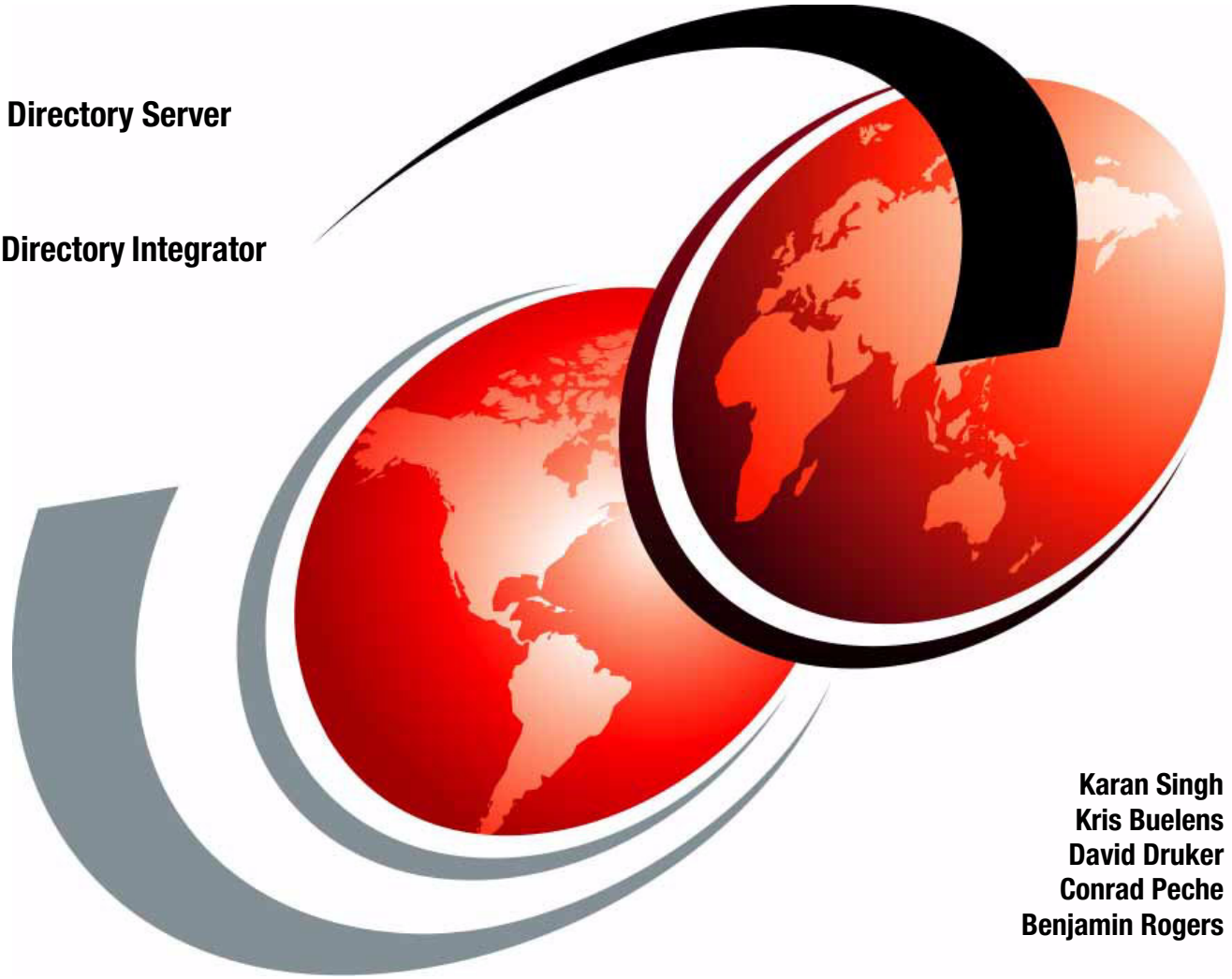
IBM

Synchronizing IBM RACF data by using IBM Tivoli Directory Integrator

- RACF password synchronization
- IBM Tivoli Directory Server
- IBM Tivoli Directory Integrator



Karan Singh
Kris Buelens
David Druker
Conrad Peche
Benjamin Rogers

Redpaper

ibm.com/redbooks

International Technical Support Organization

Synchronizing IBM RACF data using IBM Tivoli Directory Integrator

May 2013

REDP-4460-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (May 2013)

This edition applies to Version 1, Release 10, of IBM z/OS (product number 5694-A01), Version 5, Release 4, of IBM z/VM (product number 5741-A05), Version 7, Release 1, Modification 1 of Tivoli Directory Integrator (product number 5698-A82).

This document created or updated on July 25, 2016.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

DB2®
Global Technology Services®
IBM®
MVS™
RACF®
RDN®
Redbooks®
Redpaper™
Redbooks (logo) ®
Tivoli®
WebSphere®
z/OS®
z/VM®

The following terms are trademarks of other companies:

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redpaper™ publication provides an example of a solution to synchronize an IBM RACF® user ID, password, and password phrase data between IBM z/OS® and IBM z/VM® systems, or just between z/VM systems. Topics that are covered are the installation and customization of IBM Tivoli® Directory Integrator, IBM Tivoli Directory Server, and RACF. Using this basic infrastructure, a sample Tivoli Directory Integrator configuration is presented, which allows for a flexible and extensible means for synchronizing RACF information.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.

Karan Singh is a Project Leader at the International Technical Support Organization, Poughkeepsie Center.

Kris Buelens is an Advisory IT Specialist working in IBM Global Technology Services® (GTS) in Belgium. He has been a specialist in z/VM more than 28 years. For the last 20 years, he was the key person in the VM system support group of a major Belgian bank. As a result, he has much experience in managing, tuning, automating, and installing of z/VM in a distributed z/VM environment. Knowing REXX and CMS Pipelines as a second language, he is known for his numerous VM tools, many of which are popular on the VM www download library.

David Druker is an IBM Security Architect and works with IBM clients in the central and western United States. He is known worldwide as an expert in IBM Tivoli Directory Integrator and designed many solutions around this product. He has over 20 years of broad technology experience in security, programming, and enterprise architecture. David is a Senior Certified IT Specialist and holds a Ph.D. in Speech and Hearing Science from the University of Iowa.

Conrad Peche is a Systems Programmer with IBM Australia Development Labs (ADL) in Perth, Western Australia. He has been with ADL in various roles, including systems support and software development, since 2002. He is team lead of the Infrastructure Support group. He has over 30 years of mainframe experience, most as a z/OS or z/VM Systems Programmer both at IBM and as a client.

Benjamin Rogers, during the writing of this publication, was an Advisory Software Engineer for IBM Systems and Technology Group, z Lab Services organization.

Thanks to the following people for their contributions to this project:

Rich Conway and Roy Costa
IBM International Technical Support Organization, Poughkeepsie Center

Bruce Wells
IBM z/OS Security Server Design and Development

Mark Andrews and Robert E. Hart
IBM Australia Development Laboratory

Brian Hemric, John McGarvey, Michael Suszko, Jason Todoroff, and Jason Williams
IBM Tivoli

Mark McConaughy and John A. McKeeking
IBM Systems and Technology Group

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks® publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks
- Send your comments in an email to:

  redbooks@us.ibm.com
- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:

  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction

This chapter presents an overview of the software and methods that we use to create a solution to synchronize data between Resource Access Control Facility (RACF) databases. An overview is also provided of the implementation that is presented in the rest of this publication.

1.1 Overview

Automated synchronization of RACF data between systems can reduce the amount of time that administrators spend manually keeping data synchronized between RACF databases. RACF remote sharing facility (RRSF) can be used to synchronize RACF data between z/OS RACF databases but not between z/OS and z/VM RACF databases. Furthermore, RRSF is not available on z/VM to allow synchronization between z/VM RACF databases.

The objective of this document is to describe a method for synchronization of RACF user-related information, including passwords, between any combination of z/VM and z/OS operating system platforms. We provide a sample solution to show how RACF data can be propagated between RACF databases. There is a focus on synchronizing passwords and password phrases between z/OS and z/VM systems, by using IBM Tivoli Directory Integrator. In addition, to demonstrate that additional information can be synchronized, we propagate changes that are made to the name field of the user profile.

The software that is used to achieve this synchronization is RACF, IBM Tivoli Directory Server, which is an implementation of the Lightweight Directory Access Protocol (LDAP); IBM Tivoli Directory Integrator; and IBM WebSphere® MQ. See 1.3, “Software inventory” on page 4 for a list of program numbers, names, and releases required. These products are referred to by their acronyms for ease of reference.

In broad terms, RACF through LDAP provides change notifications to Tivoli Directory Integrator. Tivoli Directory Integrator then runs scripts (known as *AssemblyLines*) to drive corresponding RACF changes through LDAP on the target systems to synchronize user information.

The method that is employed on z/OS and z/VM use the RACF and LDAP (GDBM) event notification capability. This capability creates user, group, and connection change log records (notifications that a change occurred), along with password and password phrase enveloping to enable extraction of encrypted password data by authorized LDAP clients.

Tivoli Directory Integrator extraction of user information and modification is achieved by using the LDAP (SDBM) interface to RACF. Furthermore, we configure Tivoli Directory Integrator to use the RACF remote authorization and audit services, by enabling the IBM Tivoli Directory Server ICTX extended operations component to perform authorization checks against RACF general resource profiles.

Although the solution that is provided supports synchronization between any combination of z/OS and z/VM systems, the logical topology for the examples that are used throughout this publication is shown in Figure 1-1.

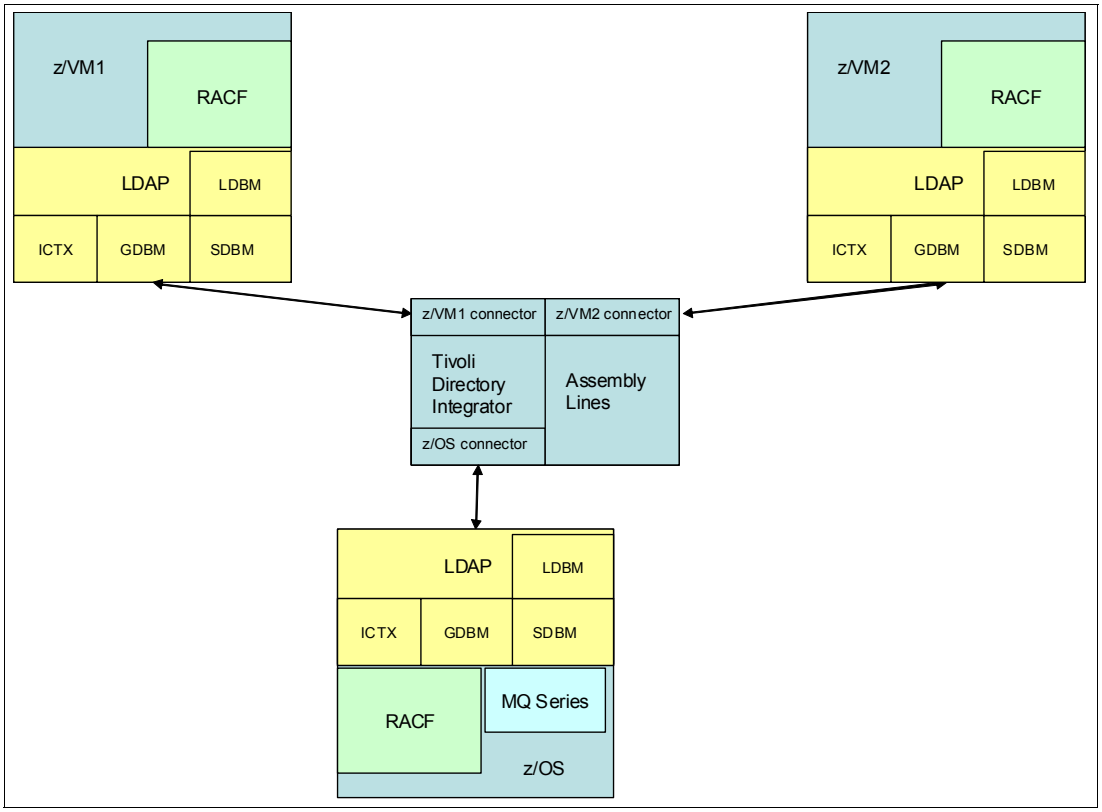


Figure 1-1 Example synchronization topology

This document provides details about how to configure RACF and LDAP on z/OS and z/VM, and how to implement our sample synchronization solution.

In this topology, Tivoli Directory Integrator instantiates AssemblyLines to monitor the changelog for systems that are defined as source systems (that is, they are the systems in which data is pushed to other systems). In addition, Tivoli Directory Integrator also instantiates AssemblyLines to push the changes to the systems defined as targets (that is, systems that receive changes to their database). It is possible for a system to be both a source and target. Changes to be made are placed by Tivoli Directory Integrator on a persistent IBM WebSphere MQ queue and processed from that queue by Tivoli Directory Integrator.

The topology (list of systems that are involved in the synchronization solution) is defined in an LDAP directory. Therefore, Tivoli Directory Integrator can dynamically read the topology information upon startup. The LDAP directory contains information about the systems, their IP addresses, LDAP bind information for Tivoli Directory Integrator, and source or target information.

1.2 Sample Tivoli Directory Integrator configuration

We present a sample Tivoli Directory Integrator configuration in this publication and ideas on how this setup can be extended to synchronize more RACF data between systems. An overview of this configuration is presented in 1.8, “Sample solution” on page 13.

This configuration provides a simple solution for synchronizing passwords, password phrases, and select user profile fields between z/OS and z/VM. In the examples that are used throughout this document, we propagate passwords, password phrases, and the NAME field of the user profile from a z/OS RACF database to two z/VM RACF databases. However, the configuration is designed to allow propagation between any combination of z/OS and z/VM systems. See 1.8, “Sample solution” on page 13 for details.

Basic setup for z/OS and z/VM is detailed in Chapter 3, “z/OS setup” on page 49 and Chapter 2, “z/VM setup” on page 21. Extra required setup is detailed in Chapter 4, “Additional configuration setup” on page 61.

1.3 Software inventory

The following product levels were used for implementing the solutions that are presented in this publication.

1.3.1 z/OS

The following products were used to perform synchronization of user data on z/OS:

- z/OS Version 1 Release 13, Program Number 5694-A01
- Security Server, RACF feature of z/OS 1.13
- IBM Tivoli Directory Server for z/OS (IBM Tivoli Directory Server, also known as LDAP), base element of z/OS 1.13
- Cryptographic Services, System Secure Sockets Layer (System SSL), base element of z/OS 1.13 (optional)

1.3.2 z/VM

The following products are required on z/VM to perform synchronizing of user data:

- z/VM Version 5 Release 4, Program Number 5741-A05
- RACF feature of z/VM Version 5 Release 4
- LDAP feature of z/VM Version 5 Release 4

1.3.3 Tivoli Directory Integrator

The Tivoli Directory Integrator 7.1.1 server component can be run on z/OS or on a distributed operating system (for example, Windows). The Tivoli Directory Integrator Configuration Editor component requires a graphical user interface (GUI) and is necessary for creating the Tivoli Directory Integrator configurations and AssemblyLines. The Tivoli Directory Integrator Configuration Editor component must be run on a distributed operating system.

1.3.4 WebSphere MQ

Our solution uses WebSphere MQ for z/OS 6.0 for a persistent queue. We used WebSphere MQ on z/OS, but a WebSphere MQ queue manager on any supported platform can be used.

1.4 Lightweight Directory Access Protocol

Although a more extensive *Lightweight Directory Access Protocol* (LDAP) introduction can be found in the IBM Redbooks publication, *Security on z/VM* SG24-7471, section *1.3.2 LDAP,* a brief overview is presented here.

LDAP is an open standard to define organizations and people in a structured way, which is organized in a directory tree. LDAP also defines standards to access this directory, for both read and write operations.

LDAP is network-based, meaning that the LDAP server can be on a different server than the LDAP client. LDAP is widely accepted in the distributed systems environment. It is used for example by web browsers and applications such as the IBM WebSphere Application Server to control application access (authorization and authentication).

1.4.1 LDAP back-end services

The LDAP servers on z/OS and z/VM provide all normal LDAP LDBM database services. In LDAP terminology, these databases are also termed *backend services*.

LDBM This is a general-purpose database that can store any type of directory information. In this publication however, this type is the least important database. We use LDAP as a means to synchronize user information that is stored in the RACF database. However, we do present a solution where Tivoli Directory Integrator reads configuration data from the LDBM.

GDBM The GDBM backend service is used to log changes that are made to LDBM, LDAP schema entries, and RACF user information. This information is what Tivoli Directory Integrator uses to synchronize changes. Tivoli Directory Integrator can listen to it for changes or poll this database at regular intervals.

SDBM Secured Database Manager (SDBM) gives access to the data stored in the RACF database. The SDBM database is not a copy of the RACFVM database. The database merely acts as a mapping tool to access the data in the RACF database.

ICTX The Identity Context Extension (ICTX) extended operation plug-in provides *RACF Remote services support*. This service enables programs, which cannot use RACROUTE because they do not run directly under z/VM or z/OS, to make authorization decisions by sending authorization requests to RACF.

1.4.2 LDAP binding and client authentication

To connect to an LDAP server, the client must authenticate itself. This process is called *binding* in LDAP terminology.

Initial setup of the LDAP server LDBM backend uses a clear text password that is stored in the configuration file (`ds.conf`) when performing a bind for the LDAP administrator. The examples that are supplied for both z/OS and z/VM configuration setup in Chapter 3, “z/OS

setup" on page 49 and Chapter 2, "z/VM setup" on page 21, include steps to enable the administrator to bind by using an encrypted LDBM stored password. Other methods of binding, including Native Authentication, are described in the LDAP administration guides for z/OS and z/VM and might be more suitable in your production environment.

In this example solution, we bind to LDAP in various ways. To access LDAP to retrieve information about the topology, we use either a simple bind with the distinguished name (DN) matching an LDAP user entry for Tivoli Directory Integrator defined in the LDBM. Or, optionally, we allow binding by using Simple Authentication and Security Layer (SASL). To access SDBM data, thus RACF data, we use a RACF user ID and password or password phrase in the bind to LDAP.

The solution is configured to require Secure Sockets Layer (SSL) connections between Tivoli Directory Integrator and LDAP. Server-only authentication or Server/Client authentication is supported.

1.4.3 Change Logging

One of the basic LDAP functions is change logging. That is, changes to an LDAP database are also written to the log database (the GDBM). This function is used by RACF. Tivoli Directory Integrator obtains and uses these change log entries to propagate changes to RACF databases on other systems.

Readers that are familiar with the RACF remote sharing facility (RRSF) feature on z/OS should note a difference between RRSF and our solution. The difference is that with RRSF, RACF commands are shipped to the remote systems for execution. However, in our solution we are not aware of the command that was issued to change the RACF data. The LDAP change log entries do not contain the actual executed RACF command. The LDAP records merely indicate something was changed in the definition of some user, group, or connection. Thus, in our configuration, for the user information, Tivoli Directory Integrator retrieves the RACF data that is associated with the change record from the source system (where the change record originated). Tivoli Directory Integrator then compares this data to the target system or systems data, and then propagates the comparison delta to the target system or systems. For passwords and password phrases, Tivoli Directory Integrator propagates the change to the target system or systems without a comparison.

There are several types of change log records that are logged by RACF, as indicated by the profiletype portion of the targetdn in a change log record. Examples of the types are USER, CONNECT, and GROUP. Our solution addresses only USER change records.

The following list provides some example change log records and partial extracts of the change log records that are generated as a result of the command issued.

```
RAC CONNECT USER094 GROUP(SYNC)
changetype=add
targetdn=RACFUSERID=USER094+RACFGROUPID=SYNC,PROFILETYPE=CONNECT,CN=RACFVM

RAC ADD USER094
changetype=add
targetdn=RACFID=USER094,PROFILETYPE=USER,CN=RACFVM

RAC ALTUSER USER094 PASSWORD(testpass)
changetype=modify
targetdn=RACFID=USER094,PROFILETYPE=USER,CN=RACFVM
changes=replace: racfPassword
racfPassword: *ComeAndGetIt*
```

```
RAC ALTUSER USER094 PHRASE('this is a phrase')
changetype=modify
targetdn=RACFID=USER094,PROFILETYPE=USER,CN=RACFVM
changes=replace: racfPassPhrase
racfPassPhrase: *NoEnvelope*
```

Not all RACF commands yield specific change log entries, like this ALTUSER

```
rac altuser operatns NAME('Dump receiver')
changenumber=7503
changetype=modify
targetdn=RACFID=OPERATNS,PROFILETYPE=USER,CN=RACFVM
ibm-changeinitiatorsname=RACFID=KRIS,PROFILETYPE=USER,CN=RACFVM
changetime=20081009150542.470000Z
```

No information other than a notification that a change occurred is available from change log records. Therefore, the Tivoli Directory Integrator server might have to pull the entire RACF database entry from the system where the change occurred before it can propagate the change. For a password or password phrase change, the password or password phrase field of the changelog record indicates *ComeAndGetIt* when a password envelope is available.

Change records: In our configuration, when Tivoli Directory Integrator queries a change record and notices that the changeinitiatorsname is equal to the Tivoli Directory Integrator user ID (which in our example, is Tivoli Directory Integrator) it skips that change record. This step is the means by which Tivoli Directory Integrator stops the change propagation loop.

1.5 Password and password phrase enveloping

RACF always stores passwords in its database by using a “one-way” encryption technique. That is, when a password is stored in its database, there is no way to unencrypt it. During password verification, RACF encrypts the password that is entered by the user and compares the result with the password in its database. Password phrases are handled in a similar manner.

When passwords must be synchronized among heterogeneous systems, RACF must have a method of enabling de-encryption of passwords. The RACF term for this method is *password enveloping*. The password of users without a password envelope cannot be synchronized to other systems.

Basically, the following steps occur when the password of a user is changed:

- RACF stores the one-way encrypted password in its database.
- If the user is authorized for password enveloping, RACF encrypts the password with the RACF private key (defined during setup). The result, the password envelope, is also stored in the RACF database.
- An authorized user can retrieve a password or password phrase envelope through an LDAP SDBM search. This requires that a public certificate of the requesting user be stored in the keyring that is used by RACF. Upon request of a password or password phrase, RACF retrieves the encrypted envelope, decrypts it, and re-encrypts it by using the public key of the requesting user (called the *recipient certificate*). RACF then signs the resulting message and returns the data to the requester through LDAP.

- In our configuration, when the Tivoli Directory Integrator server finds out that a user password is changed, it uses LDAP to request the password from RACF.
  - When there is no password envelope, the work of Tivoli Directory Integrator stops there.
  - When there is a password envelope, RACF decrypts the password envelope with its own key, and encrypts it with the Tivoli Directory Integrator public key. RACF then returns the password to the requesting Tivoli Directory Integrator server.
  - Only Tivoli Directory Integrator can decrypt the password sent by using its private key. Tivoli Directory Integrator then uses LDAP commands to change the user password on the systems that are selected for synchronization. It should be clear to the reader that the session between Tivoli Directory Integrator and the target LDAP servers should be SSL encrypted. More considerations can be found in 1.7, “Security considerations” on page 12.

There are some slight differences in the implementation of password and password phrase enveloping between z/OS and z/VM. In z/OS, RACF uses the System SSL functions to do the encryption functions. A simplified diagram of enveloping on z/OS is shown in Figure 1-2.

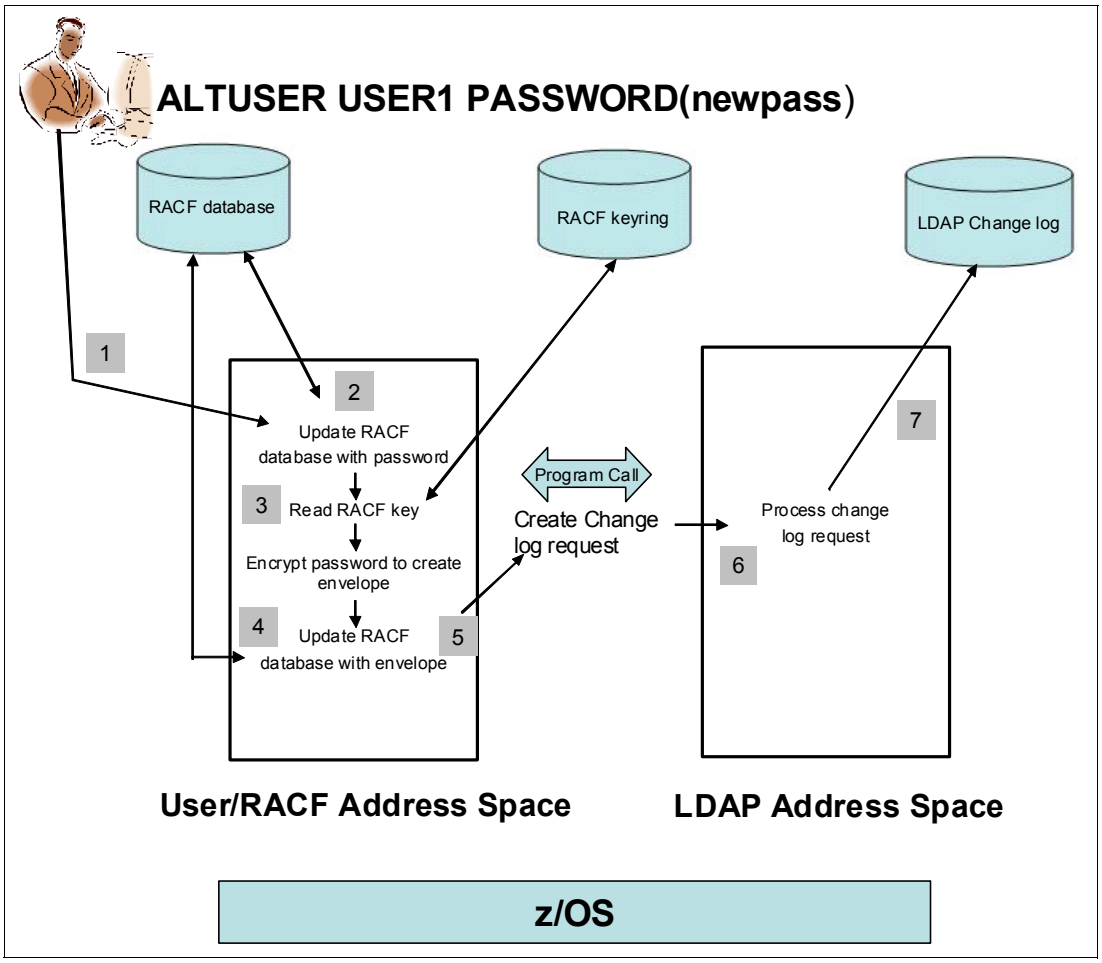


Figure 1-2 Password and password phrase enveloping process on z/OS

A simplified illustration of an enveloped password retrieval by Tivoli Directory Integrator on z/OS is shown in Figure 1-3.

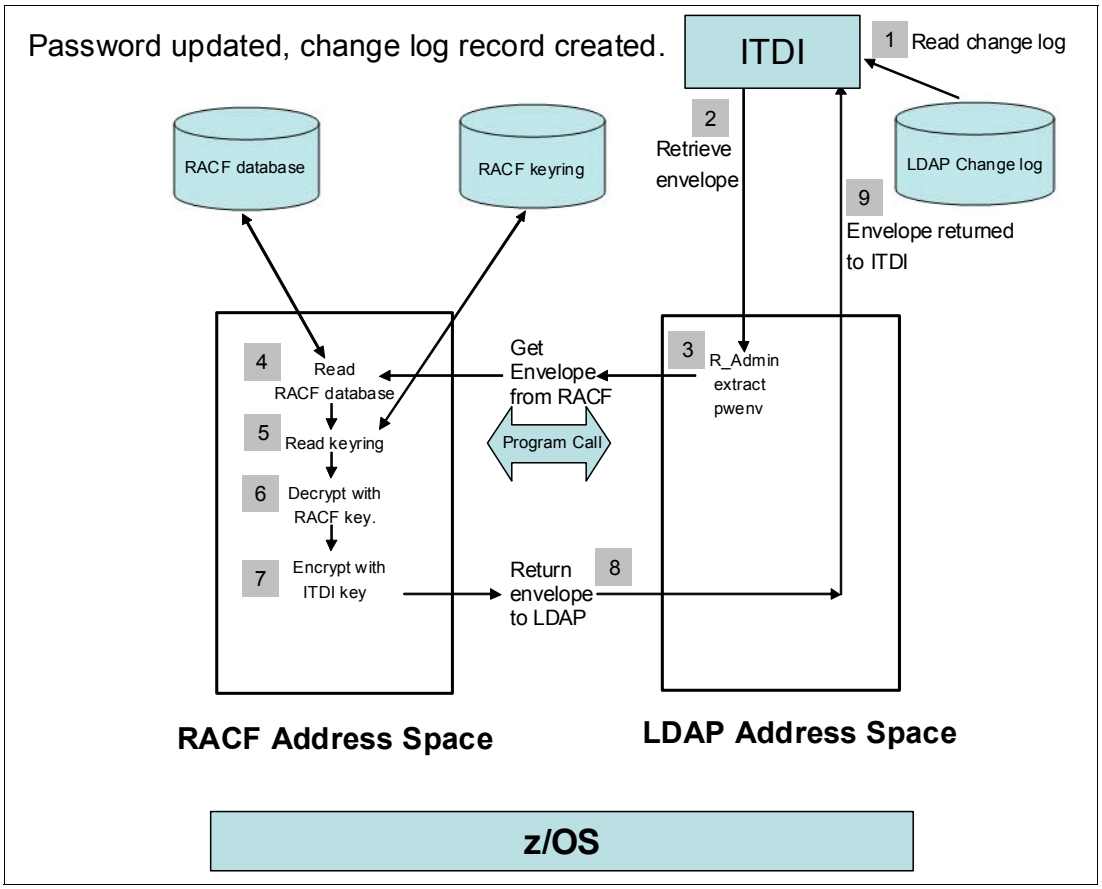


Figure 1-3 Password and password phrase retrieval on z/OS

In z/VM, RACF runs isolated in its own virtual machine, under a modified version of Conversational Monitor System (CMS). Neither System SSL or “open VM extensions” are available to RACF. Therefore, RACF sends encrypt and decrypt requests to the LDAP server, which runs under a full-function CMS.

LDAP on z/VM comes with embedded SSL support. It is, however, still RACF that stores the password envelopes in its database. A simplified example of password enveloping on z/VM is shown in Figure 1-4.

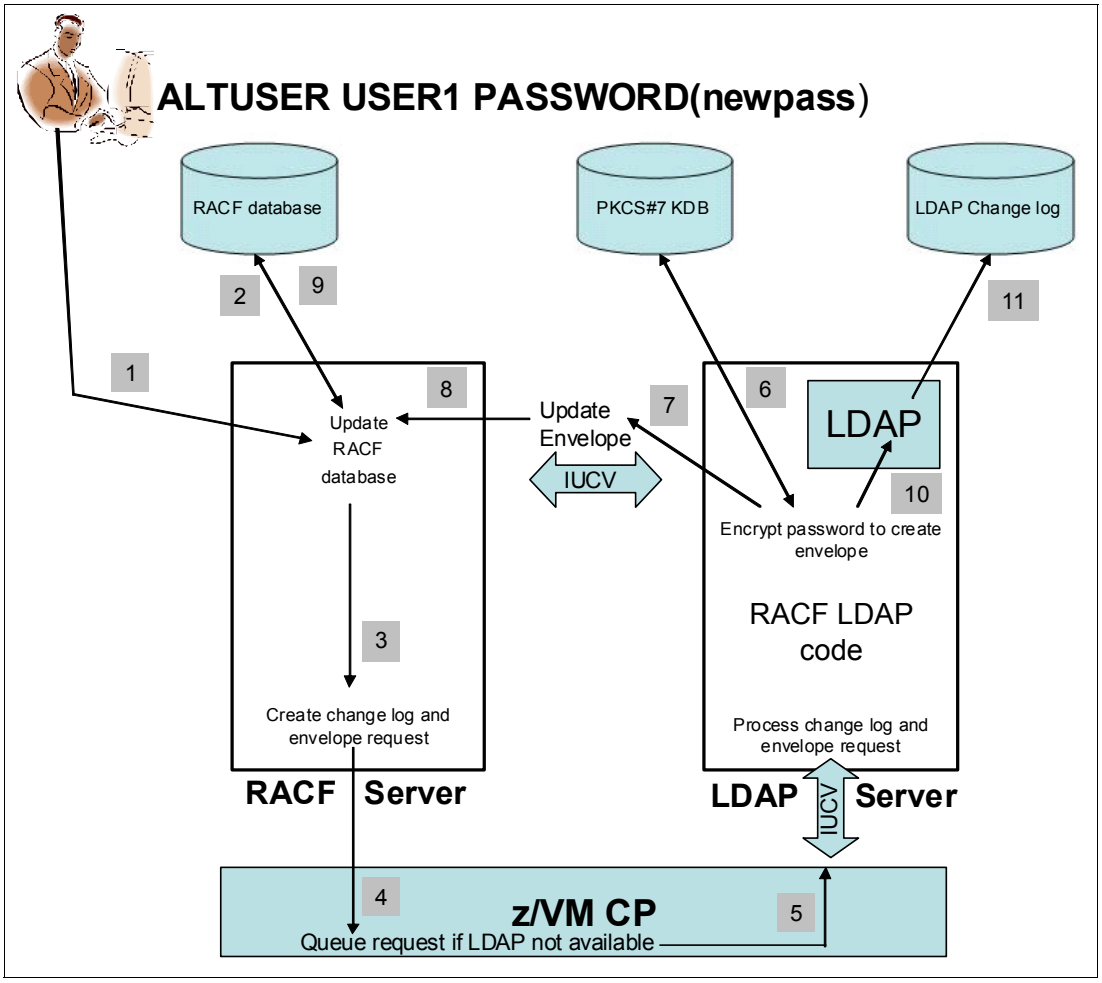


Figure 1-4 Password and password phrase enveloping process on z/VM

A simplified illustration of the retrieval of an enveloped password by Tivoli Directory Integrator on z/VM is shown in Figure 1-5.

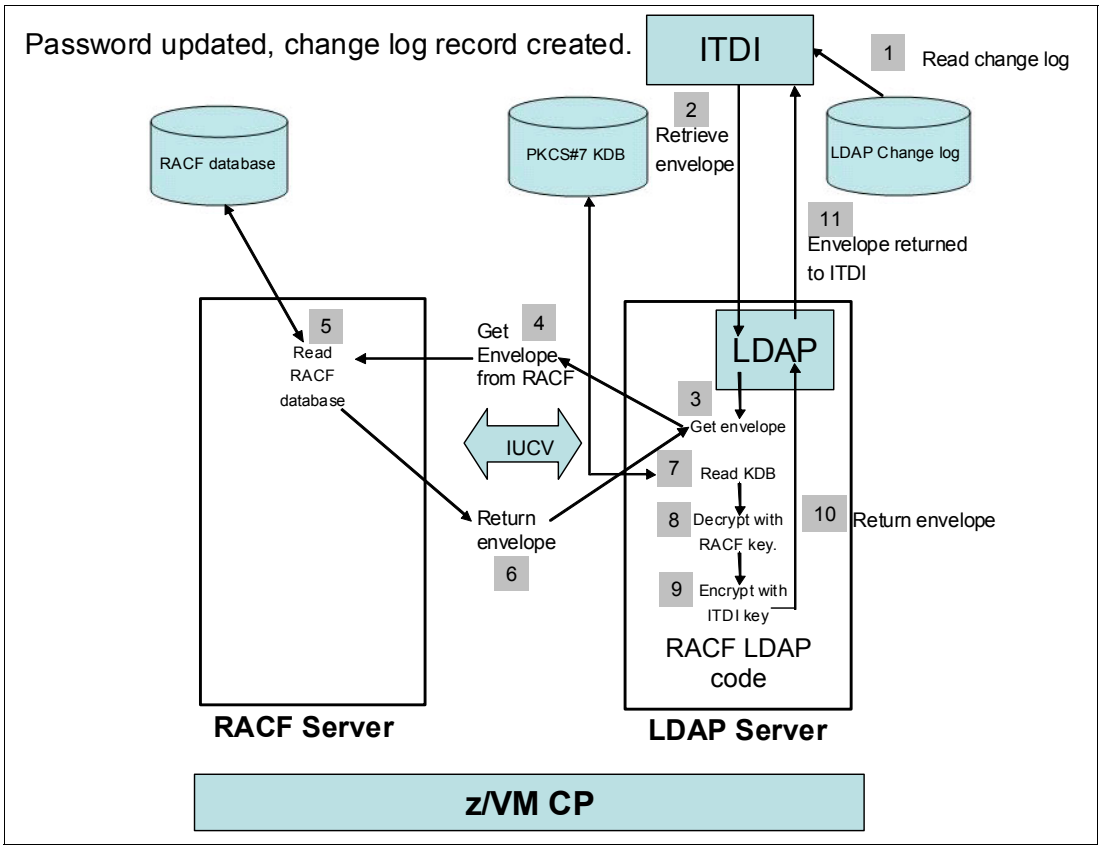


Figure 1-5 Password and password phrase retrieval on z/VM

Another difference is the management of the keys and certificates. On z/OS, keys and certificates can be managed by using RACF **RACDCERT** commands and stored in the RACF database. Or, a file base keystore can be used. On z/VM, `gskkyman` is used, and the storage is a file in a BFS file space (the VM Posix compliant file system).

1.6 Planning

A brief discussion on planning for synchronization is provided.

1.6.1 Choosing eligible users and data for synchronization

Each installation must determine, based on their own environment and needs, which RACF users should have their passwords and password phrases synchronized. If a target system has more than one source system sending changes, then the group of users should not overlap between the source systems. For example, if SC58 and SC60 are both source systems (each with its own RACF database) for target system VMLINUX5, do not have USER001 on both SC58 and SC60 configured to propagate changes to VMLINUX5. Ensure that you have no intersection between the user IDs on source systems when multiple sources exist for a potential target system.

1.6.2 Controls on z/OS and z/VM

The basic configuration of RACF and LDAP on z/OS and z/VM requires read access to RACF profiles `PASSWORD.ENVELOPE` and `PASSPHRASE.ENVELOPE` in class RACFEVNT for any user whose password or password phrase is to be enveloped for propagation to other systems.

In the case of our configuration, access is allowed by creating a RACF group that is called *SYNCH,* allowing the group to the RACFEVNT profiles and connecting selected users to the group.

As an extra filter, eligibility for synchronization is determined by RACF profiles in the FACILITY class, thus enabling a degree of granularity. The Tivoli Directory Integrator AssemblyLine is written to check the subject user's access to these FACILITY class profiles to determine whether propagation should take place. See 4.2, "RACF setup on source systems" on page 70 for details about the FACILITY class profiles that Tivoli Directory Integrator uses. Tivoli Directory Integrator uses the RACF Remote Services support through the LDAP ICTX extended operation to query the FACILITY class profiles.

FACILITY class profiles: The FACILITY class profiles affect only the actions of Tivoli Directory Integrator. Access to the RACFEVNT profiles (described in the preceding information) is still required if RACF is to perform enveloping.

1.6.3 Controls in Tivoli Directory Integrator AssemblyLines

When synchronizing data between like operating systems (that is, z/OS to z/OS, or z/VM to z/VM), segments of the base user record (for example, Time Sharing Option (TSO), Data Facility Product (DFP), and so on) can be included.

When synchronizing unlike platforms (that is, z/OS or z/VM, or vice versa), much of the segment data might be inappropriate to synchronize. For example, a TSO segment for a z/OS RACF user would have no meaning to a z/VM user. For this reason, most segment data is excluded from synchronization on unlike platforms.

In addition to the preceding platform considerations, operational considerations must be taken into account when you decide if a particular field should be synchronized. For example, care must be taken if RACF attributes are to be specialized. You might not want to propagate authorities such as the RACF SPECIAL and OPERATIONS attributes. In our configuration, RACF attributes are not synchronized in the sample Tivoli Directory Integrator AssemblyLines.

A list of user record attributes and their suggestions whether the data makes sense for synchronization can be found in Appendix B, "RACF: LDAP field mappings" on page 197.

Although we propagate changes only to the name field of the user profile, it is possible to synchronize extra fields from the user profile. Enabling this function requires modification to our sample code.

1.7 Security considerations

This section contains security-related considerations and notes on certificate usage.

1.7.1 Restricting access to RACF password and password phrase envelopes

The ability to extract envelopes from the RACF database is controlled by FACILITY class profiles `IRR.RADMIN.EXTRACT.PWENV` and `IRR.RADMIN.EXTRACT.PPENV`. It is recommended that these profiles be defined with UACC(NONE) and only authorized clients (in our case, Tivoli Directory Integrator) be granted read access. Access to envelopes along with the certificate that is used for encryption can result in exposure of password data.

1.7.2 Certificates

Digital certificates are used the following ways:

- By RACF for encryption of a password or password phrase for storing as an envelope in the RACF database. This is the RACF certificate.
- By RACF for decrypting password and password phrase envelopes that are stored in the RACF database. This is the same RACF certificate that is used for encryption.
- By RACF for re-encryption of a decrypted envelope with a recipient's (such as Tivoli Directory Integrator) public certificate for transmission. This is the recipient certificate.
- By Tivoli Directory Integrator to decrypt envelopes, which provide the source for password updates on target systems. This is the private key that is associated with its recipient certificate used by RACF.
- By Tivoli Directory Integrator to verify the digital signature of the envelope that is sent by RACF. This process requires that the RACF address space's certificate and its signing CA certificate be available to Tivoli Directory Integrator.
- For encrypting LDAP client/server communication on a secure SSL/TLS connection. These are the SSL certificates for each LDAP server and Tivoli Directory Integrator.

The setup chapters for z/OS and z/VM contain details of certificate management (creation, export, import, and so on) relevant to each platform.

1.8 Sample solution

For this scenario, we synchronize password, password phrases, and user name field changes from a z/OS LPAR to two z/VM LPARs. The sample Tivoli Directory Integrator configuration supports password and password phrase synchronization between any combination of z/OS and z/VM LPARs.

The sample solution consists of the following components:

- The z/OS and z/VM system configuration, including LDAP and RACF
- A WebSphere MQ queue manager with a queue for Tivoli Directory Integrator to place and retrieve messages
- Tivoli Directory Integrator configuration (the XML file that contains the set of AssemblyLines and components) which is executed by Tivoli Directory Integrator

1.8.1 Synchronizing RACF passwords and password phrases considerations

This implementation is designed to synchronize a select group of RACF user passwords, password phrases, and the name field of the user profile. The technical requirements for enabling RACF change logging and enveloping are covered in Chapter 3, "z/OS setup" on

page 49, and Chapter 2, “z/VM setup” on page 21. In addition, the following considerations apply for our scenario 1:

- The user ID whose password or password phrase is being synchronized should exist on all the systems where it can potentially be propagated to. If a user ID does not exist on a system that is defined as a possible target to Tivoli Directory Integrator, for a specified source system, the password or password phrase change is not attempted.
- For our example, a RACF group is created solely for grouping users whose passwords and password phrases are to be enveloped. This group is allowed with read access to RACEVNT profiles `PASSWORD.ENVELOPE` and `PASSPHRASE.ENVELOPE`. For our example, this group is called *SYNCH*. This configuration was done for all systems where password or password phrase changes are to originate. It is possible to define, for example, one system to envelop passwords and password phrases entries and have Tivoli Directory Integrator propagate to systems that do not envelop passwords or password phrases. Although, in our example, all systems are configured to envelop passwords and password phrases.
- The configuration requires the creation of RACF FACILITY class profiles, which are to be used by Tivoli Directory Integrator to determine whether the data of a user should be propagated to a particular target system. For each LPAR that is defined as a source LPAR (meaning password or password phrase changes performed on this system are to be synchronized to other LPARs), a set of RACF FACILITY class profiles is created for each target LPAR. Users must be given read access to these RACF FACILITY class profiles to have their password or password phrase changes propagated to target LPARs. This process provides extra control for the security administrator. Tivoli Directory Integrator queries these profiles on source systems to determine if a user ID should be propagated from a source system to a target system.
- No z/OS RACF protected user IDs are added to the synchronization process. The RACF administrator must take care of this function. If, for example, a z/OS protected user gets a password assigned on a z/VM system, Tivoli Directory Integrator can propagate that password to z/OS and the user would no longer be protected. The same applies to NOPASSWORD users on z/VM. It might be wise to have some automated procedure to verify that your list of PROTECTED/NOPASSWORD users are still in that state.
- On z/OS, a user ID with a password phrase must also have a password. However, on z/VM, a user ID can have a password phrase without a password. We did not add z/VM password-phrase only users to the synchronization process. If you want users to use only password phrases on z/VM, the best solution is to code a RACF exit on z/VM that refuses a login that is based on a password. This assures that passwords that are promoted from z/OS by Tivoli Directory Integrator cannot be used to log in.
- The examples that are used in this document propagate password and password phrase changes from z/OS to two z/VM LPARs.

The sequence of events is shown in Figure 1-6. The figure omits the WebSphere MQ queue to simplify the flow.

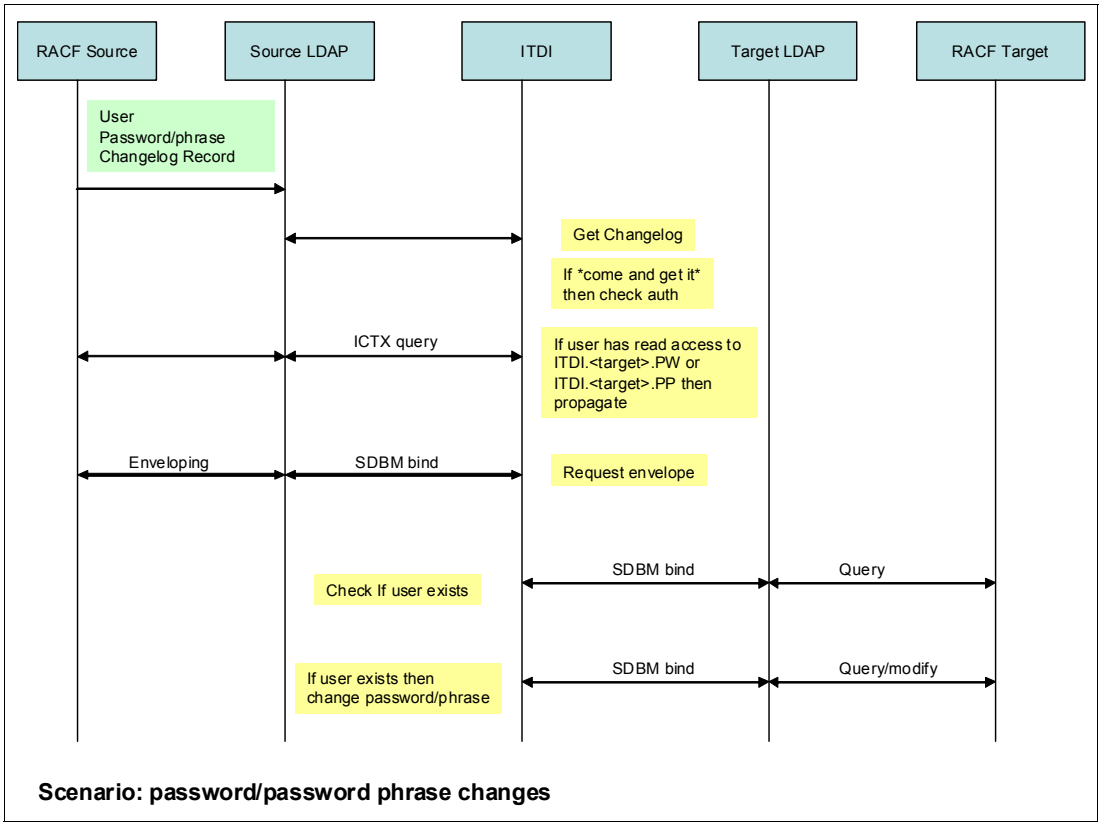


Figure 1-6 Password and password phrase sequence of events

The following sequence of events applies to the considerations described previously:

1. A user password, password phrase, or user profile is changed on one of the systems (source) that is part of the synchronization complex.
2. Tivoli Directory Integrator through its changelog connectors polls the changelog and discovers a new changelog record. Tivoli Directory Integrator checks the changelog and determines through further processing that this user password, password phrase, or user profile changes are to be synchronized to the other system or systems (target or targets). This process occurs because Tivoli Directory Integrator is set up to listen to this system's changelog. And, the changelog record indicates either a password change with *come and get it* or some change in the user profile.
3. Tivoli Directory Integrator queries the source system's RACF FACILITY class profile to determine which target systems the user is eligible to have its changed data propagated to.
4. For a password or password change, Tivoli Directory Integrator requests the enveloped password. This process is done through an LDAP SDBM bind by Tivoli Directory Integrator. RACF encrypts the user password or password phrase with the recipient's public key (Tivoli Directory Integrator). The enveloped password or password phrase is returned to Tivoli Directory Integrator.
5. Tivoli Directory Integrator checks if the user ID exits on the target system. If the user ID does not exist on a target system, then no action is taken.
6. For target systems where the user ID exists, Tivoli Directory Integrator then updates the user data, password, or password phrase on the target system.

1.8.2 Overview of the implementation steps

To implement this solution, the following actions must be taken. For our example, we are running the Tivoli Directory Integrator instance on a distributed system:

1. Complete the z/OS setup steps as described in Chapter 3, “z/OS setup” on page 49. Chapter 3 covers setup of the LDAP server on z/OS, enabling enveloping and change logging, creation of required IDs, and SSL/TLS setup.
2. Complete the z/VM setup steps as described in Chapter 2, “z/VM setup” on page 21. Chapter 2 covers setup of the LDAP server on z/VM, enabling enveloping and change logging, creation of required IDs, and SSL/TLS setup.
3. Complete the steps that are described in Chapter 4, “Additional configuration setup” on page 61. Chapter 4 covers the extra steps that are required to implement the scenario. For example, steps that are described include setting up the LDAP LDBM entries, LDAP schema modifications, WebSphere MQ requirements, definition of required RACF profiles, and Tivoli Directory Integrator setup.
4. Test the configuration. We provide extra details about the Tivoli Directory Integrator configuration and AssemblyLine logic in Chapter 5, “Testing the configuration” on page 99.

1.9 Overview of the Tivoli Directory Integrator AssemblyLines

The logic encapsulating the solution and that is executed by Tivoli Directory Integrator is called a *configuration* (*config*). The configuration is composed of AssemblyLines, connectors, scripts, and other logical resources. Although it is beyond the scope of this document to provide detailed information about these components, this section provides a brief overview of how these components are used and the logic flow of the sample configuration.

The information that Tivoli Directory Integrator requires to start the synchronization process is gathered from two sources. Information is gathered either as properties that are defined in the configuration or from entries in an LDAP directory tree. The preparation of this preliminary information is covered in Chapter 4, “Additional configuration setup” on page 61.

Our config consists of a set of AssemblyLines (AL), which are composed of various components that are used to obtain data from systems, transform this data, and push this transformed data to systems. When an Tivoli Directory Integrator configuration is started, one or more AssemblyLines are started to perform the processing of data. Our config consists of the following three ALs:

- startPutAndGetALs
- putEntryToTarget
- getEntryFromSource

The following sections provide an overview of these ALs.

1.9.1 startPutAndGetALs AssemblyLine

When the config is started, the first AL that is executed is the `startPutAndGetALs`. The purpose of this AssemblyLine is to start a new AL for each system that is defined as a target system and a new AL for each system that is defined as a source system. This AL performs the following functions:

- Using an LDAP connector, it reads a portion of a directory tree in an LDAP server to obtain the synchronization topology. This portion of the directory tree contains entries that

describe the systems participating in the solution. These entries describe whether the system is to be a source system or a target system. The entries also contain other connection information that is needed by Tivoli Directory Integrator to communicate with these systems.

- Based on the information that is retrieved from the directory entries, for each system that is defined as a target system, this AL starts a putEntryToTarget AL for each target system.
- Based on the information that is retrieved from the directory entries, for each system that is defined as a source system, this AL starts a getEntryFromSource AL for each source system.
- When the startPutAndGetALs AL parsed the directory tree and instantiated all putEntryToTarget ALs for target systems and getEntryFromSource ALs for source systems, it terminates.

1.9.2 putEntryToTarget

A putEntryToTarget AL is started for each system that is defined as a target by the startPutAndGetALs AL. Each putEntryToTarget AL is associated with a single target system. It processes work for the intended target system. The purpose of this AL is to read messages from a WebSphere MQ queue. If the message header matches the system for which this AL was instantiated for, then it processes that message. This AL pushes the password, password phrase, and user data changes to a target system. It does the following functions:

- Using a WebSphere MQ connector, it reads messages from a WebSphere MQ queue. If the message header matches the system for which the AL was created for, it processes the message.
- Using an LDAP connector and an SDBM bind to the target system for which this AL is associated with, it pushes the change to the target system.
- When a message is retrieved, the message is deleted from the WebSphere MQ queue.
- This AssemblyLine stays active until terminated by the user or encounters an unrecoverable error.

1.9.3 getEntryFromSource

One getEntryFromSource AL is created for each system that is defined as a source system. It is associated with a single source system. The main purpose of this AL is to read changelog records from the source system, determine if this is a change to be propagated, and verify that the user is authorized to have this data propagated. If the change is to be propagated, this AL pushes the change, as a message, to a WebSphere MQ queue (one message for each system that is defined as a target for this source system). This AL does the following functions:

- Using a z/OS changelog connector, retrieve changelog records for the system for which this AL is associated.
- Interrogates the changelog record to determine if it should be processed. In our solution, password, password phrase, and user data (name field) changelog records are acted upon.
- If a change is to be acted upon, query the source system by using the RACF remote authorization feature through use of the ICTX plug-in. This action allows you to check if the user has READ access to the appropriate profile in the FACILITY class.

