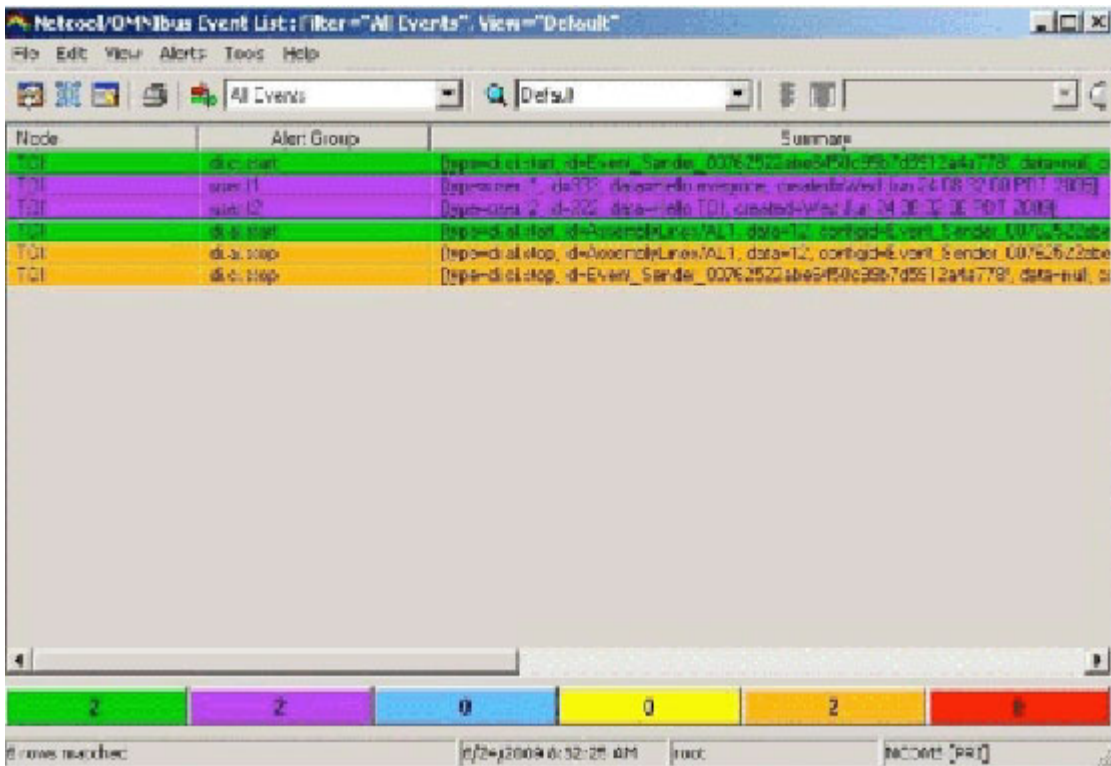


Figure 27. OMNIBus Event list

Working with the EventPropertyFile.properties file

You can do a number of things with EventPropertyFile.properties file. Learn more about it through the information provided here.

EventPropertyFile.properties provides a default set of events that can be received by the Server Notifications Connector. This will enable the user to configure the AL using the property file only. The property file has the following structure:

key=value

key determines the type of event and *value* determines if this event is received. Therefore mainly *true* and *false* values are used. However, the *event.customNotifications* key doesn't expect a Boolean value. It must be set the names of custom events that will be received. For more detailed information about custom notifications refer to Sending custom notifications to OMNIBus. A few other things should be considered as well in order to avoid any misunderstanding. To clarify, look at the following diagram showing the default set of events:


```

->event.all
|
|->event.ci.all
|   |
|   |->event.ci.start
|   |->event.ci.stop
|   |
|   |->event.ci.fileUpdated
|   |
|   |->event.al.all
|   |   |
|   |   |->event.al.start
|   |   |->event.al.stop
|   |   |
|   |   |->event.server.stop
|   |   |
|   |   |->event.hasCustomNotofications
|   |   |->event.customNotofications

```

As shown above some events include sub-events. If you enable an event then all sub-events will be received, no matter if they are set to *true* or *false*. This means that if you have

```

event.ci.all=true
event.ci.stop=false

```

then the event.ci.stop event will be received despite of it is set to *false*. In other words event.ci.all overrides its sub-events. However, if you have

```

event.ci.all=false
event.ci.stop=true
event.ci.start=false

```

then only the event.ci.stop event will be received.

By default the property file is set to provide all IBM Security Directory Integrator Server notifications. If you want to modify this set of events you need to change the Boolean values to *true* (if you want the event to be received) and *false* (if you want the event not to be received). In other words, if you want to receive all events that notify about the start of some component then your property file should look like this:

Figure 28. OMNibus Properties

For working with the property file when considering receiving custom notifications refer to the section below, "Send custom notifications to OMNibus" on page 374.

Send custom notifications to OMNIBus

You can customize the notifications sent to OMNIBus. Follow the steps listed here to perform this task.

To receive custom notifications you must set `event.hasCustomNotofications` to `true`. Then you need to specify the set of events that will be received. Note that all custom events sent by IBM Security Directory Integrator are prefixed with "user.". This means that if you send a custom event of type `myType` then you must set:
`event.customNotifications=user.myType`

To specify more than one custom event you can use ";" to separate them. To clarify, imagine the following situation. You want to receive all custom notifications of type "user.myType1", "user.myType2", "user.myType3". Then your property file opened with a text editor will look like this:

```
##Determine if Server Shutdown events are received
event.server.stop=false
##Determine what Custom Notification events are received
##This property is used only if event.hasCustomNotofications is enabled
##Note that all custom notifications are prefixed with "user."
event.customNotifications=user.myType1;user.myType2;user.myType3
##Determine if Custom Notification events are received
event.hasCustomNotifications=true
```

In order to receive any types of custom events the `event.customNotifications` value need to be set to "*". This will not specify the type of custom events that the Connector will listen to, therefore any custom event detected will be manipulated. The ITM example provides a configuration that can send custom notifications. It can be used to send custom notifications to OMNIBus, too. For more information about custom notifications refer to section "Send custom notifications to ITM" on page 370.

Appendix C. Accessibility features for IBM Security Directory Integrator

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Security Directory Integrator:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

The IBM Security Directory Integrator product documentation, and its related publications, are accessibility-enabled. The accessibility features of the product documentation are described at http://pic.dhe.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.iehsc.doc/iehs34_accessibility.html.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys for common Windows actions such as access to the File menu, and to the copy, paste, and delete actions. Actions that are unique use keyboard shortcuts. Keyboard shortcuts have been provided wherever required for all actions.

Interface information

The accessibility features of the user interface and documentation include:

- Steps for changing fonts, colors, and contrast settings in the Configuration Editor:
 1. Type **Alt-W** to access the Configuration Editor **Window** menu. Using the downward arrow, select **Preferences...** and press Enter.
 2. Under the **Appearance** tab, select **Colors and Fonts** settings to change the fonts for any of the functional areas in the Configuration Editor.
 3. Under **View and Editor Folders**, select the colors for the Configuration Editor, and by selecting colors, you can also change the contrast.
- Steps for customizing keyboard shortcuts, specific to IBM Security Directory Integrator:
 1. Type **Alt-W** to access the Configuration Editor **Window** menu. Using the downward arrow, select **Preferences...**
 2. Using the downward arrow, select the General category; right arrow to open this, and type downward arrow until you reach the entry **Keys**.
Underneath the **Scheme** selector, there is a field, the contents of which say "type filter text." Type security directory integrator in the filter text field. All specific IBM Security Directory Integrator shortcuts are now shown.

3. Assign a keybinding to any IBM Security Directory Integrator command of your choosing.
4. Click **Apply** to make the change permanent.

The Configuration Editor is a specialized instance of an Eclipse workbench. More detailed information about accessibility features of applications built using Eclipse can be found at <http://help.eclipse.org/help33/topic/org.eclipse.platform.doc.user/concepts/accessibility/accessmain.htm>

- The product documentation and its related publications are accessibility-enabled for the JAWS screen reader and the IBM Home Page Reader. You can operate all documentation features using the keyboard instead of the mouse.