  We set up profiles in the FACILITY class to represent each possible change that we are interested in: One profile for passwords, one profile for password phrases, and one profile

for user data. If the user, for whom the changelog record is associated with, has READ access to the relevant profile, we continue processing. Otherwise, we do not propagate the change.

- Using an LDAP connector and a bind to the SDBM of the source system for which this AL is associated with, retrieve the password, password phrase envelope, or user data.
- For each system that is defined as a target system for this source system, a message is placed on the WebSphere MQ queue with a header containing the name of the target system and the data that is to be propagated.
- This assembly time stays active until terminated by the user or encounters an unrecoverable error.

1.9.4 Example of AssemblyLines

Here is an example to clarify how the AssemblyLines are started. In our LDAP directory tree, SC58 is defined as a source system and VMLINUX5 and VMLINUX7 are defined as target systems.

As shown Figure 1-7 on page 19, the startPutAndGetALs AL parses the directory tree and creates a putEntryToTarget AL for each system that is defined as a target system. In this example, ALs are created for VMLINUX5 and VMLINUX7. The startPutAndGetALs AL then terminates, but all putEntryToTarget ALs remain active.

The startPutAndGetALs AL parses the directory tree and creates a getEntryFromSource AL for each system that is defined as a source. In this example, an AL is created for SC58. The startPutAndGetALs AL terminates, but all getEntryFromSource ALs remain active.

Figure 1-7 shows an example of an AssemblyLine.

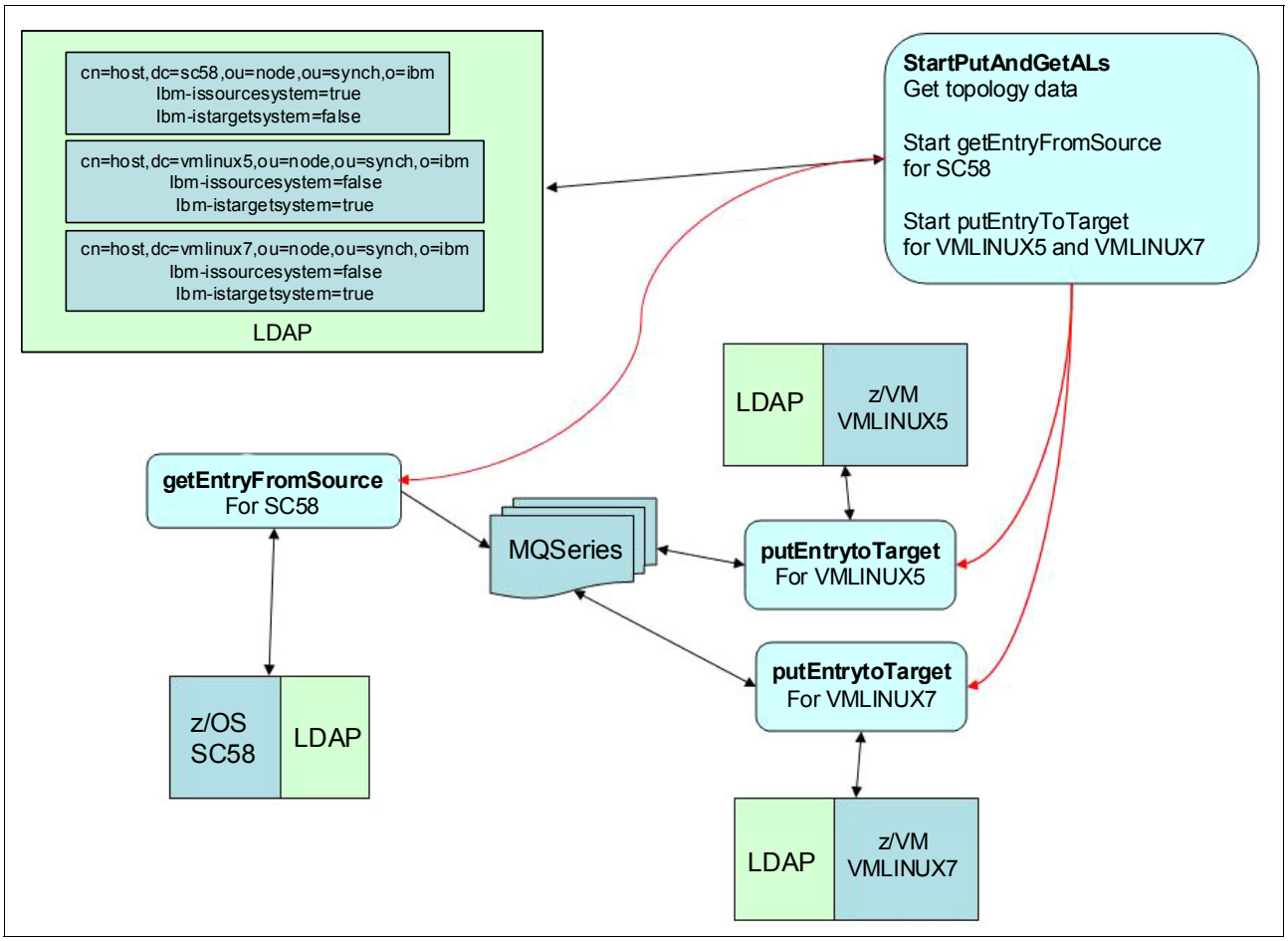


Figure 1-7 AssemblyLine example

2

z/VM setup

This chapter describes how to configure Resource Access Control Facility (RACF) and Lightweight Directory Access Protocol (LDAP) on z/VM to enable synchronization by using IBM Tivoli Directory Integrator.

2.1 Assumptions

It is assumed that RACF is already activated on your z/VM system.

If not, keep in mind that RACF is delivered as a preinstalled, but disabled feature of z/VM. The *Program Directory for RACF Security Server,* GI11-2894-00, for z/VM describes the step-by-step process to enable and configure the product. Program directories for z/VM features and products can be found on the z/VM web page at:

`http://www.vm.ibm.com/progdir/`

Furthermore, it is assumed that the RACF *list-of-group checking* is active. If not, issue `RAC SETOPTS GRPLIST`. The *list-of-group* feature makes RACF decide on resource access permissions on all groups that the user is connected to as opposed to only its default group.

The z/VM LDAP server is a ported version of the IBM Tivoli Directory Server for z/OS server. It uses z/VM OpenExtensions and BFS files.

The LDAP software comes preinstalled too. As opposed to z/OS, in z/VM, LDAP is part of the TCP/IP stack. Consequently, most LDAP code is installed on the TCPMAINT 592 minidisk. Though, other parts of the code are installed in a BFS file space in the VMSYS file pool.

2.2 Byte File System introduction

If you are relatively new in z/VM, you might be unfamiliar with the terms, SFS and BFS. Therefore, these terms and others are briefly introduced:

SFS SFS stands for *Shared File System*. SFS is implemented as a z/VM service machine. It manages files that are stored by Conversational Monitor System (CMS) users. As a storage medium, an SFS server gets a series of minidisks. When the space allocated to SFS is full, more minidisks can be added dynamically to the server. When z/VM is installed, two SFS servers come with it: *VMSERVS* and *VMSERVU*.

File pool In addition to a VM user ID, an SFS server has a *file pool ID*. This ID is the name that users specify on the CMS commands. With the default setup, *VMSYS* is the file pool ID for VMSERVS, and *VMSYSU* is the file pool ID for VMSERVU.

File space Before CMS users can use files in SFS, they must be enrolled in one or more filepools. The **`ENROLL USER`** command is used for this purpose. The space that the user can use is named a *file space*.

BFS BFS stands for *Byte File System*. It is a z/VM implementation of a Posix compliant file system. It is implemented in SFS. The BFS option that is specified on the **`ENROLL USER`** command triggers the creation of a BFS file space. An SFS server can host both BFS and classic CMS filespaces.

Tools Two FILELIST-like tools for SFS and BFS are available on the VM download library:

- **SFSULIST** Lists all users (or filespaces) in a file pool.
- **BFSLIST** Lists the Posix file tree. Allows for the rename, delete, copy, and so on, of BFS resident files.

The download library can be found on the z/VM web page at:

http://www.vm.ibm.com/download/packages

2.3 More information

To make cross-system user information synchronization work, many setup instructions are required. The instructions are provided in several IBM manuals:

- *z/VM TCP/IP Planning and Customization,* SC24-6140, Chapter: *Configuring the* LDAP Server
- *z/VM RACF Security Server Security Administrator's Guide,* SC24-6142, Chapter: *LDAP event notification* and chapter *Password and password phrase enveloping*
- *z/VM TCP/IP LDAP Administrator's Guide,* SC24-6125
- *z/VM TCP/IP User's Guide,* SC24-6127, Chapter: *Using the LDAP Operation Utilities*

2.4 Lightweight Directory Access Protocol brief introduction

A brief Lightweight Directory Access Protocol (LDAP) introduction is provided, which is oriented towards the z/VM administrator that installs LDAP.

The deliverable for LDAP in z/VM provides both LDAP client and server functions. A brief overview of these functions follows.

2.4.1 LDAP client

The LDAP client runs in a CMS virtual machine. The LDAP client programs are available on the TCPMAINT 592 minidisk. Each z/VM user can inquire, insert, update, or delete data in LDAP databases depending on its permissions. The LDAP client commands can address an LDAP server on the local z/VM system and any LDAP server that TCP/IP on z/VM can reach. For example, it can reach another z/VM or z/OS system, or LDAP servers on distributed systems.

The following list provides the LDAP client functions:

- LDAPSRCH: Performs a search by using filter parameters.
- LDAPMDFY: Updates entries. With the **-a** option, data can be added or modified.
- LDAPADD: Adds entries.
- LDAPDLET: Deletes entries.
- LDAPCMPR: Compares LDAP entries with a file or from your standard input.

Most of these command names differ a bit from the LDAP client commands that are found on distributed systems. The reason is that the z/VM LDAP client commands are REXX execs, whose names cannot be longer than eight characters. The command parameters are compatible with other LDAP implementations. See *z/VM TCP/IP User's Guide,* SC24-6127, Chapter: *Using the LDAP Operation Utilities.* Or run them without any parameter to receive a brief command help display.

The z/VM LDAP client requires no tailoring.

2.4.2 z/VM LDAP back-end services and BFS files

The z/VM LDAP server provides all normal LDAP LDBM database services. In LDAP terminology, these databases are also termed *back-end services*. See 1.4.1, “LDAP back-end services” on page 5 for a summary of these back-end services. In z/VM, all LDAP databases are implemented as BFS files.

BFS filespaces and filepools for LDAP

When using the defaults and setup instructions in the manual, *z/VM TCP/IP Planning and Customization,* SC24-6140, the LDAP databases are defined in the VMSYS file pool. The authors of this publication did not find this method to be the best choice because when a new z/VM release is installed, the VMSYS file pool is replaced. Part of the LDAP server code is stored in the ROOT BFS file space in file pool VMSYS. That part should remain there so it gets refreshed with fixes or new z/VM releases.

Our recommendation is to use two different filepools:

VMSYS The file pool to hold the LDAP code and other things.

xyz The file pool to store the LDAP databases and other tailored files such as the password encryption certificates. We decided to use the same name (VMSYSL) as was used in the Redbooks publication, *Security on z/VM* SG24-7471, but you can use any other file pool that you might already have.

Another reason to store the LDAP files in a special file pool might be security reasons. Even though no RACF passwords get stored in the BFS file space used by LDAP, you might not like that VM users without the need to know get easy access to, for example, the LDAP change log database.

2.5 LDAP setup on z/VM

First, the z/VM system must be tailored a bit (such as creation of a file pool and customization of TCP/IP). Then, LDAP is set up and finally RACF is instructed to perform what is required for user information synchronization.

2.5.1 Step-by-step system preparation for LDAP

In these steps, the system is made ready to deploy LDAP. A special SFS file pool is created and the TCP/IP configuration files get prepared.

RACF commands: Many of the RACF commands included in this section must be executed by a z/VM user that is defined as SPECIAL to RACF. These RACF commands are listed at a logical place in the step-by-step procedures.

1. **Create the VMSYSL file pool in user VMSERVL**
   To create a file pool, the easiest way is to copy the directory entry of `VMSERVS`. It is required to change the user ID and the locations of the minidisks. The “normal” user ID to run this step would be MAINT.

a. Create the directory entry, such as what is shown in Figure 2-1.

```
USER VMSERVL VMSERVL 64M 64M BG
 INCLUDE IBMDFLT
 ACCOUNT 1 VMSERVL
 MACH XC
 OPTION MAXCONN 100 NOMDCFS APPLMON ACCT QUICKDSP SVMSTAT
 SHARE REL 1500
 IUCV ALLOW
 IUCV *IDENT RESANY GLOBAL
 IPL CMS
 POSIXOPT SETIDS ALLOW
 CONSOLE 009 3215 T OPERATOR
 LINK MAINT 193 193 RR
 MDISK 191 3390 3338 0001 LX5RES  MR RSERVER  WSERVER
 MDISK 301 3390 0141 0020 LX5U1R  WR RCONTROL WCONTROL
 MINIOPT NOMDC
 MDISK 302 3390 0091 0010 LX5U1R  WR RLOG1    WLOG1
 MINIOPT NOMDC
 MDISK 303 3390 2371 0010 LX5W02  WR RLOG2    WLOG2
 MINIOPT NOMDC
 MDISK 401 3390 1079 0005 LX5U1R  WR RCATALOG WCATALOG
 MDISK 402 3390 3334 0005 LX5W02  WR RCATALOG WCATALOG
 MDISK 501 3390 0891 0050 LX5U1R  WR RDATA    WDATA
 MDISK 502 3390 3284 0050 LX5W02  WR RDATA    WDATA
```

Figure 2-1 VMSERVL directory entry

The catalog and data parts are spread over two minidisks each, on different packs, what allows I/O concurrency in the VMSERVL SFS server. The user ID, minidisks, and LINK permission must be defined in RACF too.

```
RAC ADDUSER VMSERVL PASSWORD(VMSERVL) DFLTGRP(SYS1) OWNER(SYS1)
RAC ALTUSER VMSERVL UACC(NONE)   NAME('SFS FOR LDAP')
RAC RDEFINE VMMDISK VMSERVL.191 OWNER(VMSERVL) UACC(NONE)
RAC RDEFINE VMMDISK VMSERVL.301 OWNER(VMSERVL) UACC(NONE)
RAC RDEFINE VMMDISK VMSERVL.302 OWNER(VMSERVL) UACC(NONE)
RAC RDEFINE VMMDISK VMSERVL.303 OWNER(VMSERVL) UACC(NONE)
RAC RDEFINE VMMDISK VMSERVL.401 OWNER(VMSERVL) UACC(NONE)
RAC RDEFINE VMMDISK VMSERVL.402 OWNER(VMSERVL) UACC(NONE)
RAC RDEFINE VMMDISK VMSERVL.501 OWNER(VMSERVL) UACC(NONE)
RAC RDEFINE VMMDISK VMSERVL.502 OWNER(VMSERVL) UACC(NONE)
RAC PERMIT MAINT.193 CLASS(VMMDISK) ID(VMSERVL) ACC(READ)
RAC PERMIT VMSERVS.191 CLASS(VMMDISK) ID(MAINT) ACC(ALTER)
RAC PERMIT VMSERVL.191 CLASS(VMMDISK) ID(MAINT) ACC(ALTER)
```

Password removal: After the VMSERVL ID is created and used for the initial setup and XAUTOLOGed, it is wise to remove its password with RAC ALTUSER VMSERVL NOPASSWORD. This way, VMSERVL can be only XAUTOLOGed, cannot be hacked, and is never revoked because of RACF not seeing it log on.

b. Format and populate the VMSERVL 191 minidisk.

```
LINK VMSERVS 191 1191 RR
ACCESS 1191 K
```

```
LINK VMSERVL 191 2191 M
FORMAT 2191 L
COPY * * K = = L (OLDDATE
RELEASE 1191 (DETACH
```

c. Rename the copied VMSERVS DMSPARMS file to VMSERVL DMSPARMS and change as shown in Figure 2-2.

```
ADMIN MAINT 5VMTCP40 KRIS TCPMAINT
NOBACKUP
SAVESEGID CMSFILES
FILEPOOLID VMSYSL
FORMAT
USERS 100
```

Figure 2-2 VMSERVL DMSPARMS

In Figure 2-2, the FILEPOOLID is changed to VMSYSL and the TCPMAINT ID is added as an administrator.

d. Rename the copied VMSYS POOLDEF file to VMSYSL POOLDEF and change as shown in Figure 2-3.

```
MAXUSERS=1000
MAXDISKS=500
DDNAME=CONTROL        VDEV=301
DDNAME=LOG1           VDEV=302
DDNAME=LOG2           VDEV=303
DDNAME=MDK00001       VDEV=401     GROUP=1
DDNAME=MDK00002       VDEV=402     GROUP=1
DDNAME=MDK00003       VDEV=501     GROUP=2
DDNAME=MDK00004       VDEV=502     GROUP=2
```

Figure 2-3 VMSYS POOLDEF

e. Build and start the file pool.

```
LOGON VMSERVL NOIPL
IPL CMS
ACCESS (NOPROF
ACCESS 193 R
FILESERV GENERATE VMSYSL POOLDEF
EXEC PROFILE
#CP DISC
```

f. Do not forget to include an XAUTOLOG VMSERVL statement in the PROFILE EXEC of AUTOLOG2. Because TCP/IP starts LDAPSRV, the start of TCP/IP should be postponed until the VMSYSL file pool is ready. This action can be verified by coding the AUTOLOG2 PROFILE EXEC similar to what is shown in Figure 2-4.

```
/**************************/
/*  Autolog Profile Exec  */
/**************************/
Address Command
'CP XAUTOLOG VMSERVL'
'CP XAUTOLOG VMSERVS'
'CP XAUTOLOG VMSERVU'
'CP XAUTOLOG VMSERVR'

ReqSFS='VMSYS VMSYSL'                  /* Required SFS servers */
do i=1 to 20 until sfsNtFnd=''
   'PIPE (end ? sep !) CP Q RESOURCE', /* Which SFS pools are active? */
     '!LK: LOOKUP Word2 Word1',        /* Match with those we need */
     '?VAR reqSFS!SPLIT!LK:',          /* Pass needed SFS to LOOKUP */
     '?LK:!JOIN * / /!Append Literal!VAR SfsNtFnd' /* Not found yet */
   if sfsNtFnd<>'' & i<20 then do
      say 'Waiting till pool(s)' sfsNtFnd 'get ready'
      'CP SLEEP 1 SEC'
   end
end
if sfsNtFnd<>'' then
   say 'LDAPSRV may not start: following filepool(s) not ready:' sfsNtFnd

'CP XAUTOLOG DTCVSW1'
'CP XAUTOLOG DTCVSW2'
'CP XAUTOLOG TCPIP'
```

Figure 2-4 Autolog profile exec

2. **Verify the LDAPSRV directory entry**
The CP directory entry for LDAPSRV should look like what is shown in Figure 2-5.

```
USER LDAPSRV  LDAPSRV  128M 256M BG
 INCLUDE TCPCMSU
 POSIXINFO UID 5 GID 0
 IUCV ALLOW
 IUCV *RPI MSGLIMIT 255
 IUCV VMSYSL PRIORITY MSGLIMIT 255
 OPTION QUICKDSP SVMSTAT
 LINK 5VMTCP40 491 491 RR
 LINK 5VMTCP40 492 492 RR
 LINK TCPMAINT 591 591 RR
 LINK TCPMAINT 592 592 RR
 LINK TCPMAINT 198 198 RR
 MDISK 191 3390 2366 005 LX5W02  MR RLDAPSRV WLDAPSRV MLDAPSRV
```

Figure 2-5 LDAPSRV CP Directory entry

Notice the addition of **`IUCV *RPI MSGLIMIT 255`** as an extra statement. It gives the LDAP server permission to use the *RPI system IUCV service to receive user definition change notifications from the RACF server. LDAP can then log these events in its GDBM database.

3. **Enroll the LDAP server as a BFS user in the VMSYSL file pool**
   Log on as an administrator listed in the VMSERVL DMSPARMS file. Create the BFS file space and define TCPMAINT as an SFS user by issuing:

   ```
   ENROLL USER LDAPSRV VMSYSL (BFS BLOCKS 15000
   ENROLL USER TCPMAINT VMSYSL (BLOCKS 1000
   ```

   PUBLIC not in the file pool: We do *not* enroll PUBLIC in this file pool. This process assures only explicitly enrolled users can get access to the VMSYSL file pool. An extra security measure might be to remove the `IUCV ALLOW` statement from the VMSERVL directory entry. Consequently, an `IUCV VMSYSL PRIORITY MSGLIMIT 255` must be inserted in the directory entry for LDAPSRV and any other user needing access to VMSYSL, such as TCPMAINT.

   Set up the BFS permissions for the just created BFS file space:

   ```
   openvm owner  /../VMBFS:VMSYSL:LDAPSRV/ = LDAPSRV
   ```

   By making LDAPSRV the owner of that BFS file space, all Posix file permission bits are set OK.

4. **Prepare TCPMAINT for BFS usage**

   If you decided not to create a filespool for BFS, TCPMAINT might not have sufficient SFS authorities in the file pool you use for LDAP. The following command might be used (CP class C required):

   ```
   CP SEND VMSERVxx GRANT ADMIN TCPMAINT
   ```

   This command grants administrator permissions until the SFS server is restarted.

   During the LDAP setup, TCPMAINT is also involved in BFS work. By default though, TCPMAINT has no BFS file space mounted. We added the following statement in the CP directory of TCPMAINT:

   ```
   POSIXINFO fsroot /../VMBFS:VMSYS:ROOT/ UID 0 GNAME system
   ```

   To activate this change, re-IPL CMS in TCPMAINT.

   TCPMAINT: It is suggested to use TCPMAINT for the following steps.

5. **Update the TCP/IP definitions for LDAP**

The PROFILE TCP/IP file on TCPMAINT 198 must be updated to include LDAPSRV. The updated sections of PROFILE TCP/IP look like what is shown in Figure 2-6.

```
OBEY
OPERATOR TCPMAINT MAINT MPROUTE DHCPD REXECD SNMPD SNMPQE LDAPSRV
KRIS BEN CONRAD DAVE
ENDOBEY
; -------------------------------------------------------------
PORT
  20   TCP FTPSERVE  NOAUTOLOG ; FTP Server
  21   TCP FTPSERVE            ; FTP Server
  23   TCP INTCLIEN            ; TELNET Server
--------------------  10  line(s) not displayed ---------------
  389  TCP LDAPSRV             ; LDAP Server
; 389  UDP LDAPSRV             ; LDAP Server
--------------------  5  line(s) not displayed ----------------
  636  TCP LDAPSRV             ; LDAP Server (Secure)
; 636  UDP LDAPSRV             ; LDAP Server (Secure)
--------------------  7  line(s) not displayed ----------------
; -------------------------------------------------------------
AUTOLOG
--------------------  2  line(s) not displayed ----------------
  FTPSERVE  0           ; FTP Server
; IMAP      0           ; IMAP Server
; IMAPAUTH  0           ; IMAP Authentication Virtual Machine
  LDAPSRV   0           ; LDAP Server
--------------------  12  line(s) not displayed ---------------
ENDAUTOLOG
```

Figure 2-6 Updated PROFILE TCPIP

LDAPSRV is added to the OBEY list and uncommented in the PORT section. The latter is required to allow LDAPSRV to use ports 389 and 636. Adding LDAPSRV in the AUTOLOG section ensures that when the TCP/IP server is started, it automatically starts LDAPSRV, and that TCP/IP restarts LDAPSRV in case it terminates. It is probably a good idea not to add LDAPSRV in the AUTOLOG section while still in the test phase. Therefore, only you decide whether LDAPSRV is started during testing.

We also enabled the FTP server. In later steps, it is required to exchange some files with other systems, and FTP might be helpful then.

These changes can be activated by recycling the TCP/IP server, or can be done dynamically with the **OBEYFILE** command, which is executed by a user ID in the OBEY list. The user that is executing the **OBEYFILE** command must authorize the TCP/IP server to link to its A-disk. Setting these permissions for TCPMAINT is more than appropriate:

```
RAC PERMIT TCPMAINT.191 CLASS(VMMDISK) ID(TCPIP) ACCES(READ)
```

Copy the OBEY and PORT sections *completely*, then issue the **OBEYFILE** command against the file, for example by using the file LDAP TCPIP A, issue:

```
OBEYFILE LDAP TCPIP A
```

6. **Update the DTCPARMS definitions for LDAP**

   The default DTCPARMS definitions define LDAP, but do not include RACF support. Furthermore, we want to use another file pool (VMSYSL) instead of the default (VMSYS). For these two reasons, DTCPARMS updates are required.

   Remember the general DTCPARMS rules: Never change IBM DTCPARMS; create one of your own with the required overrides; its file name can be user ID, node ID, or SYSTEM; and it must be stored on TCPMAINT 198. Refer to *z/VM TCP/IP Planning and Customization,* SC24-6140 for details.

   We preferred to use SYSTEM DTCPARMS, which was already created by IPWIZARD during the z/VM installation. The added statements look like what is shown in Figure 2-7.

```
:nick.ftp     :type.class
              :name.FTP daemon
              :command.SRVRFTP
              :runtime.PASCAL
              :diskwarn.YES
              :anonymous.YES
              :ESM_Enable.YES
              :ESM_Validate.RPIVAL
              :ESM_Racroute.RPIUCMS

:nick.ldap    :type.class
              :name.LDAP daemon
              :command.LDAPSRV
              :runtime.C
              :memory.128M
              :mixedcaseparms.YES
              :mount. /../VMBFS:VMSYS:ROOT/   /  ,
                      /../VMBFS:VMSYSL:        /var/ldap
              :ESM_Enable.YES
              :ESM_Racroute.LDAPESM
```

Figure 2-7 SYSTEM DTCPARMS, new statements for LDAP and FTP

Even though only a few highlighted items differ from the default in IBM DTCPARMS, all lines comprising a :nick. section must be included in the SYSTEM DTCPARMS file.

Readers that are familiar with BFS file space naming conventions might find that the second line on the :mount. tag in Figure 2-7 looks syntactically wrong. No user ID follows the name of the filepoolid (**VMSYSL**, above). It is however, correct. The LDAP code completes the name of the user ID in which it is running (LDAPSRV in this case). The full name becomes: /../VMBFS:VMSYSL:LDAPSRV/.

2.5.2 Preparation steps to work with the LDAP server on z/VM

Some work methods were used during this project that can make things easier. We want to share them with you.

Restarting the LDAP server

The LDAP server uses BFS resident files. To avoid problems, it is recommended that the server is stopped in a clean way. A CMS user included in the OBEY list in PROFILE TCP/IP is authorized to use LDAP operator commands, like **shutdown**.

TCPMAINT should thus be able to issue the following command:

```
CP SMSG LDAPSRV shutdown
....wait ...
CP XAUTOLOG LDAPSRV
```

When logged on to LDAPSRV, it can be stopped in a similar way:

```
#CP SMSG * shutdown
```

During the setup phase, you might find an exec like the one shown in Figure 2-8 useful. It is included in this publication's samples.

```
/* This exec can be used to restart the LDAP server
    +-----------------------------------------------------------+
    | format:  | RESLDAP                                        |
    +-----------------------------------------------------------+
  Written by: Kris Buelens IBM Belgium;  KRIS at VMLINUX5 23 Sep 2008*/
address command

do i=1 to 2
  'PIPE CP IND USER LDAPSRV EXP|FIND CPU|xlate = 40|spec w4.2|VAR CTIME1'
  if rc=45 then signal StartItUp /* RC 45 = Not logged on */
  if rc=3 then do                /* Not authorised for IND USER xxx */
     if i=1 then 'CP SET PRIVCLASS * +E'
     else say 'Cannot check LDAP restart: not auth for IND USER LDAPSRV'
  end
end i

'CP SM LDAPSRV SHUTDOWN'
if rc=45 then signal StartItUp /* RC 45 = Not logged on */
if rc<>0 then call Exit rc,'LDAPSRV does not accept SHUTDOWN request',,
                          'Maybe it is not realy started yet'
do i=1 for 30 until rc<>0
   if i//5=0 then say 'Waiting for LDAPSRV to log off'
   'CP SLEEP 1 SEC ATTN'
   if rc<>0 then call exit 0,'Abandoned' /* End-user pressed ENTER */
   'PIPE CP IND USER LDAPSRV EXP|FIND CPU|xlate = 40|spec w4.2|VAR CTIME'
   if rc=0 then                  /* Maybe TCPIP stack restarted LDAP  */
    if ctime < ctime1 then leave /*  yes it did (connectTime now less)*/
end
if i>30 then say '***>> LDAPSRV did not SHUTDOWN itself. <<***'

StartItUp:
'EXEC AUTOSCIF LDAPSRV'
exit rc
EXIT: /* general (error)exit routine */
 parse upper source . . myname mytype . syn .
 do i=2 to arg()   /* give errormessages (if any) */
    say myname':' arg(i)
 end
 exit arg(1)
```

Figure 2-8 An exec to restart the LDAP server

The AUTOSCIF exec is part of the SCIF package. It XAUTOLOGs a user and sets its secondary user to you, so you can watch it start. It can be downloaded from the z/VM web page:

http://www.vm.ibm.com/download/packages/descript.cgi?SCIF

Running LDAP commands from CMS

The parameters of LDAP commands can become long, maybe even longer than the 3270 input area. On some commands, the LDAP command syntax requires the use of double quotation mark characters. With the default CP TERMINAL settings, the double quotation mark is defined as an escape character, which means they should be doubled to keep one.

For these reasons, most LDAP commands that we used during this project were included in simple REXX execs, like the one shown in Figure 2-9.

```
 SPWENV   EXEC     A1  V 130  Trunc=130 Size=7 Line=0 Col=1
====>
* * Top of File * * *
/* Rexx - extract a password envelope from LDAP */
'ldapsrch -h 127.0.0.1' ,
   '-D racfid=itdi,profiletype=user,cn=RACFVM -w itdi' ,
   '-L -b racfid=OPERATNS,profiletype=user,cn=RACFVM' ,
   '"objectclass=*" racfpasswordenvelope' ,
   'racfpassphraseenvelope'
exit rc
* * * End of File * * *
```

Figure 2-9 Sample LDAP command that is embedded in a REXX exec

The disadvantage of the preceding sample is that for some searches there is no console output at all, and you are uncertain if something was executed. Therefore, our execs now have general structure as illustrated in Figure 2-10.

```
 IVP1     EXEC     A1  V 130  Trunc=130 Size=11 Line=0 Col=1 A
====>
* * * Top of File * * *
/* LDAP search, using LDAP's DS CONF authentication */
ex='ldapsrch -h 127.0.0.1 -D cn=Admin -w secret -s base',
                        '-b o=ibm objectclass=*'
address command
parse source . . myname mytype .
say center(' 'myname mytype' ',79,'-')
Say center(sourceline(1),79)
say left('',79,'-')
'PIPE (sep |) VAR EX|SPILL 79 offset 12|cons'
address CMS ex
exit rc
* * * End of File * * *
```

Figure 2-10 General REXX exec to execute an LDAP command

The exec displays the first (comment) line and the executed command before running the command.

2.5.3 Step-by-step setup of LDAP

Throughout this section, the LDAP server is tailored to make it usable by Tivoli Directory Integrator for user information synchronization (passwords, password phrase, RACF groups, and so on).

Create the DS ENVVARS and DS CONF files

Samples are provided on TCPMAINT 591 (LDAP-DS SAMPENVR and LDAP-DS SCONFIG). They must be copied to TCPMAINT 198 as DS ENVVARS and DS CONF and tailored to your needs.

1. **The environment variables file DS ENVVARS**

   The only update that is applied to the DS ENVVARS file is the location of certificates that are used to encrypt and decrypt the password and password phrase envelopes. This information is often termed *key ring* or *key database*. The build process of the key database is described in 2.6.1, “Creating the keys and certificates” on page 45.

   The updated section is shown in Figure 2-11.

```
NLSPATH=/usr/lib/nls/msg/%L/%N
#
LANG=En_US.IBM-1047
#
#LIBPATH=/usr/lib
#
#TZ=GMT0
#
#IRR.PWENV.KEYFILE=./IRR.PWENV.KEYRING
IRR.PWENV.KEYFILE=/var/ldap/IRR.PWENV.KEYRING
```

Figure 2-11 DS ENVVARS file, with updated keyring information

   Because the key database should not get lost when installing a new VM release, it is not recommended to store it in the VMSYS file pool. The environment variable `IRR.PWENV.KEYRING` set in Figure 2-11 makes an LDAP search for its key ring in the BFS directory `/var/ldap`. Looking back at the tailored SYSTEM DTSCPARMS file (shown in Figure 2-13 on page 35), you can see that the BFS file space `/../VMBFS:VMSYSL:LDAPSRV/` is mounted at `/var/ldap`. Hence, the keyring files are stored in VMSYSL and no longer in VMSYS.

2. **The configuration file DS CONF**

   In this tailoring pass an LDAP administrator is defined, and the three required LDAP back ends (LDBM, GDBM, and SDBM) are defined in the DS CONF file. The sample config file contains lots of comment lines. We have hidden most of them in Figure 2-12 on page 34 that is used to display the tailored lines.

   DS CONF file: The DS CONF file is somehow case-sensitive. The keywords are not case-sensitive. Some values are. Issue `SET CASE Mixed Ignore` when using XEDIT to modify DS CONF.

The first change to make (shown in Figure 2-12) is the definition of a temporary LDAP administrator.

```
 DS       CONF     B1  V 80  Trunc=80 Size=1899 Line=0 Col=1 Alt=4
====>
* * * Top of File * * *
--------------------  184  line(s) not displayed --------------------
#-----------------------------------------------------------------------
# adminDN <distinguished name>
#
# Description:
#   The adminDN option specifies the distinguished name (DN) of the
#   administrator for this LDAP server.  Normally, this DN has
--------------------  33  line(s) not displayed --------------------
adminDN "cn=Admin"
# adminDN "cn=Admin, o=Your Company"

#-----------------------------------------------------------------------
# adminPW <password>
#
# Description:
#   The adminPW option specifies the password for the administrator
#   defined by the adminDN option.
--------------------  13  line(s) not displayed --------------------
adminPW secret

--------------------  1652  line(s) not displayed --------------------
* * * End of File * * *
```

Figure 2-12 Adapted DS CONF file, part one

When configuring the LDAP server for the first time, the LDAP administrator password is defined in the DS.CONF file. This is required since no user entries have yet been created in the LDBM database. Once an administrator ID has been created in the LDBM with a password, the adminPW in the DS.CONF file can be removed.

In Figure 2-13, the three LDAP back ends are defined (by uncommenting the `#database` records in the sample file). Part of this tailoring is the selection of a suffix for the SDBM and LDBM. These suffixes are used in various LDAP commands. The suffix for the SDBM is even used by Tivoli Directory Integrator.

```
 DS       CONF     B1  V 80  Trunc=80 Size=1899 Line=953 Col=1 Alt=4
====>
#
# DATABASE-SPECIFIC SECTIONS
#
--------------------  10  line(s) not displayed --------------------

# SDBM-specific CONFIGURATION SETTINGS

--------------------  19  line(s) not displayed --------------------
database SDBM GLDBSD31
--------------------  14  line(s) not displayed --------------------
# suffix <dn-suffix>
#
# Description:
#   The suffix option specifies the root of a subtree in the namespace
#   managed by this server within this backend.
--------------------  7  line(s) not displayed --------------------
suffix "cn=RACFVM"

--------------------  49  line(s) not displayed --------------------

# LDBM-specific CONFIGURATION SETTINGS

--------------------  19  line(s) not displayed --------------------
database LDBM GLDBLD31
--------------------  16  line(s) not displayed --------------------
# suffix <dn-suffix>
#
# Description:
#   The suffix option specifies the root of a subtree in the namespace
#   managed by this server within this backend.
--------------------  8  line(s) not displayed --------------------
suffix "o=ibm"

--------------------  474  line(s) not displayed --------------------
#
# GDBM-specific CONFIGURATION SETTINGS
#
--------------------  19  line(s) not displayed --------------------
database GDBM GLDBGD31
--------------------  283  line(s) not displayed --------------------
* * * End of File * * *
```

Figure 2-13 Adapted DS CONF file, part two

RACF permissions for the LDAP server

The LDAP server needs two special permissions to enable it to perform its role in the synchronization setup. Because these permissions belong to the RACF FACILITY class, that class must be activated if it is not already.

From a RACF user with the SPECIAL attribute, issue the following commands:

1. Activate the FACILITY class:

   ```
   RAC SETROPTS CLASSACT(FACILITY)
   ```

2. Give LDAPSRV permission to use RACROUTE so that it can authenticate LDAP clients with RACF.

   ```
   RAC RDEF FACILITY ICHCONN UACC(NONE)
   RAC PERMIT ICHCONN CL(FACILITY) ID(LDAPSRV) ACCESS(UPDATE)
   ```

3. Allow LDAP to use CP's IUCV service that returns the RACF change log records to LDAP:

   ```
   RAC RDEFINE FACILITY IRR.CHANGELOG UACC(NONE)
   RAC PERMIT IRR.CHANGELOG CLASS(FACILITY) RESET ID(LDAPSRV)
   ```

Verification step: Start the LDAP server

In the step-by-step process, it should now be possible to start the LDAP server. Because this is the first time, it is recommended not to start it with XAUTOLOG, but to log on to LDAPSRV from a 3270 session. A real logon assures that you can see all messages that are signaling problems, like RACF rejecting the LINK to required minidisks.

There should not be any error messages. One important message should be present. It is displayed soon after the startup and confirms that LDAPSRV can use RACROUTE.

```
RPICMS016I USER/RACF VM Racroute communication path is established.
```

From a user that is authorized in the TCPIP OBEY list (like TCPMAINT), issue the following command:

```
smsg ldapsrv display monitor
```

It should display a response that starts with the following words:

```
Monitor Statistics
------------------

 Server Version:      z/VM Version 5 Release 4 IBM
                      LDAP Server
```

Verification step: Run LDAP searches

It should be possible now to run some simple LDAP search commands. You can use the sample **`IVP* EXEC`**, though you might have to change some user IDs to reflect your VM system.

The first command searches the LDBM database, authenticated with the LDAP administrator's user ID and password as defined in the DS CONF file (Figure 2-12 on page 34). We use the loopback address (127.0.0.1) to point to the local z/VM system. The **`-D`** and **`-w`** options are used to authenticate. The **`-b`** option defines what is searched for (sample: **`IVP1 EXEC`**):

```
ldapsrch -h 127.0.0.1 -D cn=Admin -w secret -s base -b o=ibm objectclass=*
```

Because the LDBM database is still empty (o=ibm is thus undefined), the following expected response is indicated:

```
ldap_search: No such object
ldap_search: additional info: R004071 DN 'o=ibm' does not exist
(ldbm_process_request)
```

Alternatively, an SDBM database is defined in the DS CONF file and LDAPSRV is given permission to use RACROUTE. Therefore, it should be possible to authenticate by using RACF. The same search command shown previously must then be written as follows (sample: IVP2 EXEC):

```
ldapsrch -h 127.0.0.1 -D racfid=KRIS,profiletype=user,cn=RACFVM -w KRISPW
                      -s base -b o=ibm objectclass=*
```

KRIS is a user that is defined to RACF in z/VM. *RACFVM* is the *suffix* that is assigned to the SDBM in the DS CONF file (Figure 2-13 on page 35). And, *KRISPW* is the RACF password of KRIS. The expected response is the same as in the previous search.

The LDAP SDBM database is a mapping of the RACF database, meaning that an LDAP search to it displays RACF information (sample: IVP3 EXEC):

```
ldapsrch -h 127.0.0.1 -D racfid=U80027,profiletype=user,cn=RACFVM -w U80027PW
        -L -b "racfid=U80027,profiletype=user,cn=RACFVM" "objectclass=*"
```

In this first RACF search, the user ID that is searched for is also used to authenticate the command. This way, RACF has no authorization problems. The search corresponds to a **RAC LISTUSER <userid>** executed in CMS. Therefore, replace U80027 with a user ID for which you know the password. The response should contain the following RACF information:

```
dn: racfid=U80027,profiletype=USER,cn=RACFVM
racfid: U80027
racfauthorizationdate: 09/26/08
racfowner: RACFID=SYS1,PROFILETYPE=GROUP,CN=RACFVM
racfpasswordinterval: 30
racfpasswordchangedate: 10/01/08
racfprogrammername: KRIS BUELENS
racfdefaultgroup: RACFID=SYS1,PROFILETYPE=GROUP,CN=RACFVM
racflastaccess: 10/01/08/13:49:36
racflogondays: SUNDAY
racflogondays: MONDAY
....
```

Sample: **IVP4 EXEC** is similar, but uses a password phrase instead of a password.

With the three search commands above, it is verified that LDAP can use RACF as an authenticator and that RACF information can be retrieved by using the LDAP interface. We used an LDAP client from CMS, but an **ldapsearch** command that is issued from any other system should work equally well.

Remove the LDAP administrator password from DS CONF

It is not recommended to have passwords that are stored in readable form in configuration files. Therefore, remove the LDAP administrator's password from the DS CONF file. It is stored in the LDBM database instead.

This process can be done by running the following steps:

1. **Load LDBM schemas**

   Before a user ID can be defined in the LDBM, its definition must be complemented with the schemas. Two schema input files (IBMSCHEM LDIF and USRSCHEM LDIF) are provided on TCPMAINT 591 and must be loaded, by issuing the following commands:

   ```
   ldapmdfy -h 127.0.0.1 -D cn=Admin -w secret -f //usrschem.ldif -u on
   ldapmdfy -h 127.0.0.1 -D cn=Admin -w secret -f //ibmschem.ldif -u on
   ```

2. **Define organization and LDAP administrator in an LDIF file**

   Create a CMS file named `ADDADMIN LDIF` to define the organization and the administrator user ID to be used later. Use Figure 2-14 or the sample file, `ADDAMIN.LDIF`, as an idea.

```
 ADDADMIN LDIF     A1  V 80  Trunc=80
====>
* * * Top of File * * *
dn: o=ibm
objectclass: top
objectclass: organization
o: ibm

dn: cn=ldapAdmin, o=ibm
objectclass: top
objectclass: person
objectclass: organizationalPerson
cn: ldapAdmin
sn: LDAP Administrator
userPassword: secret

* * * End of File * * *
```

Figure 2-14 LDIF file to define an LDAP administrator

3. **Import the definition into the LDBM database**

   To import the definitions from the LDIF file into the LDBM database, defining both an organization and a user ID in that organization (sample ADDADMIN.EXEC), issue the following command:

   ```
   ldapadd -h 127.0.0.1 -D "cn=Admin" -w secret -f //ADDADMIN.LDIF
   ```

   We still authenticate with the administrator that is defined in the DS CONF file.

   If the LDAP search used in the first verification test is repeated, results are reported (sample: IVP1 EXEC):

   ```
   ldapsrch -h 127.0.0.1 -D cn=Admin -w secret -s base -b o=ibm objectclass=*
   ```

   ```
   o=ibm
   objectclass=top
   objectclass=organization
   o=ibm
   ```

   This search should return the user that was added as a future LDAP administrator (sample: IVP5 EXEC):

   ```
   ldapsrch -h 127.0.0.1 -D cn=Admin -w secret-s base -b
   cn=ldapAdmin,o=ibm objectclass=*
   ```

```
cn=ldapAdmin, o=ibm
objectclass=top
objectclass=person
objectclass=organizationalPerson
cn=ldapAdmin
....
```

4. **Update the administrator information in the DS CONF file**

   When the new user, *ldapAdmin* in our example, is defined in the LDBM database, the DS CONF files can be changed to assign it the role of LDAP administrator. This change and the removal of the password is illustrated in Figure 2-15.

```
 DS       CONF     B1  V 80  Trunc=80 Size=1899 Line=0 Col=1 Alt=4
====>
* * * Top of File * * *
--------------------  184  line(s) not displayed --------------------
#------------------------------------------------------------------------
# adminDN <distinguished name>
#
# Description:
#   The adminDN option specifies the distinguished name (DN) of the
#   administrator for this LDAP server.  Normally, this DN has
--------------------  33  line(s) not displayed --------------------
# adminDN "cn=Admin"
adminDN "cn=ldapAdmin, o=ibm"

#------------------------------------------------------------------------
# adminPW <password>
#
# Description:
#   The adminPW option specifies the password for the administrator
#   defined by the adminDN option.
--------------------  13  line(s) not displayed --------------------
#adminPW secret

--------------------  1652  line(s) not displayed --------------------
* * * End of File * * *
```

Figure 2-15 Adapted DS CONF file, part one

5. **Restart LDAP and verify**

   Restart the LDAP server and verify that the previous admin is unable to be used, and that the new administrator is functional. It means that the first command shown below should fail, the second should work:

```
ldapsrch -h 127.0.0.1 -D cn=Admin -w secret -s base -b o=ibm objectclass=*
ldapsrch -h 127.0.0.1 -D cn=ldapAdmin,o=ibm -w secret -b o=ibm objectclass=*
```

The book, *z/VM TCP/IP Planning and Customization,* SC24-6140, explains how to store passwords in the LDAP LDBM in an encrypted form.

Make the LDAP server listen for change log records sent by RACF

The last update to apply to the DS CONF file is to open the LDAP Program Call interface. This interface is used by RACF to forward change log requests to LDAP. LDAP stores those

requests in its GDBM database. The update that is shown in Figure 2-16, consists of uncommenting the *Listen* record with **:pc** as port number.

LDAP Program Call interface: On z/VM, a program call is not used. Instead, change log requests are returned by using IUCV to *RPI. The configuration file changes (enabling the Program Call interface) were made to keep consistency between z/OS and z/VM.

```
 DS       CONF     B1  V 80  Trunc=80 Size=1899 Line=367 Col=1 Alt=0
====>
--------------------  366  line(s) not displayed --------------------
#--------------------------------------------------------------------
# listen <ldap-url>
-------------------- 15 line(s) not displayed --------------------
#   To listen for change log requests from RACF on the Program Call
#   interface:
#      listen ldap://:pc
--------------------  56 line(s) not displayed --------------------
listen ldap://:389
listen ldap://:pc
--------------------  1455 line(s) not displayed --------------------
```

Figure 2-16 Updated DS CONF file for RACF change log requests

When RACF starts making change log records, LDAP must be authorized to retrieve them:

```
RAC RDEFINE FACILITY IRR.CHANGELOG
RAC PERMIT IRR.CHANGELOG CLASS(FACILITY) RESET ID(LDAPSRV)
```

Failure to do so causes RPICLM002E IUCV CONNECT has been rejected during the startup of LDAP.

IRR.CHANGELOG: Documentation on IRR.CHANGELOG was omitted from the publication, SC24-6142-01 *z/VM RACF Security Administrator's Guide V5.4,* and will be added when the publication is re-issued.

To activate this change, LDAPSRV must once more be restarted.

2.5.4 Step-by-step setup of RACF

Setup for RACF change logging

To make cross system user information synchronization work, RACF must be instructed to forward the change events to LDAP so that it gets logged in the LDAP GDBM database.

This is a fairly simple step; define the following profiles in the RACFEVNT class:

```
RAC RDEFINE RACFEVNT NOTIFY.LDAP.USER
RAC RDEFINE RACFEVNT NOTIFY.LDAP.GROUP
RAC RDEFINE RACFEVNT NOTIFY.LDAP.CONNECT
```

Or, you can use a single generic profile. However, one was not used during this project. The following single generic profiles are used for an example:

```
SETROPTS GENERIC(RACFEVNT)
RDEFINE RACFEVNT NOTIFY.LDAP.*
```

A next step is to activate the RACFEVNT class and to load its profiles in storage for improved performance:

```
RAC SETROPTS CLASSACT(RACFEVNT) RACLIST(RACFEVNT)
```

If the RACFEVNT class was already raclisted, issue the following command:

```
RAC SETROPTS RACLIST(RACFEVNT) REFRESH
```

The preceding commands ensure that RACF creates the change event records. For more information, refer to *z/VM RACF Security Server Security Administrator's Guide,* SC24-6142, chapter *LDAP event notification*.

Verification step

By making a change to the RACF database followed by an LDAP query, it can be verified whether RACF is sending change events to LDAP. When the RACF profile `NOTIFY.LDAP.USER` is created, a command such as **`RAC ALTUSER OPERATNS PASSWORD(abcdef)`** creates a change log record. The following LDAP query can be used to search the LDAP GDBM changelog database (sample: **`IVP6 EXEC`**).

Example 2-1 Searching the changelog

```
ivp6
---------------------------------- IVP6 EXEC ----------------------
           /* LDAP search for one changelog                        */
---------------------------------------------------------------------
ldapsrch -h 127.0.0.1 -D cn=ldapAdmin,o=ibm -w topDog -b
            "changeNumber=1,cn=changelog" "objectclass=*"
changeNumber=1,cn=changelog
objectclass=top
objectclass=changeLogEntry
objectclass=ibm-changeLog
changenumber=1
changetype=modify
targetdn=RACFID=OPERATNS,PROFILETYPE=USER,CN=RACFVM
changes=replace: racfPassword
racfPassword: *NoEnvelope*
-

ibm-changeinitiatorsname=RACFID=KRIS,PROFILETYPE=USER,CN=RACFVM
changetime=20080926192904.890000Z
Ready; T=0.01/0.01 11:06:59
```

We explicitly request changeNumber 1. If it is not found, it might have been deleted by some other LDAP user. By changing the search parameter to

```
-b "cn=changelog" "objectclass=*"
```

All changelog records are returned.

Grant the Tivoli Directory Integrator server access to GDBM changelog

To allow the Tivoli Directory Integrator server to access the changelog entries in the GDBM, update the ACL of the GDBM. In our configuration, the Tivoli Directory Integrator server accesses the GDBM by using a bind of racfid=ITDI,profiletype=USER,cn=RACFVM. Place the following commands into a file, for example `modacl.ldif`. The Tivoli Directory Integrator user ID must be defined to RACF.

Example 2-2 shows an ACL entry.

Example 2-2 ACL entry

```
dn: cn=changelog
changetype: modify
add: aclEntry
aclEntry:access-id:racfid=ITDI,profiletype=USER,cn=RACFVM:
 normal:rsc:sensitive:rsc:critical:rsc:restricted:rsc:system:rsc
```

Then, run the following command:

```
ldapmdfy -h 127.0.0.1 -D "cn=ldapAdmin,o=ibm" -w <adminpw> -f //MODACL LDIF
```

Setup RACF profiles for password enveloping

As mentioned in the introduction of this publication, to synchronize passwords, RACF must not only send change logs to LDAP, but, it must also store the users' passwords and pass phrases in *envelopes*.

The following setup steps are required to make this process happen:

1. Create RACF profiles by which you can selectively authorize users to have an envelope created when their password or phrase is changed.

   ```
   RAC RDEFINE RACFEVNT PASSWORD.ENVELOPE UACC(NONE)
   RAC RDEFINE RACFEVNT PASSPHRASE.ENVELOPE UACC(NONE)
   RAC PERMIT PASSWORD.ENVELOPE CLASS(RACFEVNT) ID(OPERATNS) ACC(READ)
   RAC SETROPTS RACLIST(RACFEVNT) REFRESH
   ```

   Note: The permit to OPERATNS is just a sample for a test. When the password for the OPERATNS id is changed, it is enveloped.

2. Create the SYNCH group. For the scenarios described in this publication, users whose passwords and password phrases are to be synchronized, should get this authorization by a connection to group, SYNCH. We create the setup for it now:

   ```
   RAC ADDGROUP SYNCH OWNER(SYS1) SUPGROUP(SYS1) DATA('Synchronize their
   passwords')
   RAC PERMIT PASSWORD.ENVELOPE   ID(SYNCH) CLASS(RACFEVNT) ACC(READ)
   RAC PERMIT PASSPHRASE.ENVELOPE ID(SYNCH) CLASS(RACFEVNT) ACC(READ)
   ```

3. Create a user that the Tivoli Directory Integrator server uses to authenticate with z/VM:

   ```
   RAC ADDUSER ITDI PASSWORD(ITDI) DFLTGRP(SYS1) OWNER(SYS1) UACC(NONE)
   RAC ALTUSER ITDI NAME(ITDI server) PASSWORD(ITDI) NOEXPIRED SPECIAL
   ```

   If this user does not have enough privileges in RACF, the Tivoli Directory Integrator server is not able to retrieve information of all z/VM users. Our tests, which are based on several LDAP queries that are issued from CMS, reveal that it should have the RACF SPECIAL attribute.

   Although we used a trivial password for the Tivoli Directory Integrator user ID for our test system, you should not. Follow your installations security requirements for password generation.

4. Create RACF profiles and allow the Tivoli Directory Integrator user so it can retrieve the envelopes:

   ```
   RAC RDEFINE FACILITY IRR.RADMIN.EXTRACT.PWENV
   RAC RDEFINE FACILITY IRR.RADMIN.EXTRACT.PPENV
   RAC PERMIT IRR.RADMIN.EXTRACT.PWENV CLASS(FACILITY) ID(ITDI) ACC(READ)
   RAC PERMIT IRR.RADMIN.EXTRACT.PPENV CLASS(FACILITY) ID(ITDI) ACC(READ)
   ```

Creating certificates and keys for enveloping

If you are unfamiliar with encryption, keys, and certificates, a good overview can be found in the LDAP chapter in the document, *z/VM TCP/IP Planning and Customization,* SC24-6140, section *Setting up for SSL/TLS*.