Vendor software

IBM Security Directory Integrator includes certain vendor software that is not covered under the IBM license agreement. IBM makes no representation about the accessibility features of these products. Contact the vendor for the accessibility information about its products.

The installer uses the InstallAnywhere 2012 SP1 installer technology.

Related accessibility information

Softcopy Adobe Portable Document Format (PDF) documentation is also available, as an alternative to the online product documentation. You can view the PDF publications by using Adobe Acrobat Reader. With PDF documentation, you can use optional font enlargement, high-contrast display settings, and can navigate by keyboard alone; however, in this case, alternative text is not provided for screen-reader users.

You can access or download the PDF publications for IBM Security Directory Integrator from the IBM Security Directory Integrator documentation."

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law :

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to

IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.



Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Index

Special characters

.registry file
 components 55

A

access
 server API 110
accessibility ix, 375
accessibility features for this product 375
ACL
 configuration 263
Action Manager 239
 add 272
 Add/Modify action 274
 AMC 249
 command line utility 283
 configuration rules 272
 Configure trigger 273
 Configured actions 274
 delete 272
 Derby database 249
 enable 249
 event data 278
 modify 272
 Monitor Status 267
 remotely 240
 Results Table 271
 shutdown 240
 start up 240
 status 250
 substitute variable 278
 triggers 280
 event data 279
 window 250
Active Directory 86
ActiveMQ
 logging 159
add feature
 panel flow 36
Add/Modify action
 Action Manager 274
adding
 server 260
Administration and Monitoring Console
 configuration 237
 installation 237
administrator access 5
administrator privileges 5
agent configuration
 XML file 350
algorithm
 encryption 95
AMC 3, 237, 239, 240, 283
 action manager
 actions 243
 threads 243
 triggers 243
 Action Manager 237, 250, 251

AMC (*continued*)
 amc.properties 255
 config files 260, 267
 configuration 239, 242
 console properties 261
 console user authority 242
 Custom Authentication 253
 custom load 267
 deferred deployment 44
 Derby 239
 encrypted configs 255
 features 251
 filtering
 tables 259
 finding
 tables 259
 force trigger 250
 general information 44
 IBM Security Directory Integrator server 253
 Integrated Solutions Console 237
 ISC 242
 LDAP properties 239
 logging off 257
 login 255
 logout 255
 logs 242
 navigation area 257
 navigation links 242
 non SSL 253
 paging 258
 password 255
 remote servers 253
 role management 254
 Select action 258
 server
 adding 260
 modification 261
 servers 260
 solution view 263
 sorting 258
 SSL 251, 253
 SSL keystore 251
 SSL properties 239
 SSL truststore 251
 table pages 258
 tables 257, 258
 user interface 255, 257, 258
 web platform deployment 44
 work area 257
AMC authentication
 non-root user 7
AMC deployment
 ISC 47
AMC installation 47
Apache ActiveMQ
 parameters 158
Apache Derby
 networked mode 199
applyUpdates.bat(sh) 53
assemblyline 151

AssemblyLine 83, 87
 View Rules Summary 280
AssemblyLines 81, 230
auditing capability
 principles 129
auditing scope 129
authentication
 HTTP 147
Authentication 164
authentication hook
 Username/password based authentication 115
authorization roles 124

B

back up
 important data 78
Backing up
 Apache Derby databases 201
backup
 upgrade 7.0 to 7.1 79
 upgrade 7.1 to 7.1.1 79
 upgrade from 6.0 to 7.1 78
 upgrade version 6.1.x to 7.1 78
backup tools
 manual migration
 backupam/restoram 79
 backupamc/restoramc 79
 backupamcdb/restoramcdb 79

C

CE 50
CE update site
 Eclipse deployment 44
certificate
 PKI 179
 SSL 179
certificate creation 148
certificates
 CA 180
 CA signed 181
 digital 180
 PKI 180
 self signed 181
 SSL 180
Change Detection Connector
 EventHandler 86
Changelog Connector
 EventHandler 84
command line
 installation 41
Command Line Interface
 Remote Server API 209
 tdisrvctl utility 209
Command Line Reference
 general options 210
command line tool
 cryptoutils 135