Note: As opposed to z/OS, the encryption and decryption are not executed by RACF itself. In z/VM, the RACF server uses a modified version of CMS, in which not all CMS services are available. The RACF server gets help from the LDAP server in the encrypt and decrypt process. This means that the LDAP server must be running when envelopes are created or re-encrypted for use by Tivoli Directory Integrator. It also means that the keyring holding the RACF certificates must be accessible by the LDAP server.

TCPMAINT: It is suggested to use TCPMAINT for the following steps.

Create the keys with gskkyman

Take the following steps for the creation:

1. **Mount the BFS file space**

   As mentioned in"BFS filespaces and filepools for LDAP" on page 24, we recommend not to use the VMSYS file pool as the storage medium for your installation's BFS data. The certificates and keys are data that you do not want to get lost when you install a new VM release. Therefore they should be stored in a file pool of your own. In our examples, we use the VMSYSL file pool, which we created earlier.

   Consequently, before starting the **gskkyman** command, mount the LDAPSRV file space with the following command:

   ```
   openvm mount /../VMBFS:VMSYSL:LDAPSRV/ /
   ```

   Or, use the MOUNTLDP EXEC that we provide. Thanks to this mount, you can follow the examples and respond IRR.PWENV.KEYRING to the gskkyman prompt for the database name.

2. **Use gskkyman**

   Use the **gskkyman** command to do the following functions:

   – Define a local certificate authority certificate acting as the certificate authority

   – Generate an X.509 V3 certificate for the RACF use for enveloping

   – Create an X.509 V3 certificate for intended recipients or import a certificate

   The options to specify are explained in the publication, *z/VM RACF Security Server Security Administrator's Guide,* SC24-6142.

   Follow the steps in section 18.2.1.3, *Defining a local Certificate Authority certificate acting as the certificate authority*, and section 18.2.1.4, *Generating an X.509 V3 certificate for RACF's use for enveloping*.

   In addition, the recipient (in our case, Tivoli Directory Integrator) must have its certificate loaded in the KDB. We opted to import the recipient certificate that was generated for Tivoli Directory Integrator. This certificate was generated on a distributed system and transferred to z/VM, and finally imported by using gskkyman.

3. **Export and Import certificates**

   Use gskkyman to export the public part of the RACF certificate. This certificate is used by Tivoli Directory Integrator to verify the digital signature of the password envelope that is sent by RACF. What is not mentioned in current documentation is that the files, for which

gskkyman prompts, are all BFS fileids. Consequently, after an export, the file will not be found on your A-disk, and a file to import is not on your A-disk. Do not use the common CMS syntax (*filename space filetype*), but something such as *filename dot filetype* (case sensitive). The **openvm** command is an easy way to transfer files between your A-disk and BFS. Example 2-3 illustrates an export in gskkyman, followed by an openvm getbfs to extract the file from the BFS.

Example 2-3 gskkyman export example

```
Enter option number (press ENTER to return to previous menu):
6

       Export File Format

   1 - Binary ASN.1 DER
   2 - Base64 ASN.1 DER
   3 - Binary PKCS #7
   4 - Base64 PKCS #7

Select export format (press ENTER to return to menu):
2

Enter export file name (press ENTER to return to menu):
LDAPssl_VM5.b64

Certificate exported.
......
Ready; T=1.91/1.97 15:31:18
openvm getbfs LDAPssl_VM5.b64 LDAP_VM5 b64 A
Ready; T=0.01/0.01 15:31:47
```

For an import use **openvm putbfs** ***fn ft fm bfspathname***.

4. **Restart LDAP**

   When the certificates are built, restart the LDAP server. Beware however, during the project, LDAPSRV regularly reported that it could not read the key ring file.

   ```
   GLD116OE Unable to initialize the LDAP client SSL support:
   Error 113, Reason 2.
   ```

   To avoid this problem, issue IPL CMS in the virtual machine where gskkyman was run before restarting LDAPSRV.

Verification: Are envelopes created and can they be retrieved?

As the last step in the basic z/VM setup process, we use some LDAP search commands issued from CMS.

Are password envelopes being created?

In the preceding setup, OPERATNS was given permission to the `PASSWORD.ENVELOPE` profile. Consequently, when the OPERATNS' password is changed now, RACF should create a password envelope. Issue the following command:

```
RAC ALTUSER OPERATNS PASSWORD(NOBBY)
```

Wait a while and issue the following command:

```
RAC LISTUSER OPERATNS
```

And RACF should respond:

```
USER=OPERATNS  NAME=UNKNOWN  OWNER=IBMUSER   CREATED=08.261
DEFAULT-GROUP=SYS1     PASSDATE=00.000 PASS-INTERVAL= 30 PHRASEDATE=N/A
PASSWORD ENVELOPED=YES
ATTRIBUTES=NONE
REVOKE DATE=NONE   RESUME DATE=NONE
LAST-ACCESS=08.281/19:02:54
```

Can Tivoli Directory Integrator retrieve envelopes?

Issue the following LDAP search (sample: **IVP7 EXEC**):

```
ldapsrch -h 127.0.0.1 -D racfid=ITDI,profiletype=user,cn=RACFVM -w ITDI -L -b
            racfid=OPERATNS,profiletype=user,cn=RACFVM "objectclass=*"
            racfpasswordenvelope racfpassphraseenvelope
```

Tivoli Directory Integrator is used as the authentication user. This is required because it is the only user that we gave the required RACF permission to retrieve password envelopes. A response similar to the following is received:

```
dn: racfid=OPERATNS,profiletype=USER,cn=RACFVM
racfpasswordenvelope:: MIIGlQYJKoZIhvcNAQcDoIIGhjCCBoICAQAxggHqMIHdAgEAMEYwPj
E8MDoGA1UEAxMzVGl2b2xpIERpcmVjdG9yeSBJbnRlZ3JhdG9yIG9uIE5vZGUgSUJNLThENDBDQU
MwMUY1AgRI1/t2MAOGCSqGSIb3DQEBAQUABIGASMVC2bnmPngs9IqVDk+v3b1DEPauCdvyVdnmoH
u9w7aFyAAWRlLMiyH9aXLlFleeLOOotnSEF8Lyg/DXr4aJsORSd+jdY5WjJukpRCASKqhmH5+x8P
........
```

In the search parameters, the envelopes for both the password and the phrase are requested. Because OPERATNS has no pass phrase, that attribute will not be reported.

The **racfpasswordenvelope** and **racfpassphraseenvelope** parameters must be specified explicitly on the search command. LDAP does not return password envelopes as part of a general search without those parameters explicitly specified.

2.6 Setting up an SSL secured LDAP environment

This section explains the steps that are required to make the LDAP server on z/VM operate in an environment that is secured by Secure Sockets Layer (SSL). On z/VM, LDAP contains its own SSL support, so no extra SSL server must be set up.

Details are provided in the publication *z/VM TCP/IP Planning and Customization,* SC24-6140, section *Setting up for SSL/TLS,* in chapter *Configuring the LDAP Server.*

Another source of information is the publication *z/VM TCP/IP LDAP Administrator's Guide,* SC24-6125, Chapter *SSL Certificate/Key Management.*

The gskkyman SSL utility program is explained in the publication *z/VM RACF Security Server Security Administrator's Guide,* SC24-6142, chapter *Password and password phrase enveloping.*

2.6.1 Creating the keys and certificates

To generate certificates for an SSL-secured LDAP environment, follow these steps:

1. openvm mount /../VMBFS:VMSYSL:LDAPSRV/ /

2. Enter `gskkyman` on the CMS Ready prompt.
3. Select option 2 **Open database**.
4. Enter `IRR.PWENV.KEYRING` at the *Enter key database name* prompt.
5. Enter your password.
6. Select option 6 **Create a self-signed certificate.**
7. Select option 5 **User or server certificate with 1024-bit RSA key.**
8. Select option 1 **SHA-1**.
9. Enter a name, such as `LDAPsslVM5`, at the *Enter label* prompt.
10. Enter a name, such as `LDAPsslVM5`, at the *Common name* prompt.
11. Enter appropriate values or accept defaults for Organizational unit, Organization, City/Locality, State/Province, Country/Region, and Number of days valid.
12. Enter **option 0** at the *Enter 1 to specify subject alternate names or 0 to continue*: prompt.
13. You should receive the following message: *Certificate created.*

The next step is to export the just generated certificate from the z/VM key ring into a file.

14. Select option 1 **Manage keys and certificates** on the gskkyman main menu.
15. Select the just generated certificate from the list by entering its number.
16. Select option 6 **Export certificate to a file**.
17. Select option 2 **Base64 ASN.1 DER** as Export File Format.
18. Enter a valid BFS file name, such as `LDAPsslVM5.b64`
19. Leave the gskkyman application.
20. Export the file from the BFS into a regular CMS file by entering, for example:

    ```
    openvm getbfs LDAPsslVM5.b64 LDAPVM5 BASE64 A
    ```

    As an alternative, an FTP session directly to the BFS file space in z/VM might be attempted. Issue the following FTP subcommands:

    ```
    cd /../VMBFS:VMSYSL:LDAPSRV/
    ascii
    get LDAPsslVM5.b64
    ```

Remember to IPL CMS in the user where gskkyman was run before restarting LDAPSRV.

2.6.2 Updates to the DS CONF file

To enable SSL support in LDAP, the DS CONF file must be adapted, like shown in Figure 2-17.

```
listen ldaps://:636
sslAuth serverAuth

sslCertificate LDAPsslVM5

sslKeyRingFile /var/ldap/IRR.PWENV.KEYRING

sslKeyRingFilePW password
```

Figure 2-17 Updates to DS CONF to use SSL secured connections

Some changes might need clarification:

listen This statement was added to have LDAP listen on port 636 for SSL secured connections.

sslCertificate Here, the label of the certificate that is generated in “Creating the keys and certificates” on page 45 must be coded.

sslKeyRingFile This record must point to the same BFS mount point as set up in the SYSTEM DTCPARMS file, as shown in Figure 2-11 on page 33.

sslKeyRingFilePW This is the password that was specified when building the key ring with gskkyman in “Use gskkyman” on page 43.

Verification step

Activate the updated DS CONF file by restarting the LDAP server.

The following command can be used to verify, from CMS, if the SSL setup works (sample: **IVP8 EXEC**).

```
ldapsrch -h 127.0.0.1 -D cn=ldapAdmin,o=ibm -w topDog -p 636 -Z -K
            /../VMBFS:VMSYSL:LDAPSRV/IRR.PWENV.KEYRING -P password -b
            cn=ldapAdmin,o=ibm objectclass=* cn sn
```

We use a fully qualified keyring fileid. This way, we are independent of what is mounted at time of execution. A response similar to the following should be received:

```
ccn=ldapAdmin, o=ibm
cn=ldapAdmin
sn=LDAP Administrator
```

Final step

Now disable the LDAP unsecured port by removing the listen ldap://:389 statement from the DS CONF file.

3

z/OS setup

This chapter describes the steps that are required to configure Resource Access Control Facility (RACF) and Lightweight Directory Access Protocol (LDAP) on IBM z/OS for synchronization of user, group, connection, password, and password phrase data.

3.1 Configuration documentation

The following publications contain the information that is required to customize RACF and LDAP for the purposes of synchronization:

- The following chapters in SA22-7683-12, *z/OS Security Server RACF Security Administrator's Guide*, detail setting up RACF for LDAP event notification and password/phrase enveloping:
  - Chapter 22, *RACF and the z/OS LDAP* server section *LDAP event notification*
  - Chapter 23, *Password and password phrase enveloping*
- The following chapters in SC23-5191-02, *IBM Tivoli Directory Server Administration and Use for z/OS*, detail the steps that are required to customize and run LDAP:
  - Chapter 8, *Customizing the LDAP server configuration*
  - Chapter 9, *Running the LDAP server*
  - Chapter 7, *Preparing the backends, sysplex, SSL/TLS, and encryption*, section *Setting up for SSL/TLS* (optional)

This document does not attempt to reproduce the setup from the preceding publications in their entirety. Instead, it aims to document choices made and provide samples of commands and config files that are used in our configuration activity. The checklist tables in this chapter aim to provide a quick setup method of getting RACF and LDAP prepared for synchronization. For details of what the checklist tasks are trying to achieve, consult the preceding publications.

Table 3-1 on page 52 provides a step-by-step checklist for RACF users. This section is written from the perspective of an existing RACF user.

Table 3-2 on page 54 provides a step-by-step checklist for LDAP users. This section is written from the perspective on a new LDAP user.

The following publication might provide extra documentation for optional configuration tasks:

- SC31-8776, *z/OS Communications Server: IP Configuration Reference,* contains details of reserving port numbers in the TCP/IP profile data set for use by LDAP.

3.2 Assumptions

It is assumed that RACF and LDAP are installed on z/OS. z/OS Cryptographic Services System Secure Sockets Layer (SSL) is required for the task of setting up a secure connection between LDAP client and server. If not, RACF, LDAP, and optionally, System SSL should be installed following the instructions that are contained in the relevant program directories.

3.2.1 RACF assumptions

The following RACF assumptions can be made:

- RACF is active on z/OS.
- The RACF FACILITY class is active.
- The RACF profiles for managing digital certificates are defined and the user issuing the commands in this document has the appropriate level of access. See Chapter 19, RACF

and Digital Certificates, in the *Security Server RACF Security Administrator's Guide*, SA22-7683-12, for details. We assume that the DIGTCERT class is RACLISTed.

- In our example, the RACF subsystem address space is running under the authority of user ID STC, which has an OMVS segment with UID=0 and HOME=/ specified.
- For our scenarios, the Integrated Cryptographic Service Facility (ICSF) on z/OS is not used to store key information. Your installation can use ICSF to store key information if you choose.
- The user ID that is used to define required RACF profiles and digital certificates has the RACF SPECIAL attribute.
- IBM Tivoli Directory Integrator processes run under the authority of user ID ITDI, which has the RACF SPECIAL attribute.
- The certificate to be used by the envelope recipient (ITDI) is created by the recipient (in this case, on a non-z/OS platform) and is imported into the RACF keyring. In the examples that follow, the certificate file that is copied to z/OS is in the sequential data set *prefix*.ITDI.CERT.
- The RACF certificate (in data set *prefix*.RACF.CERT) created in the task, *Export RACF certificate for recipient use* of Table 3-1 on page 52, must be made available to the envelope recipient (ITDI) in the recipient key database.
- z/OS users who are subject to password and password phrase enveloping are connected to the RACF group, SYNCH. This group is an arbitrary group name that is chosen for the example. It can be changed to any group or userID of your choice as long it is permitted to `PASSWORD.ENVELOPE` or `PASSPHRASE.ENVELOPE` profiles (see Table 3-1 on page 52).

3.2.2 LDAP assumptions

The following LDAP assumptions can be made:

- LDAP product code is installed (in `/usr/lpp/ldap/` directory), but the server started task is not active.
- LDAP load library (DLL) SYS1.SIEALNKE is APF-authorized and added to the LNKLST.
- LDAP server started task is called *LDAPSRV*.
- Sample started task JCL (supplied by default in GLD.SGLDSAMP(DSSRV)) is copied to an appropriate procedure library with a member name of LDAPSRV. PGM=GLDSRVnn should be changed to GLDSRV64 to run in 64-bit addressing mode.
- Appropriate RACF authorization is granted to LDAPSRV. Guidelines for securing the LDAP server can be found in Chapter 6, Setting up the user ID and security for the LDAP server, of the *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191-02.
- LDBM, SDBM, and GDBM backends are not yet configured.
- LDAP runs in single-server mode.
- GDBM file-based (not IBM DB2®) backend is used.
- LDBM backend is not required for synchronization, but is used for authentication of the LDAP administrator ID and in scenarios that are provided in this document. If you want to use the sample scenarios, configure LDBM as shown in the following sections.

If any of the preceding assumptions are not true in your environment, consider them when you follow the checklist or run the supplied samples during setup. Skip any setup steps that are not relevant to your environment.

3.3 RACF configuration

RACF configuration for the purposes of user and password/password phrase synchronization is described in detail in the *z/OS Security Server RACF Security Administrator's Guide,* chapters 22 and 23, SA22-7683-15. There is a great deal of useful background information included in these chapters. You might want to follow the instructions that are contained in these chapters. Alternatively, take a shortcut and follow the RACF configuration checklist in Table 3-1, which results in the same configuration that is used in producing this document. Review 3.2.1, "RACF assumptions" on page 50 closely if you intend to follow the configuration checklist.

3.4 LDAP configuration

LDAP SDBM and GDBM backends are required for user and password/password phrase synchronization. The LDBM backend is optional, but is used by the example scenarios later in this document. Configuration of all backends and how to run the LDAP server is described in detail in IBM Tivoli Directory Server Administration and Use for z/OS, SC23-5191-02, Chapters 1 through to 9. There is a great deal of useful background information included in these chapters. You might want to follow the instructions that are contained in these chapters. Alternatively, take a shortcut and follow the LDAP configuration checklist in Table 3-2 on page 54, which results in the same configuration that is used in producing this document. Review 3.2.2, "LDAP assumptions" on page 51 closely if you intend to follow the configuration checklist.

3.5 RACF configuration checklist

The following table contains a list of RACF tasks to be performed (Task) and how to perform the task (Action).

Following this process, take the following steps:

- Define required RACF profiles for synchronization
- Create RACF certificates for password and phrase envelope encryption
- Import recipient's (ITDI) certificate
- Export RACF certificate for recipient's (ITDI) use
- Add RACF user ID for Tivoli Directory Integrator client

Commands: All commands are TSO commands unless otherwise stated.

Table 3-1 RACF configuration checklist

Task	Action
Define profiles for LDAP change notification	Issue commands: RDEFINE RACFEVNT NOTIFY.LDAP.USER RDEFINE RACFEVNT NOTIFY.LDAP.GROUP RDEFINE RACFEVNT NOTIFY.LDAP.CONNECT
Activate and RACLIST the RACFEVNT class	Issue command: SETROPTS CLASSACT(RACFEVNT) RACLIST(RACFEVNT)

Task	Action
Define profiles for password and phrase enveloping	Issue commands: RDEFINE RACFEVNT PASSWORD.ENVELOPE UACC(NONE) RDEFINE RACFEVNT PASSPHRASE.ENVELOPE UACC(NONE)
Define envelope retrieval profiles	Issue commands: RDEFINE FACILITY IRR.RADMIN.EXTRACT.PWENV UACC(NONE) RDEFINE FACILITY IRR.RADMIN.EXTRACT.PPENV UACC(NONE)
Define extract certificates authority for RACF subsystem	Issue commands: RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE) PERMIT IRR.DIGTCERT.LISTRING CL(FACILITY) ID(<racssuid>) ACC(READ) where <racssuid> is the RACF subsystem.
Generate RACF CA certificate	Issue commands: RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('RACFCA') O('IBM') C('US')) WITHLABEL('RACFCA') NOTAFTER(DATE(2020-12-31))
Define RACF keyring	Issue command: RACDCERT ID(STC) ADDRING(IRR.PWENV.KEYRING) Note: The ID(STC) is the ID under which the RACF subsystem runs.
Generate RACF address space certificate	Issue commands: RACDCERT ID(STC) GENCERT SUBJECTSDN(CN('RACF') O('IBM') C('US')) WITHLABEL('RACFCERT') SIGNWITH(CERTAUTH LABEL('RACFCA')) KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) NOTAFTER(DATE(2020-12-31))
Connect RACF certificate to keyring	Issue commands: RACDCERT ID(STC) CONNECT(LABEL('RACFCERT') RING(IRR.PWENV.KEYRING) DEFAULT USAGE(PERSONAL))
Import certificate for envelope recipient (Tivoli Directory Integrator)	Issue commands: RACDCERT ID(ITDI) ADD('*prefix*.ITDI.CERT') TRUST WITHLABEL('ITDICERT') Note: Replace prefix with the HLQ of the data set containing certificate.
Connect recipient certificate to keyring	Issue commands: RACDCERT ID(STC) CONNECT(ID(ITDI) LABEL('ITDICERT') RING(IRR.PWENV.KEYRING) USAGE(PERSONAL))
Export RACF certificate for recipient use. Used by Tivoli Directory Integrator to verify digital signature of envelope	Issue commands: RACDCERT ID(STC) EXPORT(LABEL('RACFCERT')) DSN('*prefix*.RACF.CERT') FORMAT(CERTB64) Note: Replace prefix with an HLQ of your choice.

Task	Action
Authorize envelope recipient	Issue commands: PERMIT IRR.RADMIN.EXTRACT.PWENV CLASS(FACILITY) ACCESS(READ) ID(ITDI) PERMIT IRR.RADMIN.EXTRACT.PPENV CLASS(FACILITY) ACCESS(READ) ID(ITDI)
Enable enveloping for users connected to SYNCH group	Issue commands: PERMIT PASSWORD.ENVELOPE CLASS(RACFEVNT) ID(SYNCH) ACCESS(READ) PERMIT PASSPHRASE.ENVELOPE CLASS(RACFEVNT) ID(SYNCH) ACCESS(READ)
Refresh in-storage profiles	Issue commands: SETROPTS RACLIST(FACILITY) REFRESH SETROPTS RACLIST(RACFEVNT) REFRESH SETROPTS RACLIST(DIGTCERT) REFRESH
Add RACF user ID for Tivoli Directory Integrator client	Issue commands: ADDUSER ITDI PASSWORD(xxxx) SPECIAL ALTUSER ITDI PASSWORD(ITDI) NOEXPIRED Note: This user must have the SPECIAL attribute. Change the password to suit your installation standards.

3.6 LDAP configuration checklist

Table 3-2 contains a list of LDAP tasks to be performed (Task) and how to perform the task (Action).

Following this process, take the following steps:

- Create and edit the LDAP config to define LDBM, SDBM, and GDBM backends
- Load user and IBM schemas into LDBM
- Change the LDAP administrator to be LDBM authenticated (remove clear text password from configuration file)
- Update the GDBM Access Control List so that the Tivoli Directory Integrator user ID can read changelog entries

Commands: All commands are TSO unless otherwise stated.

Table 3-2 LDAP configuration checklist

Task	Action
Copy LDAP server config file to /etc/ldap	From OMVS, issue the command: cp /usr/lpp/ldap/etc/ds.conf /etc/ldap/ds.conf

Task	Action
Edit /etc/ldap/ds.conf	From OMVS, issue the command: oedit /etc/ldap/ds.conf and change the following lines: Change adminDN to "cn=Admin' in global section Uncomment (remove # sign in column 1) #listen ldap://:pc in the global section Uncomment #database SDBM GLDBSD31/GLDBSD64 in the SDBM section Replace #suffix "sysplex=sysplex1" with suffix "cn=RACF" in the SDBM section Uncomment #database LDBM GLDBLD31/GLDBLD64 the LDBM section Replace #suffix "o=Your Company" with suffix "o=ibm" in the LDBM section Uncomment #database GDBM GLDBGD31/GLDBGD64 in the GDBM section
Start the LDAP server	From the Operator console, issue the command: S LDAPSRV
Verify LDBM	Issue the command: ldapsrch -h 127.0.0.1 -D “cn=Admin” -w secret -s base -b o=ibm “objectclass=*” Note: You should get the message ‘No such object’ from LDAP.
Verify SDBM	Issue the command: ldapsrch -h 127.0.0.1 -D racfid=<userid>,profiletype=user,cn=RACF -w <password> -b racfid=<userid>,profiletype=user,cn=RACF “objectclass=*” Note: Substitute a valid RACF user ID and password for <userid> and <password>.
Verify GDBM	Issue the command: ldapsrch -h 127.0.0.1 -D "cn=Admin" -w secret -s base -b cn=changelog “objectclass=*” Note: You should get the message ‘No such object’ from LDAP.
Load the LDBM User schema	Issue the command: ldapmdfy -h 127.0.0.1 -D "cn=Admin" -w secret -f /usr/lpp/ldap/etc/schema.user.ldif -u on Note: Do this BEFORE loading the IBM schema.
Load the LDBM IBM schema	Issue the command: ldapmdfy -h 127.0.0.1 -D "cn=Admin" -w secret -f /usr/lpp/ldap/etc/schema.IBM.ldif -u on Note: Do this AFTER loading the User schema.
Create .ldif file for LDBM suffix load	From OMVS, issue the command: oedit suffix.ldif ... and insert the following lines in the empty file: dn: o=ibm objectclass: top objectclass: organization o: ibm
Load LDBM suffix entries	Issue the command: ldapadd -h 127.0.0.1 -D "cn=Admin" -w secret -f <home>/suffix.ldif Note: Replace <home> with your home directory.

Task	Action
Create .ldif file for LDBM authenticated LDAP administrator	From OMVS, issue the command: oedit addldba.ldif and insert the following lines in the empty file: dn: cn=ldapAdmin, o=ibm objectclass: top objectclass: person objectclass: organizationalPerson cn: ldapAdmin sn: LDAP Administrator userPassword: <adminpw> Note: Replace <adminpw> with the new administrator password of your choice.
Load LDBM authenticated LDAP admin ID	Issue the command: ldapadd -h 127.0.0.1-D "cn=Admin" -w secret -f <home>/addldba.ldif Note: Replace <home> with your home directory.
Change LDAP administrator adminDN and remove password from ds.conf	From OMVS, issue the command: oedit /etc/ldap/ds.conf and change the following lines: Change adminDN "cn=Admin' to adminDN "cn=ldapAdmin,o=ibm" Remove adminPW secret line
Restart LDAP server to activate config change	From the Operator console, issue the command: P LDAPSRV S LDAPSRV
Create .ldif file for changelog ACL update	From OMVS, issue the command: oedit gdbmaclr.ldif and insert the following lines in the empty file: dn: cn=changelog changetype: modify add: aclEntry aclEntry:access-id:racfid=ITDI,profiletype=USER,cn=RACF: normal:rsc:sensitive:rsc:critical:rsc:restricted:rsc:system:rsc
Modify GDBM changelog ACL	Issue the command: ldapmdfy -h 127.0.0.1 -D "cn=ldapAdmin,o=ibm" -w <adminpw> -f <home>/gdbmaclr.ldif Note: Replace <adminpw> with the administrator password set above; replace <home> with your home directory.

After you complete the preceding configuration, specify the LDAP client distinguished names in the following format:

- "cn=ldapAdmin,o=ibm" when LDAP administrator access is required
- “racfid=ITDI,profiletype=user,cn=RACF” for Tivoli Directory Integrator client access

3.7 SSL/TLS secure connection

Creation of a Secure Sockets Layer/Transport Layer Security (SSL/TLS) secure connection requires the installation of z/OS Cryptographic Services System SSL. The secure connection provides the capability of using public key infrastructure (PKI) algorithms to establish and maintain an encrypted communications path between a client and the LDAP server. Our

sample requires that a secure connection is used for LDAP client/server communication because sensitive password data is transmitted during synchronization activity.

Creation of an SSL/TLS secure connection is described in detail in *IBM Tivoli Directory Server Administration and Use for z/OS*, SC23-5191-02, Chapter 7, section *Setting up for SSL/TLS.* There is a great deal of useful background information included in this chapter. You might want to follow the instructions that are contained in this chapter. Alternatively, take a shortcut and follow the SSL/TLS configuration checklist in Table 3-3, which results in the same configuration that is used in producing this document. Review 3.2.2, "LDAP assumptions" on page 51 closely if you intend to follow the configuration checklist.

In our configuration, we configure LDAP on z/OS for both server and client-based authentication. This step is required because the Tivoli Directory Integrator server reads configuration information from the z/OS LDAP LDBM back end, which includes certificates and passwords. Therefore, we want this connection to be as secure as possible.

Server authentication

The following checklist is used to create a server authentication environment with the LDAP certificate connected to a keyring owned by the LDAP started task ID (in our case, STC). Port 636 is used for secure communication (for example, specify LDAPSRCH -p 636 for an encrypted connection).

The LDAP certificate (in data set *prefix*.LDAP.CERT) that is created in the task, *Export LDAP server certificate for use by client* of Table 3-3, must be made available to LDAP clients (including Tivoli Directory Integrator) in a key database to use the SSL/TLS secure connection.

In these examples, we use self-signed certificates for testing.

Table 3-3 SSL and TLS configuration checklist

Task	Action
Generate a self-signed certificate for the LDAP server	Issue the commands: RACDCERT ID(STC) GENCERT SUBJECTSDN(CN('LDAP') O('IBM') C('US')) WITHLABEL('LDAPCERT') KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) NOTAFTER(DATE(2020-12-31)) SETROPTS RACLIST(DIGTCERT) REFRESH
Define RACF keyring	Issue the command: RACDCERT ID(STC) ADDRING(LDAP.SSL.KEYRING) Note: The ID(STC) is the ID under which the LDAP server runs.
Connect the LDAP certificate to keyring	Issue the commands: RACDCERT ID(STC) CONNECT(LABEL('LDAPCERT') RING(LDAP.SSL.KEYRING) USAGE(PERSONAL)) SETROPTS RACLIST(FACILITY) REFRESH
Export LDAP server certificate for use by client	Issue commands: RACDCERT ID(STC) EXPORT(LABEL('LDAPCERT')) DSN('*prefix*.LDAP.CERT') FORMAT(CERTB64)
Transfer to clients	Use FTP or some other means

Task	Action
Import to client's keystore	Use tools available on client's platform
Edit /etc/ldap/ds.conf and add secure connection information	From OMVS, issue the command: oedit /etc/ldap/ds.conf and make the following changes in the global section: Insert line listen ldaps://:636 after line listen ldap://:389 Uncomment and change sslAuth to (remove # sign in column 1) sslAuth serverClientAuth Replace #sslCertificate none with sslCertificate LDAPCERT Replace #sslKeyRingFile LDAPRING with sslKeyRingFile LDAP.SSL.KEYRING
Restart LDAP server to activate config change	From the Operator console, issue the commands: P LDAPSRV S LDAPSRV

Client Authentication

The following checklist is used to create an LDAP client authentication environment:

- Client generates a public/private key pair.
- Client generates a certificate request and sends to certificate authority.
- Certificate authority fulfills certificate request and sends signed certificate back to client along with certificates authority's signing certificate (and any intermediate certificates).
- Client imports certificate returned by certificate authority into keystore, which contains original public and private key pair.
- The LDAP server must have the public certificate of the signer (and any intermediates) in its keystore.

Tivoli Directory Integrator uses this client certificate to authenticate to LDAP. Client authentication by the LDAP server facilitates the use of certificate bind (SASL mechanism of EXTERNAL) by an LDAP client. The bind identity becomes the distinguished name in the client digital certificate.

The following example describes the process of creating a public and private key and JCE keystore for Tivoli Directory Integrator in a z/OS UNIX System Services directory. A CA request is then created and exported to a z/OS data set. Then, it is fulfilled by issuing a RACDCERT **`gencert`** command. Finally, the signed certificate and trust chain are exported to a z/OS data set. This data set is then transferred to the Tivoli Directory Integrator client and imported into the Tivoli Directory Integrator keystore.

For testing purposes, a self-signed certificate can be used as the LDAP client certificate.

Table 3-4 on page 59 shows an SSL/TLS client configuration checklist.

Table 3-4 SSL/TLS client configuration checklist

Task	Action
Generate a client certificate and keystore using keytool	In our case, we use dn = cn=ITDI,o=IBM for the distinguished name. From the OMVS shell command prompt, issue the following command: keytool -genkey -alias "ldapclnt" -dname "cn=ITDI,o=ibm" -validity 365 -keystore "/var/itdi/soldir/tdikeystore" -storepass somepassword
Generate a certificate request for the client certificate	From the OMVS shell command prompt, issue the following command: keytool -certreq -v -alias ldapclnt -keystore "/var/itdi/soldir/tdikeystore" -storepass somepassword -file ldapclnt.csr
Convert request to ebcdic	From the OMVS shell command prompt, issue the following command: iconv -f iso8859-1 -t ibm-1047 ldapclnt.csr > ldapclnt.csrconv
Transfer to z/OS data set	Issue from a TSO command line: GET '/var/itdi/soldir/ldapclnt.csrconv' '<prefix>.LDAPCLNT.CSR' text
Generate a client certificate with RACDCERT and specify request data set, sign with CERTAUTH	We are our own certificate authority (CA) in this case and sign it with the RACFCA cert. Issue from a TSO command line: RACDCERT ID(ITDI) GENCERT('prefix.LDAPCLNT.CSR') WITHLABEL('LDAPCLNT') SIGNWITH(CERTAUTH LABEL('RACFCA')) SETROPTS RACLIST(DIGTCERT) REFRESH
Export client certificate for use by client	Issue from a TSO command line: RACDCERT ID(ITDI) EXPORT(LABEL('LDAPCLNT')) DSN('*prefix*.LDAPCLNT.PKCS7B64') FORMAT(PKCS7B64)
Transfer to IBM Tivoli Directory Integrator client	Use FTP or some other means
Import client certificate	Use tools available on client platform
Connect CA cert to LDAP server keyring	Issue from a TSO command line: RACDCERT ID(STC) CONNECT(CERTAUTH LABEL('RACFCA') RING(LDAP.SSL.KEYRING) USAGE(CERTAUTH))

3.7.1 Reserve LDAP ports in TCP/IP configuration

The following statements might be added to the TCP/IP profile data set (ddname TCPPROF in the TCP/IP started task) to reserve the ports that LDAPSRV is using:

- 389 TCP LDAPSRV; LDAP Server
- 686 TCP LDAPSRV; LDAP Server secure communications

Refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776-21, for details of TCP/IP configuration.

4

Additional configuration setup

A detailed description is provided of configuration changes that must be made to support the sample synchronization scenario.

4.1 Lightweight Directory Access Protocol directory tree entries

This section details required Lightweight Directory Access Protocol (LDAP) configuration changes to the LDAP server containing the entries that define the synchronization topology.

4.1.1 Storing configuration information in LDAP (the sysRegistry)

To increase the flexibility and scalability of this solution, configuration information about the synchronization topology (including connection and authentication details) of the participating systems is stored within an LDAP directory tree. Since the LDBM is easily configured and does not require significant resources, we chose to use this LDAP back end to store our configuration information. An LDAP must be configured on each source or target system for IBM Tivoli Directory Integrator to read and propagate changes (to access GDBM and to access Resource Access Control Facility (RACF) via the Secured Database Manager (SDBM)). However, this storage of topology information is an extra requirement.

Storing connection information: The use of a z/OS or z/VM LDAP to store this connection information is not required. Although the few unique attribute types and object classes that we used were not tested with other directories, they should work in non-z/OS LDAP servers.

In our example, we stored the required information about the z/OS LDAP server, which is also part of our synchronization topology. This method is not a requirement. The configuration information can be stored on an LDAP server that is not part of the synchronization topology.

We refer to the LDAP server containing the entries that describe the synchronization topology as the *sysRegistry*.

As designed in our example, the Tivoli Directory Integrator AssemblyLine expects to find the z/OS and z/VM configuration information within an LDAP directory. This expectation means that the Tivoli Directory Integrator AssemblyLine must only be explicitly configured to use the directory that holds all of the connection and configuration information to each of the z/OS and z/VM LDAP servers. When the main Tivoli Directory Integrator AssemblyLines are started, they connect to the LDAP server designated to hold this information. And, the AssemblyLines read the configuration information that is stored there to construct Tivoli Directory Integrator's view of the synchronization topology.

4.1.2 LDBM structure

The following structure is created in the sysRegistry's LDBM:

- All nodes are created under a top-level node. In our example, this node is O=IBM.
- An organizational unit is created to group all other nodes that contain data for the synchronization process. In our example, this organizational unit is ou=sync.
- Another node is created under ou=sync, this node contains sub nodes that represent each system in the synchronization process. This node is called the ou=node.
- Nodes for each system are created under the ou=node. One node is created for each system.

- For each system node, GDBM, HOST, and SDBM node entries are created that contain the data that Tivoli Directory Integrator needs to properly instantiate its AssemblyLines and connectors for each system.
- A node is created to represent the Tivoli Directory Integrator user.
- A node is created to grant a group of users (including Tivoli Directory Integrator) LDBM administrators rights through access control list (ACL) entries to read and store encrypted password fields (secretkey attribute).

This scenario is depicted visually in Figure 4-1 on page 64, where:

- The number 1 is an organization. In our case, O=ibm.
- The number 2 is an organizational unit. In our case, ou=sync,O=ibm.
- The number 3 is an organizational unit. In our case, ou=node,ou=sync,O=ibm.
- The number 4 is domain components, one for each system (the system nodes). For example, z/OS system SC58 is dc=sc58,ou=node,ou=sync,O=IBM.
- The numbers 5 through 7 are common name attributes for each system and two LDAP back ends. Attribute entries that are associated with these common names contain the data that Tivoli Directory Integrator uses.
  - The number 5 is for the GDBM back end. Attribute entries are defined for the Tivoli Directory Integrator bind name and password.
  - The number 6 is a HOST entry. Attribute entries define whether the host is a source or target or both, and host IP addresses and port numbers for its LDAP server.
  - The number 7 is for the SDBM back end. Attribute entries contain SDBM bind information, which is the RACF database SDBM back-end common name and password information for Tivoli Directory Integrator to use during RACF style bind.
- The number 8 is a group of unique names that are used to grant access to secretkey attributes.
- The number 9 is an entry representing the Tivoli Directory Integrator user. This entry is entered as the value for a uniquemember attribute in the groupOfUniqueNames (see 8, above).

The sample Tivoli Directory Integrator configuration file contains an associated properties file that sets the host name or IP address of the LDAP server (SysRegistryHostName) that must contain the preceding LDAP entries.

In addition, the properties file also sets the search base (SysRegistrySearchBase) under which it expects to find the node objects. Therefore, you can create the domain components under any relative distinguished name (IBM RDN®) and set that RDN as the search base in the properties file. Therefore, instead of the O=IBM,ou=synch,ou=node, you can have the dc entries for each node under some RDN such as O=yourcompany,ou=racfsynchsystems.

Figure 4-1 shows the LDBM nodes.

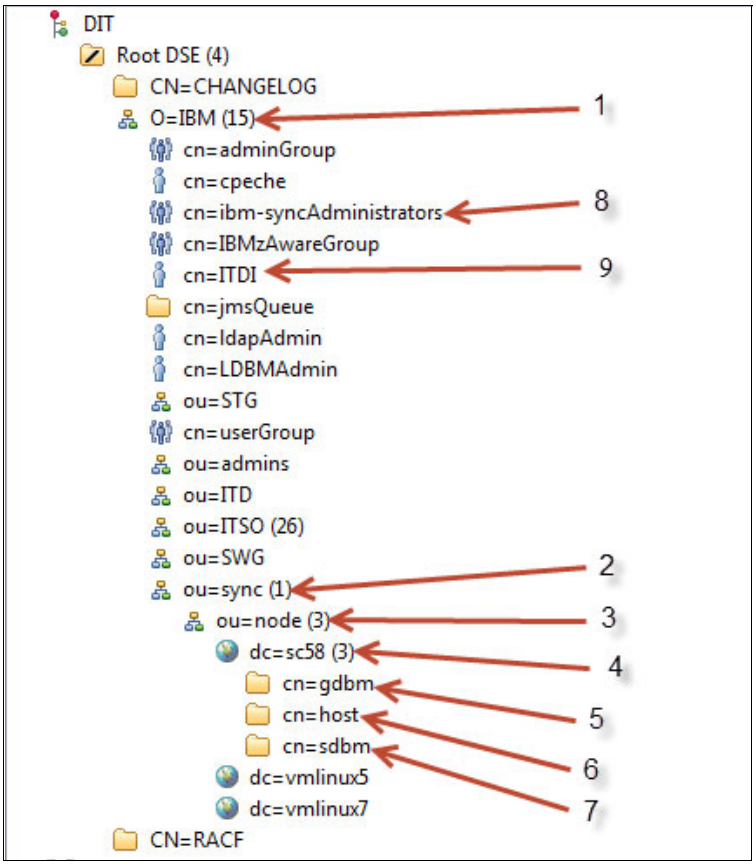


Figure 4-1 LDBM nodes

The steps that are required to create and populate the LDBM structure are now described.

4.1.3 Connection information

To meet all of our connection and processing needs, we implemented several additions to the schema, adding attribute types and objectclasses specifically for this project. Before looking at these attribute types and objectclasses, we review the connection and configuration requirements for connecting to all of our LDAP back ends.

Verify: When storing connection and configuration information in an existing LDAP, ensure to review these objectclasses and attribute types with the LDAP directory administrator to ensure that there are no name or OID conflicts.

Systems participating in the synchronization topology are identified by a common name in the LDBM tree structure. In our example, we follow the pattern dn: dc= <*system name*>,ou=node,ou=synch,o=IBM. The <*system name*> represents the participating system (z/OS or z/VM LPAR). This name is used by Tivoli Directory Integrator when constructing the name of the RACF profile to query on the source system to determine if data should be propagated to target systems.

General (Host) information

The host entry contains the following information:

- Secure Sockets Layer (SSL) URL of the LDAP server participating in the synchronization process.
- A flag to set whether this host is a source system, target system, or both.

RACF (SDBM) information

The SDBM entry contains the following information:

- Tivoli Directory Integrator Bind DN and password that is used for the SDBM style bind
- Suffix of the SDBM

Changelog (GDBM) information

The GDBM entry contains the Tivoli Directory Integrator Bind DN and password that is used for the GDBM.

ibm-synchAdministrators

The ibm-synchAdministrators entry consists of a groupofUniqueNames to allow access to secretkey attributes.

The following sections provide details on how we defined the preceding requirements to LDAP.

4.1.4 Extending the schema with new attribute types

Because we are storing data under the common name entries for each node, we extend the LDAP schema with attribute types and object classes for this data.

The following extensions, which are shown in Example 4-1, were created to store information in a way that is meaningful to the synchronization process. For example, on z/OS, place the schema updates into a UNIX System Services file and run an ldap modify command:

```
ldapmdfy -h 127.0.0.1 -D “cn=ldapAdmin” -w password' -f
/usr/lpp/ldap/etc/schema.add.ldif -u on
```

Example 4-1 Additions to schema ldif example

```
dn: cn=schema, o=ibm
changetype: modify
add:attributetypes

attributetypes: ( 1.3.18.0.2.4.5999997 NAME ( 'ibm-syncSuffix' ) DESC 'Attribute
that holds the suffix for a host.' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE
userApplications )

attributetypes: ( 1.3.18.0.2.4.5999998 NAME ( 'ibm-syncBindDN' ) DESC 'Attribute
that holds the bind DN for a host.' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 USAGE
userApplications )

attributetypes: (
  1.3.18.0.2.4.6000003
  NAME 'ibm-isSourceSystem'
  DESC 'Source for ITDI synchronization'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  USAGE userApplications
```

```
  )
attributetypes: (
  1.3.18.0.2.4.6000004
  NAME 'ibm-isTargetSystem'
  DESC 'Target for ITDI synchronziation'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
  USAGE userApplications
  )

attributetypes: (
  1.3.18.0.2.4.6000005
  NAME 'ibm-hasTheseTargets'
  DESC 'Alternate multi valued means to specify Target System.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  USAGE userApplications
  )

add: objectclasses
objectclasses: ( 1.3.18.0.2.6.59998  NAME ( 'ibmSyncNode' ) DESC 'Used to store
information needed to connect to a node. This is HOST entry' STRUCTURAL SUP ( top
) MUST ( cn ) MAY ( url $ ibm-isSourceSystem $ ibm-isTargetSystem
$ibm-hasthesetargets) )

objectclasses: ( 1.3.18.0.2.6.59999  NAME ( 'ibmSyncNodeBackend' ) DESC 'Used to
store information needed to connect to a node. This is gdbm and sdbm entry'
STRUCTURAL SUP ( top ) MUST ( cn ) MAY ( ibm-syncBindDN $ ibm-syncSuffix $
secretKey ))
```

4.1.5 LDBM entries for top-level node and subnode

First, we create the top-level node for our directory tree, as show in Example 4-2.

Example 4-2 creating the O=IBM node

```
dn: O=IBM
objectclass: organization
objectclass: top
o: ibm
```

To grant the Tivoli Directory Integrator server access to the various LDBM entries, we define a group entry, named *ibm-syncAdministrators*, that allows access to the attributes. Granting access to a group and then adding and removing IDs to this group is more efficient than defining individual user IDs to the ACL entries. This process is shown in Example 4-4 on page 67.

To access the LDAP server housing the directory tree data, the Tivoli Directory Integrator server uses either a simple bind over SSL or a Simple Authentication and Security Layer (SASL) external bind. In either case, Tivoli Directory Integrator needs an entry in the LDBM representing itself. See Example 4-3.

Example 4-3 Tivoli Directory Integrator LDBM entry

```
dn: cn=ITDI,O=IBM
objectclass: person
objectclass: organizationalPerson
objectclass: top
cn: ITDI
sn: ITDI
userpassword:: <secret>
```

We add this DN to the cn=SynchAdministrators group. In addition, the ldapAdmin ID is added to this group. See Example 4-4.

Example 4-4 ibm-syncAdministrators group

```
dn: cn=ibm-syncAdministrators,O=IBM
objectclass: groupOfUniqueNames
objectclass: top
cn: ibm-syncAdministrators
uniquemember: cn=ITDI,o=IBM
uniquemember: cn=ldapAdmin,o=IBM
```

This action allows us to restrict access to data in the LDBM to only authorized users that are members of the ibm-syncAdministrators group, particularly the secretkey attribute, which is described later.

The actual **aclentry** parameter, which grants Tivoli Directory Integrator access to the data that it requires, is defined in the ou=sync,O=IBM distinguished name entry. This entry is created with the following aclentry values, which are shown in Example 4-5. These values allow Tivoli Directory Integrator access to all the data housed in this portion of the LDBM tree.

Example 4-5 defining ou=sync node and setting aclentry values

```
dn: ou=sync,O=IBM
objectclass: organizationalUnit
objectclass: top
ou: sync
aclentry: access-id:cn=Anybody:normal:rsc:system:rsc
aclentry: access-id:cn=ibm-syncAdministrators,O=IBM:normal:rwsc:system:rwsc:
 at.secretkey:grant:rwsc
aclpropagate: TRUE
```

A node (domain component) is created for each system participating in the synchronization topology. For example, Example 4-6, shows the entry for one of the z/VM systems.

Example 4-6 vmlinux5 host entry

```
dn: dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass: domain
objectclass: top
dc: vmlinux5
```

4.1.6 LDBM entries for system and back ends

Under each system domain component, we create three common name entries: GDBM, HOST, and SDBM for each system (represented by a domain component). Each of these common name entries contains information relevant for the system and for each back end.

GDBM entry example

For example, cn=gdbm,dc=vmlinux5,ou=node,ou=sync,O=IBM is the GDBM entry for VMLINUX5. It has the attributes that are shown in Example 4-7.

- The ibm-syncbinddn is the bind name that Tivoli Directory Integrator uses when accessing the GDBM (changelog). This distinguished name must have access to the GDBM by using an aclentry. We did this step as part of the LDAP setup for both z/OS and z/VM.
- The secretkey is the password for the bind user ID.

Example 4-7 gdbm entry for VMLINUX5

```
dn: cn=gdbm,dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNodeBackend
objectclass: top
cn: gdbm
ibm-syncbinddn: racfid=ITDI,profiletype=USER,cn=RACFVM
secretkey: <ITDI userid password>
```

HOST entry example

The HOST entry for VMLINUX5 is cn=host,dc=vmlinux5,ou=node,ou=sync,O=IBM. It has the attributes that are shown in Example 4-8. The following information is required:

- The cn portion of the distinguished name for a host entry must be host.
- The ibm-issourcesystem and ibm-istargetsystem are boolean values, either `TRUE` or `FALSE`. These values determine if Tivoli Directory Integrator starts AssemblyLines to monitor changes (issourcesystem=TRUE) or to push changes (istargetsystem=true).
- The URL entries define IP addresses and port numbers for the LDAP server. One entry is defined for the SSL connection. The format must be ldaps:// + host or ip + : + port, as shown in Example 4-8.

  Example 4-8 host entry for VMLINUX5

  ```
  dn: cn=host,dc=vmlinux5,ou=node,ou=sync,O=IBM
  objectclass: ibmSyncNode
  objectclass: top
  cn: host
  ibm-issourcesystem: FALSE
  ibm-istargetsystem: TRUE
  url: ldaps://9.12.4.170:636
  ```

- The host entry can contain one other multi-valued attribute called *hasthesetargets*. If the Tivoli Directory Integrator configuration property named *SynchToAllTargets* is `FALSE`, then Tivoli Directory Integrator sends only changes originating from a source system to those target systems defined with hasthesetarget entries. If SynchtoAllTargets is `TRUE`, then Tivoli Directory Integrator sends changes to all systems that are defined as possible target systems (have ibm-istargetsystem=TRUE).

  Therefore, if SynchtoAllTargets=FALSE, for a host entry as shown in Example 4-9 on page 69, Tivoli Directory Integrator sends only changes originating from SC58 to VMLINUX5.

Example 4-9 host entry for sc58

```
dn: cn=host,dc=sc58,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNode
objectclass: top
cn: host
ibm-hasthesetargets: vmlinux5
ibm-issourcesystem: TRUE
ibm-istargetsystem: FALSE
url: ldaps://wtsc58oe.itso.ibm.com:636
```

SDBM sample entry

The SDBM entry for VMLINUX5 is cn=sdbm,dc=vmlinux5,ou=node,ou=sync,O=IBM. It has the attributes that are shown in Example 4-10. The following information is required:

- The ibm-syncsuffix is the name of the SDBM back end.
- The ibm-syncbinddn is the distinguished name that Tivoli Directory Integrator uses to bind to the SDBM.
- The secretkey is the bind user ID password.

Example 4-10 sdbm entry for VMLINUX5

```
dn: cn=sdbm,dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNodeBackend
objectclass: top
cn: sdbm
ibm-syncbinddn: racfid=ITDI,profiletype=USER,cn=RACFVM
ibm-syncsuffix: cn=RACFVM
secretkey: <ITDI userid password>
```

secretkey attribute: The secretkey attribute type is shipped in the IBM schemas. This attribute type automatically encrypts (when storing) and decrypts (when retrieving) the contents that are based on definitions in the LDAP configuration. We defined our secretkey attribute to use 3DES encryption so that we can store and retrieve passwords that are used to bind to the GDBM and SDBM back end. To control access to this attribute type, in the ou=sync,o=ibm node, we define an ACL that restricts access to the secretkey attribute to user IDs that are in the cn=ibm-syncAdministrators,o=ibm group. We can also define individual user IDs to this ACL. However, granting access to a group and then adding and removing IDs to this group is more efficient.

4.1.7 Enable ICTX

Our solution uses the ICTX plug-in to check RACF profiles on source systems as described in 4.2, “RACF setup on source systems” on page 70.

You must enable the ICTX plug-in on the LDAP server on all source systems, as shown in Example 4-11.

Example 4-11 ICTX plug-in

```
#plugin
# ICTX plugin
#
plugin clientoperation ITYBIC31/ITYBIC64 ICTX_INIT "CN=ICTX"
```

4.2 RACF setup on source systems

In addition to the RACF setup that is described in Chapters 2 and 3, there are RACF profiles that must be created for each system that is defined as a source system.

The ICTX plug-in, available through the z/OS and z/VM LDAP server, allows applications to query RACF profiles, for example to determine whether a particular user ID has access to a given RACF resource. We used this plug-in for this solution and created profiles within the FACILITY class that indicate the type of information that should be synchronized and the node to which the information should be sent.

The following syntax is developed for this solution:

PREFIX.TARGET NODE.ACTION

The PREFIX syntax is one or more qualifiers. In our examples, we use Tivoli Directory Integrator.

The TARGET NODE syntax is a reference to a particular target system. The value must match the name of a system that is defined in the topology. There must be a matching dc entry for the system in the LDAP directory tree containing the topology information: the sysRegistry.

The ACTION syntax is either PW for passwords or PP for password phrases, or USER for user profile information. Our sample solution propagates changes only to the name field of the user profile.

You define the PREFIX value to Tivoli Directory Integrator by using the FacilityClassPrefix property in the configuration.

When Tivoli Directory Integrator gets a change log record from a source system, affecting for example a user password or password phrase, it checks the profiles on the source system. Tivoli Directory Integrator checks to determine if the affected user has at least READ access to a specified profile representing a target system. If yes, the change is promoted to the target system.

The conventions are shown in Table 4-1.

Table 4-1 RACF profile naming convention

RACF profile	RACF class	Purpose
prefix.target node.PP	FACILITY	Promote password changes to system *target node*
prefix.target node.PW	FACILITY	Promote password phrase changes to system *target node*
prefix.target node.USER	FACILITY	Promote USER profile changes to system *target node*

An example best illustrates how Tivoli Directory Integrator uses these profiles. Say that we have three systems, SC58, VMLINUX5, and VMLINUX7. In the LDBM host entries, SC58 is defined as a source system, VMLINUX5 and VMLINUX7 are defined as target systems. Tivoli Directory Integrator instantiates AssemblyLines that are based on this information.

On SC58, user USER001 is connected to group SYNCH (which grants read access to RACEVNT profiles PASSWORD.ENVELOPE and PASSPHRASE.ENVELOPE). Therefore, password and password phrase envelopes are created when the user password is changed.

We define the FacilityClassPrefix property as Tivoli Directory Integrator. In addition, USER001 is given read access (either by user ID or by connection to a group with read access) to the following profiles in the FACILITY class on system SC58:

- ITDI.VMLINUX5.PW
- ITDI.VMLINUX5.PP
- ITDI.VMLINUX5.USER

When USER001's password is changed on SC58, the following steps occur:

1. Tivoli Directory Integrator finds the change log record for USER001 coming from system SC58. Consequently, the possible systems to synchronize with are VMLINUX5 and VMLINUX7.
2. Tivoli Directory Integrator checks if there is a password envelope, and continues only if there is.
3. Tivoli Directory Integrator uses the ITCX interface to send this query to RACF on SC58: has USER001 access to profile ITDI.VMLINUX5.PW?

   a. If yes, then propagate the password change to VMLINUX5.

   b. If no, then do not propagate the password change to VMLINUX5.

   In this example, USER001's password change is propagated to VMLINUX5 because the user has READ access to ITDI.VMLINUX5.PW.
4. Tivoli Directory Integrator repeats step 3 for all possible target systems.

   In this example USER001 will not have the password propagated to VMLINUX7 since the user does not have READ access to ITDI.VMLINUX7.PW.

The logic to handle a password phrase is similar, except the profile that is checked in the facility class is ITDI.<target node>.PP. Likewise, changes to USER profile information are checked against the ITDI.<target node>.USER profile.

When you decide on a prefix, you must define the required profiles on each system that is defined as a source.

4.2.1 Connect and permit users to appropriate groups and profiles

Connect the set of users who are to have their password or password phrase changes enveloped to a RACF group. In our example, we use the SYNCH group. This group must have read access to the appropriate profiles as detailed in 1.8.1, "Synchronizing RACF passwords and password phrases considerations" on page 13. This process must be done on all source systems that are part of this synchronization scheme.

In addition, permit users (or groups) to the FACILITY class profiles, described in 4.2, "RACF setup on source systems" on page 70. These profiles control whether a user's password or password phase should be synchronized to a particular target system. This process must be done on all source systems that are part of this synchronization scheme.

4.3 WebSphere MQ

It is possible that a target system becomes unavailable for a period. During that time, password or password phrase changes might have been made on a source system. To avoid losing those changes and to make this solution scalable, a WebSphere MQ persistent queue is used by Tivoli Directory Integrator.

Tivoli Directory Integrator uses a persistent WebSphere MQ queue to store changelog information. It pushes data onto this queue when it detects a changelog event that it processes and pulls data from this queue when it processes a request. For our example, we use a WebSphere MQ server on z/OS, but any WebSphere MQ server can be used. For example, you can run the WebSphere MQ application on the same server as the Tivoli Directory Integrator application.

We created a persistent queue that takes the default parameters. The following WebSphere MQ information is required to be set in the Tivoli Directory Integrator configuration's properties file:

- Server and port
- Queue manager name
- Queue name
- Channel name

4.4 Tivoli Directory Integrator

The IBM Tivoli Directory Integrator must be initially installed. The sample solution that we provide must be modified to suit your environment. This section details how to configure the Tivoli Directory Integrator server.

- Download the sample files and store on the server where Tivoli Directory Integrator is being installed. These files are used in subsequent steps. Download the following files:
  - `zRedbook.xml`
  - `racfexop.jar`
- Install Tivoli Directory Integrator. We installed on a Windows 7 workstation by using the standard ITID installer. If installing on another operating system, you must adapt the examples that follow for that platform. We recommend creating the solution directory outside of the Tivoli Directory Integrator program installation path.

  For our installation, the following paths were used:

IBM Tivoli Directory Integrator Install Directory	`C:\Program Files\IBM\TDI\V7.1.1`
Solution Directory	`C:\TDI`
Workspace	`C:\TDI\workspace`

4.4.1 Import the sample configuration file

When the installation of Tivoli Directory Integrator is complete, start the Configuration Editor (CE). Upon the first start of the Configuration Editor, the solution directory and workspace directory are populated with initial files and folders. The initial Tivoli Directory Integrator CE window is displayed, similar to Figure 4-2.

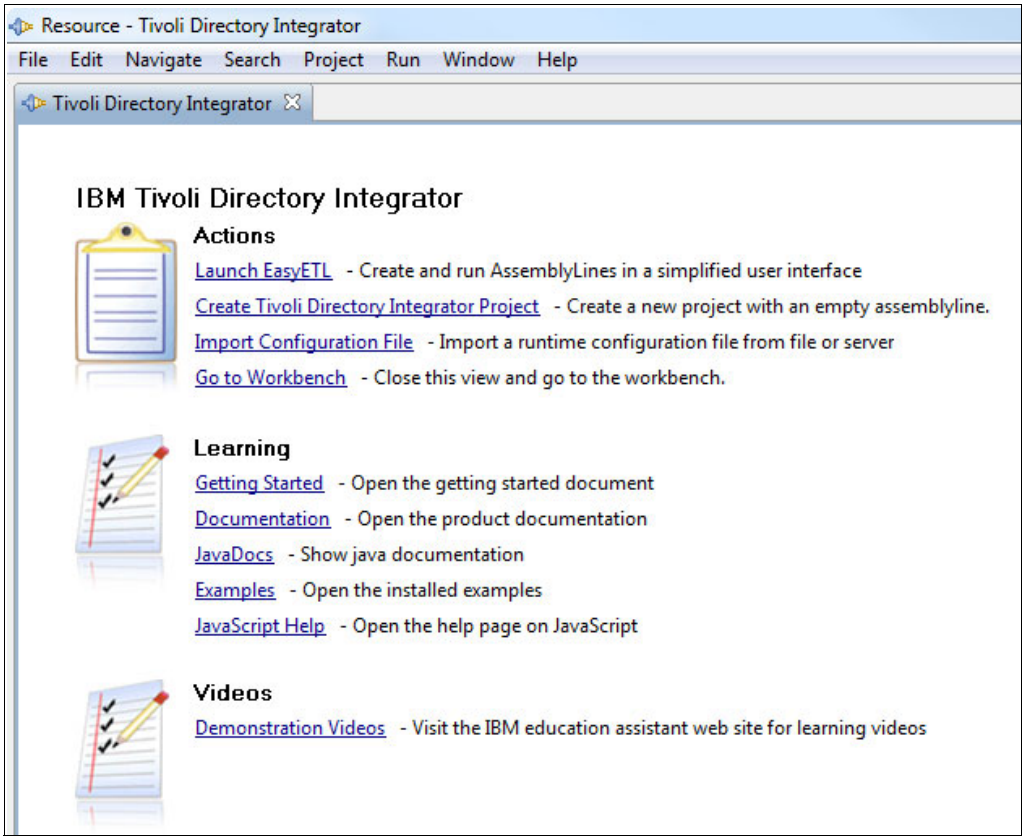


Figure 4-2 Initial CE window

You must now import the example configuration file. From the drop-down menu, select **File** → **Import**, as shown in Figure 4-3.

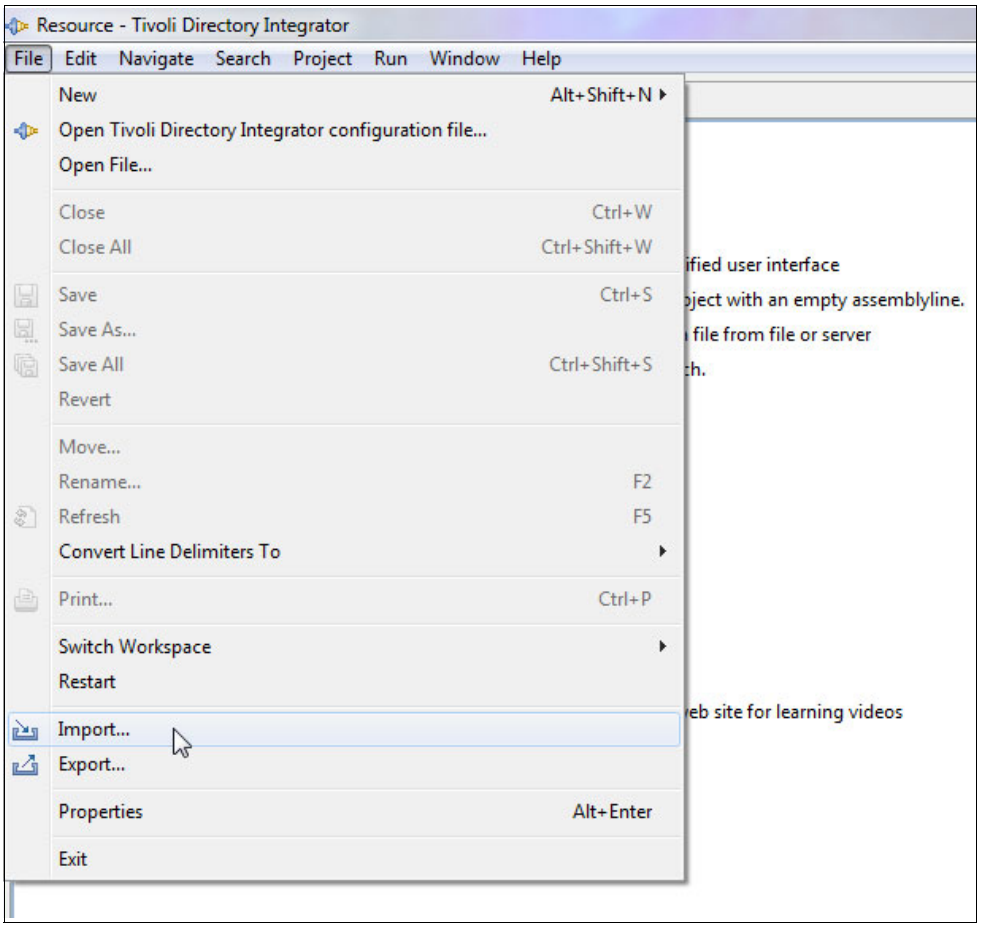


Figure 4-3 Import

The Import window opens. Expand the IBM Tivoli Directory Integrator option and select **Configuration**. Click **Next** to continue. See Figure 4-4.

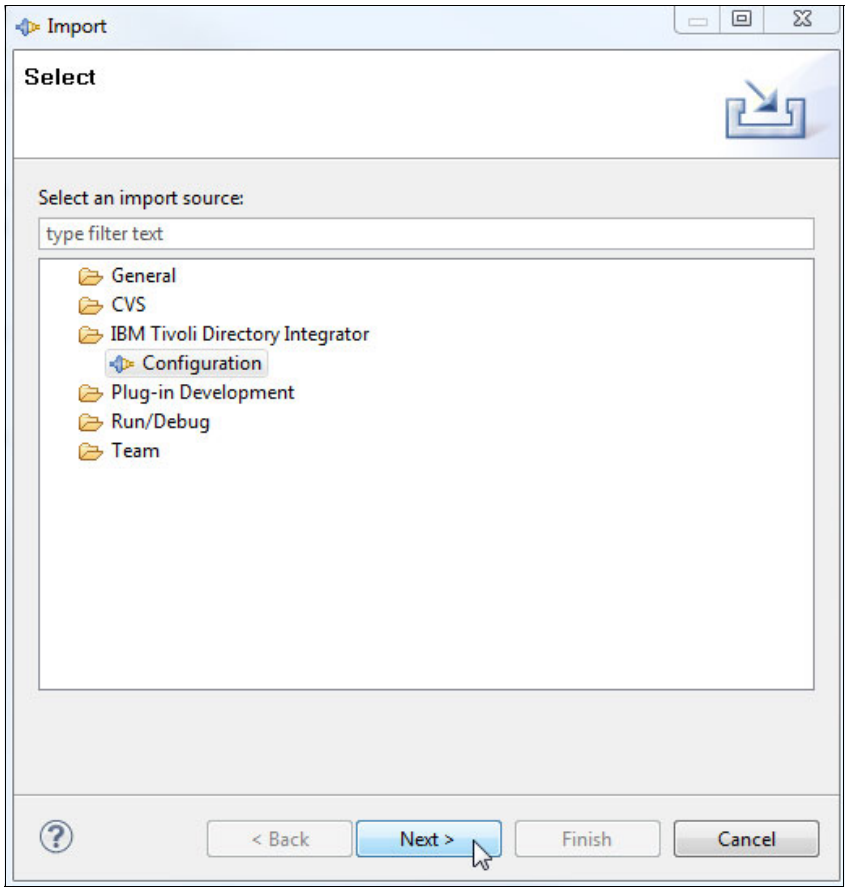


Figure 4-4 Import sources

The Select objects to import window opens. Click the button labeled**...** next to the Configuration File field to find the `zRedbook.xml` file. Leave all the selection filters checked. Click **Finish** to complete the import of the configuration file. See Figure 4-5.

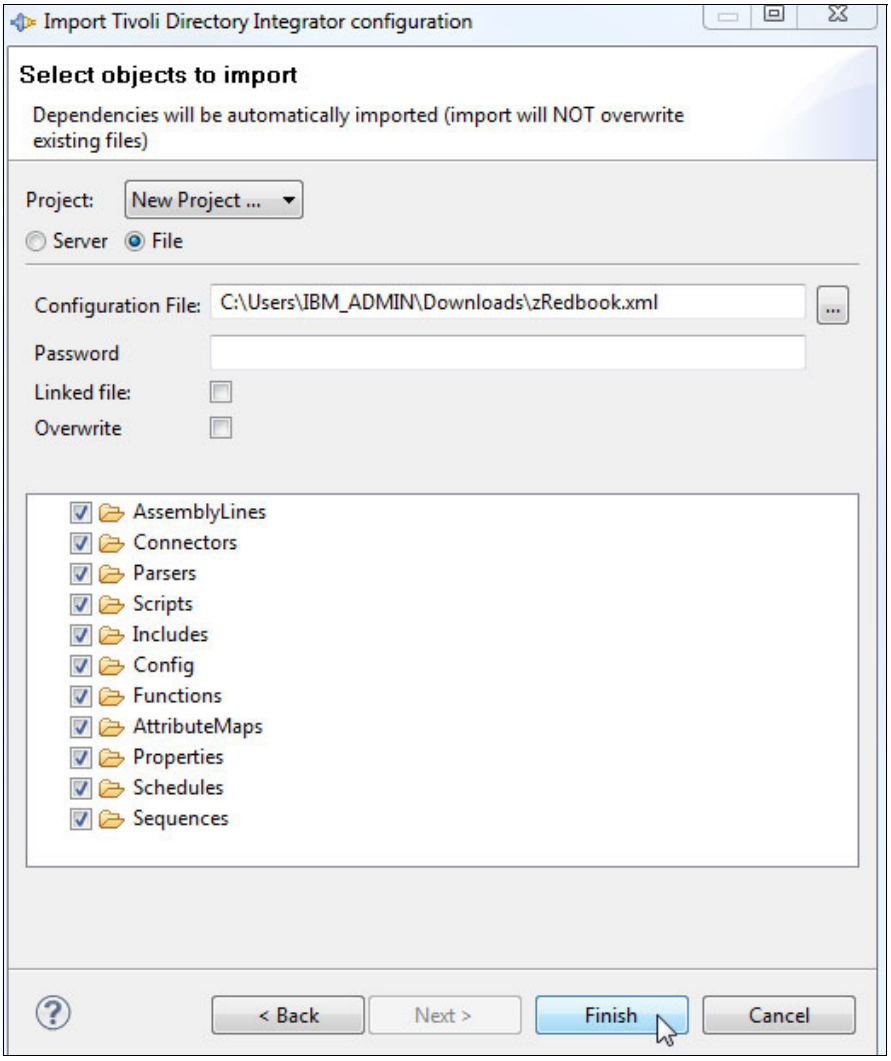


Figure 4-5 Select configuration file to import

A New Project window opens. Give the project a name. We used *zRedbook*. The location should be either the default location or change it to point to the workspace location that you specified during the Tivoli Directory Integrator installation. After you name the project, click **Finish** to complete. See Figure 4-6.

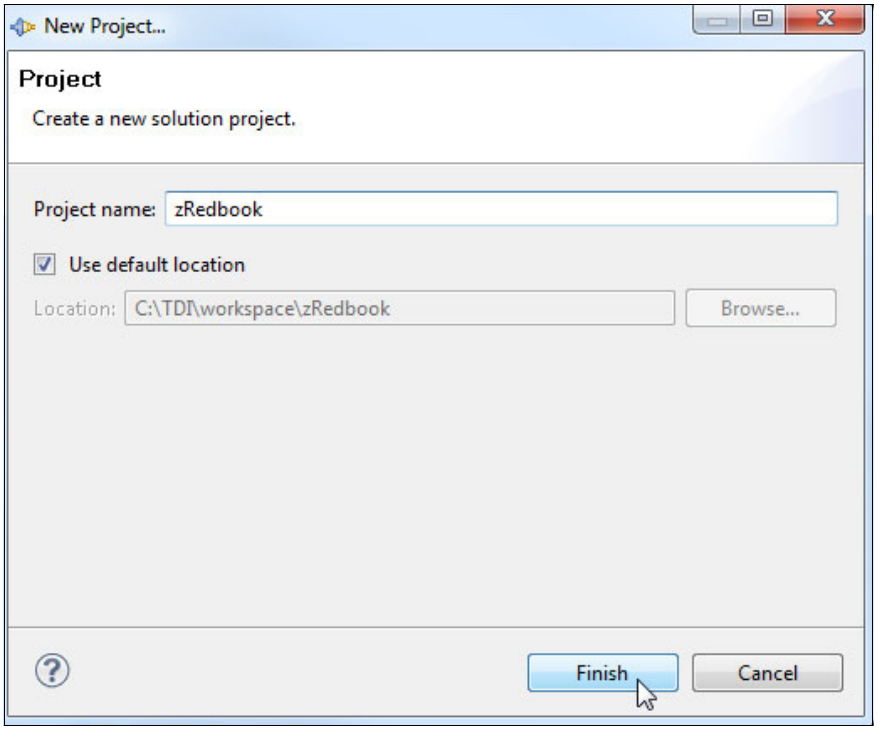


Figure 4-6 New Project

Close the Tivoli Directory Integrator window. See Figure 4-7.

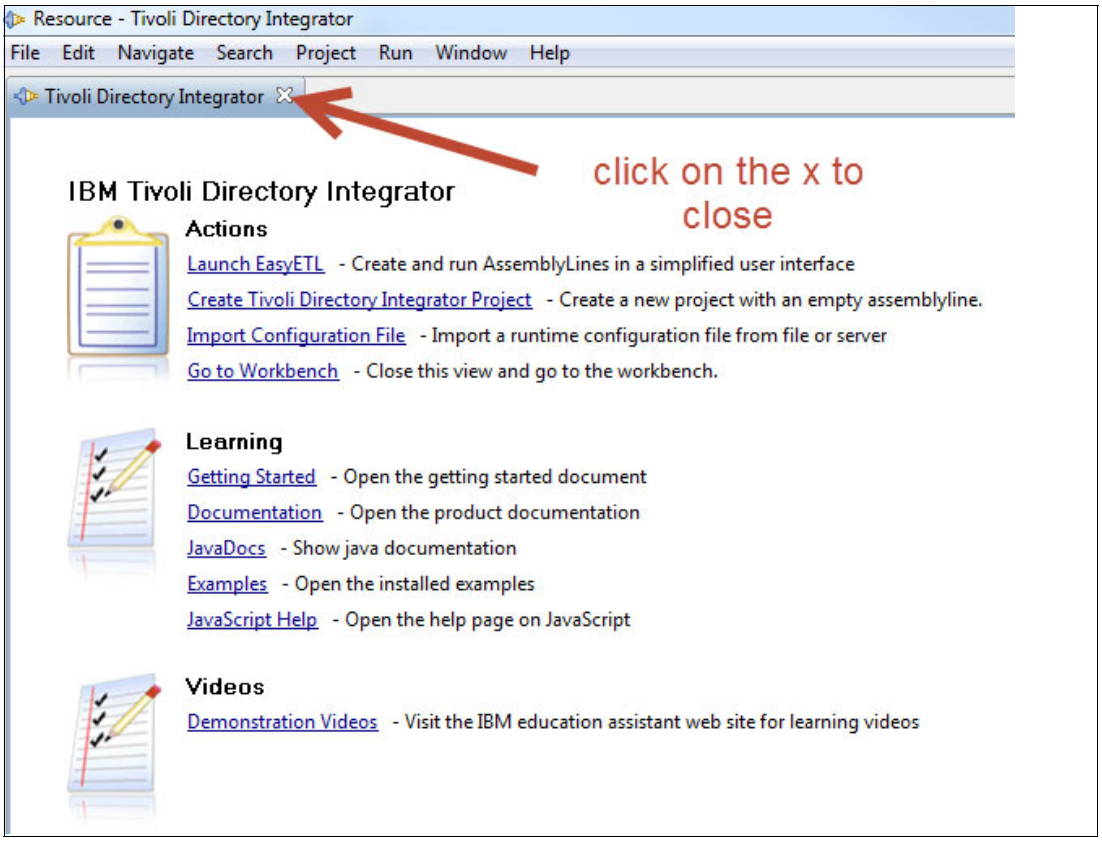


Figure 4-7 Close initial window

When the initial welcome window is closed, you are in the IBM Tivoli Directory Integrator perspective. See Figure 4-8.

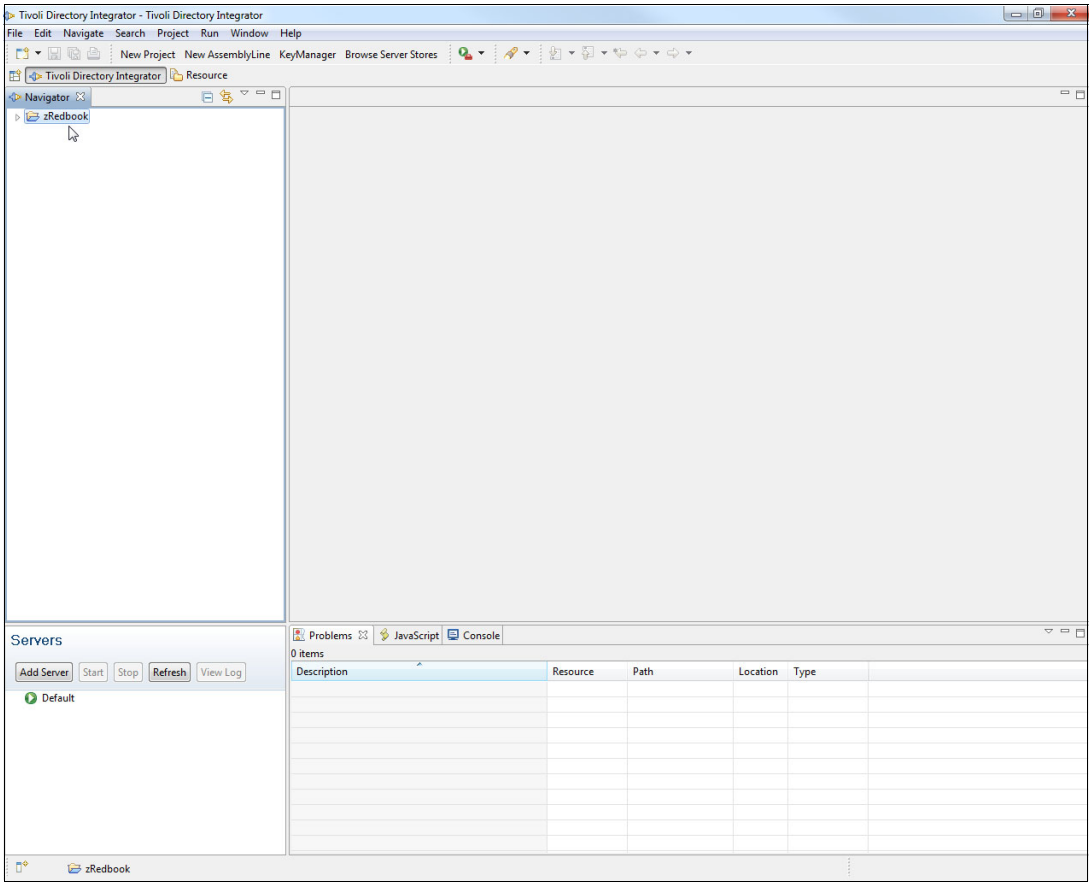


Figure 4-8 IBM Tivoli Directory Integrator perspective

4.4.2 Create extra folders

- In the solution directory, create a folder named *jars*. Place the `racfexops.jar` file in the jars folder.
- In the solution directory, create a folder name *crypto*. The following keystores are created and stored in the crypto directory:
  - `tdi_ssl_truststore`: SSL certs or CA certs for LDAP SSL connections.
  - `tdi_ssl_keystore`: SSL cert for client authentication and SASL.
  - `racf_envelope_keystore`: Contains RACF cert chain and Tivoli Directory Integrator public and private keys (recipient).

4.4.3 Create required keystores

To create the keystores, we use the KeyManager tool from the Tivoli Directory Integrator Configuration Editor. From the top menu, select KeyManager. The IBM Key Management window opens. From the menu, select **Key Database File** → **New**, as shown in Figure 4-9.

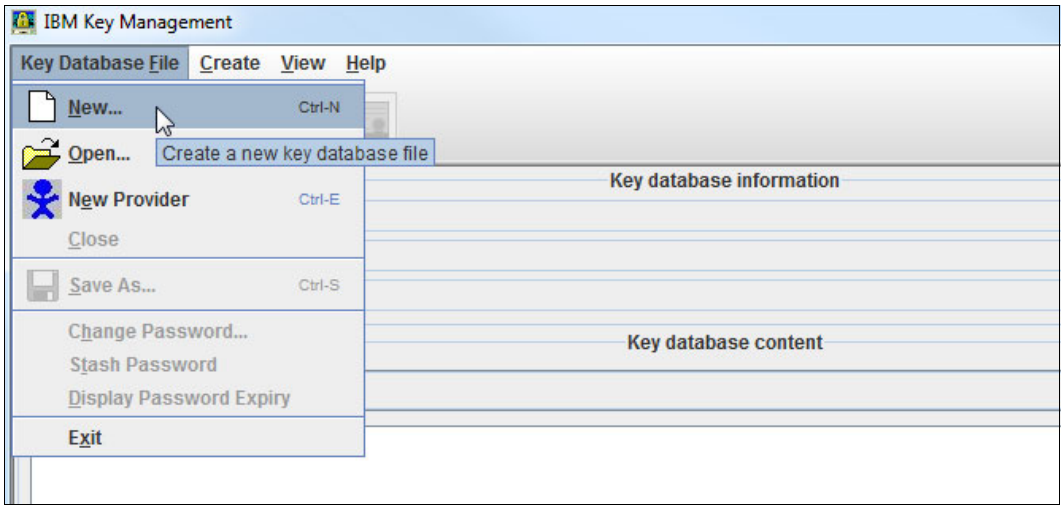


Figure 4-9 Create a new key database

The New window opens. Set the key database type as *JKS*. Set the file name as `racf_envelope_keystore.jks`. Set the location to `<solution directory>\crypto\`. See Figure 4-10.

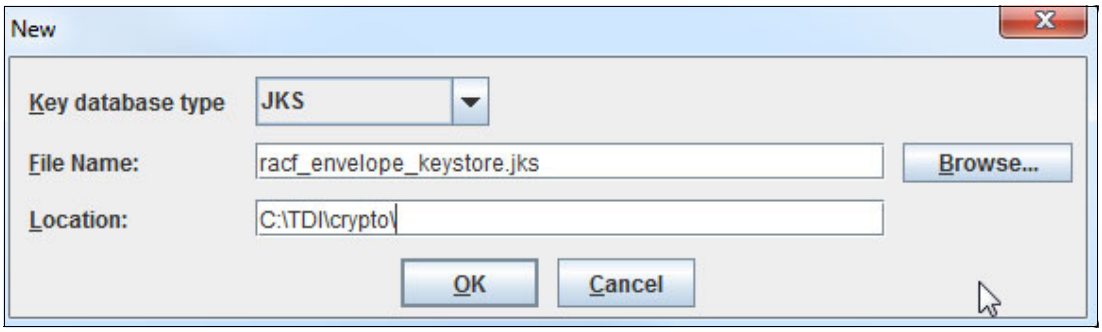


Figure 4-10 Create racf_envelope_keystore.jks

The Password Prompt window opens. Set and record the password for future use. See Figure 4-11.

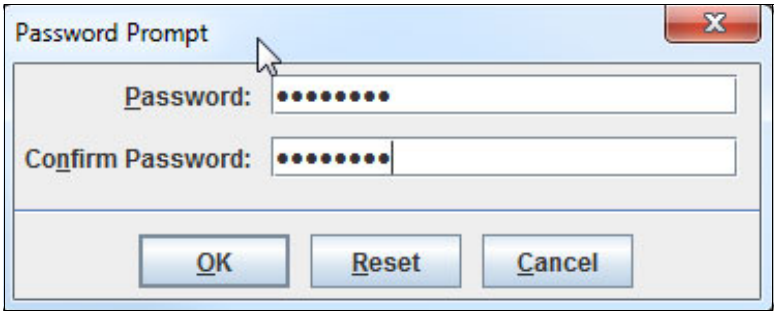


Figure 4-11 Set password for keystore

Repeat the procedure to create the `tdi_ssl_keystore` and `tdi_ssl_keystore`. You can now exit the KeyManager application. Check the <solution directory>/crypto folder to verify that the keystores were created.

Now we populate the keystores with the required certificates and key pairs.

The `racf_envelope_keystore` stores the following components:

- The RACF public certificate and any signing certificates.
- The Tivoli Directory Integrator public and private key used. The public certificate is exported and must be imported to the RACF keyring to be used as the recipient certificate by RACF.

For our example, we create a self-signed certificate for Tivoli Directory Integrator. Open the KeyManager application from the Tivoli Directory Integrator CE. Select **Key Database File** → **Open**. Browse to the crypto directory and select the *racf_envelope_keystore*. Enter the password. From the Key database content drop-down menu, select **Personal Certificates**, as shown in Figure 4-12. You can also import the public and private keypair for Tivoli Directory Integrator if you created it using another utility.

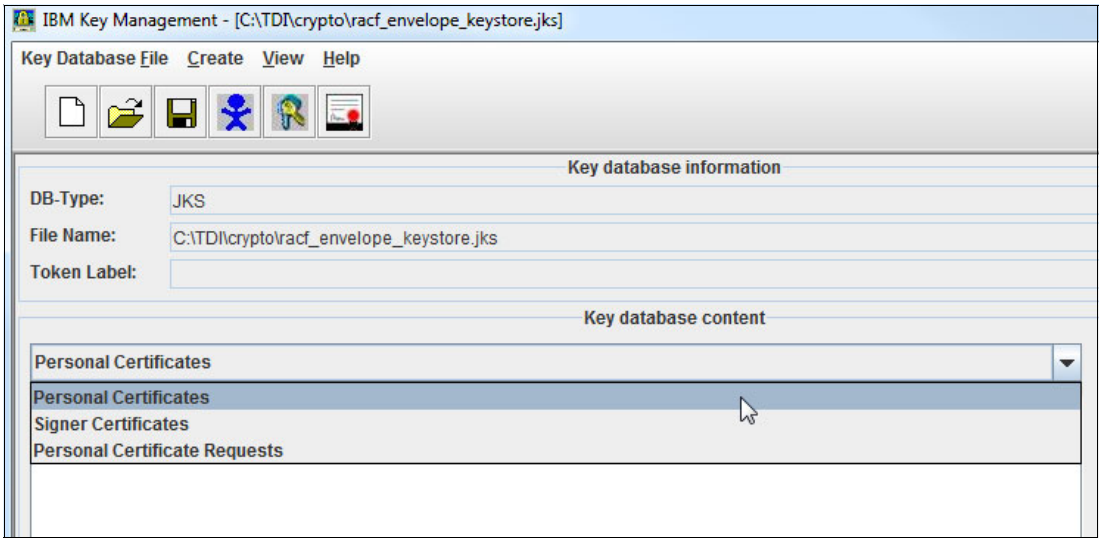


Figure 4-12 Select Personal Certificates

Click the **New Self-Signed** button to begin the process of creating a self-signed keypair. The Create New Self-Singed Certificate window opens. See Figure 4-13. Complete the values. Record the key label because this value is used later to set the TDICertAlias property. This label is used by Tivoli Directory Integrator to select the key used to decrypt the password envelope that is sent by the remote systems. Click **OK** to continue.

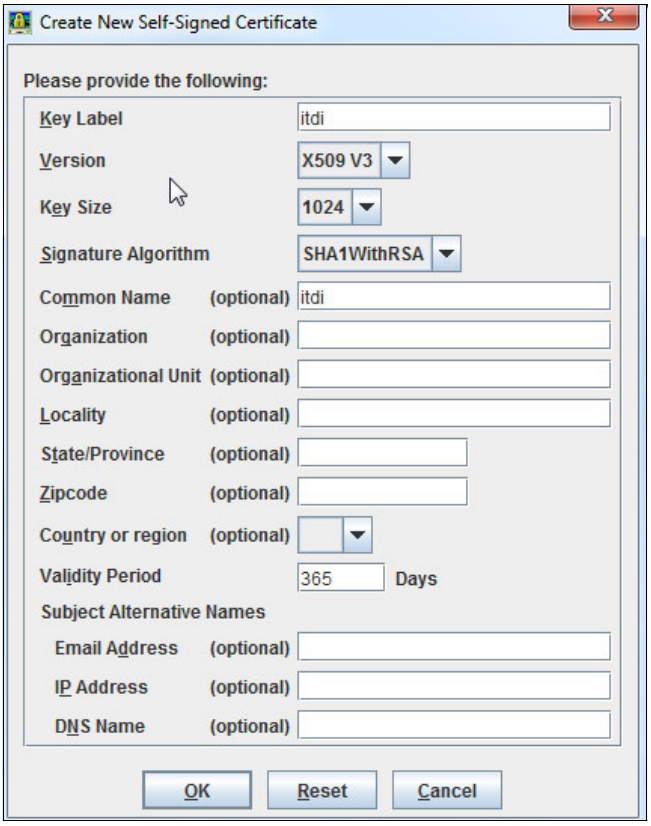


Figure 4-13 Create Tivoli Directory Integrator keypair

Extract the public certificate that will be imported into the RACF keyring for all source systems in the configuration. Open the key database containing the certificate that you want to extract. Select the certificate that you want to extract; then click on **Extract Certificate**. Either Bas64 encoded or DER binary format can be used. Name the exported file. Transfer and import this certificate to the RACF keyrings of all source systems. This is the Tivoli Directory Integrator recipient certificate. See Figure 4-14.

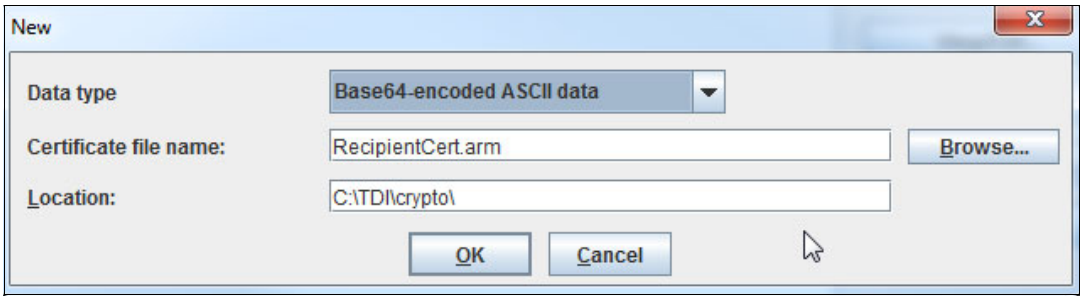


Figure 4-14 Export Tivoli Directory Integrator public cert for import into RACF keyring on source systems

The next step is to import the RACF certificate and any signing certificates. In our example, we used the same RACF public and private keys on all source systems. Therefore, we need to import only one certificate chain. Transfer the file containing the RACF certificate and any signing certificates to the Tivoli Directory Integrator system. Open the `racf_envelope_keystore`. Set the Key database content to **Signer Certificates**. Select **Add**. Browse for the file containing the RACF certificate chain. Click **OK**. The Select from Key Label List panel opens. Select all labels. Click **OK**. See Figure 4-15.

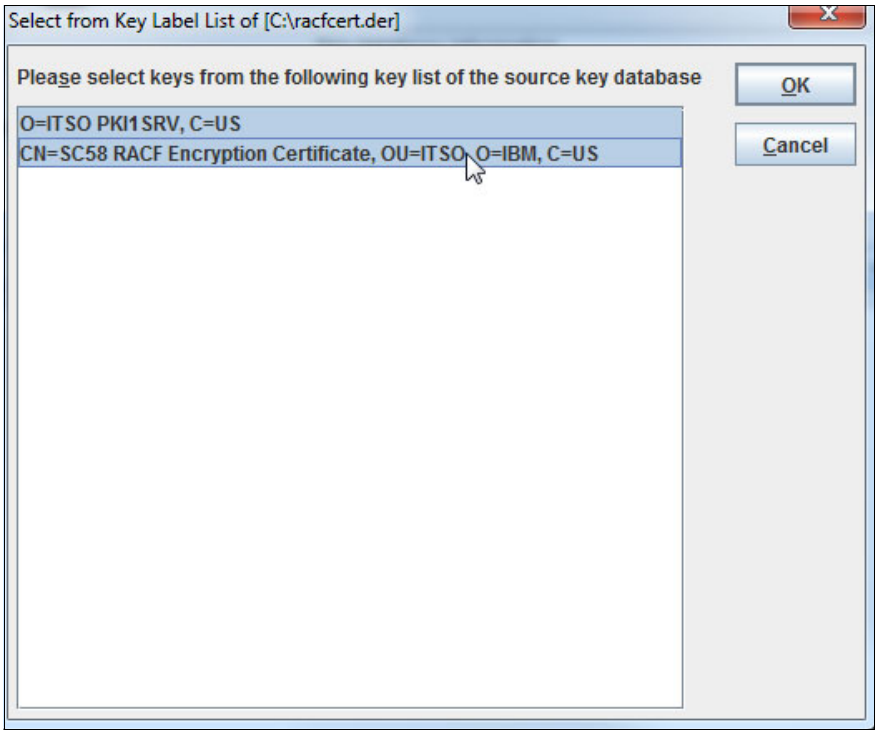


Figure 4-15 Select from Key Label List panel

You are prompted with a question asking if want to change the labels of imported certificates. The labels of the RACF public certificates are important and you have two choices for our sample configuration:

1. If you are using the same RACF certificate for all RACF systems (you created the RACF public and private keys and then exported and imported them to the other RACF systems), you can name the certificate (label) anything you want. In the properties file, set the property UseCommonRACFCert to `TRUE`, then set the property CommonRACFCertLabel to the label of the RACF certificate.
2. If you are using different RACF certificates for each RACF node, the label name must match the name of the system for which it is used. This name must match the dc= portion of the DN for the system as defined in the LDAP directory tree for the system. For example, in the directory tree we have a node entry for SC58, DN=dc=sc58,ou=node,ou=sync,o=ibm. In this case, the label of the RACF certificate for SC58 must be named SC58.

If using a common RACF certificate, make a note of the label that is used for the RACF public certificate because this label is used later to set the CommonRACFCertLabel property. In our example, we use a common certificate for all RACF systems and labeled the RACF certificate, `commonracfcert`.

The RACF certificates and any signing certs are now imported. See Figure 4-16.

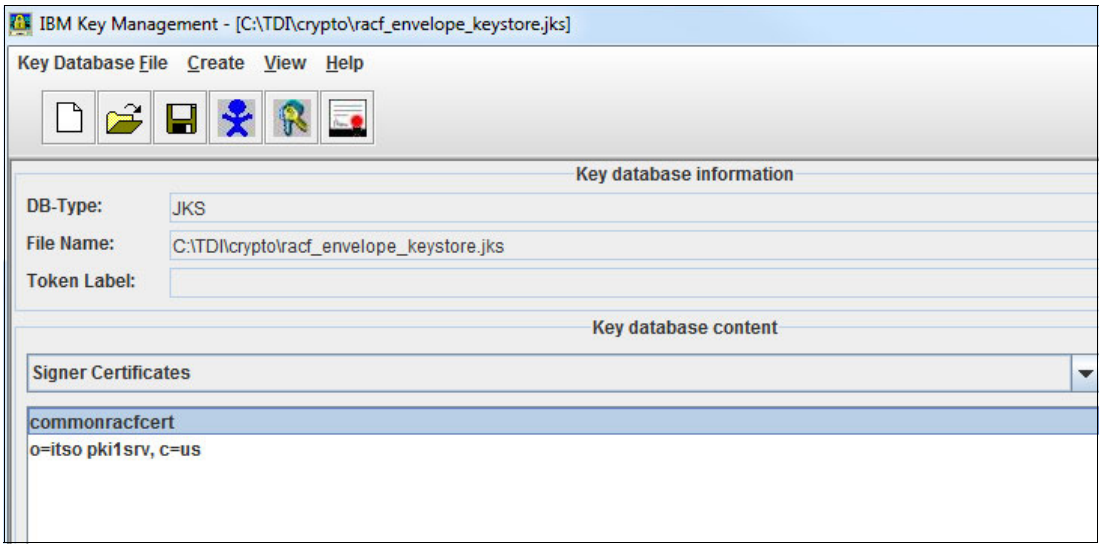


Figure 4-16 Imported RACF certificates

Repeat the procedure to import any required SSL certificates or Signing Certificates from the remote LDAP servers into the `itdi_ssl_truststore` as signer certificates.

Create an SSL certificate for Tivoli Directory Integrator in the `itdi_ssl_keystore` as a personal certificate. Export the public certificate to the LDAP servers. There can be only a single personal certificate in the `itdi_ssl_keystore`.

Important: You can use an SASL bind to the LDAP server that contains the topology entries. In our sample, the certificate that is used for SASL *must* be the same as the certificate created for the SSL connection that is used by Tivoli Directory Integrator. Therefore, ensure that the DN entered in the SSL certificate (if using for SASL) matches the ITDN DN in the LDAP registry. For example, we created an entry for ITDN in the LDAP server as DN:cn=ITDI,o=IBM. Our SSL certificate was created with DN:cn=ITDI,o=IBM. Therefore, we can use this certificate for SASL binds.

The use of an SASL bind is enabled by setting the configuration properties: sysRegistryUseSASL to `TRUE` and setting sysRegistrySASLdn to the DN of the certificate (which must match the LDBM DN entry for ITDI).

The previous example gives a brief overview of the certificate setup by using the IBM Key Manager tool. It is likely that you might have more robust certificate policies in place and need to modify the previous steps to implement your own needs.

4.4.4 Modify solution.properties file

- We must modify some settings in the `solution.properties` file. Close the Tivoli Directory Integrator Configuration Editor if you have it open. The `solution.properties` file is found in the <solution directory> path.

Open the file in a text editor and make the following changes:

– Find the `com.ibm.di.loader.userjars` property. Uncomment and change it to point to the `<solution directory>/jars` directory that you created earlier. See Example 4-12.

Example 4-12 userjars property

```
com.ibm.di.loader.userjars=c:\TDI\jars
```

– Find the dashboard.on property and set it to `false`. See Example 4-13.

Example 4-13 dashboard.on property

```
dashboard.on=false
```

– Find the systemqueue.on property and set it to `false`. See Example 4-14.

Example 4-14 systemqueue.on property

```
systemqueue.on=false
```

– Find the `javax.net.ssl.trustStore` property. Replace the property with that path to the `itdi_ssl_truststore.jks` keystore that was created earlier. Enter the password for the keystore by using the `{protect}-javax.net.ssl.trustStorePassword` property. The {protect} setting causes Tivoli Directory Integrator to encrypt the password when Tivoli Directory Integrator or the CE is next started. This process is done by using the solution directory that is associated with the `solution.properties` file being edited. In Example 4-15, the path is relative to the <solution directory>.

Example 4-15 javax.net.ssl.trustStore property

```
javax.net.ssl.trustStore=crypto/tdi_ssl_truststore.jks
{protect}-javax.net.ssl.trustStorePassword=truststorePassword
javax.net.ssl.trustStoreType=jks
```

– Find the `javax.net.ssl.keyStore` property. Replace the property with that path to the `itdi_ssl_keystore.jks` keystore that was created earlier. Enter the password for the keystore by using the {protect}-javax.net.ssl.keyStorePassword property. The {protect} setting causes Tivoli Directory Integrator to encrypt the password when Tivoli Directory Integrator or the CE is next started. This process is done by using the solution directory that is associated with the `solution.properties` file that is being edited. In Example 4-15, the path is relative to the <solution directory>.

Example 4-16 javax.net.ssl.keyStore property

```
javax.net.ssl.keyStore=crypto/tdi_ssl_keystore.jks
{protect}-javax.net.ssl.keyStorePassword=keystorePassword
javax.net.ssl.keyStoreType=jks
```

4.4.5 Create configuration properties store

We must create a property store to contain required properties and values for the configuration.

Start the Tivoli Directory Integrator Configuration Editor. Open the zRedbook project if it is not opened by default. Expand the Projects folder; expand the Resources folder; expand the Properties folder. Right click the Properties folder and select **New Property Store**. See Figure 4-17.

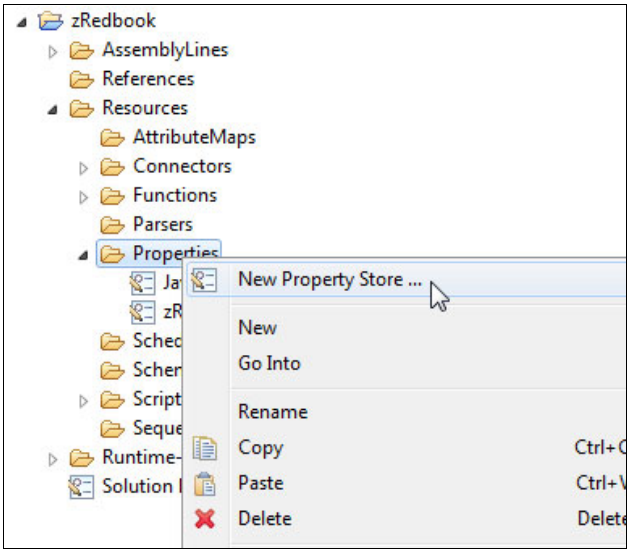


Figure 4-17 New Property Store

Name the new property store *RACF Redbook Properties*. See Figure 4-18.

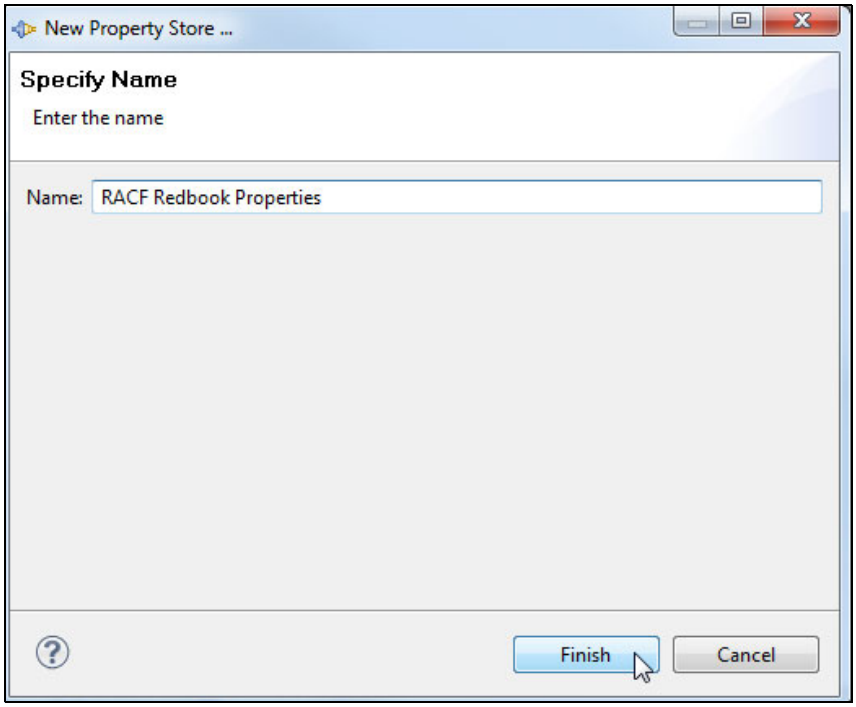


Figure 4-18 Specify property store name

Click **Add property** (shown in Figure 4-19 on page 90). Add the following properties that are shown in Table 4-2, and set their values.

Table 4-2 Table of properties

Property	Value	Protected
CryptoFolder	The name of folder under the solution directory containing the keystores. In our example, crypto.	False
EnvelopeKeyStore	The name of the keystore containing the RACF certificate chains and the IBM Tivoli Directory Integrator public and private keys. For example, racf_envelope_keystore.jks.	False
EnvelopeKSPassword	Password for the EnvelopeKeyStore property.	True
UseCommonRACFCert	`True`, if you are using one RACF certificate for all systems. False, otherwise (IBM Tivoli Directory Integrator then expects to find a label for the RACF certificate of the source system that is equal to the source system's name).	False
CommonRACFCertLabel	If UseCommonRACFCert is set to `TRUE`, this must be the label of the single RACF public certificate in the EnvelopeKeyStore. For example, commonracfcert.	False
TDICertAlias	Label of the ITDI personal certificate in EnvelopeKeyStore used to decrypt the password and password phrase envelope. For example, itdi.	False
TDICertPassword	Password for the EnvelopeKeyStore property.	True
FacilityClassPrefix	Prefix that is used for FACILITY class profiles to verify is a change is to be propagated, without any trailing periods. The suffix of these profiles is fixed as <target node>.<change type>. Therefore, for example, if you have one source system named SC58 and one target system named VMLINUX5 and you set a prefix of TDI.SYNCH, then the following profiles must be defined: TDI.SYNCH.VMLINUX5.PW TDI.SYNCH.VMLINUX5.PP TDI.SYNCH.VMLINUX5.USER In our examples, we set the prefix as ITDI.	False

Property	Value	Protected
SynchToAllTargets	`TRUE`, if you want all systems that are defined as target systems to receive changes from all systems that are defined as source systems. If the host entry for a system has istargetsystem=true, an AssemblyLine is started to push changes to that system. If a host entry has issourcesystem=true, an AssemblyLine is started to read changelogs and put changes to a WebSphere MQ queue with a header message indicating the system that the change is destined for. By setting SynchToAllTargets = TRUE, then a source system's AssemblyLine puts a change record to the WebSphere MQ queue for each system that is defined as istargetsystem = TRUE. `FALSE`, if you want to specify specific target systems for each source system. If FALSE, then the host entry in the LDBM of each source must have one or more hasthesetarget attributes with the value set as the name of the target system. The source system's AssemblyLine puts a change message to WebSphere MQ for each target that is defined with hasthesetargets.	False
MQjmsBroker	Host name and port for WebSphere MQ server. For example, wtsc58.itso.ibm.com:1610.	False
MQjmsServerChannel	Name of WebSphere MQ Channel for connection. For example, ITDI.	False
MQjmsQueueManager	WebSphere MQ Queue manager name. For example, MQ2B.	False
MQjmsQueue	WebSphere MQ Queue name. For example, ITDI.	False

The following properties define the connection to the LDAP server that contains the topology for the synchronization solution, the entries that define the systems. We refer to this LDAP server and relevant portion of the directory tree as the *sysRegistry*.

You have two choices for the bind to the sysRegistry:

1. Use a simple bind and SSL connection. This configuration requires the following steps:
   a. An entry for IBM Tivoli Directory Integrator in the sysRegistry's LDBM. The IBM Tivoli Directory Integrator DN must be part of the ibm-synchAdministrators group.
   b. Set the following properties:
      i. SysRegistryBindDN
      ii. SysRegistryBindPW
2. Use SASL. This configuration requires the following steps:
   a. Remote LDAP server must support SASL (EXTERNAL) binds.
   b. The certificate that is used for SASL *must* be the same as the certificate created for the SSL connection that is used by Tivoli Directory Integrator and be stored in the `tdi_ssl_keystore` (personal certificate). FALSE to use simple bind.

c. An entry for Tivoli Directory Integrator in the sysRegistry's LDBM. The Tivoli Directory Integrator DN must be part of the ibm-synchAdministrators group. The DN of the SASL certificate must match the DN of the LDBM entry for Tivoli Directory Integrator.

d. Set the following properties:

   i. SysRegistryUseSASL
   ii. SysRegistryUseSASLdn

For both simple bind and SASL bind to the sysRegistry, the following common properties must be set:

- SysRegistryHostName
- SysRegistrySSLport
- SysRegistrySearchBase

Table 4-3 shows the SysRegistry related properties.

Table 4-3 SysRegistry related properties

Property	Value	Protected
SysRegistryHostName	The host name or IP address of the sysRegistry. For example, wtsc58.itso.ibm.com.	False
SysRegistrySSLport	The SSL port of the sysRegistry. For example, 636.	False
SysRegistrySearchBase	The search base (portion of the directory tree) under which the system entries (dc=<system name>) are defined in the LDBM. For example, ou=node,ou=sync,o=ibm	False
SysRegistryBindDN	The bind DN which IBM Tivoli Directory Integrator uses for a simple bind to the sysRegistry. This DN must match an entry in the sysRegistry's LDBM.	False
SysRegistryBindPW	The bind password for the SysRegistryBindDN.	True
SysRegistryUseSASL	`TRUE`, to use a SASL(EXTERNAL) bind. The certificate that is used for SASL *must* be the same as the certificate created for the SSL connection that is used by IBM Tivoli Directory Integrator and be stored in the `tdi_ssl_keystore` (personal certificate). `FALSE`, to use simple bind.	False
SysRegistryUseSASLdn	If SysRegistryUseSASL is set to `TRUE`, this property must be set to the DN of the certificate that is used for SASL. In this solution, we use a single certificate for SSL and SASL. Therefore, this is the ITDI personal certificate in the `tdi_ssl_keystore`. In our example, cn=itdi,o=ibm.	False

As an example of how to create properties by using the CE, we create two properties: the EnvelopeKeyStore and the EnvelopeKSPassword. Click **Add property**, as shown in Figure 4-19.