- command line tool (*continued*)
 - editing configuration file 135
 - encryption 135
 - Command-line options
 - CE 205
 - Command Line Interface – tdisrvctl utility 205
 - Server 205
 - command-line parameters
 - cryptoutils 136
 - component password 140
 - components
 - available 3
 - config file feature 140
 - config files 142
 - solution view 266
 - configuration
 - ActiveMQ
 - parameters 158
 - AMC logs 242
 - Apache Derby Instances 199
 - interval 186
 - load time-out 186
 - microsoft active directory 103
 - password 140
 - PKI 179
 - properties 187
 - solution directory 187
 - SSL 103, 179
 - system store 199
 - configuration editor 3, 8, 143
 - AIX 7
 - Eclipse 205
 - perspective options 205
 - properties 141
 - RPM 7
 - shutdown servers 205
 - tombstone 325
 - Configuration Example
 - System Queue 163
 - configuration files
 - encryption 133
 - summary 144
 - configuration rules
 - Action Manager 272
 - configuration settings
 - migration 80
 - configuration window
 - AssemblyLine 325
 - configurations 81
 - Configure trigger
 - Action Manager 273
 - Configured actions
 - Action Manager 274
 - configuring certificates
 - PKI 181
 - SSL 181
 - connectors 170
 - console 230
 - console properties
 - general 261
 - JDBC 261
 - SSL 261
 - create key 96
 - create keys 93
 - Create statements
 - System Store 200
 - cryptographic keys 94
 - hardware devices 182
 - JRE 182
 - PKCS 182
 - RSA 182
 - SSL 182
 - cryptoutils
 - command line tool 135
 - command-line parameters 136
 - editing configuration file 135
 - encryption 135
 - CryptoUtils
 - decrypt 180
 - encrypt 180
 - Custom Authentication
 - non SSL 253
 - SSL 253
 - custom load
 - config files 267
 - custom notifications
 - OMNIBus 374
- ## D
- DB2
 - JDBC connection parameter 197
 - table statements 197
 - default installation
 - location 51
 - default parameters 229
 - deployment
 - AMC 237
 - existing environment 238
 - UNIX process 238
 - WebSphere Application 238
 - windows service 238
 - migration 45
 - derby
 - user authorization 138
 - Derby 240
 - RDBMS 198
 - System Store 198
 - Details Table
 - Solution View 269
 - disability 375
 - disk space requirements
 - link 3
 - DNS names
 - configuration
 - MQ Everyplace 166
 - documentation
 - local 45
 - online 45
 - software requirements 59
 - DSMLv2 87
- ## E
- Eclipse Update Manager
 - installation 48
 - updatation 48
 - education ix
 - EIF probe
 - OMNIBus 371
 - port 371
 - EIF rules file
 - OMNIBus 371
 - embedded web platform 3
 - encrypted configs
 - AMC 255
 - encrypted data
 - migration 63
 - encryption
 - configuration file 135
 - configuration files 133
 - global.properties 135
 - public/private key 95
 - public/private keys 133
 - RSA 133
 - secret key 95
 - server hooks 142
 - solution directory 142
 - solution.properties files 135
 - Symmetric cipher support 169
 - Encryption 169
 - encryption artifacts 183
 - encryption key 183
 - encryption utility 136
 - error action 151
 - event data
 - triggers 279
 - Event Notifier table 364
 - EventHandler 83, 84, 85, 87
 - EventHandlers 81
 - EventPropertyFile.properties file
 - structure 372
 - example
 - creating Solution View 288
 - exceptions
 - JDBC connector 154
 - LDAP connector 154
 - expired certificate 183
- ## F
- file backup
 - file list 78
 - FileAppender 229
 - Fiorano MQ system
 - JavaScript
 - configuration 163
 - FIPS
 - auxiliary tools 178
 - compliance
 - rules 172
 - configuration 169, 179
 - createtash 178
 - cryptoutils 178
 - encryption 170
 - keytool/Ikeyman 179
 - FIPS mode 169
 - fix pack 55
 - force trigger
 - AMC 250
 - function components 170
- ## G
- general concepts 1
 - graphical installer
 - installation 12

- graphical installer (*continued*)
 - instructions 12
- graphics packages
 - CE 7
 - UNIX systems 7

H

- help system 3
- High Availability
 - configuration 167
- host based authentication
 - global.properties 120
 - solution.properties 120
- HTTP 84, 295
- HTTP basic authentication 147

I

- IBM
 - Software Support ix
 - Support Assistant ix
- IBM SDI
 - client 107
 - debugging 223
 - EIF Connector 371
 - JMS messaging system 157
 - JMX interface 185
 - logging 223
 - OMNibus 371
 - server 107
 - server API 185
 - system queue 157
- IBM SDI data
 - monitoring 359
- IBM SDI installation 339
- IBM Security Directory Integrator
 - web service suite 148
- IBM Security Directory Integrator
 - installation
 - platform specific 12
- IBM Security Directory Integrator server
 - configuration files 131
 - cryptographic modules 171
 - ECB 131
 - encryption algorithms 131
 - FIPS 171
 - local client session 113
 - RSA 131
 - server API authentication 113
- IBM Security Directory Integrator service
 - ibmdiservice.props 333
 - log service 333
 - logging 333
 - properties 333
 - property file 331
 - service logs 333
 - service name 331
 - start script 337
 - starting 333
 - stop script 337
 - stopping 333
 - UNIX systems 337
 - windows service 331, 333
 - installation procedure 332
 - installation steps 332

- IBM Security Directory Integrator
 - service (*continued*)
 - windows service (*continued*)
 - uninstallation procedure 332
 - uninstallation steps 332
 - windows service installation 331
 - windows service uninstallation 331
 - windows systems 337
- IBM Security Directory Integrator services
 - i5/OS 331
 - Linux/Unix 331
 - windows 331
 - z/OS 331
- IBM Tivoli Monitoring Agent 351
- IBM Tivoli Monitoring Agent editor 357
- IBM websphere 149
- IBM WebSphere
 - MQ Everywhere
 - parameters 160
 - MQ parameters 160
- IBMPCKS11
 - certificates 183
 - SSL keys 183
- important data
 - back up 78
- installation 5
 - command line 41
 - instructions 3
 - panel flow 12
 - system requirement 3
- installation and administration 1
- installation location
 - Linux and Unix 51
 - windows 51
- installer
 - migration
 - automatic 64
 - manual 64
- installing fixes
 - components 57
 - manual steps 57
- installing help file 45
- installing IBM SDI 8
- Instance Configuration 310
- Integrated Solutions Console
 - deployment 237
- integration external tools 349
 - Tivoli monitoring 349
 - Tivoli Netcool/OMNibus 349
- Intermediary
 - Touchpoint Instance 323
- ITM
 - agent 350
 - architecture 350
 - AssemblyLine 364
 - AssemblyLine table 365
 - configuration file 370
 - custom notifications 370
 - data collection 350
 - eclipse 351
 - importing 350
 - ITM agent 350
 - link construction 365
 - monitoring 350
 - tables
 - links 364

- ITM agent
 - configuration 357, 358
 - deployment 357
 - generation 357
 - IBM SDI server 371
 - import 350
 - limitation 371
- ITM agent builder 358
- ITM agent builder 6.2 350, 351

J

- java API documentation 3
- Java properties 157
- Java system property 112
- JDBC 170
- JLOG
 - parameters 235
- JLOG logger 234
- JLOG-based 341
- JMS 170
- JMS driver 157
 - JavaScript 162
- JMS Driver
 - JMSScript Driver parameters 161
 - ret JavaScript 162
- JMX Connector 83
- JMX layer
 - server API 121
- JRE 3, 6
- JSSE 147

K

- keys
 - certificate 96
- keystore 94, 96, 183
 - JCEKS 94
 - JKS 94
 - list command 96
 - PKCS#12 94
- keytool 96