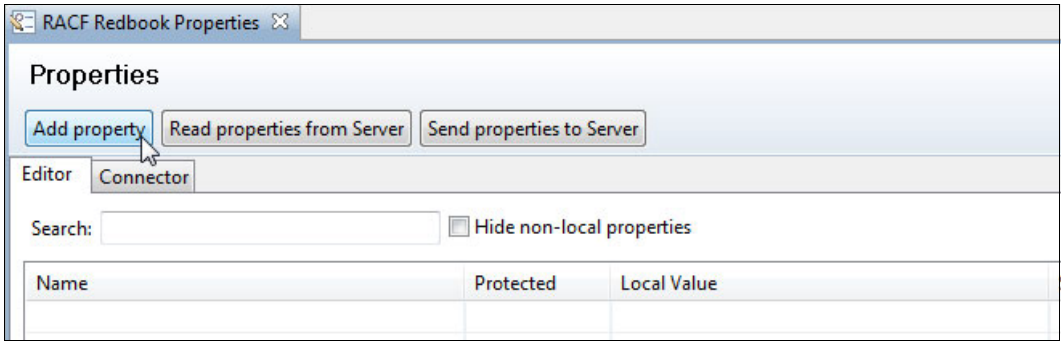


Figure 4-19 Add property

Name the new property and click **OK**, as shown in Figure 4-20.

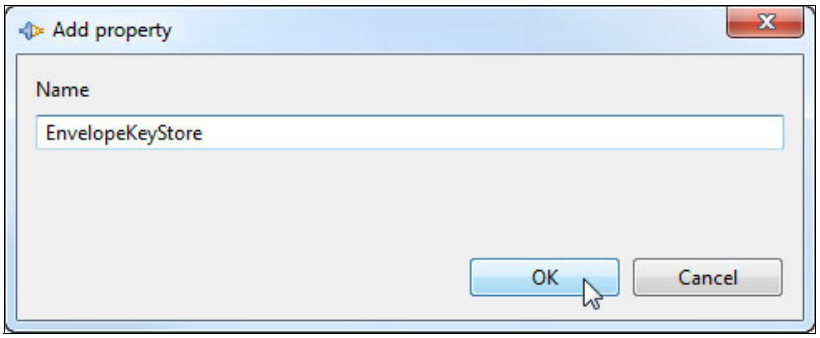


Figure 4-20 Naming a property

The new property is now displayed in the Properties window. Click the field under the Local Value column in the row of the property you are changing. Enter the value of the property. In this example, we enter the name of the keystore. Press Enter when the data entry is completed. See Figure 4-21 for how to enter a value.

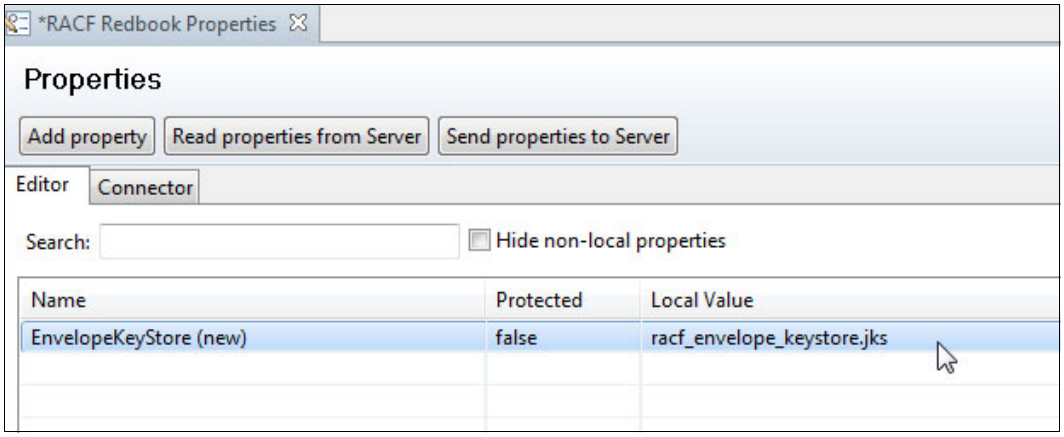


Figure 4-21 Entering a value

Repeat the same process to add other properties and values. The Protected field triggers Tivoli Directory Integrator to encrypt the value if the property store is written to clear text.

To enable a Protected field, toggle the Protected value in the row of the value you want to protect. Clicking the Protected field enables a field menu with two options (false and true). Select **true** to enable protection for the field. For example, we define the EnvelopeKSPassword property and set the field to `true`, as shown in Figure 4-22.

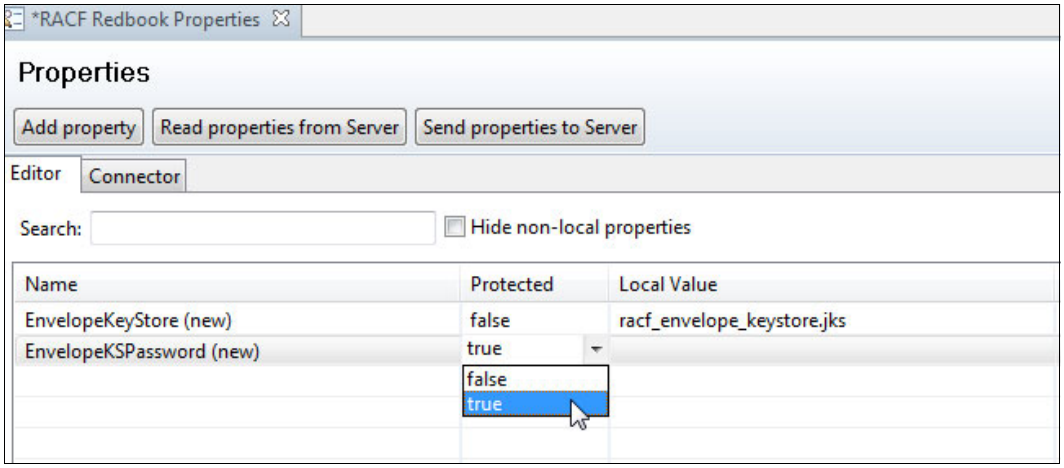


Figure 4-22 Setting Protected value

You enter the password or value for a protected value in the same way as an unprotected value. Click in the Local Value field and enter the value.

Follow the sample procedure to create all the other required properties and enter the required values. When all the properties and values are entered, select **File** → **Save All** from the drop-down menu to save the changes.

Example 4-17 is a sample of the exported properties file that we used.

Example 4-17 Sample Properties

```
CryptoFolder=crypto
{protect}-EnvelopeKSPassword={encr}RMGzjTMsfegbUKggFC1mDnnO8ZFhzAUG8i2Enf5TEh+d
EnvelopeKeyStore=racf_envelope_keystore.jks
SysRegistrySearchBase=ou=node,ou=sync,o=ibm
TDICertAlias=itdi
{protect}-TDICertPassword={encr}A+i6PF8j7B4ElgyM+XFYI+Aqkcej5m9OkRreH51r7C8mRo4
MQjmsBroker=wtsc58.itso.ibm.com:1610
MQjmsQueue=ITDI
MQjmsQueueManager=MQ2B
MQjmsServerChannel=ITDISERV
SysRegistryHostName=wtsc58.itso.ibm.com
SysRegistryBindDN=cn=itdi,o=ibm
SysRegistryUseSASL=true
SysRegistrySSLport=636
{protect}-SysRegistryBindPW={encr}Yl7zTifUvFrzEpxMEnBhrlPEd2kRr99S9LOzjz24Ob+cT
SysRegistryUseSASLdn=cn=itdi,o=ibm
FacilityClassPrefix=ITDI
SynchToAllTargets=TRUE
UseCommonRACFCert=TRUE
CommonRACFCertLabel=commonRACFCert
```

4.5 Summary of LDBM entries and example ldif file

The following tables summarize the entries that are required in the LDBM.

Table 4-4 shows the entries for the top-level organization.

Table 4-4 Top-level entry

Attributes	Our example
dn	O=IBM
objectclass	organization
objectclass	top
o	ibm

Table 4-5 shows the entry for the ibm-synchAdministrators group. This entry should have the LDAP administrator and IBM Tivoli Directory Integrator user entries as uniquemembers.

Table 4-5 ibm-synchAdministrators group

Attribute	Our example
dn	cn=ibm-syncAdministrators,O=IBM
objectclass	groupOfUniqueNames
objectclass	top
cn	ibm-syncAdministrators
uniquemember	cn=itdi,o=ibm
uniquemember	cn=ldapAdmin,o=IBM

Table 4-6 shows the entries for the organizational unit ou=sync. Note the aclentry, which allows Tivoli Directory Integrator (through the ibm-synchAdministrators entry) access to this entry and all child entries.

Table 4-6 Organizational unit entry for ou=sync

Attribute	Our example
dn	ou=sync,O=IBM
objectclass	organizationalUnit
objectclass	top
ou	sync
aclentry	access-id:cn=Anybody:normal:rsc:system:rsc
aclentry	access-id:cn=ibm-syncAdministrators,O=IBM:normal:rwsc:system:rwsc: at.secretkey:grant:rwsc
aclpropagate	TRUE

Table 4-7 shows the entries for organizational unit ou=node.

Table 4-7 Organizational unit entry for ou=node

Attribute	Our example
dn	ou=node,ou=sync,O=IBM
objectclass	organizationalUnit
objectclass	top
ou	node

Table 4-8 shows the entries for the domain component. In this example, we show the entry for VMLINUX5.

Table 4-8 Domain component entry (for VMLINUX5)

Attribute	Our example
dn	dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass	domain
objectclass	top
dc	vmlinux5

Table 4-9 shows the entries for common name cn=gdbm. In this example, VMLINUX5 is used as the domain component.

Table 4-9 GDBM entry (for VMLINUX5)

Attribute	Our example
dn	cn=gdbm,dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass	ibmSyncNodeBackend
objectclass	top
cn	gdbm
ibm-syncbinddn	racfid=ITDI,profiletype=USER,cn=RACFVM
secretkey	<ITDI userid password>

Table 4-10 shows the entries for common name cn=host. In this example, VMLINUX5 is used as the domain component.

Table 4-10 Host entry (for VMLINUX5)

Attribute	Our example
dn	cn=host,dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass	ibmSyncNode
objectclass	top
cn	host
ibm-issourcesystem	FALSE

Attribute	Our example
ibm-istargetsystem	TRUE
url	ldaps://9.12.4.129:636

Table 4-11 shows another example of a common name cn=host entry for SC58.

Table 4-11 Host entry (for SC58)

Attribute	Our example
dn	cn=host,dc=sc58,ou=node,ou=sync,O=IBM
objectclass	ibmSyncNode
objectclass	top
cn	host
ibm-issourcesystem	TRUE
ibm-istargetsystem	FALSE
ibm-hasthesetargets	vmlinux5
ibm-hasthesetargets	vmlinux7
url	ldaps://wtsc58oe.itso.ibm.com:636

Table 4-12 shows the entries for common name cn=sdbm. In this example, VMLINUX5 is used as the domain component.

Table 4-12 SDBM entry (for VMLINUX5)

Attribute	Our example
dn	cn=sdbm,dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass	ibmSyncNodeBackend
objectclass	top
cn	sdbm
ibm-syncbinddn	racfid=ITDI,profiletype=USER,cn=RACFVM
ibm-syncsuffix	cn=RACFVM
secretkey	<ITDI password>

4.5.1 Sample LDAP Data Interchange Format file

A sample LDAP Data Interchange Format (ldif) file, which we used to create all the required entries, is shown in Example 4-18. The schema must be extended, as shown in 4.1.4, “Extending the schema with new attribute types” on page 65 before you define most of these entries.

Example 4-18 Sample ldif file to create entries

```
dn: O=IBM
objectclass: organization
objectclass: top
o: ibm

dn: cn=ITDI,O=IBM
objectclass: person
objectclass: organizationalPerson
objectclass: top
cn: ITDI
sn: ITDI
userpassword:: aXRkaQ==

dn: cn=ibm-syncAdministrators,O=IBM
objectclass: groupOfUniqueNames
objectclass: top
cn: ibm-syncAdministrators
uniquemember: cn=ITDI,o=IBM
uniquemember: cn=ldapAdmin,o=IBM

dn: ou=sync,O=IBM
objectclass: organizationalUnit
objectclass: top
ou: sync
aclentry: access-id:cn=Anybody:normal:rsc:system:rsc
aclentry: access-id:cn=ibm-syncAdministrators,O=IBM:normal:rwsc:system:rwsc:
 at.secretkey:grant:rwsc
aclpropagate: TRUE

dn: ou=node,ou=sync,O=IBM
objectclass: organizationalUnit
objectclass: top
ou: node

dn: dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass: domain
objectclass: top
dc: vmlinux5

dn: cn=host,dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNode
objectclass: top
cn: host
ibm-issourcesystem: FALSE
ibm-istargetsystem: TRUE
url: ldaps://9.12.4.170:636
```

```
dn: cn=gdbm,dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNodeBackend
objectclass: top
cn: gdbm
ibm-syncbinddn: racfid=ITDI,profiletype=USER,cn=RACFVM
secretkey:: aXRkaQ==

dn: cn=sdbm,dc=vmlinux5,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNodeBackend
objectclass: top
cn: sdbm
ibm-syncbinddn: racfid=ITDI,profiletype=USER,cn=RACFVM
ibm-syncsuffix: cn=RACFVM
secretkey:: aXRkaQ==

dn: dc=sc58,ou=node,ou=sync,O=IBM
objectclass: domain
objectclass: top
dc: sc58

dn: cn=host,dc=sc58,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNode
objectclass: top
cn: host
ibm-hasthesetargets: vmlinux5
ibm-hasthesetargets: vmlinux7
ibm-issourcesystem: TRUE
ibm-istargetsystem: FALSE
url: ldaps://wtsc58oe.itso.ibm.com:636

dn: cn=gdbm,dc=sc58,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNodeBackend
objectclass: top
cn: gdbm
ibm-syncbinddn: racfid=ITDI,profiletype=USER,cn=RACF
secretkey:: aXRkaQ==

dn: cn=sdbm,dc=sc58,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNodeBackend
objectclass: top
cn: sdbm
ibm-syncbinddn: racfid=ITDI,profiletype=USER,cn=RACF
ibm-syncsuffix: cn=RACF
secretkey:: aXRkaQ==

dn: dc=vmlinux7,ou=node,ou=sync,O=IBM
objectclass: domain
objectclass: top
dc: vmlinux7

dn: cn=host,dc=vmlinux7,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNode
objectclass: top
cn: host
ibm-issourcesystem: FALSE
```

```
ibm-istargetsystem: TRUE
url: ldaps://9.12.4.171:636

dn: cn=gdbm,dc=vmlinux7,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNodeBackend
objectclass: top
cn: gdbm
ibm-syncbinddn: racfid=ITDI,profiletype=USER,cn=RACFVM
secretkey:: aXRkaQ==

dn: cn=sdbm,dc=vmlinux7,ou=node,ou=sync,O=IBM
objectclass: ibmSyncNodeBackend
objectclass: top
cn: sdbm
ibm-syncbinddn: racfid=ITDI,profiletype=USER,cn=RACFVM
ibm-syncsuffix: cn=RACFVM
secretkey:: aXRkaQ==
```

5

Testing the configuration

Guidance is provided on testing and deploying the sample configuration. We describe the execution of the AssemblyLines and provide commentary on the components that are used in the configuration.

5.1 Testing the sample configuration by using the CE

When the prerequisite setup is complete, you can test the solution by using the IBM Tivoli Directory Integrator Configuration Editor (CE).

The feeds and flows of the various AssemblyLines of the configuration are described by using an example as a use case.

This example propagates changes from z/OS system sc58 to the z/VM systems, vmlinux5 and vmlinux7.

The z/OS system sc58 is configured as detailed in Chapter 3, “z/OS setup” on page 49. The z/VM target systems require only that an ID for Tivoli Directory Integrator is defined with adequate privileges to update the Resource Access Control Facility (RACF) database. It is also required that the Lightweight Directory Access Protocol (LDAP) server is configured to allow Tivoli Directory Integrator to use this ID with an SDBM style bind.

For the following example, we modify the attribute values in our LDAP directory tree to enable sc58 as the only source system, and vmlinux5 and vmlinux7 as the target systems.

To start running the configuration for using the CE, open the Tivoli Directory Integrator CE. Wait for the default server to start. When started, its color is green in the server window.

To start the configuration, open the project in the CE, as shown in Figure 5-1. Select the startPutandGetALs AssemblyLine.

Figure 5-1 Select AssemblyLine

Double-click the startPutandGetALs AssemblyLine and it opens in the editor window. To start the AssemblyLine, click **Run in console,** which is shown in Figure 5-2.

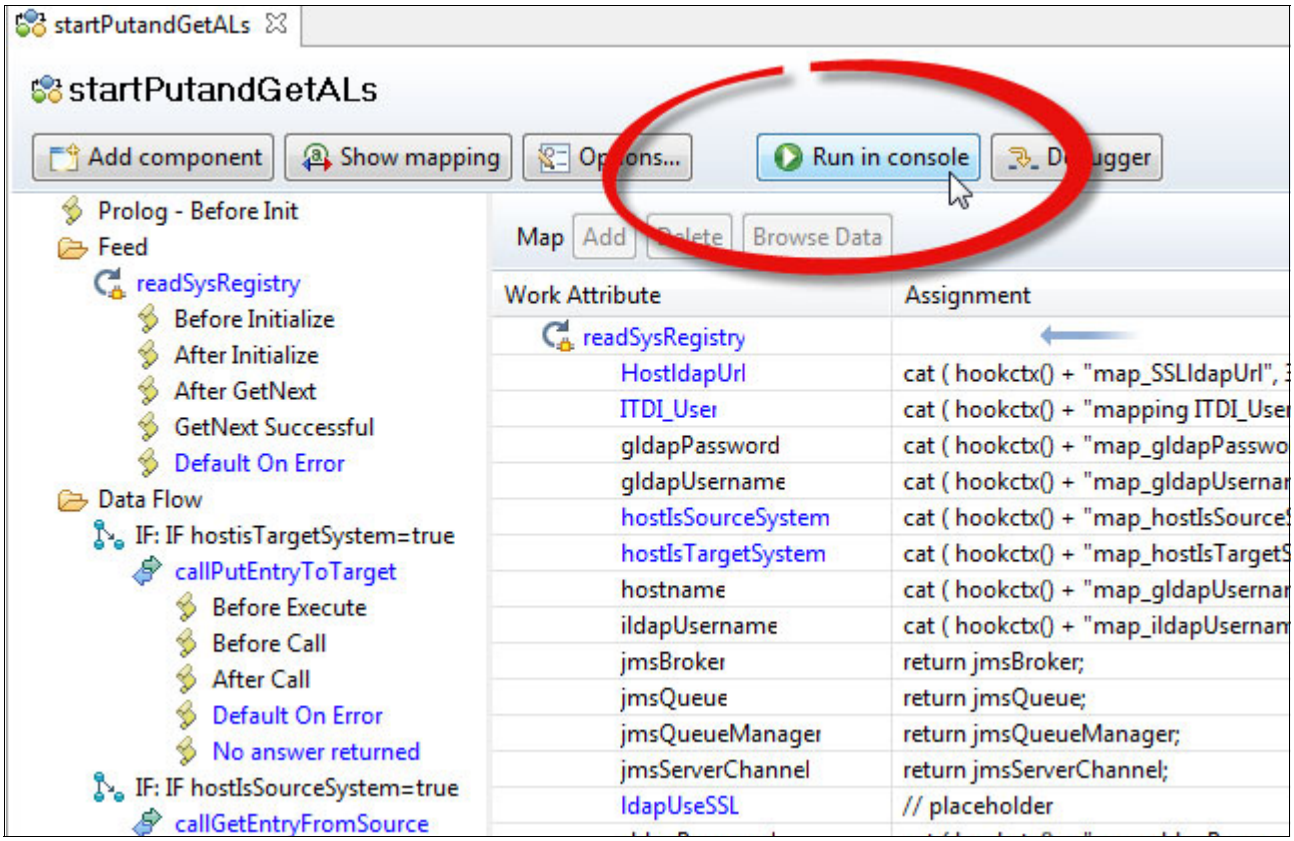


Figure 5-2 Run in console

An output window for the startPutAndGetALs AssemblyLine opens and logging output is displayed. If everything went well, you see new AssemblyLines started under the default server and project name, as shown in Figure 5-3.

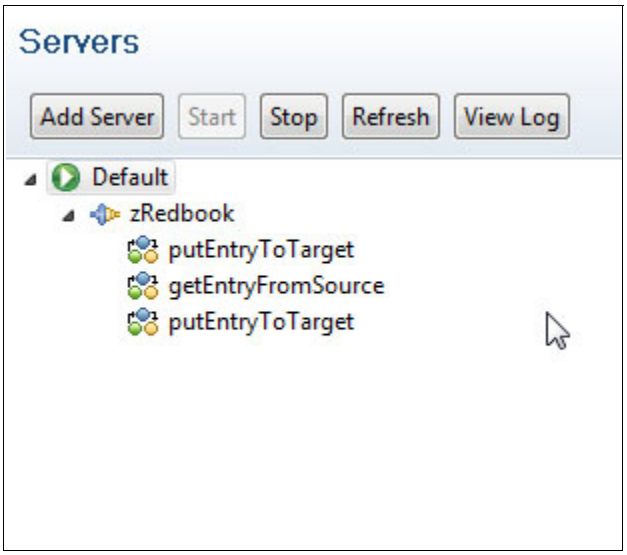


Figure 5-3 Started AssemblyLines

If there are any issues, check the output in the startPutAndGetALs AssemblyLines (ALs) output window and server log (View Log button on server window). Additionally, check the log files for the ALs that are started as described below.

For the AssemblyLines that are started from startPutAndGetALs (the putEntryToTarget and getEntryFromSource), the log files are written to an external file. The log settings are configured to write logs for these ALs to the following locations relative to the solution directory:

- `system_logs/<project name>/AL_getEntryFromSource/`
- `system_logs/<project name>/AL_putEntrytoTarget/`

There is one log file for each AssemblyLine.

If there are no errors, you can test a simple change for a user that is set up for propagation between source and target systems.

5.2 The AssemblyLine feeds and flow

We provide a tour of the AssemblyLine feeds and flows, and comment on the configuration.

5.2.1 General configuration settings

Our configuration consists of a set of AssemblyLines that use various containers, functions, and scripts.

Each AssemblyLine is composed of a set of components. Typically, the AL contains a connector that gathers data (feeds) for the rest of the AssemblyLine logic (the flow). The other components in the AssemblyLine process the data and there is usually another connector that sends the data somewhere.

We use several javascript functions (scripts) that are defined globally (available to all the components). Of note is the global_Prolog script, which sets many variables for use by the components in the configuration. Most of these variables are set from attribute values from the properties store.

Before the configuration can be successfully run, the properties store must be defined as detailed in 4.4.5, “Create configuration properties store” on page 85. The properties store for this discussion is shown in Figure 5-4.

Name	Protected	Local Value
CommonRACFCertLabel	false	commonRACFCert
CryptoFolder	false	crypto
EnvelopeKSPassword	true	********
EnvelopeKeyStore	false	racf_envelope_keystore.jks
FacilityClassPrefix	false	ITDI
MQjmsBroker	false	wtsc58.itso.ibm.com:1610
MQjmsQueue	false	ITDI
MQjmsQueueManager	false	MQ2B
MQjmsServerChannel	false	ITDISERV
SynchToAllTargets	false	FALSE
SysRegistryBindDN	false	cn=itdi,o=ibm
SysRegistryBindPW	true	********
SysRegistryHostName	false	wtsc58.itso.ibm.com
SysRegistrySSLport	false	636
SysRegistrySearchBase	false	ou=node,ou=sync,o=ibm
SysRegistryUseSASL	false	FALSE
SysRegistryUseSASLdn	false	cn=itdi,o=ibm
TDICertAlias	false	itdi
TDICertPassword	true	********
UseCommonRACFCert	false	TRUE

Figure 5-4 properties

5.2.2 startPutAndGetALs

The startPutAndGetALs is the first AssemblyLine that is executed and creates the AssemblyLines that do the real work of getting and propagating changes.

The startPutAndGetALs AL consists of the readSysRegistry component as the data feed.

The startPutAndGetALs AL has the following components as part of its flow:

- IF:IF hostisTargetSystem=true control component
- callPutEntryToTarget Function
- IF:IF hostisSourceSystem=true control component
- callGetEntryFromSource Function

readSysRegistry

The readSysRegistry component is through inheritance of an LDAP connector in iterator mode. The connector first connects to the defined LDAP server hosting the topology information.

Before Initialize hook

In the Before Initialize hook, we set the connector parameters that are based on the values that are defined in the properties store, see 4.4.5, “Create configuration properties store” on page 85.

After Initialize hook

The After Initialize hook prints the connector information for debugging as shown in Example 5-1 (some data removed for clarity), by using a simple bind to the LDAP server.

Example 5-1 readSysRegistry After Initialize hook output

```
2013-03-05 15:08:57,576 INFO  ... - [readSysRegistry] Before Initialize: Setting Parameters
2013-03-05 15:08:57,587 INFO  ... - readSysRegistry ldapUrl: ldaps://wtsc58.itso.ibm.com:636
2013-03-05 15:08:57,588 INFO  ... - readSysRegistry ldapUsername: cn=itdi,o=ibm
2013-03-05 15:08:57,589 INFO  ... - readSysRegistry ldapPassword: secretpw
2013-03-05 15:08:57,590 INFO  ... - readSysRegistry ldapAuthenticationMethod: simple
2013-03-05 15:08:57,591 INFO  ... - readSysRegistry jndiExtraProviderParams:
2013-03-05 15:08:57,592 INFO  ... - readSysRegistry ldapSearchBase: ou=node,ou=sync,o=ibm
2013-03-05 15:08:57,593 INFO  ... - readSysRegistry ldapSearchFilter: dc=*
2013-03-05 15:08:57,594 INFO  ... - readSysRegistry ldapSearchScope: subtree
```

The search filter is set as dc=* and the search scope is subtree. These connector parameters are inherited from and set in the connector tab of the sysRegistry connector. The search base is set from the value of the SysRegistrySearchBase property.

After connection, for each iteration, the connector will retrieve a dc= entry under the defined SysRegistrySearch Base.

Our directory tree contains the dc= nodes under ou=node,ou=sync,o=ibm, as shown in Figure 5-5. Our SysRegistrySearchBase is set to ou=node,ou=sync,o=ibm.

Figure 5-5 dc= nodes

After GetNext hook

After retrieving a dc= entry, the After GetNext hook is executed. In this section, we issue more searches and retrieve the SDBM, GDBM, and host entries for the returned dc= node. For example, for each dc=<node> in our directory tree, there is a cn=gdbm, cn=host and cn=sdbm subentry. These entries contain the attributes that define our topology. See Figure 5-6 on page 105.

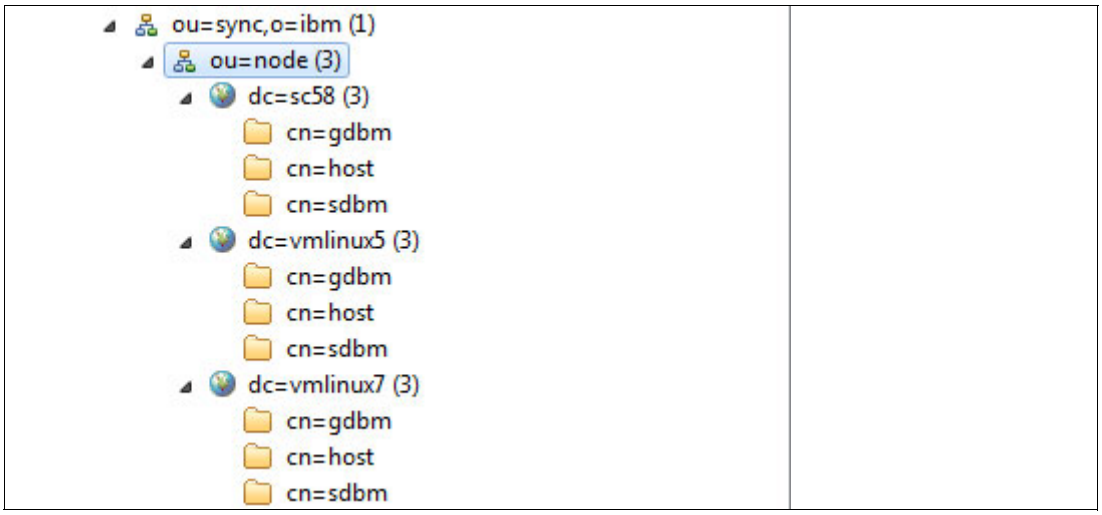


Figure 5-6 dc= subentries

We create variables for the gdbm, sdbm, and host entries. The input Attribute Map phase occurs next. Here, the AssemblyLines work entry is populated with data. This work entry (containing attribute and values) is passed to the rest of the AssemblyLine (our data flow components) to use. The work entry values are populated from the data we retrieved from LDAP.

By using the CE, you can view the data that is part of the work entry and see how the data is set. Click **Show Mapping** in the startPutAndGetALs to show the work entry display. Double-click any attribute to display another window to show the code that is doing the mapping. For example, in Figure 5-7, the gladpUsername attribute of the work entry is set to the value of the ibm-syncbinddn attribute from the gdbm object that is returned from the LDAP search.

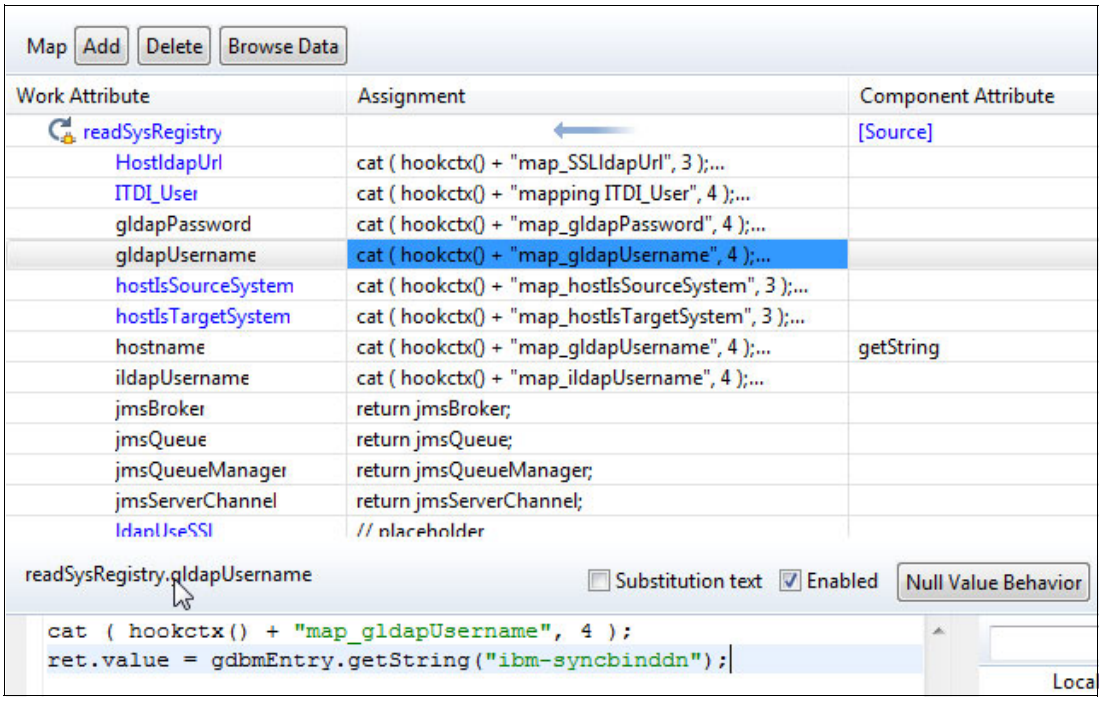


Figure 5-7 Work attributes and values

GetNext Successful hook

After the inputs of the readSysRegistry are mapped to the work entry, in the GetNext Successful hook, we print the work entry. For example, the first entry that is returned from the LDAP server is for dc=vmlinux5. Example 5-2 shows the work entry attributes and values for vmlinux5, a target system.

Example 5-2 Work entry for target system vmlinux5

```
...Entry attributes:
...   gldapPassword (replace):'itdi'
...   ITDI_User (replace):'ITDI'
...   jmsBroker (replace):'wtsc58.itso.ibm.com:1610'
...   hostIsSourceSystem (replace):'FALSE'
...   sldapPassword (replace):'itdi'
...   HostldapUrl (replace):'ldaps://9.12.4.170:636'
...   jmsServerChannel (replace):'ITDISERV'
...   jmsQueueManager (replace):'MQ2B'
...   hostIsTargetSystem (replace):'TRUE'
...   ildapUsername (replace):'RACFID=ITDI,CN=ICTX'
...   gldapUsername (replace):'racfid=ITDI,profiletype=USER,cn=RACFVM'
...   jmsQueue (replace):'ITDI'
...   hostname (replace):'vmlinux5'
...   sldapUsername (replace):'racfid=ITDI,profiletype=USER,cn=RACFVM'
...   sldapSearchBase (replace):'cn=RACFVM'
...CTGDIS004I *** Finished dumping Entry
```

Example 5-3 shows the work entry attributes for a source system, sc58.

Example 5-3 Work entry for source system sc58

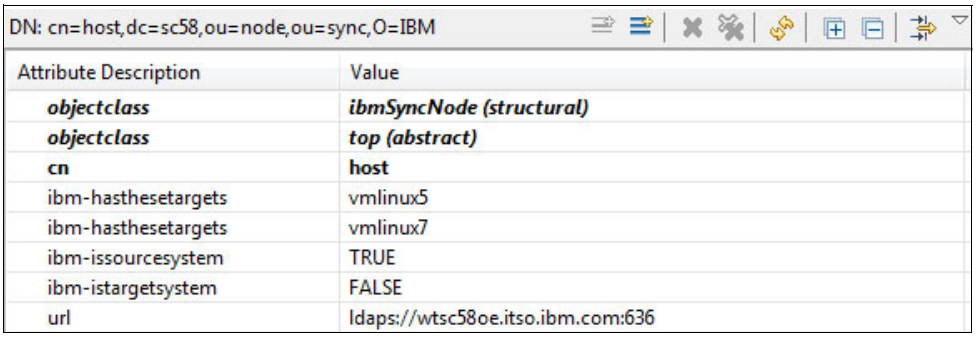
```
...Entry attributes:
...gldapPassword (replace):'itdi'
...ITDI_User (replace):'ITDI'
...jmsBroker (replace):'wtsc58.itso.ibm.com:1610'
...hostIsSourceSystem (replace):'TRUE'
...sldapPassword (replace):'itdi'
...HostldapUrl (replace):'ldaps://wtsc58oe.itso.ibm.com:636'
...jmsServerChannel (replace):'ITDISERV'
...jmsQueueManager (replace):'MQ2B'
...hostIsTargetSystem (replace):'FALSE'
...ildapUsername (replace):'RACFID=ITDI,CN=ICTX'
...gldapUsername (replace):'racfid=ITDI,profiletype=USER,cn=RACF'
...jmsQueue (replace):'ITDI'
...hostname (replace):'sc58'
...sldapUsername (replace):'racfid=ITDI,profiletype=USER,cn=RACF'
...sldapSearchBase (replace):'cn=RACF'
...targetList (replace):'vmlinux7''vmlinux5'
```

As can be seen in Example 5-3, for a source system an attribute (targetList) is created whose values are the intended target systems for which changes are destined. The values of the targetList can be derived from one of two ways.

If the SynchToAllTargets property is `FALSE`, the code parses the multi-valued ibm-hasthesetargets entry from the source system's host entry in the LDAP directory tree. An object named targetlist is created to contain these values. If ibm-hasthesetargets is null, this source system is skipped, thus no getEntryFromSourceAL is started.

If the SynchToAllTargets property is `TRUE`, meaning any system that is defined with ibm-istargetsystem=true is a target for this source system, we call the get_targets function. The get_targets function parses the LDAP directory tree to get a list of the system names of all systems that have a host entry attribute of ibm-istargetsystem=true. An object named *targetlist* is created to contain these values.

In this example, we have SynchToAllTargets set to `FALSE`, so the targetList values are derived from the ibm-hasthesetargets multi-valued attribute from system sc58's cn=host entry in the LDAP directory. See Figure 5-8.


DN: cn=host,dc=sc58,ou=node,ou=sync,O=IBM

Attribute Description	Value
objectclass	***ibmSyncNode (structural)***
objectclass	***top (abstract)***
cn	**host**
ibm-hasthesetargets	vmlinux5
ibm-hasthesetargets	vmlinux7
ibm-issourcesystem	TRUE
ibm-istargetsystem	FALSE
url	ldaps://wtsc58oe.itso.ibm.com:636

Figure 5-8 sc58 LDAP host entry

The targetlist attribute is passed to the AL (getEntryFromSource) created for the source system.

When an iteration of the readSysRegistry is completed, processing passes to the components in the data flow section.

IF hostisTargetSystem=true

The first component in the data flow section is a control flow component. If the host isTargetSystem work attribute is equal to `true`, processing flows to the call PutEntryToTarget function. Otherwise, the next IF component is called.

callPutEntryToTarget

The callPutEntryToTarget function is an instance, through inheritance, of an AssemblyLine Function Component. This component is a means of starting another AssemblyLine.

If a system is defined as being a target system, we start another AssemblyLine, the putEntryToTarget AL.

Before Call Hook

Using the attribute values that we gathered from the LDAP directory tree (and mapped as work entry attributes), in the Before Call hook we set various parameters for the target AL we create. Connection parameters are set for the getFromMQ component and the updateLDAP component of the target AL.

These parameters are set in the Function Component's task control block (TCB), which is used by the called AL to initialize itself. In the Before Call hook, we dump the TCB for debugging. For vmlinux5, we see the following parameters (what we set in the Before Call hook and what is merged from the AL settings), as shown in Example 5-4 on page 108.

Example 5-4 Settings for the PutEntryToTarget AL

```
[callPutEntryToTarget] BeforeCall: Creating TCB
...message filter for target is: target='vmlinux5'
...jmsmessageFilter is set as: target='vmlinux5'
...Calling AL putEntryToTarget with these parameters:
...CTGDIS003I *** Start dumping Entry
...Operation: generic
...Entry attributes:
...Entry properties:
...   $metamerge.tcb.connectorParameters:'{
   "MapDN": "{}",
   "getFromMQ": "{
   "jms.sslCipher": "SSL_RSA_WITH_RC4_128_MD5",
   "jms.lookupRetries": "10",
   "connectorType": "com.ibm.di.connector.IBMMQConnector",
   "jms.lookupConsumesMessage": "false",
   "jms.driver": "IBMMQ",
   "jms.connectionType": "Queue",
   "jms.messageFilter": "target='vmlinux5'",
   "jms.getnextTimeout": "-1",
   "jms.specificPropertiesAsAttributes": "",
   "jms.lookupTimeout": "1000",
   "jms.serverChannel": "ITDISERV",
   "jms.autoAcknowledge": "true",
   "jms.topic": "ITDI",
   "debug": "true",
   "meta.description": "Java Message Server client",
   "inheritFrom": "[parent]",
   "jms.broker": "wtsc58.itso.ibm.com:1610",
   "jms.headersAsAttributes": "false",
   "jms.durable": "false",
   "jms.usetextmessages": "false",
   "jms.qManager": "MQ2B",
   "jms.sslUseFlag": "false",
   "parserOption": "Optional",
   "jms.propertiesAsAttributes": "false",
   "jms.specificHeadersAsAttributes": ""
}",
   "updateLDAP": "{
   "ldapAuthenticationMethod": "Simple",
   "ldapUsername": "racfid=ITDI,profiletype=USER,cn=RACFVM",
   "connectorType": "com.ibm.di.connector.LDAPConnector",
   "ldapTimeLimit": "0",
   "supportsSkipLookup": "true",
   "ldapUseSSL": "true",
   "ldapSizeLimit": "0",
   "ldapSearchFilter": "racfid=*",
   "ldapAddAttr": "false",
   "automapADPassword": "false",
   "ldapVLVPageSize": "0",
   "connectorFlags": "{deleteEmptyStrings}",
   "debug": "true",
   "inheritFrom": "[parent]",
   "ldapSearchScope": "subtree",
   "setOprAttributes": "false",
```

```
   "ldapUrl": "ldaps://9.12.4.170:636",
   "ldapPageSize": "0",
   "ldapReferrals": "follow",
   "parserOption": "Useless",
   "ldapPassword": "itdi",
   "simulateRename": "false",
   "ldapSearchBase": "cn=RACFVM"
}"
}'
```

The putEntryToTarget AL is then started as a separate task (the AssemblyLine Function Component Execution mode is set as Run in the background).

After Call Hook

In the After Call hook, we set a short delay before proceeding with processing. The same Function Component TCB is used for each new AL that is created. Therefore, if the Tivoli Directory Integrator server is on a network with low latency and the LDAP server responds quickly, there is a possibility that the created AL will have its TCB information overlaid with the data from the next iteration of the readSysRegistry processing. This action can occur because the new AL is run as a separate task and control immediately returns to the Function Component, which resumes processing. Therefore, we insert a small delay to allow the created AL to initialize before proceeding.

When the new AL is called, processing for the startPutAndGetALs returns to the readSysRegistry feed to process the next entry.

IF hostisSourceSystem=true

The next component in the data flow section is another control flow component. If the hostisSourceSystem work attribute is equal to `true`, processing flows to the callGetEntryFromSource function. Otherwise, the processing flow is complete and the next iteration of the readSysRegistry connector begins.

callGetEntryFromSource

The callGetEntryFromSource function is an instance, through inheritance, of an AssemblyLine Function Component. This component is a means of starting another AL.

If a system is defined as being a source system, we start another AL, the getEntryFromSource AL.

Before Call Hook

Using the attribute values that we gathered from the LDAP directory tree (and mapped as work entry attributes) in the Before Call hook, we set various parameters for the AL that we create. Connection parameters are set for the readChangeLog, chkICTX, lookupAttributes, and the putToMQ components.

The attribute that is named, *targetList*, is also passed to the new AL. This object contains the list of systems for which this source system creates WebSphere MQ messages. These are the messages that the putEntryToTarget AL reads and processes to push changes to the destination systems.

These parameters are set in the function components task control block (TCB) which is used by the called AL to initialize itself. In the Before Call hook, we dump the TCB for debugging. For our example, sc58 is the source system. Therefore, when the readSysRegistry connector iterates to the dc=sc58 entry, the flow eventually reaches this point and we see the following

parameters (what we set in the Before Call hook and what is merged from the called AL component settings). See Example 5-5.

Example 5-5 Settings for the GetEntryFromSource AL

```
...[callGetEntryFromSource] BeforeCall: Creating TCB
...Calling AL startPutandGetALs with these parameters:
... CTGDIS003I *** Start dumping Entry
... Operation: generic
... Entry attributes:
...   ITDI_User (replace):'ITDI'
...   targetList (replace):'vmlinux7''vmlinux5'
... Entry properties:
...   $metamerge.tcb.connectorParameters:'{
   "lookupAttributes": "{
   "ldapAuthenticationMethod": "Simple",
   "ldapUsername": "racfid=ITDI,profiletype=USER,cn=RACF",
   "ldapReturnAttributes": "racfPasswordEnvelope
racfPassphraseEnvelope
racfid
racfProgrammername",
   "connectorType": "com.ibm.di.connector.LDAPConnector",
   "ldapTimeLimit": "0",
   "supportsSkipLookup": "true",
   "ldapBinaryAttributes": "racfPasswordEnvelope
racfPassphraseEnvelope",
   "ldapUseSSL": "false",
   "ldapSizeLimit": "500",
   "ldapSearchFilter": "",
   "ldapAddAttr": "false",
   "automapADPassword": "false",
   "ldapVLVPageSize": "0",
   "connectorFlags": "{deleteEmptyStrings}",
   "debug": "false",
   "inheritFrom": "[parent]",
   "ldapSearchScope": "subtree",
   "setOprAttributes": "false",
   "ldapUrl": "ldaps://wtsc58oe.itso.ibm.com:636",
   "ldapPageSize": "0",
   "ldapReferrals": "follow",
   "ldapPassword": "itdi",
   "parserOption": "Useless",
   "simulateRename": "false",
   "ldapSearchBase": ""
}",
   "chkICTX": "{
   "ldapAuthenticationMethod": "Simple",
   "ldapUsername": "racfid=ITDI,cn=ictx",
   "connectorType": "com.ibm.di.connector.LDAPConnector",
   "ldapTimeLimit": "0",
   "supportsSkipLookup": "true",
   "ldapUseSSL": "true",
   "ldapSizeLimit": "0",
   "ldapSearchFilter": "objectclass=*",
   "ldapAddAttr": "false",
   "automapADPassword": "false",
```

```
   "ldapVLVPageSize": "0",
   "connectorFlags": "{deleteEmptyStrings}",
   "debug": "false",
   "inheritFrom": "[parent]",
   "ldapSearchScope": "subtree",
   "setOprAttributes": "false",
   "ldapUrl": "ldaps://wtsc58oe.itso.ibm.com:636",
   "ldapPageSize": "0",
   "ldapReferrals": "follow",
   "parserOption": "Useless",
   "ldapPassword": "itdi",
   "simulateRename": "false",
   "ldapSearchBase": "CN=ICTX"
}",
   "readChangelog": "{
   "debug": "false",
   "nsTimeout": "0",
   "ldapAuthenticationMethod": "Simple",
   "nsChangenumber": "EOD",
   "ldapPassword": "itdi",
   "connectorType": "com.ibm.di.connector.ZOSChangelogConnector",
   "mergeMode": "Merge changelog and changed data",
   "iteratorStateKey": "iteratorStateKeysc58",
   "ldapUseSSL": "true",
   "parserOption": "Useless",
   "ldapSearchBase": "cn=changelog",
   "stateKeyPersistence": "End of cycle",
   "inheritFrom": "[parent]",
   "ldapUrl": "ldaps://wtsc58oe.itso.ibm.com:636",
   "nsSleepInterval": "1",
   "jndiExtraProviderParams": "",
   "ldapUsername": "racfid=ITDI,profiletype=USER,cn=RACF"
}",
   "putToMQ": "{
   "jms.sslCipher": "SSL_RSA_WITH_RC4_128_MD5",
   "jms.lookupRetries": "10",
   "connectorType": "com.ibm.di.connector.IBMMQConnector",
   "jms.driver": "IBMMQ",
   "jms.lookupConsumesMessage": "false",
   "jms.connectionType": "Queue",
   "jms.getnextTimeout": "-1",
   "jms.specificPropertiesAsAttributes": "",
   "jms.lookupTimeout": "1000",
   "jms.serverChannel": "ITDISERV",
   "jms.username": "",
   "jms.autoAcknowledge": "true",
   "jms.topic": "ITDI",
   "debug": "false",
   "meta.description": "Java Message Server client",
   "inheritFrom": "[parent]",
   "jms.broker": "wtsc58.itso.ibm.com:1610",
   "jms.headersAsAttributes": "false",
   "jms.durable": "false",
   "jms.usetextmessages": "false",
   "jms.qManager": "MQ2B",
```

```
   "jms.sslUseFlag": "false",
   "parserOption": "Optional",
   "jms.propertiesAsAttributes": "false"
}"
}'
15:52:25,615 INFO  - $metamerge.tcb.alSettings:'{
   "hostname": "sc58"
}'
15:52:25,615 INFO  - CTGDIS004I *** Finished dumping Entry
```

After Call hook

A small delay is introduced in the After Call hook to allow the new AL to properly initialize. When the new AL is called, processing for the startPutAndGetALs returns to the readSysRegistry feed to process the next entry.

When there is no more data for the readSysRegistry connector, this AL terminates.

If there were no errors, there is a putEntryToTarget AL for each target system and a getEntryFromSource for each source system. In our example, there are three ALs running. One getEntryFromSource for sc58 and two putEntryToTarget, one for vmlinux5 and one for vmlinux7, as shown in Figure 5-9.

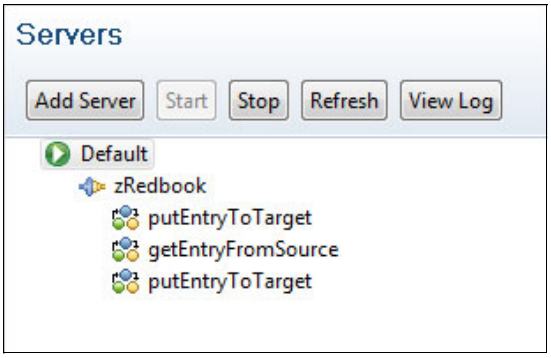


Figure 5-9 ALs when startPutandGetALs complete

5.2.3 getEntryFromSource

For each system that is defined as a source system (the cn=host entry LDAP directory tree entry has issourcesystem=true attribute), a getEntryFromSource AL is started. This AL retrieves changelog records, interrogates those records to see if they are of interest, verifies that the target user of the change is authorized to have the data propagated to target systems, and pushes the change as a message to an WebSphere MQ queue. The message header indicates the system that the change is intended for (target=<system name>) and the message body contains the data that is to be propagated.

The putEntryToTarget AL has the readChangelog component as the data feed.

The putEntryToTarget AL has the following components in the data flow:

- chkICTX
- lookupAttributes
- FOR-EACH Attribute: ForEachTarget attribute loop
- putToMQ

Prolog - Before Init hook

In the Prolog - Before Init hook for the AL, we set various variables from the passed TCB entry, such as the source host name that this AL is intended for, and its list of target systems.

The ITDI_User attribute is the Tivoli Directory Integrator user name on the host system. The targetlist is the list of target systems for this source. These targets are set from parameters that are passed to this AL in this AL's Prolog-Before Init hook.

Also, the system that this AL is associated with is removed from the list of target systems that are received in the targetlist attribute. It is possible, if SynchToAllTarget=true and this system is defined both as a source and target (for some other source), that the source system's name is also in the targetlist. We remove it to avoid any loops. This process is shown in Example 5-6.

Example 5-6 Prolog - Before Init for sc58

```
...sc58 [Prolog0] Prolog - Before Init
... Starting ALgetEntryFromSource for sc58
...hostInTargetList = false
...targetList: [vmlinux7, vmlinux5]
```

readChangelog

The readChangelog component is, through inheritance, an instance of a z/OS changelog connector in iterator mode. This connector polls the changelog of the source system for new changelog records.

When the connector gets a changelog record, the format of the returned changelog record is composed of attributes that are defined on the host LDAPs schema. For example, for a system that is configured for password enveloping, when a password is changed for a user the connector retrieves the data, as shown in Figure 5-10 on page 114.

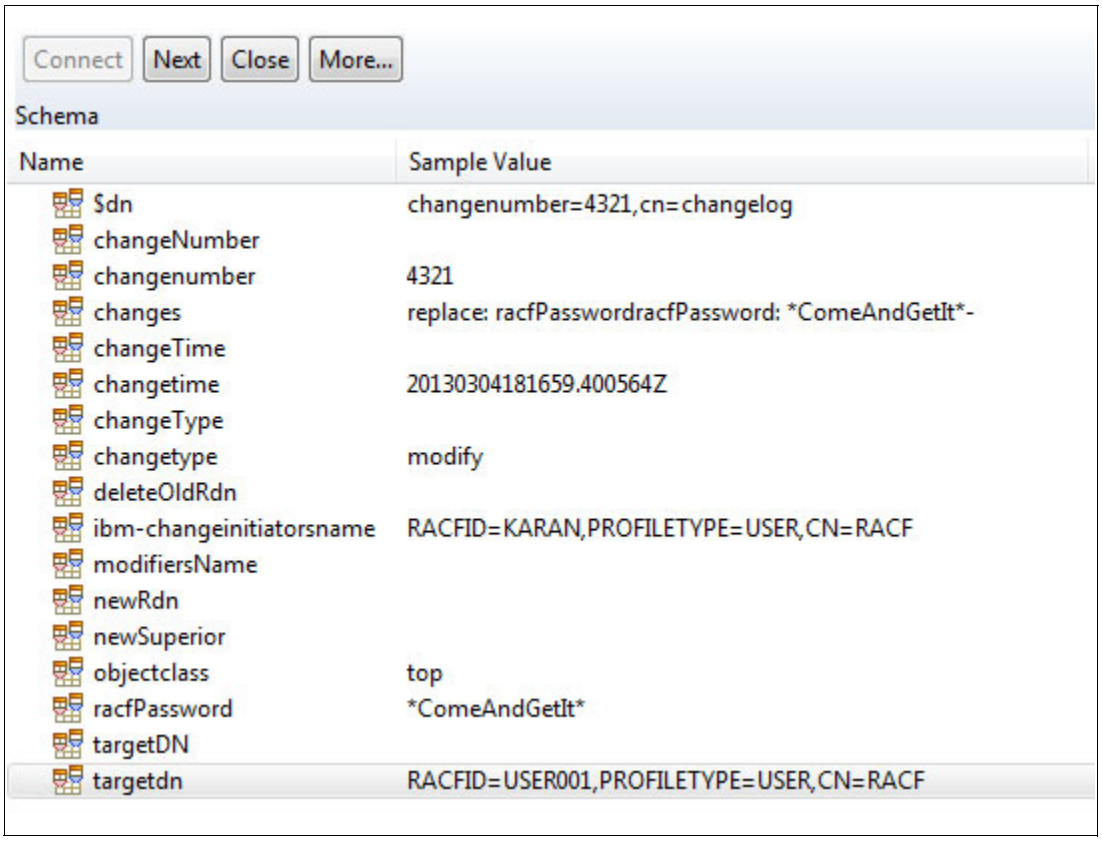


Figure 5-10 changelog record for password change

After GetNext hook

After a changelog record is retrieved, we run code in the After GetNext hook to evaluate the changelog record.