L

- launching installer
 - direct installer 9
 - launchpad 9
- launching uninstaller
 - procedure 50
 - results 50
- LDAP 85
- LDAP authentication
 - global.properties 116
 - password 118
 - solution.properties 116
 - username 118
- LDAP authentication support 116
- LDAP connector
 - SSL 105, 106
- LDAP group support
 - authentication process 118
 - group 118
 - user registry 118
 - users 118
- link creation 364

- Linux/Unix service
 - deployment methods 335
 - shutdown 335
 - tailoring 335
- list command 93, 96
- local client session
 - IBM Security Directory Integrator server 113
 - JVM 113
- local variable
 - solution view 264
- Log Level control 228
- Log Levels 228
- Log Management
 - Solution View 281
- log strategies 230
- Log4J 229
- logging
 - AssemblyLine 224
 - CE 224
 - default Log4J class 224
 - FFDC 233
 - script-based 224
 - Tracing 233
- logging in 239
- Lotus Domino 147

M

- Mailbox Connector 83
- manage keys 93
- Manage Tivoli Monitoring Enterprise Services 358
- manual backup
 - Derby database files 80
 - LDAP 80
 - OSGI 80
 - queue manager files 80
 - SCIM 80
 - workspace files 80
- manual migration 65
 - backup tools 79
 - components 65
 - configurations 65
 - files 65
 - properties 65
 - scripts 65
- microsoft active directory 103
- migration
 - another location 61, 62
 - BTree Connector 88
 - BTree tables 88
 - Cloudscape database 88
 - components 61
 - configuration settings 80
 - deployment 45
 - Derby 88
 - encrypted data 63
 - file 61
 - file types 61, 62, 63
 - installer
 - automatic 64
 - manual 64
 - installer-assisted
 - manual 65
 - modification 62
 - newer version 64

- migration (*continued*)
 - panel flow 39
 - scenario 61
 - scripts 63
 - System Store 88
 - ways to migrate 64
 - workspace 63
- mini-certificates 149
- modification
 - server 261
- Monitor Status
 - Action Manager 267, 268
 - Health Check 268
 - Health Check Result 268
- MQ Everyplace 164
 - configuration utility 164, 167
 - Mini-Certificate Server 165
 - password changes 167
- MQ Everyplace authentication 149
- MQ Queue Manager 164
- MS SQL Server
 - JDBC connection parameter 196
 - table statements 196

N

- non-silent installation 43
- notification
 - notification types 129
 - suppression 129

O

- OMNIBus
 - custom notifications 374
- operations
 - configFile 211
 - event 211
 - prop 211
 - queryop 211
- oracle
 - JDBC driver
 - client library 196
 - connection parameters 196

P

- padding
 - disable 183
 - enable 183
- panel flow
 - add feature 36
 - installation 12
 - migration 39
 - uninstallation 31
- parameter type 140
 - password 140
- parameters
 - MQ Everyplace 160
- parsers 170
- password configuration 140
- password protected Configs
 - exception 186
- password protection 142
- password store 141
- password synchronization 3

- password synchronization (*continued*)
 - plug-ins 44
- password synchronizer 55
- Persistence
 - Touchpoint Instance 305
- PKCS#11 104
- platform specific
 - installation 12
- plug-ins
 - password synchronization 44
- post installation
 - steps 44
- post-installation 50
- private keys 94
- problem-determination ix
- properties
 - CE 187
 - global 188
 - Java 189
 - Jlog file 189
 - JVM 189
 - PKCS# 104
 - Solution 189
 - solution directory 187
 - SSL 104
 - System 190
- Property
 - global 280
 - Java 280
 - solution 280
- property file 372
 - Derby parameters 342
 - derby.properties 342
 - global.properties 342
 - ibmdisrv 340
 - jlog.properties 341
 - Log options 340
 - Log4J.properties 340
 - solution directory 342
- property files
 - encryption
 - property files 136
 - external 136
 - Solution Directory 339
- Property stores
 - Password Store 194
 - User property stores 194
- Property Stores
 - Global Properties 280
 - Java Properties 280
 - Solution Properties 280
 - Solution View 280
- provisioning protocol 295
- public key certificate 94, 96
- public keys 94