Because this solution example is only interested in changes to USER profiles and only for RACF IDs, we skip this changelog entry if the targetdn does not meet our criteria. We also skip the changelog record if the ibm-changeinitiator's name is equal to ITDI_User. In essence, if Tivoli Directory Integrator last changed the user entry, we do not process the record, thus avoiding a loop condition.

Subsequent to the After GetNext hook, the returned data in is mapped to work entry attributes. The input map for the readChangelog connector is shown in Figure 5-11.

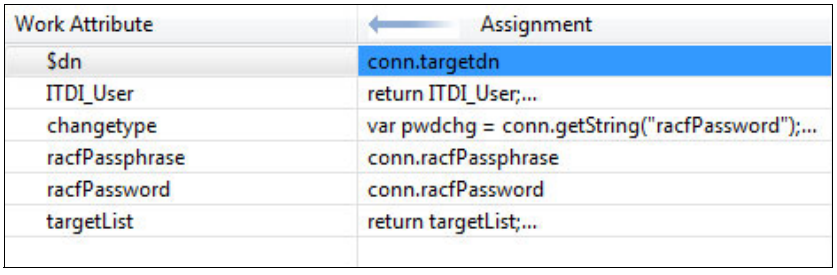


Figure 5-11 readChangelog input

During input mapping, we determine if the changelog is for a password, password phrase, or some other user change and set the changetype attribute accordingly. When an iteration of

the readChangelog connector is complete, processing passes to the data flow component chkICTX.

chkICTX

The chkICTX component is, through inheritance, an instance of an LDAP connector in lookup mode. We use this connector to query the RACF database on the source system to verify whether a user is authorized to have the change propagated to target systems.

Override Lookup hook

We override the connector's default lookup processing in the Override Lookup hook. This hook has the following functions:

- Using two Java classes from the `racfexop.jar` file and the ICTX LDAP plug-in on the host system, we check if the user has read access to the profiles defined in the facility class for the target system and type of change. The returned access check data is stored in a work variable called *ICTXresponse*.
- Those targets systems for which the user is not authorized to have the change propagated are removed from the list of valid targets, for this iteration of the AL. The list of valid targets for this changelog record is stored in loopTargetList, which is used by later connectors.
- If the user is not authorized to have the change propagated to any of the defined target systems, the change is skipped and processing returns to readChangelog to get the next changelog record.
- Because we use the Override hook, work attribute mapping is done in the Override hook.

For example, our FacilityClassPrefix property is set to ITDI. If we give user001 READ access to profile ITDI.VMLINUX5.USER but not ITDI.VMLINUX7.USER, the debugging output in the getEntryFromSource log file when user001's name is changed on sc58 is shown in Example 5-7. As shown in the example, we check access to PW, PP, and USER profiles at one time and then verify authorization for the particular changetype.

Example 5-7 ICTX RACF Facility class check

```
... checking resource access for target: vmlinux7
... UserOrGroup: USER001
... Resource: ITDI.VMLINUX7.PW
... UserOrGroup: USER001
... Resource: ITDI.VMLINUX7.PP
... UserOrGroup: USER001
... Resource: ITDI.VMLINUX7.USER
... checking resource access for target: vmlinux5
... UserOrGroup: USER001
... Resource: ITDI.VMLINUX5.PW
... UserOrGroup: USER001
... Resource: ITDI.VMLINUX5.PP
... UserOrGroup: USER001
... Resource: ITDI.VMLINUX5.USER
...
... function getItemAccess: target = vmlinux7
... function getItemAccess: resource = USER
... function getItemAccess: tindex = 0
... function getItemAccess: rindex = 2
... function getItemAccess: tagval = 701501
... function getItemAccess: MajorCode = 8
... USER001 has access to ICTX facility resource USER : false
... function getItemAccess: target = vmlinux5
```

```
... function getItemAccess: resource = USER
... function getItemAccess: tindex = 1
... function getItemAccess: rindex = 2
... function getItemAccess: tagval = 702502
... function getItemAccess: MajorCode = 0
... USER001 has access to ICTX facility resource USER : true
```

If we have a valid change record to process, the flow continues to the lookupAttributes component.

lookupAttributes

The lookupAttributes component is, through inheritance, an LDAP connector in lookup mode. At this point in the processing, we know we have a changelog record that is of interest to us and the user is authorized to have the change propagated. This component retrieves the user entry that is associated with the changelog record.

We know which user to look up based on the targetdn attribute that is returned with the changelog record (which we stored as a work entry named $dn). We set the link criteria for this connector to the value of $dn. We request the attributes and binary attributes that we are interested in the connector's connection parameters configuration panel (racfPasswordEnvelope, and so on).

Figure 5-12 is an example of the data that would be returned for our example user, USER001.

Name	Sample Value
$dn	racfid=USER001, profiletype=USER, CN=RACF
objectclass	TOP
racfattributes	OPERATIONS
racfauthorizationdate	01/03/13
racfconnectgroupname	RACFID=SYNCH,PROFILETYPE=GROUP,CN=RACF
racfdefaultgroup	RACFID=SYNCH,PROFILETYPE=GROUP,CN=RACF
racfhavepassphraseenvelope	NO
racfhavepasswordenvelope	YES
racfid	USER001
racflastaccess	03/07/13/09:53:15
racflogondays	SUNDAY
racflogontime	ANYTIME
racfowner	RACFID=KARAN,PROFILETYPE=USER,CN=RACF
racfpasswordchangedate	03/04/13
racfpasswordenvelope	[B@f2e0f2e
racfpasswordinterval	90
racfprogrammername	ELMER

Figure 5-12 Sample lookup

During the input mapping phase of the connector, we set attributes in the work entry that are based on the returned data, as shown in Figure 5-13.

lookupAttributes	←	[Source]
havePassphraseEnvelope	conn.racfhavepassphraseenvelope	racfhavepassphraseenvelope
havePasswordEnvelope	conn.racfhavepasswordenvelope	racfhavepasswordenvelope
racfPassphrase	//Set in After Lookup...	racfPassphrase
racfPassword	//Set in After Lookup...	racfPassword
racfProgrammername	conn.racfProgrammername	racfProgrammername
racfattributes	conn.racfattributes	racfattributes
racfid	conn.racfid	racfid

Figure 5-13 lookupAttributes map

Customization: If you want to propagate more data, this feature is one of the places where you can do so. Set work attributes to contain the data you are interested in, and then modify the putToMQ connector to add those fields to the message. Other changes must be made in the putEntryToSource AL's getFromMQ connector to retrieve the fields from the message and then to the updateLDAP connector to add those fields to its update map.

For a change to user profile field properties, there is no indication as to what field changed. Therefore, you must retrieve the fields of interest to be propagated, and then either update the target fields blindly or do a comparison and if different, push the changed fields to the target.

After Lookup hook

When the input map is completed, the After Lookup hook is executed. In this phase, we perform some simple checks to verify that the user has an enveloped password or passphrase, if that is the type of change that is indicated. If not, we skip this entry.

If an envelope exists, then we call the GetCred function in the processPass script. The functions in this script decrypt the password envelope and place the value in the work entry as an attribute, either racfPassword or racfPassphrase.

When the user data is retrieved, control passes to the control flow component, FOR-EACH Attribute: ForEach Target.

FOR-EACH Attribute: ForEach Target

This is a control flow component that runs the putToMQ component for each value in an attribute. In this case, we run the putToMQ component for each value in loopTargetList, which is the list of systems for which this user is authorized to have changes propagated.

putToMQ

The final component in the getEntryFromSource AL is the putToMQ component, which is through inheritance, an instance of an IBM-MQ connector in add-only mode.

Before Add hook

For each iteration, in the Before Add hook, we set the Java Message Service (JMS) message selector to the intended target system, which is 'target=<system name>'. The connector's Add phase places the entries that are defined in the component's output map as a message to the WebSphere MQ queue defined in the property store.

The values of interest are shown in Figure 5-14.

Assignment	Component Attribute
work.changetype	changetype
work.racfPassphrase	racfPassphrase
work.racfPassword	racfPassword
work.racfProgrammername	racfProgrammername
work.racfattributes	racfattributes
work.racfid	racfid

Figure 5-14 putToMQ output map

5.2.4 putEntryToTarget

For each system that is defined as a target system, (the cn=host entry LDAP directory tree entry has istargetsystem=true attribute) a putEntryToTarget AL is started. This AL retrieves messages from the defined WebSphere MQ queue. The message header indicates the system that the change is intended for (target=<system name>) and the message body contains the data that is to be propagated.

The putEntryToTarget AL has the getFromMQ component as the data feed.

The putEntryToTarget AL has the following components in the data flow:

- MapDN
- updateLDAP

getFromMQ

The getFromMQ component is, through inheritance, an instance of the WebSphere MQ connector in iterator mode. The connection is to an WebSphere MQ queue, for which the connection parameters were set in the startPutandGetALs. The actual values that are used are set through the properties store.

In the After Initialize hook, we print the relevant connector parameters as shown in Example 5-8.

Example 5-8 getFromMQ connection parameters

```
2013-03-05 17:27:08,886 DEBUG - vmlinux5 [getFromMQ] After Initialize
2013-03-05 17:27:08,886 DEBUG - broker is wtsc58.itso.ibm.com:1610
2013-03-05 17:27:08,886 DEBUG - server channel is ITDISERV
2013-03-05 17:27:08,886 DEBUG - qManager is MQ2B
2013-03-05 17:27:08,886 DEBUG - topic is ITDI
2013-03-05 17:27:08,886 DEBUG - message filter is target='vmlinux5'
2013-03-05 17:27:08,886 DEBUG - autoAck is true
```

This connector polls messages from the WebSphere MQ queue and retrieves those messages that match the connector's message filter. The message filter is always for the target system for which the AL was created. For example, the putEntryToSource AL for vmlinux5 has a message filter of target=vmlinux5.

The attributes from the WebSphere MQ message are mapped to work entry attributes as shown in Figure 5-15.

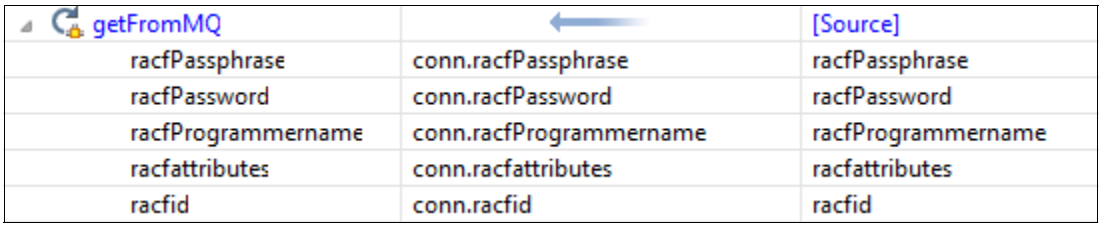


Figure 5-15 getFromMQ input map

When the connector retrieves a record, processing passes to the data flow components. After a change record is retrieved, the message is acknowledged and thus removed from the WebSphere MQ queue.

MapDN

This component sets a work entry attribute, $DN, whose value is the SDBM style DN of the user whose change is updated on the target system.

updateLDAP

The updateLDAP component is, through inheritance, an instance of an LDAP connector in update mode. This connector's parameters were set by the startPutAndGetALs AL.

This connector uses an SDBM style bind to the target system's LDAP server with the Tivoli Directory Integrator user ID and password. The search filter is set to the $DN work entry attribute (initially defined in the MapDN component) through the connector's link criteria settings panel. The connector searches and if found, retrieves the non-null SDBM attributes associated with the user that we are interested in. These returned attributes are stored in an object called *current*. Another entry that is called *conn* is also created and contains the same attributes as the current entry.

For a modify operation, attributes from the work entry are compared to the conn object. If the attribute is null (or does not exist), that attribute is dropped from the conn entry. Other attributes are compared. If they are different, the work entry attribute is merged into the conn object (replaces the conn entry value). If they are the same, then the attribute is dropped from the conn entry.

The resulting conn object contains the attributes that are updated by the connector through and LDAP modify command.

In Figure 5-16, the output map is shown. As shown, all fields are set to `false` for Add operations. And, only the racfattributes, racfpassword, racfPassphrase, and racfprogrammername are set to `true` for modify operations.

Assignment	Add	Mod	Component Attribute
work["$dn"]	false	false	$dn
work.objectclass	false	false	objectclass
work.racfPassphrase	false	true	racfPassphrase
work.racfattributes	false	true	racfattributes
work.racfdefaultgroup	false	false	racfdefaultgroup
work.racfid	false	false	racfid
work.racflogondays	false	false	racflogondays
work.racflogontime	false	false	racflogontime
work.racfowner	false	false	racfowner
work.racfpassword	false	true	racfpassword
work.racfprogrammern	false	true	racfprogrammername

Figure 5-16 updateLDAP output map

Override Add hook

The Add function of the connector is only called if the user does not exist (the search that uses the $DN value does not find a match). If the user does not exist and because we are not interested in adding new users, in the Override Add hook, we simply skip this entry.

5.3 Concluding notes

Whether you use the sample configuration (modified for your environment), or as an example for the creation of your own configuration, we hope that it provided insight on how to use Tivoli Directory Integrator to synchronize RACF data between z/OS and z/VM systems. Of course, it can be used to synchronize data between any type of host system that supports the basic infrastructure. There are many more deployment concerns to consider before you implement such a solution. In this example solution, make the following considerations:

- Establish a Secure Sockets Layer (SSL) connection between Tivoli Directory Integrator and the WebSphere MQ server.
- Encrypt the password in the WebSphere MQ message.
- Comment and disable all logging that displays user passwords in the clear (console output and in log files).
- Have a policy in place to deal with certificate management.
- Have a policy in place to deal with the Tivoli Directory Integrator user password.
- Decide on how to automate the start and monitor the Tivoli Directory Integrator instance, which is running the synchronization configuration.

A

Sample files

This appendix provides the sample IBM Tivoli Directory Integrator configuration file and example properties file.

Tivoli Directory Integrator configuration sample file

The configuration file and sample properties file are provided here and are available as downloads from the additional materials repository that is associated with this IBM Redbooks publication. In addition, the `RACFEXOP.jar` file is also available.

The configuration file is shown in Example A-1. The properties file is shown in Example A-2 on page 195.