R

- reconnect action 151
- reconnect rule
 - CE 155
 - configuration 155
- reconnect rule engine 151
- reconnect rules
 - built-in 151
 - user-defined 151

- refreshing
 - Solution View Details 272
- remote CE
 - restrictions 143
- remote CE limitations 143
- remote client session
 - authentication methods 113
 - server API authentication 113
- remote configuration
 - MQ Everyplace 167
- remote configuration editor 142
- remote server API 107
- Remote Server API 110
- requirements 5
- Results Table
 - Action Manager 271
- RHEL 6
- RMI 107
- role management
 - admin 254
 - config admin 254
 - execute 254
 - read 254
- rollback
 - update installer 57
- RPM
 - AIX 7
- runtime server 3

S

- SDI
 - editions 1
 - SDI loader 50
- secret keys
 - keystore 94
 - SSL 94
- security
 - API 93
 - config files 93
 - manage keys 93
 - SSL 93
 - web admin console 93
- Security
 - Encryption 164
- security aspects
 - various 147
- security concepts 94
- Security Directory Server 84
- security enhanced 6
- security properties
 - summary 144
- security tools
 - Ikeyman 96
 - JVM 96
 - keytool 96
- Select action
 - AMC 258
- SELinux 6
- sending notification
 - delivery parameters 130
 - listener 130
- Server
 - command line options 206
 - IBM SDI 206
- server API 111, 112, 116
 - access 110
- server API (*continued*)
 - authentication 122
 - configuration 108
 - examples 122
 - properties 108
 - user registry 125
- Server API
 - authentication 113
 - JMX 121
 - JMX layer 121
- server API authentication
 - JAAS authentication 113
 - remote client session 113, 114
 - SSL based authentication 114
- server API authorization
 - client server API session 123
 - remote API 123
- Server API security model 124
 - authorization roles 124
- server audit capabilities
 - authentication 128
 - authorization 128
 - notifications 128
- server authentication
 - host based 121
 - LDAP 121
 - SSL-based 121
 - username/password based 121
- server connector
 - TCP 83
- Server Connector 84, 85, 87
- server ID
 - AMC 185
 - IP address 185
- Server Notifications Connector 372
- Server RMI
 - SSL access 186
- server security mode
 - secure 132
 - standard 132
- service name
 - UNIX 43
- severity of events 371
- Show Preferred
 - Solution Views 271
- silent installation 43
- silent uninstallation
 - console uninstallation 51
 - GUI 51
- SNMP Server Connector
 - AssemblyLine 83
 - EventHandler 83
- SOAP 87
- solidDB
 - JAR 197
 - JDBC connection parameter 197
 - table statements 197
- solution directory
 - server hooks 142
- Solution Directory 164
- solution view
 - add 263
 - addition 264
 - config files 266
 - configuration file 264
 - configure users 263
 - local variable 264
- solution view (*continued*)
 - modify 263
- Solution View
 - Details Table 269
 - Monitor Status 268
 - Preferred 282
- Solution View Details
 - refreshing 272
 - View components 271
- Solution Views
 - Show Preferred 271
- SSL 94, 112, 142, 170
 - ActiveMQ 159
 - client 101, 105, 106
 - client authentication 102
 - configuration 99, 103, 147
 - connectors 99
 - example 105
 - IBM Security Directory Integrator
 - component 105, 106
 - IBM Security Directory Integrator
 - components 100, 101
 - JSSE 112
 - JVM 112
 - keys 95
 - keystore 95, 100, 102
 - LDAP connector 105, 106
 - local access 111
 - properties 104
 - remote access 111
 - server 100, 105
 - system property 112
 - truststore 95, 100, 101, 102
- SSL based authentication 142
- SSL client authentication 142
- starting 239
- stash file
 - keypassword 132
 - keystore password 132
 - server instance security 132
- Sun Directory Change Detection
 - Connector 85
- supported platforms
 - link 59
- Symmetric cipher support
 - encryption 169
- system memory requirement
 - link 3
- system platform requirement
 - link 3
- System Queue
 - Configuration Example 163
 - Microbroker parameters 161
- System Queue Configuration 157
- system store
 - Cloudscape 191
 - JVM 191
 - solution directories 191
- System Store
 - DDL 195
 - JDBC Driver 195
 - RDBMS 195
 - user authentication 199
- system store security
 - Built-in Derby Users 138
 - External Directory Service 138
 - User-defined class 138