Example: A-1 Sample configuration file

```
<?xml version="1.0" encoding="UTF-8"?><MetamergeConfig IDIversion="Created by
TDI7.1.1.2 - 2012-10-15" created="Tue Mar 19 16:15:14 EDT 2013"
createdBy="karansin" modified="Tue Mar 19 16:15:14 EDT 2013" modifiedBy="karansin"
version="7.1.1">
    <Folder name="AssemblyLines">
        <AssemblyLine name="putEntryToTarget">
            <ModTime>1362809582398</ModTime>
            <Settings/>
            <Hooks>
                <Hook name="prologO">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>prologO</Name>
              <Script><![CDATA[/*
*
 *--------------------------------------------------------------------------*
 * DISCLAIMER OF WARRANTIES:                                                *
 *                                                                          *
 *   The following enclosed code is sample code created by IBM              *
 *   Corporation.  This sample code is not part of any standard IBM         *
 *   product and is provided to you solely for the purpose of assisting     *
 *   you in the development of your applications.  The code is provided     *
 *   "AS IS", without warranty of any kind.  IBM shall not be liable        *
 *   for any damages arising out of your use of the sample code, even       *
 *   if they have been advised of the possibility of such damages.          *
 *                                                                          *
 *--------------------------------------------------------------------------*
 *                                                                          */
LOGDEPTH = 4;
LOGLEVEL = "DEBUG";

// name of system we are monitoring
hostname = task.getConfigStr("hostname");
cat( hookctx(hostname, false) + "Prolog - Before Init", 2 );
cat( " Starting AL" + task.getShortName() + " for " + hostname, 1 );

//cat("***** Dumping AL tcb Prolog before init ****" + task.getTCB()
);]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
            </Hooks>
            <CheckpointConfig/>
            <SandboxConfig/>
            <SimulationConfig>
                <SimulationStates>
```

```
                    <Component name="MapDN" state="Enabled"/>
                    <Component name="updateLDAP" state="Simulated"/>
                    <Component name="getFromMQ" state="Enabled"/>
                </SimulationStates>
                <ProxySettings/>
            </SimulationConfig>
            <LogConfig>
                <Logger name="SystemLogAppender">

<InheritFrom>system:/Loggers/ibmdi.SystemLogAppender</InheritFrom>
                    <parameter name="SystemLog.LogPattern">%d{ISO8601} %-5p -
%m%n</parameter>
                    <parameter name="com.ibm.di.log.level">DEBUG</parameter>
                    <parameter name="enabled">true</parameter>
                </Logger>
            </LogConfig>
            <ContainerEF name="EntryFeedContainer">
                <Connector name="getFromMQ">
                    <InheritFrom>/Connectors/getFromMQ_template</InheritFrom>
                    <ModTime>1362808771752</ModTime>
                    <ConnectorMode>Iterator</ConnectorMode>
                    <ConnectorState>Enabled</ConnectorState>
                    <Configuration>
                        <InheritFrom>[parent]</InheritFrom>
                        <parameter name="jms.broker"/>
                        <parameter name="jms.getnextTimeout">-1</parameter>
                        <parameter name="jms.messageFilter"/>
                        <parameter
name="jms.propertiesAsAttributes">false</parameter>
                        <parameter name="jms.qManager"/>
                        <parameter name="jms.serverChannel"/>
                        <parameter name="jms.specificHeadersAsAttributes"/>
                        <parameter name="jms.topic"/>
                    </Configuration>
                    <Parser>
                        <InheritFrom>[parent]</InheritFrom>
                        <Schema name="Input">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                        <Schema name="Output">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                    </Parser>
                    <AttributeMap name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                        <AttributeMapItem>
                            <Name>changenumber</Name>
                            <Type>simple</Type>
                            <Simple>changenumber</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>racfPassphrase</Name>
                            <Type>simple</Type>
                            <Simple>racfPassphrase</Simple>
                        </AttributeMapItem>
```

```
                            <AttributeMapItem>
                                <Name>racfPassword</Name>
                                <Type>simple</Type>
                                <Simple>racfPassword</Simple>
                            </AttributeMapItem>
                            <AttributeMapItem>
                                <Name>racfProgrammername</Name>
                                <Type>simple</Type>
                                <Simple>racfProgrammername</Simple>
                            </AttributeMapItem>
                            <AttributeMapItem>
                                <Name>racfid</Name>
                                <Type>simple</Type>
                                <Simple>racfid</Simple>
                            </AttributeMapItem>
                        </AttributeMap>
                        <AttributeMap name="Output">
                            <InheritFrom>[parent]</InheritFrom>
                        </AttributeMap>
                        <DeltaSettings>
                            <UniqueAttribute/>
                            <WhenToCommit>After every database
operation</WhenToCommit>
                            <RowLocking>SERIALIZABLE</RowLocking>
                            <ChangeDetectionMode>DETECT_ALL</ChangeDetectionMode>
                        </DeltaSettings>
                        <Schema name="Input">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                        <Schema name="Output">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                        <LinkCriteria>
                            <InheritFrom>[parent]</InheritFrom>
                        </LinkCriteria>
                        <Hooks>
                            <InheritFrom>[parent]</InheritFrom>
                            <Hook name="after_initialize">
                                <InheritFrom>[no inheritance]</InheritFrom>
                                <Name>after_initialize</Name>
                                <Script><![CDATA[cat( hookctx(hostname, false) +
"After Initialize", 2 );

cat ( "broker is " + thisConnector.getConnectorParam("jms.broker") );
cat ( "server channel is " + thisConnector.getConnectorParam("jms.serverChannel")
);
cat ( "qManager is " + thisConnector.getConnectorParam("jms.qManager") );
cat ( "topic is " + thisConnector.getConnectorParam("jms.topic") );
cat ( "message filter is " + thisConnector.getConnectorParam("jms.messageFilter")
);
cat ( "autoAck is " + thisConnector.getConnectorParam("jms.autoAcknowledge") );
//cat("***** Dumping AL tcb after getFromMQ init ****" + task.getTCB()
);]]></Script>
                                <Enabled>true</Enabled>
                            </Hook>
```

```
                        <Hook name="get_ok">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>get_ok</Name>
                            <Script><![CDATA[cat( hookctx(hostname, false) +
"GetNext Successful", 2 );

cat("this is work:", 3);
cat( work, 3 );]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                    </Hooks>
                    <CheckpointConfig/>
                    <SandboxConfig/>
                    <Reconnect>
                        <InheritFrom>[parent]</InheritFrom>
                        <ReconnectRules/>
                    </Reconnect>
                    <Operations/>
                    <PoolDefinition>
                        <InheritFrom>[parent]</InheritFrom>
                    </PoolDefinition>
                    <PoolInstance/>
                    <InitializeOption>0</InitializeOption>
                </Connector>
            </ContainerEF>
            <ContainerDF name="DataFlowContainer">
                <ALMap name="MapDN">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <ModTime>1362809410909</ModTime>
                    <AttributeMap name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                        <AttributeMapItem>
                            <Name>$dn</Name>
                            <Type>advanced</Type>
                            <Script><![CDATA[cat( hookctx(hostname, false) +
"MapDN Attribute Map", 2 );

var racfid = work.getString("racfid");
var suffix = updateLDAP.getConnectorParam("ldapSearchBase");
var dn = "racfid=" + racfid + ",profiletype=USER," + suffix;
cat ("Updating " + dn, 3);
return dn;]]></Script>
                        </AttributeMapItem>
                    </AttributeMap>
                    <State>Enabled</State>
                </ALMap>
                <Connector name="updateLDAP">
                    <InheritFrom>/Connectors/updateLDAP_template</InheritFrom>
                    <ModTime>1362809582398</ModTime>
                    <ConnectorMode>Update</ConnectorMode>
                    <ConnectorState>Enabled</ConnectorState>
                    <Configuration>
                        <InheritFrom>[parent]</InheritFrom>
                        <parameter name="debug">true</parameter>
                        <parameter name="ldapPassword"/>
```

```
    <parameter name="ldapSearchBase"/>
    <parameter name="ldapUrl">ldaps://</parameter>
    <parameter name="ldapUseSSL">true</parameter>
    <parameter name="ldapUsername"/>
</Configuration>
<Parser>
    <InheritFrom>[parent]</InheritFrom>
    <Schema name="Input">
        <InheritFrom>[parent]</InheritFrom>
    </Schema>
    <Schema name="Output">
        <InheritFrom>[parent]</InheritFrom>
    </Schema>
</Parser>
<AttributeMap name="Input">
    <InheritFrom>[parent]</InheritFrom>
</AttributeMap>
<AttributeMap name="Output">
    <InheritFrom>[parent]</InheritFrom>
    <AttributeMapItem>
        <Name>$dn</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Script>work["$dn"]</Script>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>objectclass</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Script>work.objectclass</Script>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfPassphrase</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Script>work.racfPassphrase</Script>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfattributes</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Script>work.racfattributes</Script>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfdefaultgroup</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Modify>false</Modify>
        <Script>work.racfdefaultgroup</Script>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfid</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Script>work.racfid</Script>
```

```
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racflogondays</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Modify>false</Modify>
        <Script>work.racflogondays</Script>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racflogontime</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Modify>false</Modify>
        <Script>work.racflogontime</Script>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfowner</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Modify>false</Modify>
        <Script>work.racfowner</Script>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfpassword</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Script>work.racfpassword</Script>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfprogrammername</Name>
        <Type>advanced</Type>
        <Add>false</Add>
        <Script>work.racfprogrammername</Script>
    </AttributeMapItem>
</AttributeMap>
<DeltaSettings/>
<Schema name="Input">
    <InheritFrom>[parent]</InheritFrom>
</Schema>
<Schema name="Output">
    <InheritFrom>[parent]</InheritFrom>
    <SchemaItem>
        <Name>$dn</Name>
    </SchemaItem>
</Schema>
<LinkCriteria>
    <InheritFrom>[parent]</InheritFrom>
</LinkCriteria>
<Hooks>
    <InheritFrom>[parent]</InheritFrom>
    <Hook name="after_initialize">
        <InheritFrom>[no inheritance]</InheritFrom>
        <Name>after_initialize</Name>
        <Script><![CDATA[Init_Hook( "AfterInitialize");
```

```
cat ( "URL is " + thisConnector.getConnectorParam("ldapUrl") );
cat ( "LdapUsername is " + thisConnector.getConnectorParam("ldapUsername")
);]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="after_lookup">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>after_lookup</Name>
                            <Script/>
                            <Enabled>false</Enabled>
                        </Hook>
                        <Hook name="after_modify">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>after_modify</Name>
                            <Script><![CDATA[cat( hookctx(hostname, false) +
"After Modify", 2 );

cat( "Modified " + work.getString("$dn"), 3 );]]></Script>
                        </Hook>
                        <Hook name="before_initialize">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>before_initialize</Name>
                            <Script><![CDATA[Init_Hook( "BeforeInitialize");

catt("hostname is " + hostname, 2);

//cat("***** Dumping AL tcb UpdateLdap before init ****" + task.getTCB()
);]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="before_modify">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>before_modify</Name>
                            <Script><![CDATA[cat( hookctx(hostname, false) +
"Before_Modify: This is conn for " + work.getString("$dn") + " :", 3 );

cat( "This is conn:" );
cat( conn, 3 );]]></Script>
                        </Hook>
                        <Hook name="default_ok">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>default_ok</Name>
                            <Script><![CDATA[cat( hookctx(hostname, false) +
"Default Success", 2 );

cat ( "Processed " + work.getString("$dn"), 1);]]></Script>
                        </Hook>
                        <Hook name="initialize_fail">
                            <Name>initialize_fail</Name>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="modify_nochange">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>modify_nochange</Name>
```

```
                            <Script><![CDATA[cat( hookctx(hostname, false) + "On
No Changes", 2 );

cat( "No updates to " + work.getString("$dn"), 3 );]]></Script>
                        </Hook>
                        <Hook name="override_add">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>override_add</Name>
                            <Script><![CDATA[//If we are here then the user does
not exist on Target System
// Going to skip
cat("LDAP search did not find the user. Skipping Change: " + work);
system.skipEntry();]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                    </Hooks>
                    <CheckpointConfig/>
                    <SandboxConfig/>
                    <Reconnect>
                        <InheritFrom>[parent]</InheritFrom>
                        <ReconnectRules/>
                    </Reconnect>
                    <Operations/>
                    <PoolDefinition>
                        <InheritFrom>[parent]</InheritFrom>
                    </PoolDefinition>
                    <PoolInstance/>
                    <InitializeOption>0</InitializeOption>
                </Connector>
            </ContainerDF>
            <ThreadOptions/>
            <Operations/>
            <InitParams>
                <Schema name="AssemblyLineInitParams"/>
            </InitParams>
        </AssemblyLine>
        <AssemblyLine name="getEntryFromSource">
            <ModTime>1363034153430</ModTime>
            <Settings/>
            <Hooks>
                <Hook name="prolog0">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>prolog0</Name>
              <Script><![CDATA[/*
*
 *--------------------------------------------------------------------*
 * DISCLAIMER OF WARRANTIES:                                          *
 *                                                                    *
 *   The following enclosed code is sample code created by IBM        *
 *   Corporation.  This sample code is not part of any standard IBM   *
 *   product and is provided to you solely for the purpose of assisting  *
 *   you in the development of your applications.  The code is provided  *
 *   "AS IS", without warranty of any kind.  IBM shall not be liable   *
 *   for any damages arising out of your use of the sample code, even  *
 *   if they have been advised of the possibility of such damages.     *
```

```
 *                                                                          *
 *--------------------------------------------------------------------------*
 *                                                                          */
LOGDEPTH = 4;
LOGLEVEL = "DEBUG";

// name of system we are monitoring
hostname = task.getConfigStr("hostname"); //used globally in this AL
cat( hookctx(hostname, false) + "Prolog - Before Init", 2 );
cat( " Starting AL" + task.getShortName() + " for " + hostname, 1 );

targetList = task.getTCB().getAttribute("targetList");
hostInTargetList = targetList.removeValue(hostname);
cat("hostInTargetList = " + hostInTargetList,3);
cat ("targetList: " + targetList, 3);

ITDI_User = task.getTCB().getAttribute("ITDI_User");]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
            </Hooks>
            <CheckpointConfig/>
            <SandboxConfig/>
            <SimulationConfig>
                <SimulationStates>
                    <Component name="chkICTX" state="Enabled"/>
                    <Component name="lookupAttributes" state="Enabled"/>
                    <Component name="ForEachTarget" state="Enabled"/>
                    <Component name="putToMQ" state="Simulated"/>
                    <Component name="readChangelog" state="Enabled"/>
                </SimulationStates>
                <ProxySettings/>
            </SimulationConfig>
            <LogConfig>
                <Logger name="SystemLogAppender">

<InheritFrom>system:/Loggers/ibmdi.SystemLogAppender</InheritFrom>
                    <parameter name="SystemLog.LogPattern">%d{ISO8601} %-5p -
%m%n</parameter>
                    <parameter name="com.ibm.di.log.level">DEBUG</parameter>
                    <parameter name="enabled">true</parameter>
                </Logger>
                <Logger name="ConsoleAppender">

<InheritFrom>system:/Loggers/ibmdi.ConsoleAppender</InheritFrom>
                   <parameter name="Pattern.ConversionPattern">%d{ISO8601} %-5p -
%m%n</parameter>
                    <parameter name="com.ibm.di.log.layout">Pattern</parameter>
                    <parameter name="com.ibm.di.log.level">DEBUG</parameter>
                    <parameter name="enabled">true</parameter>
                </Logger>
            </LogConfig>
            <ContainerEF name="EntryFeedContainer">
                <Connector name="readChangelog">
                    <InheritFrom>/Connectors/read_clog</InheritFrom>
                    <ModTime>1363034129998</ModTime>
```

```
                        <ConnectorMode>Iterator</ConnectorMode>
                        <ConnectorState>Enabled</ConnectorState>
                        <Configuration>
                            <InheritFrom>[parent]</InheritFrom>
                            <parameter name="iteratorStateKey"/>
                        </Configuration>
                        <Parser>
                            <InheritFrom>[parent]</InheritFrom>
                            <Schema name="Input">
                                <InheritFrom>[parent]</InheritFrom>
                            </Schema>
                            <Schema name="Output">
                                <InheritFrom>[parent]</InheritFrom>
                            </Schema>
                        </Parser>
                        <AttributeMap name="Input">
                            <InheritFrom>[parent]</InheritFrom>
                            <AttributeMapItem>
                                <Name>$dn</Name>
                                <Type>advanced</Type>
                                <Script>conn.targetdn</Script>
                            </AttributeMapItem>
                            <AttributeMapItem>
                                <Name>ITDI_User</Name>
                                <Type>advanced</Type>
                                <Script><![CDATA[return ITDI_User;
//from Polog - Before Init]]></Script>
                                <Simple>sdbmldapUsername</Simple>
                            </AttributeMapItem>
                            <AttributeMapItem>
                                <Name>changenumber</Name>
                                <Type>simple</Type>
                                <Simple>changenumber</Simple>
                            </AttributeMapItem>
                            <AttributeMapItem>
                                <Name>changetype</Name>
                                <Type>advanced</Type>
                                <Script><![CDATA[//check returned values - set
changetype
//if ComeAndGetIt then we have an enveloped password
//otherwise indicates some other user profile change
var pwdchg = conn.getString("racfPassword");
var pphchg = conn.getString("racfPassphrase");
if ( pwdchg != null && pwdchg.equals("*ComeAndGetIt*") ) {
   ret.value = "PW";
} else if ( pphchg != null && pphchg.equals("*ComeAndGetIt*") ) {
   ret.value = "PP";
} else {
   cat( hookctx(hostname, false) + "Other change");
   ret.value = "USER";
}]]></Script>
                            </AttributeMapItem>
                            <AttributeMapItem>
                                <Name>racfPassphrase</Name>
                                <Type>advanced</Type>
```

```
                            <Script>conn.racfPassphrase</Script>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>racfPassword</Name>
                            <Type>advanced</Type>
                            <Script>conn.racfPassword</Script>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>targetList</Name>
                            <Type>advanced</Type>
                            <Script><![CDATA[return targetList;
// targetList captured from TCB in Prolog - Before Init; this makes it a work
attribute]]></Script>
                            <Simple>targetList</Simple>
                        </AttributeMapItem>
                    </AttributeMap>
                    <AttributeMap name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </AttributeMap>
                    <DeltaSettings>
                        <UniqueAttribute/>
                        <WhenToCommit>After every database
operation</WhenToCommit>
                        <RowLocking>SERIALIZABLE</RowLocking>
                        <ChangeDetectionMode>DETECT_ALL</ChangeDetectionMode>
                    </DeltaSettings>
                    <Schema name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <Schema name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <LinkCriteria>
                        <InheritFrom>[parent]</InheritFrom>
                    </LinkCriteria>
                    <Hooks>
                        <InheritFrom>[parent]</InheritFrom>
                        <Hook name="after_getnext">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>after_getnext</Name>
                            <Script><![CDATA[cat( hookctx(hostname, false) +
"After GetNext", 2 );

/*
   Decide whether to process this change
   Steps:
      1. Skip if changelog entry is not a user profile change.
      2. Skip if changed entry RDN is not racfid.
      3. Skip if this change initiatied by TDI (which suppresses echos)
*/
var ptype = system.getX400Attribute( conn.getString("targetdn"), "," ,
"PROFILETYPE");
if ( ptype==null || !ptype.equalsIgnoreCase("USER") ) {
   cat( "Processing change number " + conn.getString("changenumber") + ": " +
work.getString("targetdn"), 3 );
```

```
   cat( "Skipping entry because targetdn is not a user profile", 3 );
   system.skipEntry();
}

var racfid = system.getX400Attribute( conn.getString("targetdn"), "," , "racfid");
if ( racfid==null ) {
   cat( hookctx(hostname, false) + "AfterGetNext: Processing change number "
      + conn.getString("changenumber") + ", user " + racfid, 3 );
   cat( "Skipping entry because RDN is not 'racfid'", 3 );
   system.skipEntry();
}

// ITDIUSER is read from a properties file by the global_Prolog script
// using sdbm bind name - since that is the name making changes
if ( conn.getString("ibm-changeinitiatorsname").startsWith("RACFID=" + ITDI_User)
) {
   cat( hookctx(hostname, false) + "AfterGetNext: Processing change number "
      + conn.getString("changenumber") + ", user " + racfid, 2 );
   cat( "Skipping entry because ibm-changeinitiatorsname is the ITDI user", 3 );
   system.skipEntry();
}

cat("Processing change number " + conn.getString("changenumber") + "; user " +
racfid, 2 );
cat( "This is conn:", 3 );
cat( conn, 3 );]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="after_initialize">
                            <Name>after_initialize</Name>
                           <Script><![CDATA[// Function code in 'connector_hooks'
script
Init_Hook( "AfterInitialize");]]></Script>
                        </Hook>
                        <Hook name="before_initialize">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>before_initialize</Name>
                           <Script><![CDATA[// Function code in 'connector_hooks'
script
Init_Hook( "BeforeInitialize");]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="before_selectEntries">
                            <Name>before_selectEntries</Name>
                            <Enabled>false</Enabled>
                        </Hook>
                        <Hook name="connect_init">
                            <Name>connect_init</Name>
                           <Script><![CDATA[// Function code in 'connector_hooks'
script
Critical_Error_Handler();]]></Script>
                        </Hook>
                        <Hook name="default_fail">
                            <Name>default_fail</Name>
```

```
                            <Script><![CDATA[// Function code in 'connector_hooks'
script
Error_Handler();]]></Script>
                        </Hook>
                        <Hook name="get_fail">
                            <Name>get_fail</Name>
                            <Script><![CDATA[// Function code in 'connector_hooks'
script
Error_Handler();]]></Script>
                        </Hook>
                        <Hook name="get_ok">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>get_ok</Name>
                            <Script><![CDATA[cat( hookctx(hostname, false) +
"GetNext Successful", 2 );

var ctype = work.getString("changetype");
if ( ctype != null ) {
   cat( "Changetype: " + ctype, 1 );
}
cat( "This is work:", 3 );
cat(work, 3);]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="initialize_fail">
                            <Name>initialize_fail</Name>
                            <Script><![CDATA[// Function code in 'connector_hooks'
script
Critical_Error_Handler();]]></Script>
                        </Hook>
                        <Hook name="on_connection_failure">
                            <Name>on_connection_failure</Name>
                            <Script><![CDATA[// Function code in 'connector_hooks'
script
On_Connection_Lost();]]></Script>
                        </Hook>
                    </Hooks>
                    <CheckpointConfig/>
                    <SandboxConfig/>
                    <Reconnect>
                        <InheritFrom>[parent]</InheritFrom>
                        <ReconnectRules/>
                    </Reconnect>
                    <Operations/>
                    <PoolDefinition>
                        <InheritFrom>[parent]</InheritFrom>
                    </PoolDefinition>
                    <PoolInstance/>
                    <InitializeOption>0</InitializeOption>
                </Connector>
            </ContainerEF>
            <ContainerDF name="DataFlowContainer">
                <Connector name="chkICTX">
                    <InheritFrom>/Connectors/chkICTX</InheritFrom>
                    <ModTime>1363034153430</ModTime>
```

```
                    <ConnectorMode>Lookup</ConnectorMode>
                    <ConnectorState>Enabled</ConnectorState>
                    <Configuration>
                        <InheritFrom>[parent]</InheritFrom>
                        <parameter name="ldapPassword"/>
                        <parameter
name="ldapUrl">ldaps://localhost:389</parameter>
                        <parameter name="ldapUseSSL">true</parameter>
                        <parameter name="ldapUsername"/>
                    </Configuration>
                    <Parser>
                        <InheritFrom>[parent]</InheritFrom>
                        <Schema name="Input">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                        <Schema name="Output">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                    </Parser>
                    <AttributeMap name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                        <AttributeMapItem>
                            <Name>ICTXresponse</Name>
                            <Type>advanced</Type>
                            <Script>// placeholder; actual value set in Override
Lookup hook</Script>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>authorizedTargetList</Name>
                            <Type>advanced</Type>
                            <Script>//placeholder; actual value set in Override
Lookup</Script>
                            <Simple>authorizedTargetList</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>source</Name>
                            <Type>advanced</Type>
                            <Enabled>true</Enabled>
                            <Script>// placeholder; actual value set in Override
Lookup hook</Script>
                        </AttributeMapItem>
                    </AttributeMap>
                    <AttributeMap name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </AttributeMap>
                    <DeltaSettings/>
                    <Schema name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <Schema name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <LinkCriteria>
                        <InheritFrom>[parent]</InheritFrom>
                    </LinkCriteria>
```

```
                    <Hooks>
                        <InheritFrom>[parent]</InheritFrom>
                        <Hook name="default_ok">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>default_ok</Name>
                            <Script><![CDATA[cat ( hookctx(hostname, true) +
"DefaultSuccess", 2);

cat ( "changetype: " + work.getString("changetype" ), 2 );
cat ( "This is work:", 3  );
cat(work, 3);]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="override_lookup">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>override_lookup</Name>
                            <Script><![CDATA[cat( hookctx(hostname, false) +
"Override Lookup", 2 );

/*
   Purpose: Query ICTX to determine what systems get this change
   Steps: Call Java Class getRACFAuthRequest to perform an LDAP extended operation
*/

ctx = thisConnector.getConnector().getLdapContext();
env = ctx.getEnvironment();
cat("These are the LDAP environment variables:", 4)
envenum = env.elements();
while ( envenum.hasMoreElements()) {
   cat("    " + envenum.nextElement(), 4);
}

UserOrGroup = system.getX400Attribute(work.getString("$dn"), ",", "RACFID");
RequestVersion = java.lang.Integer(1);
ItemVersion = java.lang.Integer(1);
Access = java.lang.Integer(1);
Class = java.lang.String("FACILITY");

var targets = work.getAttribute("targetList").getValues();
Items = java.util.ArrayList();
for (i=0; i<targets.length; i++) {
   cat("checking resource access for target: " + targets[i], 3);
   for (j=0; j<ResSuf.length; j++ ) {
      Item = java.util.HashMap();
      Item.put("ItemVersion", ItemVersion);
      Item.put("ItemTag", java.lang.Integer(calctag(i,j)));
      Item.put("UserOrGroup", UserOrGroup);
      cat( "UserOrGroup: " + UserOrGroup, 2);
      Item.put("Resource", facilityClass_Prefix + "." + targets[i].toUpperCase() +
"." + ResSuf[j]);
      cat( "Resource: " + facilityClass_Prefix + "." + targets[i].toUpperCase() +
"." + ResSuf[j], 3);
      Item.put("Class", Class);
      Item.put("Access", Access);
      Item.put("LogString", "ITDI lookup for " + UserOrGroup);
```

```
      Items.add( Item );
   }
}

// 'local' is the package name for custom classes getRACFAuthRequest
// and getRACFAuthResponse
ICTXrequest = local.getRACFAuthRequest( RequestVersion, Items );
ICTXresponse = ctx.extendedOperation( ICTXrequest );
work.setAttribute("ICTXresponse", ICTXresponse);

for (i=0; i< ICTXresponse.getNumberOfItems(); i++) {
   cat( ICTXresponse.getItemByIndex(i).toString(), 3 );
}

// test if user has access to this resource (changetype) for any target
// for those sytems where user does not have access , remove from targetlist of
systems for putToMQ Loop
targetList = work.getAttribute("targetList");
// make a copy of target list
var authorizedTargetList = new com.ibm.di.entry.Attribute();
for (i=0; i<targetList.size(); i++) {
   var loopstr = new String(targetList.getValue(i));
   authorizedTargetList.addValue(loopstr);
}
//Check the user access for changetype against ICTX data
//If no access to changetype for system - remove system from authorizedTargetList
resource = work.getString("changetype");
var hasAccess = false;
for ( i=0; i<targetList.size(); i++ ) {
   hasAccess = getItemAccess( targetList.getValue(i), resource , ICTXresponse );
   cat( UserOrGroup + " has access to ICTX facility resource " + resource + " : "
+ hasAccess, 3);
   if (hasAccess == false) {
      authorizedTargetList.removeValue(targetList.getValue(i));
   }
}
//if denied access to all targets then skip this entry
if ( authorizedTargetList == null || authorizedTargetList.size() == 0 ) {
   cat( UserOrGroup + " has no access to ICTX facility resource for any systems" +
resource, 3 );
   cat( "Skipping Entry");
   system.skipEntry();
}
work.setAttribute("authorizedTargetList", authorizedTargetList);
work.setAttribute("source", hostname );  // do this here because AttMap overridden
cat("The authorizedTargetList: " +
work.getAttribute("authorizedTargetList"));]]></Script>
                           <Enabled>true</Enabled>
                       </Hook>
                   </Hooks>
                   <CheckpointConfig/>
                   <SandboxConfig/>
                   <Reconnect>
                       <InheritFrom>[parent]</InheritFrom>
                       <ReconnectRules/>
```

```
        </Reconnect>
        <Operations/>
        <PoolDefinition>
            <InheritFrom>[parent]</InheritFrom>
        </PoolDefinition>
        <PoolInstance/>
        <InitializeOption>0</InitializeOption>
    </Connector>
    <Connector name="lookupAttributes">
        <InheritFrom>/Connectors/lookup_template</InheritFrom>
        <ModTime>1362811117930</ModTime>
        <ConnectorMode>Lookup</ConnectorMode>
        <ConnectorState>Enabled</ConnectorState>
        <Configuration>
            <InheritFrom>[parent]</InheritFrom>
        </Configuration>
        <Parser>
            <InheritFrom>[parent]</InheritFrom>
            <Schema name="Input">
                <InheritFrom>[parent]</InheritFrom>
            </Schema>
            <Schema name="Output">
                <InheritFrom>[parent]</InheritFrom>
            </Schema>
        </Parser>
        <AttributeMap name="Input">
            <InheritFrom>[parent]</InheritFrom>
            <AttributeMapItem>
                <Name>havePassphraseEnvelope</Name>
                <Type>advanced</Type>
                <Script>conn.racfhavepassphraseenvelope</Script>
                <Simple>havePassphraseEnvelope</Simple>
            </AttributeMapItem>
            <AttributeMapItem>
                <Name>havePasswordEnvelope</Name>
                <Type>advanced</Type>
                <Script>conn.racfhavepasswordenvelope</Script>
                <Simple>havePasswordEnvelope</Simple>
            </AttributeMapItem>
            <AttributeMapItem>
                <Name>racfPassphrase</Name>
                <Type>advanced</Type>
                <Script>conn.racfPassphrase</Script>
                <Simple>racfPassphrase</Simple>
            </AttributeMapItem>
            <AttributeMapItem>
                <Name>racfPassword</Name>
                <Type>advanced</Type>
                <Script>conn.racfPassword</Script>
                <Simple>racfPassword</Simple>
            </AttributeMapItem>
            <AttributeMapItem>
                <Name>racfProgrammername</Name>
                <Type>advanced</Type>
                <Script>conn.racfProgrammername</Script>
```

```
                            <Simple>racfProgrammername</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>racfid</Name>
                            <Type>simple</Type>
                            <Simple>racfid</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>racfpassphraseenvelope</Name>
                            <Type>simple</Type>
                            <Simple>racfpassphraseenvelope</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>racfpasswordenvelope</Name>
                            <Type>simple</Type>
                            <Simple>racfpasswordenvelope</Simple>
                        </AttributeMapItem>
                    </AttributeMap>
                    <AttributeMap name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </AttributeMap>
                    <DeltaSettings/>
                    <Schema name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <Schema name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <LinkCriteria>
                        <InheritFrom>[parent]</InheritFrom>
                        <LinkCriteriaItem>
                            <Key>12bea411a4b</Key>
                            <Attribute>$dn</Attribute>
                            <Operator>equals</Operator>
                            <Value>$$dn</Value>
                        </LinkCriteriaItem>
                    </LinkCriteria>
                    <Hooks>
                        <InheritFrom>[parent]</InheritFrom>
                        <Hook name="after_initialize">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>after_initialize</Name>
                            <Script><![CDATA[Init_Hook( "AfterInitialize");
cat( hookctx() + ": Connector initialized", 1 );
//thisConnector.getConnector().showServerInfo();

ctx = thisConnector.getConnector().getLdapContext();
env = ctx.getEnvironment();

cat("These are LDAP environment variables:", 4)
envenum = env.elements();
while ( envenum.hasMoreElements()) {
   cat(envenum.nextElement(), 4);
}
```

```
cat("Configuration : " + thisConnector.getConnector.getConfiguration() , 4);
cat("ldapReturnAttributes: " +
thisConnector.connector.getParam("ldapReturnAttributes"));
cat("ldapBinaryAttributes: " +
thisConnector.connector.getParam("ldapBinaryAttributes"));]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="after_lookup">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>after_lookup</Name>
                            <Script><![CDATA[cat( hookctx(hostname, false) +
"After Lookup", 2 );
cat( "This is current:", 3);
cat( current, 3 );


if ( work.getString("changetype").equals("PW") &&
conn.getString("racfhavepasswordenvelope").equalsIgnoreCase("YES")  ) {
   conn.setAttribute("racfPassword", GetCred(hostname));
} else if ( work.getString("changetype").equals("PP") &&
conn.getString("racfhavepassphraseenvelope").equalsIgnoreCase("YES")) {
   conn.setAttribute("racfPassphrase", GetCred(hostname));
} else if ( work.getString("changetype").equals("USER")) {
   cat(hookctx() + "something else");
   conn.setAttribute("racfAttributes", null);
} else {
   cat("error in " + hookctx() + "After Lookup");
   cat("Maybe user does not have a password or passphrase envelope");
   cat("Skipping entry");
   system.skipEntry();
}

cat( "This is conn:", 3);
cat( conn, 3 );]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="before_execute">
                            <Name>before_execute</Name>
                            <Script/>
                            <Enabled>false</Enabled>
                        </Hook>
                        <Hook name="before_initialize">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>before_initialize</Name>
                            <Script>Init_Hook( "BeforeInitialize");</Script>
                            <Enabled>true</Enabled>
                        </Hook>
                        <Hook name="before_lookup">
                            <InheritFrom>[no inheritance]</InheritFrom>
                            <Name>before_lookup</Name>
                            <Script/>
                            <Enabled>false</Enabled>
                        </Hook>
                        <Hook name="lookup_ok">
                            <InheritFrom>[no inheritance]</InheritFrom>
```

```
                            <Name>lookup_ok</Name>
                            <Script><![CDATA[//all for debugging purposes
cat( hookctx(hostname, false) + "Lookup Successful", 3 );

cat( "work entry:", 4);
cat( work, 4);

if ( (pwd=work.getString("racfPassword")) != null ) {
   cat ( hookctx(hostname, false) + "LookupSuccessful: This is the password:", 3
);
   cat(pwd, 3);
}

if ( (pph=work.getString("racfPassphrase")) != null ) {
   cat ( hookctx(hostname, false) + "LookupSuccessful: This is the passphrase:", 3
);
   cat(pph, 3 );
}

cat ( hookctx(hostname, false) + "LookupSuccessful: This is the work object mapped
from SDBM:", 3 );
cat(work, 3 );]]></Script>
                            <Enabled>true</Enabled>
                        </Hook>
                    </Hooks>
                    <CheckpointConfig/>
                    <SandboxConfig/>
                    <Reconnect>
                        <InheritFrom>[parent]</InheritFrom>
                        <ReconnectRules/>
                    </Reconnect>
                    <Operations/>
                    <PoolDefinition>
                        <InheritFrom>[parent]</InheritFrom>
                    </PoolDefinition>
                    <PoolInstance/>
                    <InitializeOption>0</InitializeOption>
                </Connector>
                <Loop name="ForEachTarget">
                    <ModTime>1362808753862</ModTime>
                    <LoopType>2</LoopType>
                    <WorkAttributeName>authorizedTargetList</WorkAttributeName>
                    <LoopAttributeName>targethost</LoopAttributeName>
                    <Branch name="ForEachTarget">
                        <ModTime>1362808753862</ModTime>
                        <Connector name="putToMQ">

<InheritFrom>/Connectors/putToMQ_template</InheritFrom>
                            <ModTime>1362808753862</ModTime>
                            <ConnectorMode>AddOnly</ConnectorMode>
                            <ConnectorState>Enabled</ConnectorState>
                            <Configuration>
                                <InheritFrom>[parent]</InheritFrom>
                                <parameter
name="jms.propertiesAsAttributes">false</parameter>
```

```
</Configuration>
<Parser>
    <InheritFrom>[parent]</InheritFrom>
    <Schema name="Input">
        <InheritFrom>[parent]</InheritFrom>
    </Schema>
    <Schema name="Output">
        <InheritFrom>[parent]</InheritFrom>
    </Schema>
</Parser>
<AttributeMap name="Input">
    <InheritFrom>[parent]</InheritFrom>
</AttributeMap>
<AttributeMap name="Output">
    <InheritFrom>[parent]</InheritFrom>
    <AttributeMapItem>
        <Name>changenumber</Name>
        <Type>simple</Type>
        <Simple>changenumber</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>changetype</Name>
        <Type>simple</Type>
        <Simple>changetype</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfPassphrase</Name>
        <Type>simple</Type>
        <Enabled>true</Enabled>
        <Add>true</Add>
        <Modify>true</Modify>
        <Simple>racfPassphrase</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfPassword</Name>
        <Type>simple</Type>
        <Enabled>true</Enabled>
        <Add>true</Add>
        <Modify>true</Modify>
        <Simple>racfPassword</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfProgrammername</Name>
        <Type>simple</Type>
        <Simple>racfProgrammername</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfid</Name>
        <Type>simple</Type>
        <Simple>racfid</Simple>
    </AttributeMapItem>
</AttributeMap>
<DeltaSettings/>
<Schema name="Input">
    <InheritFrom>[parent]</InheritFrom>
```

```
                            </Schema>
                            <Schema name="Output">
                                <InheritFrom>[parent]</InheritFrom>
                            </Schema>
                            <LinkCriteria>
                                <InheritFrom>[parent]</InheritFrom>
                            </LinkCriteria>
                            <Hooks>
                                <InheritFrom>[parent]</InheritFrom>
                                <Hook name="after_initialize">
                                    <InheritFrom>[no inheritance]</InheritFrom>
                                    <Name>after_initialize</Name>
                                   <Script><![CDATA[cat( hookctx(hostname, false)
+ "After Initialize", 2 );
// needed to use JMS message selectors
thisConnector.getConnector().getSendQueue().setStringProperty("targetClient",
0);]]></Script>
                                    <Enabled>true</Enabled>
                                </Hook>
                                <Hook name="before_add">
                                    <Name>before_add</Name>
                                   <Script><![CDATA[cat( hookctx(hostname, false)
+ "Before Add", 2 );

var targethost = work.getString("targethost");
catt("setting jms.target to " + targethost, 2);

// this will be the message selection filter
conn.setProperty("jms.target", targethost );


catt("this is conn:", 3);
cat( conn, 3 );

catt("this is the JMS info in conn:", 4);
cat(thisConnector.getConnector().entry2message(conn), 4);]]></Script>
                                </Hook>
                            </Hooks>
                            <CheckpointConfig/>
                            <SandboxConfig/>
                            <Reconnect>
                                <InheritFrom>[parent]</InheritFrom>
                                <ReconnectRules/>
                            </Reconnect>
                            <Operations/>
                            <PoolDefinition>
                                <InheritFrom>[parent]</InheritFrom>
                            </PoolDefinition>
                            <PoolInstance/>
                            <InitializeOption>0</InitializeOption>
                        </Connector>
                        <Conditions/>
                        <MatchAny>false</MatchAny>
                        <Enabled>true</Enabled>
                        <Type>0</Type>
```

```
                    </Branch>
                </Loop>
            </ContainerDF>
            <ThreadOptions/>
            <Operations/>
            <InitParams>
                <Schema name="AssemblyLineInitParams"/>
            </InitParams>
        </AssemblyLine>
        <AssemblyLine name="startPutandGetALs">
            <ModTime>1362797536408</ModTime>
            <Settings>
                <parameter
name="ALPoolSettingsDialog">showALPoolSettings</parameter>
                <parameter name="automapattributes">false</parameter>
                <parameter name="createTombstones">false</parameter>
                <parameter name="includeGlobalPrologs">true</parameter>
                <parameter name="nullBehaviorDialog">showNullBehavior</parameter>
            </Settings>
            <Hooks>
                <Hook name="prolog0">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>prolog0</Name>
              <Script><![CDATA[/*
*
 *--------------------------------------------------------------------------*
 * DISCLAIMER OF WARRANTIES:                                                *
 *                                                                          *
 *   The following enclosed code is sample code created by IBM              *
 *   Corporation.  This sample code is not part of any standard IBM         *
 *   product and is provided to you solely for the purpose of assisting     *
 *   you in the development of your applications.  The code is provided     *
 *   "AS IS", without warranty of any kind.  IBM shall not be liable        *
 *   for any damages arising out of your use of the sample code, even       *
 *   if they have been advised of the possibility of such damages.          *
 *                                                                          *
 *--------------------------------------------------------------------------*
 *                                                                          */
LOGDEPTH = 4;]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
            </Hooks>
            <CheckpointConfig/>
            <SandboxConfig/>
            <SimulationConfig>
                <SimulationStates>
                    <Component name="IF hostisTargetSystem=true" state="Enabled"/>
                    <Component name="callPutEntryToTarget" state="Simulated"/>
                    <Component name="IF hostIsSourceSystem=true" state="Enabled"/>
                    <Component name="callGetEntryFromSource" state="Simulated"/>
                    <Component name="readSysRegistry" state="Enabled"/>
                </SimulationStates>
                <ProxySettings/>
            </SimulationConfig>
            <LogConfig/>
```

```
            <ContainerEF name="EntryFeedContainer">
                <Connector name="readSysRegistry">
                    <InheritFrom>/Connectors/readSysRegistry</InheritFrom>
                    <ModTime>1362797240867</ModTime>
                    <ConnectorMode>Iterator</ConnectorMode>
                    <ConnectorState>Enabled</ConnectorState>
                    <Configuration>
                        <InheritFrom>[parent]</InheritFrom>
                        <parameter name="debug">false</parameter>
                        <parameter name="ldapBERTrace"/>
                    </Configuration>
                    <Parser>
                        <InheritFrom>[parent]</InheritFrom>
                        <Schema name="Input">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                        <Schema name="Output">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                    </Parser>
                    <AttributeMap name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                        <AttributeMapItem>
                            <Name>gldapPassword</Name>
                            <Type>advanced</Type>
                            <Script><![CDATA[cat ( hookctx() +
"map_gldapPassword", 4 );
ret.value = gdbmEntry.getString("secretkey");]]></Script>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>gldapUsername</Name>
                            <Type>advanced</Type>
                            <Script><![CDATA[cat ( hookctx() +
"map_gldapUsername", 4 );
ret.value = gdbmEntry.getString("ibm-syncbinddn");]]></Script>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>hostname</Name>
                            <Type>advanced</Type>
                            <Script><![CDATA[cat ( hookctx() +
"map_gldapUsername", 4 );
return conn.getString("dc");]]></Script>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>ildapUsername</Name>
                            <Type>advanced</Type>
                            <Script><![CDATA[cat ( hookctx() +
"map_ildapUsername", 4 );
racfid = system.getX400Attribute(sdbmEntry.getString("ibm-syncbinddn"), ",",
"RACFID");
ret.value = "RACFID=" + racfid + ",CN=ICTX";]]></Script>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>jmsBroker</Name>
                            <Type>advanced</Type>
```

```
                            <Script>return jmsBroker;</Script>
                            <Simple>jmsBroker</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>jmsQueue</Name>
                            <Type>advanced</Type>
                            <Script>return jmsQueue;</Script>
                            <Simple>jmsQueue</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>jmsQueueManager</Name>
                            <Type>advanced</Type>
                            <Script>return jmsQueueManager;</Script>
                            <Simple>jmsQueueManager</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>jmsServerChannel</Name>
                            <Type>advanced</Type>
                            <Script>return jmsServerChannel;</Script>
                            <Simple>jmsServerChannel</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>sldapPassword</Name>
                            <Type>advanced</Type>
                            <Script><![CDATA[cat ( hookctx() +
"map_sldapPassword", 4 );
ret.value = sdbmEntry.getString("secretkey");;]]></Script>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>sldapSearchBase</Name>
                            <Type>advanced</Type>
                            <Script><![CDATA[cat ( hookctx() +
"map_slapSearchBase", 4 );
return sdbmEntry.getString("ibm-syncsuffix");]]></Script>
                            <Simple>sldapSearchBase</Simple>
                        </AttributeMapItem>
                        <AttributeMapItem>
                            <Name>sldapUsername</Name>
                            <Type>advanced</Type>
                            <Script><![CDATA[cat ( hookctx() +
"map_sldapUsername", 4 );
ret.value = sdbmEntry.getString("ibm-syncbinddn");]]></Script>
                            </AttributeMapItem>
                    </AttributeMap>
                    <AttributeMap name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </AttributeMap>
                    <DeltaSettings>
                        <UniqueAttribute/>
                        <WhenToCommit>After every database
operation</WhenToCommit>
                        <RowLocking>SERIALIZABLE</RowLocking>
                        <ChangeDetectionMode>DETECT_ALL</ChangeDetectionMode>
                    </DeltaSettings>
                    <Schema name="Input">
```

```
    <InheritFrom>[parent]</InheritFrom>
    <SchemaItem>
        <Name>$dn</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
</Schema>
<Schema name="Output">
    <InheritFrom>[parent]</InheritFrom>
</Schema>
<LinkCriteria>
    <InheritFrom>[parent]</InheritFrom>
</LinkCriteria>
<Hooks>
    <InheritFrom>[parent]</InheritFrom>
    <Hook name="after_getnext">
        <Name>after_getnext</Name>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="after_initialize">
        <Name>after_initialize</Name>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="before_execute">
        <InheritFrom>[no inheritance]</InheritFrom>
        <Name>before_execute</Name>
        <Script/>
        <Enabled>false</Enabled>
    </Hook>
    <Hook name="before_getnext">
        <Name>before_getnext</Name>
        <Script/>
        <Enabled>false</Enabled>
    </Hook>
    <Hook name="before_initialize">
        <Name>before_initialize</Name>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="default_fail">
        <Name>default_fail</Name>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="get_ok">
        <Name>get_ok</Name>
        <Enabled>true</Enabled>
    </Hook>
</Hooks>
<CheckpointConfig/>
<SandboxConfig>
    <parameter name="sbPlayback">false</parameter>
    <parameter name="sbRecord">false</parameter>
</SandboxConfig>
<Reconnect>
    <InheritFrom>[parent]</InheritFrom>
    <ReconnectRules/>
</Reconnect>
```

```
                        <Operations/>
                        <PoolDefinition>
                            <InheritFrom>[parent]</InheritFrom>
                        </PoolDefinition>
                        <PoolInstance/>
                        <InitializeOption>0</InitializeOption>
                    </Connector>
                </ContainerEF>
                <ContainerDF name="DataFlowContainer">
                    <Branch name="IF hostisTargetSystem=true">
                        <ModTime>1362797536408</ModTime>
                        <Function name="callPutEntryToTarget">
                            <InheritFrom>/Functions/callAL_template</InheritFrom>
                            <ModTime>1362797536408</ModTime>
                            <ConnectorState>Enabled</ConnectorState>
                            <Schema name="Input">
                                <InheritFrom>[parent]</InheritFrom>
                            </Schema>
                            <Schema name="Output">
                                <InheritFrom>[parent]</InheritFrom>
                            </Schema>
                            <Hooks>
                                <InheritFrom>[parent]</InheritFrom>
                                <Hook name="after_functioncall">
                                    <InheritFrom>[no inheritance]</InheritFrom>
                                    <Name>after_functioncall</Name>
                                    <Script><![CDATA[//Add a delay to allow
putEntryToTarget AL to initialize
//If TDI is on a low latency network, it is possible for the FC taskcontrolblock
//to be overwritten with new data (new iteration) before the previously spawned AL
//is properly initialized (the updateLDAP connector will have the wrong ldapURL)
system.sleep(1);]]></Script>
                                    <Enabled>true</Enabled>
                                </Hook>
                                <Hook name="before_execute">
                                    <InheritFrom>[no inheritance]</InheritFrom>
                                    <Name>before_execute</Name>
                                    <Script><![CDATA[//var PrintTcb =
thisComponent.getFunction().getTCB();
//cat("Before Execute Dumping FC tcb: " + PrintTcb);]]></Script>
                                    <Enabled>true</Enabled>
                                </Hook>
                                <Hook name="before_functioncall">
                                    <InheritFrom>[no inheritance]</InheritFrom>
                                    <Name>before_functioncall</Name>
                                    <Script><![CDATA[cat( hookctx() + "BeforeCall:
Creating TCB", 2 );

var tcb = thisComponent.getFunction().getTCB();

tcb.setALSetting("hostname", work.getString("hostname") );

tcb.setConnectorParameter ( "getFromMQ", "jms.broker", work.getString("JMSBROKER")
);
```

```
tcb.setConnectorParameter ( "getFromMQ", "jms.serverChannel",
work.getString("JMSSERVERCHANNEL") );
tcb.setConnectorParameter ( "getFromMQ", "jms.qManager",
work.getString("JMSQUEUEMANAGER") );
tcb.setConnectorParameter ( "getFromMQ", "jms.topic", work.getString("JMSQUEUE")
);


messageFilter = "target='" + work.getString("hostname") + "'";
tcb.setConnectorParameter ( "getFromMQ", "jms.messageFilter", messageFilter );
cat("message filter for target is: " + messageFilter);
jmsfilter = tcb.getConnectorParameter("getFromMQ", "jms.messageFilter");
cat("jmsmessageFilter is set as: " + jmsfilter);

tcb.setConnectorParameter ( "updateLDAP", "ldapUrl", work.getString("HostldapUrl")
);
tcb.setConnectorParameter ( "updateLDAP", "ldapUsername",
work.getString("sldapUsername") );
tcb.setConnectorParameter ( "updateLDAP", "ldapPassword",
work.getString("sldapPassword" ) );
tcb.setConnectorParameter ( "updateLDAP", "ldapSearchBase",
work.getString("sldapSearchBase") );

cat("Calling AL putEntryToTarget with these parameters:", 3);
cat(tcb, 3);]]></Script>
                                <Enabled>true</Enabled>
                            </Hook>
                        </Hooks>
                        <Configuration>
                            <InheritFrom>[parent]</InheritFrom>
                            <parameter
name="assemblyLine">putEntryToTarget</parameter>
                        </Configuration>
                        <SandboxConfig/>
                        <AttributeMap name="Input">
                            <InheritFrom>[parent]</InheritFrom>
                        </AttributeMap>
                        <AttributeMap name="Output">
                            <InheritFrom>[parent]</InheritFrom>
                        </AttributeMap>
                    </Function>
                    <Conditions>
                        <BranchCondition>
                            <LeftHand>hostIsTargetSystem</LeftHand>
                            <Operator>equals</Operator>
                            <RightHand>TRUE</RightHand>
                            <CaseSensitive>false</CaseSensitive>
                            <MatchAny>false</MatchAny>
                        </BranchCondition>
                    </Conditions>
                    <MatchAny>false</MatchAny>
                    <Enabled>true</Enabled>
                    <Type>0</Type>
                </Branch>
                <Branch name="IF hostIsSourceSystem=true">
```

```
                        <ModTime>1362796021420</ModTime>
                        <Function name="callGetEntryFromSource">
                            <InheritFrom>/Functions/callAL_template</InheritFrom>
                            <ModTime>1362796021420</ModTime>
                            <ConnectorState>Enabled</ConnectorState>
                            <Schema name="Input">
                                <InheritFrom>[parent]</InheritFrom>
                                <SchemaItem>
                                    <Name>ITDI_User</Name>
                                </SchemaItem>
                                <SchemaItem>
                                    <Name>havePassphraseEnvelope</Name>
                                </SchemaItem>
                                <SchemaItem>
                                    <Name>havePasswordEnvelope</Name>
                                </SchemaItem>
                                <SchemaItem>
                                    <Name>loopTargetList</Name>
                                </SchemaItem>
                                <SchemaItem>
                                    <Name>sdbmldapUsername</Name>
                                </SchemaItem>
                                <SchemaItem>
                                    <Name>targetlist</Name>
                                </SchemaItem>
                            </Schema>
                            <Schema name="Output">
                                <InheritFrom>[parent]</InheritFrom>
                                <SchemaItem>
                                    <Name>ITDI_User</Name>
                                </SchemaItem>
                                <SchemaItem>
                                    <Name>sdbmldapUsername</Name>
                                </SchemaItem>
                                <SchemaItem>
                                    <Name>targetList</Name>
                                </SchemaItem>
                            </Schema>
                            <Hooks>
                                <InheritFrom>[parent]</InheritFrom>
                                <Hook name="after_functioncall">
                                    <InheritFrom>[no inheritance]</InheritFrom>
                                    <Name>after_functioncall</Name>
                                    <Script><![CDATA[//Add a delay to allow
GetEntryFromSource AL to initialize
//If TDI is on a low latency network, it is possible for the FC taskcontrolblock
//to be overwritten with new data (new iteration) before the previously spawned AL
//is properly initialized (the connectors will have the wrong ldapURL etc..)
system.sleep(1);]]></Script>
                                    <Enabled>true</Enabled>
                                </Hook>
                                <Hook name="before_execute">
                                    <InheritFrom>[no inheritance]</InheritFrom>
                                    <Name>before_execute</Name>
```

```
                                <Script><![CDATA[//var PrintTcb =
thisComponent.getFunction().getTCB();
//cat("Before Execute Dumping FC tcb: " + PrintTcb);]]></Script>
                                <Enabled>false</Enabled>
                            </Hook>
                            <Hook name="before_functioncall">
                                <InheritFrom>[no inheritance]</InheritFrom>
                                <Name>before_functioncall</Name>
                                <Script><![CDATA[cat( hookctx() + "BeforeCall:
Creating TCB", 2 );

var tcb = thisComponent.getFunction().getTCB();

tcb.setALSetting( "hostname", work.getString("hostname") );
tcb.setAttribute( "targetList", work.getAttribute("targetList") );
tcb.setAttribute( "ITDI_User", work.getAttribute("ITDI_User"));


tcb.setConnectorParameter ( "readChangelog", "ldapUrl",
work.getString("HostldapUrl") );
tcb.setConnectorParameter ( "readChangelog", "ldapUsername",
work.getString("gldapUsername") );
tcb.setConnectorParameter ( "readChangelog", "ldapPassword",
work.getString("gldapPassword") );
tcb.setConnectorParameter ( "readChangelog", "iteratorStateKey",
"iteratorStateKey" + work.getString("hostname") );

tcb.setConnectorParameter ( "chkICTX", "ldapUrl", work.getString("HostldapUrl") );
tcb.setConnectorParameter ( "chkICTX", "ldapUsername", "racfid=" +
work.getString("ITDI_user") + ",cn=ictx");
tcb.setConnectorParameter ( "chkICTX", "ldapPassword",
work.getString("sldapPassword") );

tcb.setConnectorParameter ( "lookupAttributes", "ldapUrl",
work.getString("HostldapUrl") );
tcb.setConnectorParameter ( "lookupAttributes", "ldapUsername",
work.getString("sldapUsername") );
tcb.setConnectorParameter ( "lookupAttributes", "ldapPassword",
work.getString("sldapPassword") );

tcb.setConnectorParameter ( "putToMQ", "jms.broker", work.getString("JMSBROKER")
);
tcb.setConnectorParameter ( "putToMQ", "jms.serverChannel",
work.getString("JMSSERVERCHANNEL") );
tcb.setConnectorParameter ( "putToMQ", "jms.qManager",
work.getString("JMSQUEUEMANAGER") );
tcb.setConnectorParameter ( "putToMQ", "jms.topic", work.getString("JMSQUEUE") );


cat("Calling AL " + task.getShortName() + " with these parameters:", 3);
cat(tcb, 3);]]></Script>
                                <Enabled>true</Enabled>
                            </Hook>
                        </Hooks>
                        <Configuration>
```

```
                            <InheritFrom>[parent]</InheritFrom>
                            <parameter
name="assemblyLine">getEntryFromSource</parameter>
                        </Configuration>
                        <SandboxConfig/>
                        <AttributeMap name="Input">
                            <InheritFrom>[parent]</InheritFrom>
                        </AttributeMap>
                        <AttributeMap name="Output">
                            <InheritFrom>[parent]</InheritFrom>
                        </AttributeMap>
                    </Function>
                    <Conditions>
                        <BranchCondition>
                            <LeftHand>hostIsSourceSystem</LeftHand>
                            <Operator>equals</Operator>
                            <RightHand>TRUE</RightHand>
                            <CaseSensitive>false</CaseSensitive>
                            <MatchAny>false</MatchAny>
                        </BranchCondition>
                    </Conditions>
                    <MatchAny>false</MatchAny>
                    <Enabled>true</Enabled>
                    <Type>0</Type>
                </Branch>
            </ContainerDF>
            <ThreadOptions/>
            <Operations/>
            <InitParams>
                <Schema name="AssemblyLineInitParams"/>
            </InitParams>
        </AssemblyLine>
    </Folder>
    <Folder name="Connectors">
        <Connector name="getFromMQ_template">
            <InheritFrom>system:/Connectors/ibmdi.IBM-MQ</InheritFrom>
            <ModTime>1362252923516</ModTime>
            <ConnectorMode>Iterator</ConnectorMode>
            <ConnectorState>Enabled</ConnectorState>
            <Configuration>
                <InheritFrom>[parent]</InheritFrom>
                <parameter name="debug">true</parameter>
                <parameter name="jms.broker"/>
                <parameter name="jms.connectionType">Queue</parameter>
                <parameter name="jms.driver">IBMMQ</parameter>
                <parameter name="jms.getnextTimeout">0</parameter>
                <parameter name="jms.messageFilter"/>
                <parameter name="jms.propertiesAsAttributes">true</parameter>
                <parameter name="jms.qManager"/>
                <parameter name="jms.serverChannel"/>
                <parameter name="jms.specificPropertiesAsAttributes"/>
                <parameter name="jms.topic"/>
            </Configuration>
            <Parser>
                <InheritFrom>[parent]</InheritFrom>
```

```
    <Schema name="Input">
        <InheritFrom>[parent]</InheritFrom>
    </Schema>
    <Schema name="Output">
        <InheritFrom>[parent]</InheritFrom>
    </Schema>
</Parser>
<AttributeMap name="Input">
    <InheritFrom>[parent]</InheritFrom>
</AttributeMap>
<AttributeMap name="Output">
    <InheritFrom>[parent]</InheritFrom>
</AttributeMap>
<DeltaSettings>
    <UniqueAttribute/>
    <WhenToCommit>After every database operation</WhenToCommit>
    <RowLocking>SERIALIZABLE</RowLocking>
    <ChangeDetectionMode>DETECT_ALL</ChangeDetectionMode>
</DeltaSettings>
<Schema name="Input">
    <InheritFrom>[parent]</InheritFrom>
</Schema>
<Schema name="Output">
    <InheritFrom>[parent]</InheritFrom>
</Schema>
<LinkCriteria>
    <InheritFrom>[parent]</InheritFrom>
</LinkCriteria>
<Hooks>
    <InheritFrom>[parent]</InheritFrom>
    <Hook name="default_ok">
        <Name>default_ok</Name>
        <Script/>
        <Enabled>false</Enabled>
    </Hook>
    <Hook name="get_ok">
        <Name>get_ok</Name>
        <Enabled>false</Enabled>
    </Hook>
</Hooks>
<CheckpointConfig/>
<SandboxConfig/>
<Reconnect>
    <InheritFrom>[parent]</InheritFrom>
    <ReconnectRules/>
</Reconnect>
<Operations/>
<PoolDefinition>
    <InheritFrom>[parent]</InheritFrom>
    <Enabled>false</Enabled>
    <MinPoolSize>0</MinPoolSize>
    <MaxPoolSize>0</MaxPoolSize>
    <PurgeInterval>0</PurgeInterval>
    <InitializeAttempts>1</InitializeAttempts>
    <InitializeSleepInterval>0</InitializeSleepInterval>
```

```
                </PoolDefinition>
                <PoolInstance/>
                <InitializeOption>0</InitializeOption>
            </Connector>
            <Connector name="chkICTX">
                <InheritFrom>system:/Connectors/ibmdi.LDAP</InheritFrom>
                <ModTime>1361476020460</ModTime>
                <ConnectorMode>Lookup</ConnectorMode>
                <ConnectorState>Enabled</ConnectorState>
                <Configuration>
                    <InheritFrom>[parent]</InheritFrom>
                    <parameter name="automapADPassword">false</parameter>
                    <parameter name="debug">false</parameter>
                    <parameter name="ldapAddAttr">false</parameter>
                    <parameter name="ldapAuthenticationMethod">Simple</parameter>
                    <parameter name="ldapPageSize">0</parameter>
                    <parameter name="ldapReferrals">follow</parameter>
                    <parameter name="ldapSearchBase">CN=ICTX</parameter>
                    <parameter name="ldapSearchFilter">objectclass=*</parameter>
                    <parameter name="ldapSearchScope">subtree</parameter>
                    <parameter name="ldapSizeLimit">0</parameter>
                    <parameter name="ldapTimeLimit">0</parameter>
                    <parameter name="ldapUseSSL">false</parameter>
                    <parameter name="ldapVLVPageSize">0</parameter>
                    <parameter name="simulateRename">false</parameter>
                </Configuration>
                <Parser>
                    <InheritFrom>[parent]</InheritFrom>
                    <Schema name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <Schema name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                </Parser>
                <AttributeMap name="Input">
                    <InheritFrom>[parent]</InheritFrom>
                    <AttributeMapItem>
                        <Name>ICTXresponse</Name>
                        <Type>advanced</Type>
                        <Script><![CDATA[// placeholder; actual value set in Override
Lookup hook
ret.value = ICTXresponse;]]></Script>
                        <Simple>ICTXresponse</Simple>
                    </AttributeMapItem>
                    <AttributeMapItem>
                        <Name>source</Name>
                        <Type>advanced</Type>
                        <Script><![CDATA[// placeholder; actual value set in Override
Lookup hook
ret.value = sourceHostname;]]></Script>
                        <Simple>sourcehostname</Simple>
                    </AttributeMapItem>
                </AttributeMap>
                <AttributeMap name="Output">
```

```
                <InheritFrom>[parent]</InheritFrom>
            </AttributeMap>
            <DeltaSettings>
                <WhenToCommit>After every database operation</WhenToCommit>
                <Driver>CloudScape</Driver>
            </DeltaSettings>
            <Schema name="Input">
                <InheritFrom>[parent]</InheritFrom>
            </Schema>
            <Schema name="Output">
                <InheritFrom>[parent]</InheritFrom>
            </Schema>
            <LinkCriteria>
                <InheritFrom>[parent]</InheritFrom>
                <LinkCriteriaItem>
                    <Key>11d0cc4b926</Key>
                    <Attribute>true</Attribute>
                    <Operator>equals</Operator>
                    <Value>true</Value>
                </LinkCriteriaItem>
            </LinkCriteria>
            <Hooks>
                <InheritFrom>[parent]</InheritFrom>
                <Hook name="after_initialize">
                    <Name>after_initialize</Name>
                    <Script>Init_Hook( "AfterInitialize");</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="before_initialize">
                    <Name>before_initialize</Name>
                    <Script>Init_Hook( "BeforeInitialize");</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="connect_init">
                    <Name>connect_init</Name>
                    <Script>Critical_Error_Handler();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="default_fail">
                    <Name>default_fail</Name>
                    <Script>Critical_Error_Handler();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="default_ok">
                    <Name>default_ok</Name>
                    <Script><![CDATA[cat ( hookctx(sourceHostname, false) +
"DefaultSuccess: changetype: " + work.getString("changetype" ), 2 );
cat ( hookctx(sourceHostname, false) + "DefaultSuccess: This is work:", 3 );
cat(work, 3);]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="initialize_fail">
                    <Name>initialize_fail</Name>
                    <Script>Critical_Error_Handler();</Script>
                    <Enabled>true</Enabled>
```

```
                </Hook>
                <Hook name="on_connection_failure">
                    <Name>on_connection_failure</Name>
                    <Script>On_Connection_Lost();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="override_lookup">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>override_lookup</Name>
                    <Script><![CDATA[/*
   Purpose: Query ICTX to determine what systems get this change
   Steps: Call Java Class getRACFAuthRequest to perform an LDAP extended operation
*/

ctx = thisConnector.getConnector().getLdapContext();

UserOrGroup = system.getX400Attribute(work.getString("$dn"), ",", "RACFID");
RequestVersion = java.lang.Integer(1);
ItemVersion = java.lang.Integer(1);
Access = java.lang.Integer(1);
Class = java.lang.String("FACILITY");

Items = java.util.ArrayList();
for (i=0; i<targets.length; i++) {
   if ( sourceHostname.equalsIgnoreCase(targets[i]) ) {
      continue;
   }
   for (j=0; j<ResSuf.length; j++ ) {
      Item = java.util.HashMap();
      Item.put("ItemVersion", ItemVersion);
      Item.put("ItemTag", java.lang.Integer(calctag(i,j)));
      Item.put("UserOrGroup", UserOrGroup);
      Item.put("Resource", facilityClass_Prefix + "." + targets[i] + "." +
ResSuf[j]);
      Item.put("Class", Class);
      Item.put("Access", Access);
      Item.put("LogString", "ITDI lookup for " + UserOrGroup);
      Items.add( Item );
   }
}

// 'local' is the package name for custom classes getRACFAuthRequest
// and getRACFAuthResponse
ICTXrequest = local.getRACFAuthRequest( RequestVersion, Items );
ICTXresponse = ctx.extendedOperation( ICTXrequest );
work.setAttribute("ICTXresponse", ICTXresponse);

for (i=0; i< ICTXresponse.getNumberOfItems(); i++) {
   cat( ICTXresponse.getItemByIndex(i).toString(), 3 );
}

work.setAttribute("source", sourceHostname );  // do this here because AttMap
overridden]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
```

```
    </Hooks>
    <CheckpointConfig/>
    <SandboxConfig/>
    <Reconnect>
        <InheritFrom>[parent]</InheritFrom>
        <parameter name="autoreconnect">true</parameter>
        <parameter name="initreconnect">false</parameter>
        <parameter name="numberOfRetries">1</parameter>
        <parameter name="retryDelay">10</parameter>
        <ReconnectRules/>
    </Reconnect>
    <Operations/>
    <PoolDefinition>
        <InheritFrom>[parent]</InheritFrom>
        <Enabled>false</Enabled>
        <MinPoolSize>0</MinPoolSize>
        <MaxPoolSize>0</MaxPoolSize>
        <PurgeInterval>0</PurgeInterval>
        <InitializeAttempts>1</InitializeAttempts>
        <InitializeSleepInterval>0</InitializeSleepInterval>
    </PoolDefinition>
    <PoolInstance>
        <Enabled>false</Enabled>
        <ExhaustedPoolBehavior>Wait</ExhaustedPoolBehavior>
    </PoolInstance>
    <InitializeOption>0</InitializeOption>
</Connector>
<Connector name="putToMQ_template">
    <InheritFrom>system:/Connectors/ibmdi.IBM-MQ</InheritFrom>
    <ModTime>1361369518827</ModTime>
    <ConnectorMode>AddOnly</ConnectorMode>
    <ConnectorState>Enabled</ConnectorState>
    <Configuration>
        <InheritFrom>[parent]</InheritFrom>
        <parameter name="debug">false</parameter>
        <parameter name="jms.broker">wtsc58.itso.ibm.com:1610</parameter>
        <parameter name="jms.connectionType">Queue</parameter>
        <parameter name="jms.propertiesAsAttributes">true</parameter>
        <parameter name="jms.qManager">MQ2B</parameter>
        <parameter name="jms.serverChannel">ITDISERV</parameter>
        <parameter name="jms.specificPropertiesAsAttributes"/>
        <parameter name="jms.topic">ITDI</parameter>
        <parameter name="jms.username"/>
    </Configuration>
    <Parser>
        <InheritFrom>[parent]</InheritFrom>
        <Schema name="Input">
            <InheritFrom>[parent]</InheritFrom>
        </Schema>
        <Schema name="Output">
            <InheritFrom>[parent]</InheritFrom>
        </Schema>
    </Parser>
    <AttributeMap name="Input">
        <InheritFrom>[parent]</InheritFrom>
```

```
                </AttributeMap>
                <AttributeMap name="Output">
                    <InheritFrom>[parent]</InheritFrom>
                </AttributeMap>
                <DeltaSettings/>
                <Schema name="Input">
                    <InheritFrom>[parent]</InheritFrom>
                </Schema>
                <Schema name="Output">
                    <InheritFrom>[parent]</InheritFrom>
                </Schema>
                <LinkCriteria>
                    <InheritFrom>[parent]</InheritFrom>
                </LinkCriteria>
                <Hooks>
                    <InheritFrom>[parent]</InheritFrom>
                    <Hook name="after_initialize">
                        <Name>after_initialize</Name>

<Script>thisConnector.getConnector().getSendQueue().setStringProperty("targetClien
t", 0);</Script>
                        <Enabled>true</Enabled>
                    </Hook>
                    <Hook name="before_add">
                        <Name>before_add</Name>
                        <Script><![CDATA[task.logmsg("setting jms.target to sc58");
task.logmsg("this will be the message selection filter")
conn.setAttribute("jms.target","sc58");

task.logmsg("setting jms.line1 and line2 to strings");
conn.setAttribute("jms.line1","this is line1");
conn.setAttribute("jms.line2","this is line2");

conn.setAttribute("cn", "somecn");

task.logmsg("this is conn in Before Add:")
task.dumpEntry(conn);

task.logmsg("this is the JMS info in conn:")
task.logmsg(J.getConnector().entry2message(conn));]]></Script>
                        <Enabled>true</Enabled>
                    </Hook>
                </Hooks>
                <CheckpointConfig/>
                <SandboxConfig/>
                <Reconnect>
                    <InheritFrom>[parent]</InheritFrom>
                    <ReconnectRules/>
                </Reconnect>
                <Operations/>
                <PoolDefinition>
                    <InheritFrom>[parent]</InheritFrom>
                </PoolDefinition>
                <PoolInstance/>
                <InitializeOption>0</InitializeOption>
```

```
</Connector>
<Connector name="updateLDAP_template">
    <InheritFrom>system:/Connectors/ibmdi.LDAP</InheritFrom>
    <ModTime>1362424526537</ModTime>
    <ConnectorMode>Update</ConnectorMode>
    <ConnectorState>Enabled</ConnectorState>
    <Configuration>
        <InheritFrom>[parent]</InheritFrom>
        <parameter name="debug">true</parameter>
        <parameter name="ldapSearchBase"/>
        <parameter name="ldapSearchFilter">racfid=*</parameter>
    </Configuration>
    <Parser>
        <InheritFrom>[parent]</InheritFrom>
        <Schema name="Input">
            <InheritFrom>[parent]</InheritFrom>
        </Schema>
        <Schema name="Output">
            <InheritFrom>[parent]</InheritFrom>
        </Schema>
    </Parser>
    <AttributeMap name="Input">
        <InheritFrom>[parent]</InheritFrom>
    </AttributeMap>
    <AttributeMap name="Output">
        <InheritFrom>[parent]</InheritFrom>
        <AttributeMapItem>
            <Name>$dn</Name>
            <Type>simple</Type>
            <Enabled>true</Enabled>
            <Add>true</Add>
            <Modify>false</Modify>
            <Simple>$dn</Simple>
        </AttributeMapItem>
        <AttributeMapItem>
            <Name>objectclass</Name>
            <Type>simple</Type>
            <Enabled>true</Enabled>
            <Add>true</Add>
            <Modify>false</Modify>
            <Simple>objectclass</Simple>
        </AttributeMapItem>
        <AttributeMapItem>
            <Name>racfPassphrase</Name>
            <Type>simple</Type>
            <Enabled>true</Enabled>
            <Add>true</Add>
            <Modify>true</Modify>
            <Simple>racfPassphrase</Simple>
        </AttributeMapItem>
        <AttributeMapItem>
            <Name>racfattributes</Name>
            <Type>simple</Type>
            <Enabled>true</Enabled>
            <Add>true</Add>
```

```
        <Modify>true</Modify>
        <Simple>racfattributes</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfdefaultgroup</Name>
        <Type>simple</Type>
        <Enabled>true</Enabled>
        <Add>true</Add>
        <Modify>true</Modify>
        <Simple>racfdefaultgroup</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfid</Name>
        <Type>simple</Type>
        <Enabled>true</Enabled>
        <Add>true</Add>
        <Modify>false</Modify>
        <Simple>racfid</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racflogondays</Name>
        <Type>simple</Type>
        <Enabled>true</Enabled>
        <Add>true</Add>
        <Modify>true</Modify>
        <Simple>racflogondays</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racflogontime</Name>
        <Type>simple</Type>
        <Enabled>true</Enabled>
        <Add>true</Add>
        <Modify>true</Modify>
        <Simple>racflogontime</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfowner</Name>
        <Type>simple</Type>
        <Enabled>true</Enabled>
        <Add>true</Add>
        <Modify>true</Modify>
        <Simple>racfowner</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfpassword</Name>
        <Type>simple</Type>
        <Enabled>true</Enabled>
        <Add>true</Add>
        <Modify>true</Modify>
        <Simple>racfpassword</Simple>
    </AttributeMapItem>
    <AttributeMapItem>
        <Name>racfprogrammername</Name>
        <Type>simple</Type>
        <Enabled>true</Enabled>
```

```
                        <Add>true</Add>
                        <Modify>true</Modify>
                        <Simple>racfprogrammername</Simple>
                    </AttributeMapItem>
                </AttributeMap>
                <DeltaSettings>
                    <WhenToCommit>After every database operation</WhenToCommit>
                    <Driver>CloudScape</Driver>
                </DeltaSettings>
                <Schema name="Input">
                    <InheritFrom>[parent]</InheritFrom>
                </Schema>
                <Schema name="Output">
                    <InheritFrom>[parent]</InheritFrom>
                    <SchemaItem>
                        <Name>$dn</Name>
                    </SchemaItem>
                </Schema>
                <LinkCriteria>
                    <InheritFrom>[parent]</InheritFrom>
                    <LinkCriteriaItem>
                        <Key>11c9af0493e</Key>
                        <Attribute>$dn</Attribute>
                        <Operator>equals</Operator>
                        <Value>$$dn</Value>
                    </LinkCriteriaItem>
                </LinkCriteria>
                <Hooks>
                    <InheritFrom>[parent]</InheritFrom>
                    <Hook name="after_initialize">
                        <InheritFrom>[no inheritance]</InheritFrom>
                        <Name>after_initialize</Name>
                        <Script><![CDATA[Init_Hook( "AfterInitialize");

cat ( "URL is " + thisConnector.getConnectorParam("ldapUrl") );]]></Script>
                        <Enabled>true</Enabled>
                    </Hook>
                    <Hook name="after_modify">
                        <Name>after_modify</Name>
                        <Script>Update_AfterModify();</Script>
                        <Enabled>true</Enabled>
                    </Hook>
                    <Hook name="before_execute">
                        <Name>before_execute</Name>
                        <Script/>
                        <Enabled>false</Enabled>
                    </Hook>
                    <Hook name="before_initialize">
                        <InheritFrom>[no inheritance]</InheritFrom>
                        <Name>before_initialize</Name>
                        <Script>Init_Hook( "BeforeInitialize");</Script>
                        <Enabled>true</Enabled>
                    </Hook>
                    <Hook name="before_modify">
                        <Name>before_modify</Name>
```

```
                    <Script><![CDATA[Update_BeforeModify();
]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="connect_init">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>connect_init</Name>
                    <Script>Critical_Error_Handler();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="default_fail">
                    <Name>default_fail</Name>
                    <Script>Error_Handler();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="default_ok">
                    <Name>default_ok</Name>
                    <Script>Update_DefaultSuccess();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="initialize_fail">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>initialize_fail</Name>
                    <Script>Critical_Error_Handler();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="modify_nochange">
                    <Name>modify_nochange</Name>
                    <Script>Update_OnNoChanges();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="on_connection_failure">
                    <Name>on_connection_failure</Name>
                    <Script>On_Connection_Lost();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="update_fail">
                    <Name>update_fail</Name>
                    <Script>Error_Handler();</Script>
                    <Enabled>true</Enabled>
                </Hook>
            </Hooks>
            <CheckpointConfig/>
            <SandboxConfig/>
            <Reconnect>
                <InheritFrom>[parent]</InheritFrom>
                <parameter name="autoreconnect">true</parameter>
                <parameter name="numberOfRetries">1</parameter>
                <parameter name="retryDelay">10</parameter>
                <ReconnectRules/>
            </Reconnect>
            <Operations/>
            <PoolDefinition>
                <InheritFrom>[parent]</InheritFrom>
            </PoolDefinition>
```

```
            <PoolInstance>
                <Enabled>false</Enabled>
                <ExhaustedPoolBehavior>Wait</ExhaustedPoolBehavior>
            </PoolInstance>
            <InitializeOption>0</InitializeOption>
        </Connector>
        <Connector name="read_clog">
            <InheritFrom>system:/Connectors/ibmdi.zOSChangelog</InheritFrom>
            <ModTime>1362422786948</ModTime>
            <ConnectorMode>Iterator</ConnectorMode>
            <ConnectorState>Enabled</ConnectorState>
            <Configuration>
                <InheritFrom>[parent]</InheritFrom>
                <parameter name="debug">false</parameter>
                <parameter name="iteratorStateKey"/>
                <parameter name="ldapUseSSL">true</parameter>
                <parameter name="mergeMode">Merge changelog and changed
data</parameter>
                <parameter name="nsChangenumber">EOD</parameter>
                <parameter name="nsSleepInterval">1</parameter>
                <parameter name="nsTimeout">0</parameter>
                <parameter name="stateKeyPersistence">End of cycle</parameter>
            </Configuration>
            <Parser>
                <InheritFrom>[parent]</InheritFrom>
                <Schema name="Input">
                    <InheritFrom>[parent]</InheritFrom>
                </Schema>
                <Schema name="Output">
                    <InheritFrom>[parent]</InheritFrom>
                </Schema>
            </Parser>
            <AttributeMap name="Input">
                <InheritFrom>[parent]</InheritFrom>
                <AttributeMapItem>
                    <Name>$dn</Name>
                    <Type>simple</Type>
                    <Simple>targetdn</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>changetype</Name>
                    <Type>advanced</Type>
                    <Script>Read_Clog_MapChangeType();</Script>
                    <Simple>changetype</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>racfPassphrase</Name>
                    <Type>simple</Type>
                    <Simple>racfPassphrase</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>racfPassword</Name>
                    <Type>simple</Type>
                    <Simple>racfPassword</Simple>
                </AttributeMapItem>
```

```
</AttributeMap>
<AttributeMap name="Output">
    <InheritFrom>[parent]</InheritFrom>
</AttributeMap>
<DeltaSettings>
    <UniqueAttribute/>
    <WhenToCommit>After every database operation</WhenToCommit>
    <Driver>CloudScape</Driver>
    <RowLocking>SERIALIZABLE</RowLocking>
    <ChangeDetectionMode>DETECT_ALL</ChangeDetectionMode>
</DeltaSettings>
<Schema name="Input">
    <InheritFrom>[parent]</InheritFrom>
</Schema>
<Schema name="Output">
    <InheritFrom>[parent]</InheritFrom>
</Schema>
<LinkCriteria>
    <InheritFrom>[parent]</InheritFrom>
</LinkCriteria>
<Hooks>
    <InheritFrom>[parent]</InheritFrom>
    <Hook name="after_getnext">
        <Name>after_getnext</Name>
        <Script>Listener_AfterGetNext();</Script>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="after_initialize">
        <InheritFrom>[no inheritance]</InheritFrom>
        <Name>after_initialize</Name>
        <Script>Init_Hook( "AfterInitialize");</Script>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="before_initialize">
        <InheritFrom>[no inheritance]</InheritFrom>
        <Name>before_initialize</Name>
        <Script>Init_Hook( "BeforeInitialize");</Script>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="connect_init">
        <InheritFrom>[no inheritance]</InheritFrom>
        <Name>connect_init</Name>
        <Script>Critical_Error_Handler();</Script>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="default_fail">
        <Name>default_fail</Name>
        <Script>Error_Handler();</Script>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="get_fail">
        <Name>get_fail</Name>
        <Script>Error_Handler();</Script>
        <Enabled>true</Enabled>
    </Hook>
```

```
                    <Hook name="get_ok">
                        <Name>get_ok</Name>
                        <Script>Listener_GetNextSuccessful();</Script>
                        <Enabled>true</Enabled>
                    </Hook>
                    <Hook name="initialize_fail">
                        <InheritFrom>[no inheritance]</InheritFrom>
                        <Name>initialize_fail</Name>
                        <Script>Critical_Error_Handler();</Script>
                        <Enabled>true</Enabled>
                    </Hook>
                    <Hook name="on_connection_failure">
                        <Name>on_connection_failure</Name>
                        <Script>On_Connection_Lost();</Script>
                        <Enabled>true</Enabled>
                    </Hook>
                </Hooks>
                <CheckpointConfig/>
                <SandboxConfig/>
                <Reconnect>
                    <InheritFrom>[parent]</InheritFrom>
                    <parameter name="autoreconnect">true</parameter>
                    <ReconnectRules/>
                </Reconnect>
                <Operations/>
                <PoolDefinition>
                    <InheritFrom>[parent]</InheritFrom>
                    <Enabled>false</Enabled>
                    <MinPoolSize>0</MinPoolSize>
                    <MaxPoolSize>0</MaxPoolSize>
                    <PurgeInterval>0</PurgeInterval>
                    <InitializeAttempts>1</InitializeAttempts>
                    <InitializeSleepInterval>0</InitializeSleepInterval>
                </PoolDefinition>
                <PoolInstance>
                    <Enabled>false</Enabled>
                    <ExhaustedPoolBehavior>Wait</ExhaustedPoolBehavior>
                </PoolInstance>
                <InitializeOption>0</InitializeOption>
            </Connector>
            <Connector name="lookup_template">
                <InheritFrom>system:/Connectors/ibmdi.LDAP</InheritFrom>
                <ModTime>1362807853262</ModTime>
                <ConnectorMode>Lookup</ConnectorMode>
                <ConnectorState>Enabled</ConnectorState>
                <Configuration>
                    <InheritFrom>[parent]</InheritFrom>
                    <parameter name="automapADPassword">false</parameter>
                    <parameter name="debug">true</parameter>
                    <parameter name="ldapAddAttr">false</parameter>
                    <parameter name="ldapAuthenticationMethod">Simple</parameter>
                    <parameter
name="ldapBinaryAttributes"><![CDATA[racfPasswordEnvelope
racfPassphraseEnvelope]]></parameter>
                    <parameter name="ldapPageSize">0</parameter>
```

```
                <parameter name="ldapPassword"/>
                <parameter name="ldapReferrals">follow</parameter>
                <parameter
name="ldapReturnAttributes"><![CDATA[racfPasswordEnvelope
racfPassphraseEnvelope
racfProgrammername
racfhavepasswordenvelope
racfhavepassphraseenvelope
racfid]]></parameter>
                <parameter name="ldapSearchBase"/>
                <parameter name="ldapSearchFilter"/>
                <parameter name="ldapSearchScope">subtree</parameter>
                <parameter name="ldapSizeLimit">500</parameter>
                <parameter name="ldapTimeLimit">0</parameter>
                <parameter name="ldapUrl">ldaps://localhost:389</parameter>
                <parameter name="ldapUseSSL">true</parameter>
                <parameter name="ldapUsername"/>
                <parameter name="ldapVLVPageSize">0</parameter>
                <parameter name="simulateRename">false</parameter>
            </Configuration>
            <Parser>
                <InheritFrom>[parent]</InheritFrom>
                <Schema name="Input">
                    <InheritFrom>[parent]</InheritFrom>
                </Schema>
                <Schema name="Output">
                    <InheritFrom>[parent]</InheritFrom>
                </Schema>
            </Parser>
            <AttributeMap name="Input">
                <InheritFrom>[parent]</InheritFrom>
            </AttributeMap>
            <AttributeMap name="Output">
                <InheritFrom>[parent]</InheritFrom>
            </AttributeMap>
            <DeltaSettings>
                <WhenToCommit>After every database operation</WhenToCommit>
                <Driver>CloudScape</Driver>
            </DeltaSettings>
            <Schema name="Input">
                <InheritFrom>[parent]</InheritFrom>
                <SchemaItem>
                    <Name>$dn</Name>
                    <Syntax>java.lang.String</Syntax>
                </SchemaItem>
                <SchemaItem>
                    <Name>objectclass</Name>
                    <Syntax>java.lang.String</Syntax>
                </SchemaItem>
                <SchemaItem>
                    <Name>racfattributes</Name>
                    <Syntax>java.lang.String</Syntax>
                </SchemaItem>
                <SchemaItem>
                    <Name>racfauthorizationdate</Name>
```

```
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfconnectgroupname</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfdefaultgroup</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfhavepassphraseenvelope</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfhavepasswordenvelope</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfid</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racflastaccess</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racflogondays</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racflogontime</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfowner</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfpasswordchangedate</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfpasswordenvelope</Name>
        <Syntax>[B</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfpasswordinterval</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfprogrammername</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
</Schema>
```

```
<Schema name="Output">
    <InheritFrom>[parent]</InheritFrom>
    <SchemaItem>
        <Name>$dn</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>objectclass</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfattributes</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfauthorizationdate</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfconnectgroupname</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfdefaultgroup</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfhavepassphraseenvelope</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfhavepasswordenvelope</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfid</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racflastaccess</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racflogondays</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racflogontime</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
        <Name>racfowner</Name>
        <Syntax>java.lang.String</Syntax>
    </SchemaItem>
    <SchemaItem>
```

```
                    <Name>racfpasswordchangedate</Name>
                    <Syntax>java.lang.String</Syntax>
                </SchemaItem>
                <SchemaItem>
                    <Name>racfpasswordenvelope</Name>
                    <Syntax>[B</Syntax>
                </SchemaItem>
                <SchemaItem>
                    <Name>racfpasswordinterval</Name>
                    <Syntax>java.lang.String</Syntax>
                </SchemaItem>
                <SchemaItem>
                    <Name>racfprogrammername</Name>
                    <Syntax>java.lang.String</Syntax>
                </SchemaItem>
            </Schema>
            <LinkCriteria>
                <InheritFrom>[parent]</InheritFrom>
            </LinkCriteria>
            <Hooks>
                <InheritFrom>[parent]</InheritFrom>
                <Hook name="after_initialize">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>after_initialize</Name>
                    <Script><![CDATA[Init_Hook( "AfterInitialize");
cat( hookctx() + ": Connector initialized", 1 );
//thisConnector.getConnector().showServerInfo();

ctx = thisConnector.getConnector().getLdapContext();
env = ctx.getEnvironment();

cat("These are LDAP environment variables:", 4)
envenum = env.elements();
while ( envenum.hasMoreElements()) {
   cat(envenum.nextElement(), 4);
}]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="before_initialize">
                    <Name>before_initialize</Name>
                    <Script>Init_Hook( "BeforeInitialize");</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="connect_init">
                    <Name>connect_init</Name>
                    <Script>Critical_Error_Handler();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="initialize_fail">
                    <Name>initialize_fail</Name>
                    <Script>Critical_Error_Handler();</Script>
                    <Enabled>true</Enabled>
                </Hook>
            </Hooks>
            <CheckpointConfig/>
```

```
        <SandboxConfig/>
        <Reconnect>
            <InheritFrom>[parent]</InheritFrom>
            <parameter name="autoreconnect">true</parameter>
            <parameter name="initreconnect">false</parameter>
            <parameter name="numberOfRetries">999</parameter>
            <parameter name="retryDelay">10</parameter>
            <ReconnectRules/>
        </Reconnect>
        <Operations/>
        <PoolDefinition>
            <InheritFrom>[parent]</InheritFrom>
            <Enabled>false</Enabled>
            <MinPoolSize>0</MinPoolSize>
            <MaxPoolSize>0</MaxPoolSize>
            <PurgeInterval>0</PurgeInterval>
            <InitializeAttempts>1</InitializeAttempts>
            <InitializeSleepInterval>0</InitializeSleepInterval>
        </PoolDefinition>
        <PoolInstance/>
        <InitializeOption>0</InitializeOption>
    </Connector>
    <Connector name="readSysRegistry">
        <InheritFrom>system:/Connectors/ibmdi.LDAP</InheritFrom>
        <ModTime>1362797084756</ModTime>
        <ConnectorMode>Iterator</ConnectorMode>
        <ConnectorState>Enabled</ConnectorState>
        <Configuration>
            <InheritFrom>[parent]</InheritFrom>
            <parameter name="debug">false</parameter>
            <parameter name="jndiExtraProviderParams"/>
            <parameter name="ldapAuthenticationMethod">Simple</parameter>
            <parameter name="ldapBERTrace"/>
            <parameter name="ldapPassword"/>
            <parameter name="ldapSearchBase"/>
            <parameter name="ldapSearchFilter">dc=*</parameter>
            <parameter name="ldapUrl">ldaps://localhost:636</parameter>
            <parameter name="ldapUseSSL">true</parameter>
            <parameter name="ldapUsername"/>
        </Configuration>
        <Parser>
            <InheritFrom>[parent]</InheritFrom>
            <Schema name="Input">
                <InheritFrom>[parent]</InheritFrom>
            </Schema>
            <Schema name="Output">
                <InheritFrom>[parent]</InheritFrom>
            </Schema>
        </Parser>
        <AttributeMap name="Input">
            <InheritFrom>[parent]</InheritFrom>
            <AttributeMapItem>
                <Name>HostldapUrl</Name>
                <Type>advanced</Type>
                <Enabled>true</Enabled>
```

```
                    <Add>true</Add>
                    <Modify>true</Modify>
                    <Script><![CDATA[cat ( hookctx() + "map_SSLldapUrl", 3 );
ret.value = hostEntry.getString("url");]]></Script>
                    <Simple>gldapUrl</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>ITDI_User</Name>
                    <Type>advanced</Type>
                    <Script><![CDATA[cat ( hookctx() + "mapping ITDI_User", 4 );
ret.value = system.
getX400Attribute(sdbmEntry.getString("ibm-syncbinddn"), ",",
"RACFID");]]></Script>
                    <Simple>ITDI_User</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>gldapPassword</Name>
                    <Type>advanced</Type>
                    <Enabled>true</Enabled>
                    <Add>true</Add>
                    <Modify>true</Modify>
                    <Script><![CDATA[cat ( hookctx() + "mapping gldapPassword", 4
);
secretkeybin = gdbmEntry.getObject("secretkey");
secretkey = system.arrayToString(secretkeybin);
ret.value = secretkey;]]></Script>
                    <Simple>gldapPassword</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>gldapUsername</Name>
                    <Type>advanced</Type>
                    <Enabled>true</Enabled>
                    <Add>true</Add>
                    <Modify>true</Modify>
                    <Script><![CDATA[cat ( hookctx() + "map_gldapUsername", 3 );
ret.value = gdbmEntry.getString("ibm-syncbinddn");]]></Script>
                    <Simple>gldapUsername</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>hostIsSourceSystem</Name>
                    <Type>advanced</Type>
                   <Script><![CDATA[cat ( hookctx() + "map_hostIsSourceSystem", 3
);
ret.value = hostEntry.getString("ibm-issourcesystem");]]></Script>
                    <Simple>hostIsSourceSystem</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>hostIsTargetSystem</Name>
                    <Type>advanced</Type>
                   <Script><![CDATA[cat ( hookctx() + "map_hostIsTargetSystem", 3
);
ret.value = hostEntry.getString("ibm-istargetsystem");]]></Script>
                    <Simple>hostIsTargetSystem</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
```

```
                    <Name>hostname</Name>
                    <Type>advanced</Type>
                    <Enabled>true</Enabled>
                    <Add>true</Add>
                    <Modify>true</Modify>
                    <Script><![CDATA[cat ( hookctx() + "map_gldapUsername", 4 );
return conn.getString("dc");]]></Script>
                    <Simple>dc</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>ildapUsername</Name>
                    <Type>advanced</Type>
                    <Script><![CDATA[cat ( hookctx() + "map_ildapUsername", 3 );
racfid = system.getX400Attribute(sdbmEntry.getString("ibm-syncbinddn"), ",",
"RACFID");
ret.value = "RACFID=" + racfid + ",CN=ICTX";
]]></Script>
                    <Simple>ildapUsername</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>jmsBroker</Name>
                    <Type>advanced</Type>
                    <Script>return jmsBroker;</Script>
                    <Simple>jmsBroker</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>jmsQueue</Name>
                    <Type>advanced</Type>
                    <Script>return jmsQueue;</Script>
                    <Simple>jmsQueue</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>jmsQueueManager</Name>
                    <Type>advanced</Type>
                    <Script>return jmsQueueManager;</Script>
                    <Simple>jmsQueueManager</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>jmsServerChannel</Name>
                    <Type>advanced</Type>
                    <Script>return jmsServerChannel;</Script>
                    <Simple>jmsServerChannel</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>sldapPassword</Name>
                    <Type>advanced</Type>
                    <Enabled>true</Enabled>
                    <Add>true</Add>
                    <Modify>true</Modify>
                    <Script><![CDATA[cat ( hookctx() + "map_sldapPassword", 3 );
ret.value = sdbmEntry.getString("secretkey");;]]></Script>
                    <Simple>rldapPassword</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>sldapSearchBase</Name>
```

```
                    <Type>advanced</Type>
                    <Script><![CDATA[cat ( hookctx() + "map_slapSearchBase", 4 );
return sdbmEntry.getString("ibm-syncsuffix");]]></Script>
                    <Simple>sldapSearchBase</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>sldapUsername</Name>
                    <Type>advanced</Type>
                    <Enabled>true</Enabled>
                    <Add>true</Add>
                    <Modify>true</Modify>
                    <Script><![CDATA[cat ( hookctx() + "map_sldapUsername", 3 );
ret.value = sdbmEntry.getString("ibm-syncbinddn");]]></Script>
                    <Simple>rldapUsername</Simple>
                </AttributeMapItem>
                <AttributeMapItem>
                    <Name>targetList</Name>
                    <Type>advanced</Type>
                    <Script><![CDATA[// placeholder set in After GetNext
conn.targetList]]></Script>
                    <Simple>targetList</Simple>
                </AttributeMapItem>
            </AttributeMap>
            <AttributeMap name="Output">
                <InheritFrom>[parent]</InheritFrom>
            </AttributeMap>
            <DeltaSettings>
                <UniqueAttribute/>
                <WhenToCommit>After every database operation</WhenToCommit>
                <Driver>CloudScape</Driver>
                <RowLocking>SERIALIZABLE</RowLocking>
                <ChangeDetectionMode>DETECT_ALL</ChangeDetectionMode>
            </DeltaSettings>
            <Schema name="Input">
                <InheritFrom>[parent]</InheritFrom>
                <SchemaItem>
                    <Name>$dn</Name>
                    <Syntax>java.lang.String</Syntax>
                </SchemaItem>
                <SchemaItem>
                    <Name>dc</Name>
                    <Syntax>java.lang.String</Syntax>
                   <NativeSyntax>MUST/Directory String/Specifies one component of
a domain name.</NativeSyntax>
                </SchemaItem>
                <SchemaItem>
                    <Name>objectclass</Name>
                    <Syntax>java.lang.String</Syntax>
                </SchemaItem>
            </Schema>
            <Schema name="Output">
                <InheritFrom>[parent]</InheritFrom>
                <SchemaItem>
                    <Name>$dn</Name>
                    <Syntax>java.lang.String</Syntax>
```

```
                    </SchemaItem>
                    <SchemaItem>
                        <Name>dc</Name>
                        <Syntax>java.lang.String</Syntax>
                    </SchemaItem>
                    <SchemaItem>
                        <Name>objectclass</Name>
                        <Syntax>java.lang.String</Syntax>
                    </SchemaItem>
                </Schema>
                <LinkCriteria>
                    <InheritFrom>[parent]</InheritFrom>
                </LinkCriteria>
                <Hooks>
                    <InheritFrom>[parent]</InheritFrom>
                    <Hook name="after_getnext">
                        <InheritFrom>[no inheritance]</InheritFrom>
                        <Name>after_getnext</Name>
                        <Script><![CDATA[/*
The connector returns the DC=<sysname> from sysRegsitry entry.
Now we use connector to get the ibm-syncnodes (host)
and ibm-synchnodebackend (gdbm and sdbm) entries.
*/
numGet = task.getStats().numGet() + 1;
cat("",2);
cat("readSysRegistry: Iteration " + numGet, 4);

var EQUALS = Packages.com.ibm.di.server.SearchCriteria.EXACT;

var NodeHostname = conn.getString("dc");
var searchbase = conn.getString("$dn");
catt ("processing hostname " + NodeHostname, 2 );

// get host connection information
var searchcrit = system.newSearchCriteria();
searchcrit.addCriteria( "$dn", EQUALS, "cn=host," + searchbase );
var hostEntry = thisConnector.getConnector().findEntry(searchcrit);

// get gbdm information
var searchcrit = system.newSearchCriteria();
searchcrit.addCriteria( "$dn", EQUALS, "cn=gdbm," + searchbase );
var gdbmEntry = thisConnector.getConnector().findEntry(searchcrit);

// get sbdm information
var searchcrit = system.newSearchCriteria();
searchcrit.addCriteria( "$dn", EQUALS, "cn=sdbm," + searchbase );
var sdbmEntry = thisConnector.getConnector().findEntry(searchcrit);

//set targetList (MQ messages with headers=target will be queued)
//If sychToAllTargets = true then all nodes with isTargetSystem=true host
attribute are valid targets
//otherwise
//targetList is specified by ibm-hasthesetargets multi valued attribute in source
host
var isSource = hostEntry.getString("ibm-issourcesystem");
```

```
if (isSource == !null || isSource.equalsIgnoreCase("TRUE")) {
   if (synchToAll_Targets.equalsIgnoreCase("TRUE")) {
      var targetList = Get_Targets( "hostname" );
      conn.setAttribute("targetList", targetList);
      }
   else {
      var targetList = hostEntry.getAttribute("ibm-hasthesetargets");
      // remove whitespace and replace value in targetList
      for (var i=0; i<targetList.size(); i++) {
         var tmpTarget = targetList.getValue(i);
         var strTarget = tmpTarget.toString();
         strTarget = strTarget.replace(/^\s+|\s+$/g,'');
         targetList.setValue(i,strTarget);
         }
      conn.setAttribute("targetList", targetList);
      }
   cat("targetList " + targetList);
}]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="after_initialize">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>after_initialize</Name>
                   <Script><![CDATA[cat( hookctx() + ": Connector initialized", 1
);
//thisConnector.getConnector().showServerInfo();

ctx = thisConnector.getConnector().getLdapContext();
env = ctx.getEnvironment();

cat("These are LDAP environment variables:", 4)
envenum = env.elements();
while ( envenum.hasMoreElements()) {
   cat(envenum.nextElement(), 4);
}]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="before_execute">
                    <Name>before_execute</Name>
                    <Script/>
                    <Enabled>false</Enabled>
                </Hook>
                <Hook name="before_initialize">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>before_initialize</Name>
                    <Script><![CDATA[cat( hookctx() + "Before Initialize: Setting
Parameters", 2 );
/* variables (from properties) set in global_Prolog */
/*
sysRegistry_HostName = SysRegistryHostName
sysRegistry_SearchBase = SysRegistrySearchBase
sysRegistryUse_SASL = SysRegistryUseSASL
sysRegistryUse_SASLdn = SysRegistryUseSASLdn
sysRegistry_BindDN = SysRegsitryBindDN
sysRegsitry_BindPW = SysRegistryBindPW
```

```
sysRegistry_SSLport = SysRegistrySSLport
*/

if ((sysRegistryUse_SASL.toUpperCase()) == "TRUE") {
   /* Set SASL Parameters */
   var longUrl = "ldaps://" + sysRegistry_HostName + ":" + sysRegistry_SSLport;
   thisConnector.connector.setParam("ldapUrl", longUrl);
   thisConnector.connector.setParam("ldapSearchBase", sysRegistry_SearchBase);
   thisConnector.connector.setParam("ldapAuthenticationMethod", "EXTERNAL");
   thisConnector.connector.setParam("ldapUsername", sysRegistryUse_SASLdn);
   thisConnector.connector.setParam("jndiExtraProviderParams",
"SECURITY_AUTHENTICATION:EXTERNAL");
    thisConnector.connector.setParam("ldapPassword", "");
   }
else {
   /* Set SSL Parameters */
   var longUrl = "ldaps://" + sysRegistry_HostName + ":" + sysRegistry_SSLport;
   thisConnector.connector.setParam("ldapUrl", longUrl);
   thisConnector.connector.setParam("ldapSearchBase", sysRegistry_SearchBase);
   thisConnector.connector.setParam("ldapAuthenticationMethod", "simple");
   thisConnector.connector.setParam("ldapUsername", sysRegistry_BindDN);
   thisConnector.connector.setParam("ldapPassword", sysRegistry_BindPW);
   }

cat("readSysRegistry ldapUrl: " + thisConnector.connector.getParam("ldapUrl"));
cat("readSysRegistry ldapUsername: " +
thisConnector.connector.getParam("ldapUsername"));
cat("readSysRegistry ldapPassword: " +
thisConnector.connector.getParam("ldapPassword"));
cat("readSysRegistry ldapAuthenticationMethod: " +
thisConnector.connector.getParam("ldapAuthenticationMethod"));
cat("readSysRegistry jndiExtraProviderParams: " +
thisConnector.connector.getParam("jndiExtraProviderParams"));
cat("readSysRegistry ldapSearchBase: " +
thisConnector.connector.getParam("ldapSearchBase"));
cat("readSysRegistry ldapSearchFilter: " +
thisConnector.connector.getParam("ldapSearchFilter"));
cat("readSysRegistry ldapSearchScope: " +
thisConnector.connector.getParam("ldapSearchScope"));]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="default_fail">
                    <Name>default_fail</Name>
                    <Script>Error_Handler();</Script>
                    <Enabled>true</Enabled>
                </Hook>
                <Hook name="default_ok">
                    <Name>default_ok</Name>
                    <Script/>
                    <Enabled>false</Enabled>
                </Hook>
                <Hook name="get_ok">
                    <InheritFrom>[no inheritance]</InheritFrom>
                    <Name>get_ok</Name>
```

```
                    <Script><![CDATA[cat( hookctx() + "GetNextSuccessful work:",
3);
cat( work, 3);]]></Script>
                    <Enabled>true</Enabled>
                </Hook>
            </Hooks>
            <CheckpointConfig/>
            <SandboxConfig/>
            <Reconnect>
                <InheritFrom>[parent]</InheritFrom>
                <ReconnectRules/>
            </Reconnect>
            <Operations/>
            <PoolDefinition>
                <InheritFrom>[parent]</InheritFrom>
                <MaxPoolSize>0</MaxPoolSize>
                <InitializeSleepInterval>0</InitializeSleepInterval>
            </PoolDefinition>
            <PoolInstance>
                <Enabled>false</Enabled>
                <ExhaustedPoolBehavior>Wait</ExhaustedPoolBehavior>
            </PoolInstance>
            <InitializeOption>0</InitializeOption>
        </Connector>
        <Connector name="LDAP_template">
            <InheritFrom>system:/Connectors/ibmdi.LDAP</InheritFrom>
            <ModTime>1361369518885</ModTime>
            <ConnectorMode>Iterator</ConnectorMode>
            <ConnectorState>Enabled</ConnectorState>
            <Configuration>
                <InheritFrom>[parent]</InheritFrom>
                <parameter name="automapADPassword">false</parameter>
                <parameter name="debug">false</parameter>
                <parameter name="ldapAddAttr">false</parameter>
                <parameter name="ldapAuthenticationMethod">Simple</parameter>
                <parameter name="ldapPageSize">0</parameter>
                <parameter name="ldapPassword">itdi</parameter>
                <parameter name="ldapReferrals">follow</parameter>
                <parameter name="ldapSearchBase">CN=RACFVM</parameter>
                <parameter name="ldapSearchFilter">racfid=*</parameter>
                <parameter name="ldapSearchScope">subtree</parameter>
                <parameter name="ldapSizeLimit">0</parameter>
                <parameter name="ldapTimeLimit">0</parameter>
                <parameter name="ldapUrl">ldap://9.12.4.129:389</parameter>
                <parameter name="ldapUseSSL">false</parameter>
                <parameter
name="ldapUsername">racfid=itdi,profiletype=user,cn=RACFVM</parameter>
                <parameter name="ldapVLVPageSize">0</parameter>
                <parameter name="simulateRename">false</parameter>
            </Configuration>
            <Parser>
                <InheritFrom>[parent]</InheritFrom>
                <Schema name="Input">
                    <InheritFrom>[parent]</InheritFrom>
                </Schema>
```

```
    <Schema name="Output">
        <InheritFrom>[parent]</InheritFrom>
    </Schema>
</Parser>
<AttributeMap name="Input">
    <InheritFrom>[parent]</InheritFrom>
</AttributeMap>
<AttributeMap name="Output">
    <InheritFrom>[parent]</InheritFrom>
</AttributeMap>
<DeltaSettings>
    <WhenToCommit>After every database operation</WhenToCommit>
    <Driver>CloudScape</Driver>
</DeltaSettings>
<Schema name="Input">
    <InheritFrom>[parent]</InheritFrom>
</Schema>
<Schema name="Output">
    <InheritFrom>[parent]</InheritFrom>
</Schema>
<LinkCriteria>
    <InheritFrom>[parent]</InheritFrom>
</LinkCriteria>
<Hooks>
    <InheritFrom>[parent]</InheritFrom>
    <Hook name="after_initialize">
        <Name>after_initialize</Name>
        <Script>Init_Hook( "AfterInitialize");</Script>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="before_initialize">
        <Name>before_initialize</Name>
        <Script>Init_Hook( "BeforeInitialize");</Script>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="connect_init">
        <Name>connect_init</Name>
        <Script>Critical_Error_Handler();</Script>
        <Enabled>true</Enabled>
    </Hook>
    <Hook name="initialize_fail">
        <Name>initialize_fail</Name>
        <Script>Critical_Error_Handler();</Script>
        <Enabled>true</Enabled>
    </Hook>
</Hooks>
<CheckpointConfig/>
<SandboxConfig/>
<Reconnect>
    <InheritFrom>[parent]</InheritFrom>
    <ReconnectRules/>
</Reconnect>
<Operations/>
<PoolDefinition>
    <InheritFrom>[parent]</InheritFrom>
```

```
                <Enabled>false</Enabled>
                <MinPoolSize>0</MinPoolSize>
                <MaxPoolSize>4</MaxPoolSize>
                <PurgeInterval>0</PurgeInterval>
                <InitializeAttempts>1</InitializeAttempts>
            </PoolDefinition>
            <PoolInstance/>
            <InitializeOption>0</InitializeOption>
        </Connector>
    </Folder>
    <Folder name="Parsers"/>
    <Folder name="Scripts">
        <Script name="get_targets">
            <ModTime>1362794916939</ModTime>
            <parameter name="autoInclude">true</parameter>
            <parameter name="includeFiles"/>
      <parameter name="script"><![CDATA[/*
*
 *--------------------------------------------------------------------------*
 * DISCLAIMER OF WARRANTIES:                                                *
 *                                                                          *
 *   The following enclosed code is sample code created by IBM              *
 *   Corporation.  This sample code is not part of any standard IBM         *
 *   product and is provided to you solely for the purpose of assisting     *
 *   you in the development of your applications.  The code is provided     *
 *   "AS IS", without warranty of any kind.  IBM shall not be liable        *
 *   for any damages arising out of your use of the sample code, even       *
 *   if they have been advised of the possibility of such damages.          *
 *                                                                          *
 *--------------------------------------------------------------------------*
 *                                                                          */
/*
   This JNDI code checks the cn=host child of each node and
   adds the node to the list of targets if the cn=host ibm-isTargetSystem
attribute is TRUE
*/
function Get_Targets( sourcehost ) {
   cat ("Entering function Get_Targets( sourcehost )", 3);
   var searchbase = thisConnector.getConnectorParam("ldapSearchBase")
   catt ("searchbase is " + searchbase, 3);

   var JNDI = Packages.javax.naming;
   var ctx = thisConnector.getConnector().getLdapContext();
   var targetList = system.newAttribute("targetList");
   var ctls = new JNDI.directory.SearchControls();
   ctls.setSearchScope(JNDI.directory.SearchControls.ONELEVEL_SCOPE);
   var nodefilt = java.lang.String( "(objectclass=domain)" );
   var targfilt = java.lang.String(
"(&(objectclass=ibmSyncNode)(ibm-isTargetSystem=TRUE))" );

   var node, istarg, nodename, err;
   try {
      var nodes = ctx.search(searchbase, nodefilt, ctls);
      while (nodes.hasMore()) {
         node = nodes.next();
```

```
         istarg = ctx.search(node.getName() + "," + searchbase, targfilt, ctls);
         if (istarg.hasMore() == true) {
            nodeName = node.getAttributes().get("dc").get();
            catt ( nodeName + " is a target", 3 );
            if ( !nodeName.equalsIgnoreCase(sourcehost) )
               targetList.addValue( nodeName );
         }
       }
   } catch(err) {
      cat (err);
   }

   cat ("Exiting function Get_Targets( sourcehost )", 3);
   return targetList;
}]]></parameter>
        </Script>
        <Script name="common_connector_hooks">
            <ModTime>1362794916933</ModTime>
            <parameter name="autoInclude">true</parameter>
            <parameter name="includeFiles"/>
      <parameter name="script"><![CDATA[/*
*
 *-------------------------------------------------------------------------*
 * DISCLAIMER OF WARRANTIES:                                               *
 *                                                                         *
 *   The following enclosed code is sample code created by IBM             *
 *   Corporation.  This sample code is not part of any standard IBM        *
 *   product and is provided to you solely for the purpose of assisting    *
 *   you in the development of your applications.  The code is provided    *
 *   "AS IS", without warranty of any kind.  IBM shall not be liable       *
 *   for any damages arising out of your use of the sample code, even      *
 *   if they have been advised of the possibility of such damages.         *
 *                                                                         *
 *-------------------------------------------------------------------------*
 *                                                                         */
//*****************************************
/*
   Initialization hooks
   Purpose: Debug logging
   Used: Every Connector
*/
function Init_Hook( hookname ) {
   cat( hookctx(hostname, false) + hookname, 2 );
   if (hookname.equals("BeforeInitialize"))
      dump_connector_params( 3 );
}

//*****************************************
function Error_Handler() {
  cat( hookctx(hostname, false) + "Error: " + error.getString("message"), 1 );
  cat( error, 3 );
  cat( "Skipping entry due to exception", 1 );
  system.skipEntry();
}
```

```
//*****************************************
function Critical_Error_Handler() {
   cat( hookctx(hostname, false) + "Error: " + error.getString("message"), 1 );
   cat( error, 2 );
   cat( "Critical Error. Terminating AssemblyLine", 1 );
   task.shutdown();
}

//*****************************************
/*
   Simple Connector Hooks
   Purpose: Debug logging and branch control
   Used: Update Connectors, BeforeModify, OnNoChanges, AfterModify, Default
Success Hooks
*/
function Update_BeforeModify() {
   cat( hookctx(hostname, false) + "Before_Modify: This is conn for " +
work.getString("$dn") + " :", 3 );
   cat( conn, 3 );
}

function Update_OnNoChanges() {
   cat( hookctx(hostname, false) + "On_No_Changes: No updates to " +
work.getString("$dn"), 3 );
}

function Update_AfterModify() {
   cat( hookctx(hostname, false) + "AfterModify: Modified " +
work.getString("$dn"), 3 );
}

function Update_DefaultSuccess() {
   cat ( hookctx(hostname, false) + "DefaultSuccess: Processed " +
work.getString("$dn"), 1);
   system.exitBranch();
}

function On_Connection_Lost() {
   cat ( hookctx(hostname, false) + "OnConnectionLost", 1);
}

//*****************************************
// dump connector params for deep debugging
function dump_connector_params( logdepth ) {
   cat( "ldapUrl: " + thisConnector.getConnectorParam("ldapUrl"), logdepth );
   cat( "ldapUsername: " + thisConnector.getConnectorParam("ldapUsername"),
logdepth );
   cat( "ldapPassword: " + thisConnector.getConnectorParam("ldapPassword"),
logdepth );
   cat( "ldapUseSSL: " + thisConnector.getConnectorParam("ldapUseSSL"), logdepth
);
}]]></parameter>
        </Script>
        <Script name="hookctx">
            <ModTime>1362794916841</ModTime>
```

```
            <parameter name="autoInclude">true</parameter>
            <parameter name="includeFiles"/>
        <parameter name="script"><![CDATA[/*
*
 *-------------------------------------------------------------------------*
 * DISCLAIMER OF WARRANTIES:                                               *
 *                                                                         *
 *   The following enclosed code is sample code created by IBM             *
 *   Corporation.  This sample code is not part of any standard IBM        *
 *   product and is provided to you solely for the purpose of assisting    *
 *   you in the development of your applications.  The code is provided    *
 *   "AS IS", without warranty of any kind.  IBM shall not be liable       *
 *   for any damages arising out of your use of the sample code, even      *
 *   if they have been advised of the possibility of such damages.         *
 *                                                                         *
 *-------------------------------------------------------------------------*
 *                                                                         */

// hookctx (hook context) JavaScript function
//print useful information to log files
function hookctx ( /* optional string */ prefix, /* optional boolean */ printAL )
{

   var SEPARATOR = " ";

   // put any useful debugging code or name in prefix; e.g. target hostname
   if (!prefix) {
      var prefix = "";
   } else {
      prefix += " ";
   }

   if (!printAL)
      var ALname = "";
   else
      var ALname = task.getShortName() + " ";

   try {
      var componentName = thisConnector.getName();
      return ( prefix + ALname + "[" + componentName + "]" + SEPARATOR);
   } catch(e) {
      return ( prefix + ALname +  "[" + task.getCurrentState()  + "]" +
SEPARATOR);
   }
}]]></parameter>
        </Script>
        <Script name="ictx functions">
            <ModTime>1362858268583</ModTime>
            <parameter name="autoInclude">true</parameter>
            <parameter name="includeFiles"/>
        <parameter name="script"><![CDATA[/*
*
 *-------------------------------------------------------------------------*
 * DISCLAIMER OF WARRANTIES:                                               *
 *                                                                         *
```

```
 *   The following enclosed code is sample code created by IBM          *
 *   Corporation.  This sample code is not part of any standard IBM     *
 *   product and is provided to you solely for the purpose of assisting  *
 *   you in the development of your applications.  The code is provided  *
 *   "AS IS", without warranty of any kind.  IBM shall not be liable    *
 *   for any damages arising out of your use of the sample code, even    *
 *   if they have been advised of the possibility of such damages.       *
 *                                                                       *
 *-----------------------------------------------------------------------*
 *                                                                       */
// ICTX helper functions

// given a target name, get its index
function gettargindex(target) {
   var tmap = java.util.HashMap();
   for (var i=0; i<targets.length; i++) {
      tmap.put(targets[i], i);
   }
   return( tmap.get(target) );
}

// given a resource name, get its index
function getresindex(target) {
   var rmap = java.util.HashMap();
   for (var i=0; i<ResSuf.length; i++) {
      rmap.put(ResSuf[i], i);
   }
   return( rmap.get(target) );
}

function calcTagFromNames( target, resource ) {
   tindex = gettargindex( target );
   rindex = getresindex( resource );
   tagval = calctag( tindex, tindex);
   return( tagval );
}


// generate a 6-digit number to uniquely identify each request item tag
function calctag(targindex, resindex) {
   var targetValue = targetRange + 1000 * ( targindex + 1 );
   var resourceValue = resourceRange + ( resindex + 1 );
   return java.lang.Integer(targetValue + resourceValue);
}

// decode the tag
function decodetag( tag ) {
   var bv = tag - targetRange - resourceRange;
   var rv = bv % 1000 - 1;
   var tv = (bv - rv)/1000 - 1;
   target = targets[tv];
   resource = ResSuf[rv];
   return(target + "."  + resource);
}
```

```
// get value of a target & resource combination
function getItemAccess( target, resource, /* optional */ ICTXresponse ) {
   cat("function getItemAccess: target = " + target, 3);
   cat("function getItemAccess: resource = " + resource, 3);
   tindex = gettargindex( target );
   cat("function getItemAccess: tindex = " + tindex, 3);
   rindex = getresindex( resource );
   cat("function getItemAccess: rindex = " + rindex, 3);
   tagval = calctag( tindex, rindex);
   cat("function getItemAccess: tagval = " + tagval, 3);
   if (!ICTXresponse) {
      cat("function getItemAccess: no ICTXresponse supplied", 3);
      var ICTXresponse = work.getObject("ICTXresponse");
   }
   MajorCode = ICTXresponse.getItem( tagval, "MajorCode" );
   cat("function getItemAccess: MajorCode = " + MajorCode, 3);
   if ( MajorCode == 0 ) {
      return true;
   } else {
      return false;
   }
}

// test whether to perform an update, given a specific changetype
function testAction( target, resource ) {
   access = getItemAccess( target, resource );
   changetype = work.getString("changetype");
   if ( access==true && resource.equalsIgnoreCase(changetype) ) {
      return true;
   } else {
      return false;
   }
}

// test whether to perform an update for this changetype for any target other than
this source
// called by IF components in SyncAny
function testActionForTargets( resource ) {
   var doaction = false;
   for (var i=0; i<targets.length; i++ ) {
      if ( targets[i].equalsIgnoreCase(sourceHostname) ) {
         continue;
      }
      doaction = testAction( targets[i], resource );
   }
   return doaction;
}]]></parameter>
       </Script>
       <Script name="cat">
           <ModTime>1362794916926</ModTime>
           <parameter name="autoInclude">true</parameter>
           <parameter name="includeFiles"/>
     <parameter name="script"><![CDATA[/*
*
 *-------------------------------------------------------------------------*
```

```
 * DISCLAIMER OF WARRANTIES:                                              *
 *                                                                        *
 *   The following enclosed code is sample code created by IBM            *
 *   Corporation.  This sample code is not part of any standard IBM       *
 *   product and is provided to you solely for the purpose of assisting   *
 *   you in the development of your applications.  The code is provided   *
 *   "AS IS", without warranty of any kind.  IBM shall not be liable      *
 *   for any damages arising out of your use of the sample code, even     *
 *   if they have been advised of the possibility of such damages.        *
 *                                                                        *
 *------------------------------------------------------------------------*
 *                                                                        */

/*
   Function cat prints strings or dump objects to the console and/or logs

   LOGDEPTH must be defined globally. Do this in each AssemblyLine Prolog or
below.
   It can take any non-negative integer. Zero suppresses all logging. One is the
   ordinary default. Higher valuesprint objects whose depth equals that value or
lower.

   In the function cat, the default value of the optional argument 'depth' is one
(1).
   Depth can be any positive integer. Higher values suppress printing of the first
argument
   unless LOGDEPTH matches or exceeds 'depth'.

   Standard values for LOGDEPTH:
   0 no logging
   1 production logging
   2 basic debug logging
   3 heavy debug logging

*/


LOGDEPTH = 1;// override in global_Prolog or AL prologs
LOGLEVEL = "INFO";// override in global_Prolog or AL prologs
function cat ( arg, /* optional argument */ depth, /* optional argument */
loglevel ) {
   if (!depth) var depth = 1;
   if (!loglevel) var loglevel = LOGLEVEL;
   if ( LOGDEPTH >= depth ) {
      try {
         if ( typeof arg == "string" || arg.getClass() == "class java.lang.String"
) {
            task.logmsg( loglevel, arg );
         } else {
            task.dump( arg );
         }
      } catch (e) {
         task.dump( arg );  // dump() calls dumpEntry() for TDI entry objects
      }
   }
```

```
}

function catt ( arg, /* optional argument */ depth, /* optional argument */
loglevel ) {
   if (!depth) var depth = 1;
   if (!loglevel) var loglevel = LOGLEVEL;
   cat( "\t" + arg, depth, loglevel );
}]]></parameter>
        </Script>
        <Script name="global_Prolog">
            <ModTime>1362794916827</ModTime>
            <parameter name="autoInclude">true</parameter>
            <parameter name="includeFiles"/>
       <parameter name="script"><![CDATA[/*
*
 *--------------------------------------------------------------------------*
 * DISCLAIMER OF WARRANTIES:                                                *
 *                                                                          *
 *   The following enclosed code is sample code created by IBM              *
 *   Corporation.  This sample code is not part of any standard IBM         *
 *   product and is provided to you solely for the purpose of assisting     *
 *   you in the development of your applications.  The code is provided     *
 *   "AS IS", without warranty of any kind.  IBM shall not be liable        *
 *   for any damages arising out of your use of the sample code, even       *
 *   if they have been advised of the possibility of such damages.          *
 *                                                                          *
 *--------------------------------------------------------------------------*
 *                                                                          */
task.logmsg("Processing Global Prolog");
/*
   Global settings for all AssemblyLines
   Customize this prolog to set local defaults and configure diagnostic logging
*/

// Various Path names
SEP = system.getJavaProperty("file.separator");
SOLUTIONDIR = system.getJavaProperty("user.dir");
//PROPSDIR = system.getTDIProperty("PropertiesFolder");
//PROPSPATH = SOLUTIONDIR + SEP + PROPSDIR + SEP;
CRYPTODIR = system.getTDIProperty("CryptoFolder");
CRYPTOPATH = SOLUTIONDIR + SEP + CRYPTODIR + SEP;

// Java key store and parameters used for RACF password envelope decryption
KEYSTORE = CRYPTOPATH + system.getTDIProperty("EnvelopeKeyStore");
KEYSTORE_PASSWD = system.getTDIProperty("EnvelopeKSPassword");
TDICERT_ALIAS = system.getTDIProperty("TDICertAlias");
TDICERT_PASSWD = system.getTDIProperty("TDICertPassword");

// Get RACF Cert Info
useCommonRACF_Cert = system.getTDIProperty("UseCommonRACFCert");
commonRACFCert_Label = system.getTDIProperty("CommonRACFCertLabel");

//task.logmsg( "global variable KEYSTORE = " + KEYSTORE);
//task.logmsg( "global variable KEYSTORE_PASSWD = " + KEYSTORE_PASSWD);
//task.logmsg( "global variable TDICERT_ALIAS = " + TDICERT_ALIAS);
```

```
//task.logmsg( "global variable TDICERT_PASSWD = " + TDICERT_PASSWD);

// Get MQ info
jmsBroker = system.getTDIProperty("MQjmsBroker");
jmsServerChannel = system.getTDIProperty("MQjmsServerChannel");
jmsQueueManager = system.getTDIProperty("MQjmsQueueManager");
jmsQueue = system.getTDIProperty("MQjmsQueue");

//sysRegistry data
sysRegistry_HostName = system.getTDIProperty("SysRegistryHostName");
sysRegistry_SearchBase = system.getTDIProperty("SysRegistrySearchBase");
sysRegistryUse_SASL = system.getTDIProperty("SysRegistryUseSASL");
sysRegistryUse_SASLdn = system.getTDIProperty("SysRegistryUseSASLdn");
sysRegistry_BindDN = system.getTDIProperty("SysRegistryBindDN");
sysRegistry_BindPW = system.getTDIProperty("SysRegistryBindPW");
sysRegistry_SSLport = system.getTDIProperty("SysRegistrySSLport");

//set targets
//if synchToALL_Targets is true then targetList = isTargetSystem=true nodes
//otherwise use hasTheseTargets from host entry
synchToAll_Targets = system.getTDIProperty("SynchToAllTargets");

// data used by ICTX helper functions
ResSuf = [ "PW", "PP", "USER" ];
targetRange = 700000;
resourceRange = 500;

// used in ICTX lookup.  facilityClass_Prefix + "." + target + resSuf[]
facilityClass_Prefix = system.getTDIProperty("FacilityClassPrefix");

// racfid of itdi process (ldap logins in SyncAny AL)
// ITDIUSER = system.getTDIProperty("ITDIUser");


// Configure flexible logging
// Uncomment only one of these; TDI default is INFO
//LOGLEVEL = "DEBUG"
LOGLEVEL = "INFO";
// See comments in cat script (script library) for details on LOGDEPTH
// value in AL prologs override this value
LOGDEPTH = 1;]]></parameter>
        </Script>
        <Script name="processPass">
            <ModTime>1362794916834</ModTime>
            <parameter name="autoInclude">true</parameter>
            <parameter name="includeFiles"/>
      <parameter name="script"><![CDATA[/*
*
 *--------------------------------------------------------------------*
 * DISCLAIMER OF WARRANTIES:                                          *
 *                                                                    *
 *   The following enclosed code is sample code created by IBM        *
 *   Corporation.  This sample code is not part of any standard IBM   *
 *   product and is provided to you solely for the purpose of assisting  *
 *   you in the development of your applications.  The code is provided  *
```

```
 *   "AS IS", without warranty of any kind.  IBM shall not be liable     *
 *   for any damages arising out of your use of the sample code, even    *
 *   if they have been advised of the possibility of such damages.       *
 *                                                                       *
 *-----------------------------------------------------------------------*
 *                                                                       */
//****************************************
/*
   Purpose: Get RACFCredential and extract the cred
   Used: Used: AL(getEntryFromSource), Connector(lookupAttributes), After Lookup
hook
*/

function GetCred(sysname) {
   cat( "Entering function GetCred(sysname)", 4);
   cat( "syname = " + sysname, 4);
   if ( work.getString("changetype").equals("PP") ) {
      pwbytes = conn.getObject("racfpassphraseenvelope");
   } else if ( work.getString("changetype").equals("PW")) {
      pwbytes = conn.getObject("racfpasswordenvelope");
   } else {
      cat("error in GetCred");
   }
   racfpwobj = processPass( sysname, pwbytes );
   if ( racfpwobj.getExpired() == false ) {
      conn.setAttribute("racfAttributes", "noexpired");
      }
   else {
      conn.setAttribute("racfAttributes", null);
   }
   credString = racfpwobj.getCredentialString();
   cat( "Exiting function GetCred(sysname)", 4);
   return credString;
}


/*
   Process the racfPasswordEnvelope or racfPassphraseEnvelope and extract the
encrypted value.
*/

function processPass( source, pwbytes ) {
   cat( "Entering function processPass( source, pwbytes )", 4);
   cat( "source = " + source, 4);
   cat( "pwbytes = ", 4);
   cat( system.base64Encode(pwbytes), 4);

   if ( useCommonRACF_Cert == null || useCommonRACF_Cert.equalsIgnoreCase("FALSE")
) {
      var RACFCERT_ALIAS = source;
      }
   else {
      var RACFCERT_ALIAS = commonRACFCert_Label;
   }
```

```
   try {
      var racfpwobj = system.getRacfCredentialObject (
         pwbytes,
         KEYSTORE, KEYSTORE_PASSWD,
         TDICERT_ALIAS, TDICERT_PASSWD,
         KEYSTORE, KEYSTORE_PASSWD, RACFCERT_ALIAS
         );
   } catch (e) {
      cat(hookctx() + "Function processPass: Error decrypting password or
passphrase envelope", 2);
      cat( "pwbytes: ", 3 );
      cat( system.base64Encode(pwbytes), 3 );
      cat( "KEYSTORE:\t\t" + KEYSTORE, 3 );
      cat( "KEYSTORE_PASSWD:\t" + KEYSTORE_PASSWD, 3 );
      cat( "TDICERT_ALIAS:\t" + TDICERT_ALIAS, 3 );
      cat( "TDICERT_PASSWD:\t" + TDICERT_PASSWD, 3 );
      cat( "RACFCERT_ALIAS:\t" + RACFCERT_ALIAS, 3);
      cat( "Exception: " + e, 3);
      system.skipEntry();
   }

   cat( hookctx() + "function processPass: This is the decrypted racfPassword or
racfPassphrase envelope: \n" +
     racfpwobj.dumpContents(), 3 );

   cat( "Exiting function processPass()", 4 );
   return( racfpwobj );

}]]></parameter>
        </Script>
    </Folder>
    <JavaLibraries/>
    <JavaProperties/>
    <Folder name="Includes"/>
    <Folder name="Config">
        <LogConfig name="Logging">
            <Logger name="IDIFileRoller">
                <parameter name="IDIFileRoller.File">logs/zRedbook.log</parameter>
                <parameter name="IDIFileRoller.RollCount">5</parameter>
                <parameter name="Pattern.ConversionPattern">%d{ISO8601} %-5p [%c]
- %m%n</parameter>
                <parameter
name="com.ibm.di.log.appender">IDIFileRoller</parameter>
                <parameter name="com.ibm.di.log.layout">Pattern</parameter>
                <parameter name="com.ibm.di.log.level">INFO</parameter>
                <parameter name="enabled">true</parameter>
            </Logger>
        </LogConfig>
        <InstanceProperties name="AutoStart">
            <AutoStart>
                <ParameterList name="startGetALs">
                    <parameter name="Name">AssemblyLines/startGetALs</parameter>
                </ParameterList>
                <ParameterList name="startPutALs">
                    <parameter name="Name">AssemblyLines/startPutALs</parameter>
```

```
            </ParameterList>
        </AutoStart>
    </InstanceProperties>
    <TombstonesConfig name="Tombstones">
        <ModTime>1363724114283</ModTime>
        <parameter name="AssemblyLines">false</parameter>
        <parameter name="Configuration">false</parameter>
        <parameter name="EventHandlers">false</parameter>
    </TombstonesConfig>
    <SolutionInterface name="SolutionInterface">
        <PollInterval>-1</PollInterval>
        <InstanceID>zRedbook</InstanceID>
        <enabled>true</enabled>
    </SolutionInterface>
    <Container name="SystemStore">
        <ParameterList name="Default">
            <parameter name="enabled">true</parameter>
        </ParameterList>
    </Container>
</Folder>
<Folder name="Functions">
    <Function name="callAL_template">
        <InheritFrom>system:/Functions/ibmdi.AssemblyLineFC</InheritFrom>
        <ModTime>1362797389918</ModTime>
        <ConnectorState>Passive</ConnectorState>
        <Schema name="Input">
            <InheritFrom>[parent]</InheritFrom>
            <SchemaItem>
                <Name>$dn</Name>
            </SchemaItem>
            <SchemaItem>
                <Name>ICTXresponse</Name>
            </SchemaItem>
            <SchemaItem>
                <Name>changetype</Name>
            </SchemaItem>
            <SchemaItem>
                <Name>objectclass</Name>
            </SchemaItem>
            <SchemaItem>
                <Name>racfPassphrase</Name>
            </SchemaItem>
            <SchemaItem>
                <Name>racfPassword</Name>
            </SchemaItem>
            <SchemaItem>
                <Name>racfattributes</Name>
            </SchemaItem>
            <SchemaItem>
                <Name>racfdefaultgroup</Name>
            </SchemaItem>
            <SchemaItem>
                <Name>racfid</Name>
            </SchemaItem>
            <SchemaItem>
```

```
            <Name>racflogondays</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racflogontime</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racfowner</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racfprogrammername</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>source</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>targetdn</Name>
        </SchemaItem>
    </Schema>
    <Schema name="Output">
        <InheritFrom>[parent]</InheritFrom>
        <SchemaItem>
            <Name>$dn</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>ICTXresponse</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>changetype</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>objectclass</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racfPassphrase</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racfPassword</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racfattributes</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racfdefaultgroup</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racfid</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racflogondays</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racflogontime</Name>
        </SchemaItem>
        <SchemaItem>
            <Name>racfowner</Name>
```

```
                    </SchemaItem>
                    <SchemaItem>
                        <Name>racfprogrammername</Name>
                    </SchemaItem>
                    <SchemaItem>
                        <Name>targetdn</Name>
                    </SchemaItem>
                </Schema>
                <Hooks>
                    <InheritFrom>[parent]</InheritFrom>
                    <Hook name="after_functioncall">
                        <InheritFrom>[no inheritance]</InheritFrom>
                        <Name>after_functioncall</Name>
                        <Script><![CDATA[//Add a delay to allow GetEntryFromSource AL
to initialize
//If TDI is on a low latency network, it is possible for the FC taskcontrolblock
//to be overwritten with new data (new iteration) before the previously spawned AL
//is properly initialized (the connectors will have the wrong ldapURL etc..)
system.sleep(1);]]></Script>
                        <Enabled>true</Enabled>
                    </Hook>
                    <Hook name="before_functioncall">
                        <InheritFrom>[no inheritance]</InheritFrom>
                        <Name>before_functioncall</Name>
                        <Script/>
                        <Enabled>false</Enabled>
                    </Hook>
                    <Hook name="default_fail">
                        <Name>default_fail</Name>
                        <Script>Critical_Error_Handler();</Script>
                        <Enabled>true</Enabled>
                    </Hook>
                    <Hook name="no_reply">
                        <Name>no_reply</Name>
                        <Script>Critical_Error_Handler();</Script>
                        <Enabled>true</Enabled>
                    </Hook>
                </Hooks>
                <Configuration>
                    <InheritFrom>[parent]</InheritFrom>
                    <parameter name="assemblyLine">SyncAny</parameter>
                    <parameter name="debug">false</parameter>
                    <parameter name="mode">1</parameter>
                    <parameter name="shareLog">false</parameter>
                    <parameter name="useTCBAttributes">false</parameter>
                </Configuration>
                <SandboxConfig/>
                <AttributeMap name="Input">
                    <InheritFrom>[parent]</InheritFrom>
                </AttributeMap>
                <AttributeMap name="Output">
                    <InheritFrom>[parent]</InheritFrom>
                </AttributeMap>
            </Function>
        </Folder>
```

```
    <Folder name="AttributeMaps"/>
    <Properties name="Properties">
        <Stores>
            <PropertyStore name="Solution-Properties">
                <Parser>
                    <Schema name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <Schema name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                </Parser>
                <RawConnector>
                    <InheritFrom>system:/Connectors/ibmdi.Properties</InheritFrom>
                    <parameter
name="collectionType">Solution-Properties</parameter>
                </RawConnector>
                <Key>key</Key>
                <Value>value</Value>
                <ReadOnly>false</ReadOnly>
                <InitialLoad>true</InitialLoad>
                <CacheTimeout>0</CacheTimeout>
            </PropertyStore>
            <PropertyStore name="Global-Properties">
                <Parser>
                    <Schema name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <Schema name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                </Parser>
                <RawConnector>
                    <InheritFrom>system:/Connectors/ibmdi.Properties</InheritFrom>
                    <parameter name="collectionType">Global-Properties</parameter>
                </RawConnector>
                <Key>key</Key>
                <Value>value</Value>
                <ReadOnly>false</ReadOnly>
                <InitialLoad>true</InitialLoad>
                <CacheTimeout>0</CacheTimeout>
            </PropertyStore>
            <PropertyStore name="System-Properties">
                <Parser>
                    <Schema name="Input">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                    <Schema name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                </Parser>
                <RawConnector>
                    <InheritFrom>system:/Connectors/ibmdi.Properties</InheritFrom>
                    <parameter name="collectionType">System-Properties</parameter>
                </RawConnector>
```

```
                    <Key>key</Key>
                    <Value>value</Value>
                    <ReadOnly>false</ReadOnly>
                    <InitialLoad>true</InitialLoad>
                    <CacheTimeout>0</CacheTimeout>
                </PropertyStore>
                <PropertyStore name="zRedbook">
                    <Parser>
                        <Schema name="Input">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                        <Schema name="Output">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                    </Parser>
                    <RawConnector>
                        <InheritFrom>system:/Connectors/ibmdi.Properties</InheritFrom>
                        <parameter
name="collection">@SUBSTITUTE{config.$directory}/zRedbook.properties</parameter>
                        <parameter name="collectionType">zRedbook</parameter>
                    </RawConnector>
                    <Key>key</Key>
                    <Value>value</Value>
                    <ReadOnly>false</ReadOnly>
                    <InitialLoad>true</InitialLoad>
                    <CacheTimeout>0</CacheTimeout>
                </PropertyStore>
                <PropertyStore name="RACF Redbook Properties">
                    <ModTime>1361382919812</ModTime>
                    <Parser>
                        <Schema name="Input">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                        <Schema name="Output">
                            <InheritFrom>[parent]</InheritFrom>
                        </Schema>
                    </Parser>
                    <RawConnector>
                        <InheritFrom>system:/Connectors/ibmdi.Properties</InheritFrom>
                        <parameter
name="collection">@SUBSTITUTE{config.$directory}/RACF Redbook
Properties.properties</parameter>
                        <parameter name="collectionType">Default</parameter>
                        <parameter name="keyAlias"/>
                    </RawConnector>
                    <Key>key</Key>
                    <Value>value</Value>
                    <ReadOnly>false</ReadOnly>
                    <InitialLoad>true</InitialLoad>
                    <CacheTimeout>0</CacheTimeout>
                </PropertyStore>
                <PropertyStore name="Java-Properties">
                    <Parser>
                        <Schema name="Input">
                            <InheritFrom>[parent]</InheritFrom>
```

```
                    </Schema>
                    <Schema name="Output">
                        <InheritFrom>[parent]</InheritFrom>
                    </Schema>
                </Parser>
                <RawConnector>
                    <InheritFrom>system:/Connectors/ibmdi.Properties</InheritFrom>
                    <parameter
name="collection">@SUBSTITUTE{config.$directory}/zRedbook.properties</parameter>
                    <parameter name="collectionType">Java-Properties</parameter>
                </RawConnector>
                <Key>key</Key>
                <Value>value</Value>
                <ReadOnly>false</ReadOnly>
                <InitialLoad>true</InitialLoad>
                <CacheTimeout>0</CacheTimeout>
            </PropertyStore>
        </Stores>
    </Properties>
    <Folder name="Schedules"/>
    <Folder name="Sequences"/>
</MetamergeConfig>
```