T

- tables 364
 - AMC 259
- TCB 142
- TCP
 - server connector 83
- temporary file space
 - UNIX/LINUX 43
 - windows 43
- TEP agents 359
- threshold
 - assemblyline table 360
 - defining 360
 - example 360
- Tivoli Enterprise Portal 359
- Tivoli monitoring 349
- Tivoli Netcool/OMNIBus 349
- tombstone
 - AssemblyLine 327
 - attributes 327
 - configuration 327
 - configuration editor 325
 - configuration window 325
 - records 327
 - statistics 327
- tombstone manager
 - AssemblyLines 325
- Tombstone Manager
 - AssemblyLines 328
 - Config instance 328
 - configuration properties 328
- tombstones
 - configuration 325
 - configuration editor 325
 - switches 325
- touchpoint 295
- Touchpoint
 - authentication 319
 - communication protocol 312
 - configuration 295
 - Entry objects 313
 - HTTP 312, 313, 314
 - HTTP content 313
 - HTTP server 319
 - Initiator 313
 - Instance 312
 - instances 295
 - Intermediary 313
 - location 316
 - Property sheet
 - definitions 315
 - Provider 312
 - RMI Server API 319
 - server 295
 - Status Entry 314
 - Status Entry schema 314
 - Touchpoint Instances 313
 - XML Schema 313, 316
- Touchpoint Configuration
 - namespace 310
 - Touchpoint Instance 310
- Touchpoint instance
 - HTTP 298
 - initiator 298
 - intermediary 298
 - provider 298
- Touchpoint Instance 310

- Touchpoint Instance (*continued*)
 - Entry resource 320, 321
 - example 319
 - HTTP POST 320
 - Initiator 319, 321
 - Intermediary 323
 - Provider 319, 320
 - Resource Persistence 305
 - resources 305
 - Shipped example 319
 - URL 320
- Touchpoint provider
 - instances 296
 - JVM 296
 - Touchpoint Server 296
- Touchpoint Schema
 - HTTP 306
 - instances 306
 - resource 306
 - schema tree 306
 - server 306
- Touchpoint server
 - access 295
 - components 295
 - configuration 317
 - HTTP basic authentication 317
 - ReSTful communication protocol 295
 - web container 317
- Touchpoint Server
 - Touchpoint provider 296
- Touchpoint Template
 - AssemblyLines 301
 - initiator 301
 - IntermediaryHandler 301
 - ProviderHandler 301
 - resources 301
- touchpoint type
 - custom 296
 - standard 296
 - virtual 296
- Touchpoints
 - Atom document 311
 - Destination Configuration 311
 - Error flows 316
 - XML document 316
- trace levels 234
- trace properties
 - dynamically 234
- tracing 142
 - configuration 234
 - JLOG log level 234
 - JLOG's PDLogger object 233
 - JlogSnapHandler 233
 - SnapMemory 233
- Tracing Enhancements
 - connectors 233
 - parsers 233
- training ix
- troubleshooting ix
- Troubleshooting
 - Apache Derby issues 201

U

- uninstallation
 - IBM Security Directory Integrator 50
 - panel flow 31

- UNIX
 - service name 43
- update installer 53, 55
 - rollback 57
 - troubleshooting 58
- update site 3
- upgrade 8
 - version 7.1.1 to 7.2 79
- user authentication
 - System Store 199
- user registry
 - server API 125
- user-defined rules
 - examples 153
 - format 153
- Username/password based authentication
 - authentication hook 115

V

- View components
 - Solution View Details 271
- VPN
 - properties 110

W

- walkthrough
 - creating Solution View 288
- Web Admin Console Security
 - details 147
- windows operating system 8
- windows service 331



Printed in USA

SC27-2705-03