Example: A-2 Sample properties file

```
##{PropertiesConnector} savedBy=karansin, saveDate=Wed Mar 06 15:52:20 EST 2013
CryptoFolder=crypto
{protect}-EnvelopeKSPassword={encr}dGJvvJDeYSTghbEho1CCjWX38235L6YzQCBKHs1qs8y62xi
OO7pZWmaMrNIdOOBpUuV71pxLDqTPeH2uE8Hn/GM62BU/qOO7adQjobTb9FZCpl5sUtqQcePIigRjXJF3S
LYdp54xIl1s7OvlPRlZ38qCAxOv7Jfa+DkrAnIzsOs=
EnvelopeKeyStore=racf_envelope_keystore.jks
SysRegistrySearchBase=ou=node,ou=sync,o=ibm
TDICertAlias=itdi
{protect}-TDICertPassword={encr}U4wdZOXwpWnaGajx5FoqXHTwjqQQ/TmkHnwXoGVEjbA1j+rjqH
jVih8oVtx+Z1IaD1ayzJh8D562oU48H/j4MtnXH8IQ4j+Y1Md/Rlgd5479OcPMhy8ODItIYehGiWFBHSCD
WuZcZdY5wINd/lhomzhphoEolE9N5R4JoMJfIQw=
MQjmsBroker=wtsc58.itso.ibm.com:1610
MQjmsQueue=ITDI
MQjmsQueueManager=MQ2B
MQjmsServerChannel=ITDISERV
SysRegistryHostName=wtsc58.itso.ibm.com
SysRegistryBindDN=cn=itdi,o=ibm
SysRegistryUseSASL=FALSE
SysRegistrySSLport=636
{protect}-SysRegistryBindPW={encr}VSXI+kuFDIlaKZkrMc7zE5XaT/C9OX4VlJ5Wr9dSW/CZYoy8
Bdq/fyGf+kUK3EMEI97lFAuLGTVxZPUmbWmcEiJxE4zzExWz2zJFhmv1hiaQzL1AsF64Onl3Acd2AugaW6
dfroYcyxUlyq3n5W/5l6sdOwPqButHcTcfIahIDuU=
SysRegistryUseSASLdn=cn=itdi,o=ibm
FacilityClassPrefix=ITDI
UseCommonRACFCert=TRUE
CommonRACFCertLabel=commonRACFCert
SynchToAllTargets=FALSE
```

B

RACF: LDAP field mappings

This appendix lists the mappings between Resource Access Control Facility (RACF) and Lightweight Directory Access Protocol (LDAP) database fields.

User field mapping

Table B-1 displays the mappings between RACF and LDAP. The Synchronize column indicates whether the data can be synchronized between z/OS and z/VM by IBM Tivoli Directory Integrator. This table is not a comprehensive listing, but suggests fields that are possible candidates for synchronization. The Tivoli Directory Integrator AssemblyLines can be modified to allow synchronization of these fields.

Table B-1 Table mapping LDAP names to RACF in user domain: common z/OS z/VM fields

LDAP name	RACF term	Synchronize	Remarks
racfSecurityCategoryList	ADDCATEGORY	Yes	
racfAttributes	SPECIAL, OPERATIONS	No	Maybe unwanted (1)
racfConnectGroupAuthority	AUTH not displayed LDAP	No	Maybe unwanted (1)
racfClassName	CLAUTH	No	Maybe unwanted (1)
racfDefaultGroup	DFLTGRP	Yes	
racfConnectGroupName	GROUP	No	Handled by Connect
racfLastAccess	Not modifiable	No	
racfProgrammerName	NAME	Yes	
racfPasswordChangeDate	Not modifiable	No	
racfPasswordInterval	Not modifiable	No	
racfPassword	PASSWORD	No	Handled specially
racfPasswordEnvelope	Not modifiable	No	
racfHavePasswordEnvelope	Not modifiable	No	
racfPassPhraseEnvelope	Not modifiable	No	
racfPassPhrase	PHRASE	No	Handled specially
racfPassPhraseChangeDate	Not modifiable	No	
racfHavePassPhraseEnvelope	Not modifiable	No	
racfResumeDate	RESUME	Yes	
racfRevokeDate	REVOKE	Yes	
racfSecurityLabel	SECLABEL	Yes	
racfSecurityLevel	SECLEVEL	Yes	
racfConnectGroupUACC	UACC - Not displayed	No	
racfLogonDays	WHEN(DAYS())	Yes	
racfLogonTime	WHEN(TIME())	Yes	
racfAuthorizationDate	Not modifiable	No	
racfInstallationData	DATA	Yes	
racfDatasetModel	MODEL	Yes	
racfOwner	OWNER	Yes	

SPECIAL attribute: Some users might have the SPECIAL attribute on test systems, but not on production systems. Hence, a blind synchronization of these fields might be unwanted.

Table B-2 provides table mapping LDAP names to RACF in the user domain.

Table B-2 Table mapping LDAP names to RACF in user domain: almost common open VM: open IBM MVS™ fields

LDAP name	RACF term	Synchronize	Remarks
racfOvmFileSystemRoot	FSROOT	No	
racfOvmHome	HOME	No	
racfOvmInitialProgram	PROGRAM	No	
racfOvmUid	UID	No	
racfOmvsHome	HOME	No	
racfOmvsInitialProgram	PROGRAM	No	
racfOmvsUid	UID	No	

We decided not to synchronize these “open” fields because of the following reasons:

- Not all fields exist on both platforms.
- The LDAP names are different. Some Tivoli Directory Integrator remapping would be required.
- The values might be different between the platforms.

More customization of the sample Tivoli Directory Integrator configuration can synchronize these fields.

Group field mapping

Table B-3 displays the mappings between RACF and LDAP. The Synchronize column indicates whether the data can be synchronized between z/OS and z/VM by Tivoli Directory Integrator. This table is not a comprehensive listing, but suggests fields that are possible candidates for synchronization. The Tivoli Directory Integrator AssemblyLines can be modified to allow synchronization of these fields.

Table B-3 Table mapping LDAP names to RACF in group domain: common open VM: open MVS fields

LDAP name	RACF term	Synchronize	Remarks
racfSuperiorGroup	SUPGROUP	Yes	
racfSubGroupName	SUBGROUP(s) Not modifiable	No	
racfGroupNoTermUAC	TERMUACC	No	
racfGroupUniversal	UNIVERSAL	Yes	
racfGroupUserids	USER(S) Not modifiable	No	
racfAuthorizationDate	CREATED Not modifiable	No	

LDAP name	RACF term	Synchronize	Remarks
racfInstallationData	DATA	Yes	
racfDatasetModel	MODEL	No	Not used on VM
racfOwner	OWNER	Yes	

C

Additional material

This paper refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this paper is available in softcopy on the Internet from the IBM Redbooks Web server. Point your web browser at:

ftp://www.redbooks.ibm.com/redbooks/REDP-4460-00

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redpaper form number, REDP4460.

Using the Web material

The additional Web material that accompanies this paper includes the following files:

File name	*Description*
zRedbook.xmit	Sample Tivoli Directory Integrator configuration xml file
RACF Redbook Properties.txt	Sample properties file
racfexop.jar	java classes

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 203. Note that some of the documents referenced here may be available in softcopy only.

- *IBM Tivoli Directory Server for z/OS,* SG24-7849
- *Robust Integration with Tivoli Directory Integrator 7.0,* REDP-4672
- *z/OS Security Server RACF Security Administrator's Guide,* SA22-7683
- *IBM Tivoli Directory Server Administration and Use for z/OS,* SC23-5191

Other publications

These publications are also relevant as further information sources:

- *z/VM RACF Security Server Security Administrator's Guide,* SC24-6218
- *z/VM TCP/IP User's Guide,* SC24-6240

Online resources

This website is also relevant as a further information source:

- Tivoli Directory Integrator Information Center

  http://pic.dhe.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.IBMDI.doc_7.1.1/welcome.htm

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications, and Additional materials, as well as order hardcopy Redbooks, at this website:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

M

O

P

R

S

T

U

V

Z

Synchronizing IBM RACF data by using IBM Tivoli Directory Integrator

RACF Password Synchronization

IBM Tivoli Directory Server

IBM Tivoli Directory Integrator

This IBM Redpaper publication provides an example of a solution to synchronize an IBM RACF user ID, password, and password phrase data between IBM z/OS and IBM z/VM systems, or just between z/VM systems. Topics that are covered are the installation and customization of IBM Tivoli Directory Integrator, IBM Tivoli Directory Server, and RACF. Using this basic infrastructure, a sample Tivoli Directory Integrator configuration is presented, which allows for a flexible and extensible means for synchronizing RACF information.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

REDP-4460-00