Novell Developer Kit

www.novell.com

October 5, 2005

NOVELL SECRETSTORE®
DEVELOPER KIT FOR JAVA*

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export, or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to www.novell.com/info/exports/ (http://www.novell.com/info/exports/) for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.novell.com/company/legal/patents/ and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell developer products, and to get updates, see developer.novell.com/ndk. To access online documentation for Novell products, see www.novell.com/documentation.

Novell Trademarks

AppNotes is a registered trademark of Novell, Inc.
AppTester is a registered trademark of Novell, Inc. in the United States.
ASM is a trademark of Novell, Inc.
BorderManager is a registered trademark of Novell, Inc.
BrainShare is a registered service mark of Novell, Inc. in the United States and other countries.
C3PO is a trademark of Novell, Inc.
Certified Novell Engineer is a service mark of Novell, Inc.
Client32 is a trademark of Novell, Inc.
CNE is a registered service mark of Novell, Inc.
ConsoleOne is a registered trademark of Novell, Inc.
Controlled Access Printer is a trademark of Novell, Inc.
Custom 3rd-Party Object is a trademark of Novell, Inc.
DeveloperNet is a registered trademark of Novell, Inc., in the United States and other countries.
DirXML is a registered trademark of Novell, Inc.
eDirectory is a trademark of Novell, Inc.
Excelerator is a trademark of Novell, Inc.
exteNd is a trademark of Novell, Inc.
exteNd Director is a trademark of Novell, Inc.
exteNd Workbench is a trademark of Novell, Inc.
FAN-OUT FAILOVER is a trademark of Novell, Inc.
GroupWise is a registered trademark of Novell, Inc., in the United States and other countries.
Hardware Specific Module is a trademark of Novell, Inc.
Hot Fix is a trademark of Novell, Inc.
iChain is a registered trademark of Novell, Inc.
Internetwork Packet Exchange is a trademark of Novell, Inc.
IPX is a trademark of Novell, Inc.
IPX/SPX is a trademark of Novell, Inc.
jBroker is a trademark of Novell, Inc.
Link Support Layer is a trademark of Novell, Inc.
LSL is a trademark of Novell, Inc.
ManageWise is a registered trademark of Novell, Inc., in the United States and other countries.
Mirrored Server Link is a trademark of Novell, Inc.
Mono is a registered trademark of Novell, Inc.
MSL is a trademark of Novell, Inc.
My World is a registered trademark of Novell, Inc., in the United States.
NCP is a trademark of Novell, Inc.
NDPS is a registered trademark of Novell, Inc.
NDS is a registered trademark of Novell, Inc., in the United States and other countries.
NDS Manager is a trademark of Novell, Inc.
NE2000 is a trademark of Novell, Inc.
NetMail is a registered trademark of Novell, Inc.
NetWare is a registered trademark of Novell, Inc., in the United States and other countries.
NetWare/IP is a trademark of Novell, Inc.
NetWare Core Protocol is a trademark of Novell, Inc.
NetWare Loadable Module is a trademark of Novell, Inc.

NetWare Management Portal is a trademark of Novell, Inc.

NetWare Name Service is a trademark of Novell, Inc.

NetWare Peripheral Architecture is a trademark of Novell, Inc.

NetWare Requester is a trademark of Novell, Inc.

NetWare SFT and NetWare SFT III are trademarks of Novell, Inc.

NetWare SQL is a trademark of Novell, Inc.

NetWire is a registered service mark of Novell, Inc., in the United States and other countries.

NLM is a trademark of Novell, Inc.

NMAS is a trademark of Novell, Inc.

NMS is a trademark of Novell, Inc.

Novell is a registered trademark of Novell, Inc., in the United States and other countries.

Novell Application Launcher is a trademark of Novell, Inc.

Novell Authorized Service Center is a service mark of Novell, Inc.

Novell Certificate Server is a trademark of Novell, Inc.

Novell Client is a trademark of Novell, Inc.

Novell Cluster Services is a trademark of Novell, Inc.

Novell Directory Services is a registered trademark of Novell, Inc.

Novell Distributed Print Services is a trademark of Novell, Inc.

Novell iFolder is a registered trademark of Novell, Inc.

Novell Labs is a trademark of Novell, Inc.

Novell SecretStore is a registered trademark of Novell, Inc.

Novell Security Attributes is a trademark of Novell, Inc.

Novell Storage Services is a trademark of Novell, Inc.

Novell, Yes, Tested & Approved logo is a trademark of Novell, Inc.

Nsure is a registered trademark of Novell, Inc.

Nterprise is a trademark of Novell, Inc.

Nterprise Branch Office is a trademark of Novell, Inc.

ODI is a trademark of Novell, Inc.

Open Data-Link Interface is a trademark of Novell, Inc.

Packet Burst is a trademark of Novell, Inc.

PartnerNet is a registered service mark of Novell, Inc., in the United States and other countries.

Printer Agent is a trademark of Novell, Inc.

QuickFinder is a trademark of Novell, Inc.

Red Box is a trademark of Novell, Inc.

Red Carpet is a registered trademark of Novell, Inc., in the United States and other countries.

Sequenced Packet Exchange is a trademark of Novell, Inc.

SFT and SFT III are trademarks of Novell, Inc.

SPX is a trademark of Novell, Inc.

Storage Management Services is a trademark of Novell, Inc.

SUSE is a registered trademark of SUSE AG, a Novell business.

System V is a trademark of Novell, Inc.

Topology Specific Module is a trademark of Novell, Inc.

Transaction Tracking System is a trademark of Novell, Inc.

TSM is a trademark of Novell, Inc.

TTS is a trademark of Novell, Inc.

Universal Component System is a registered trademark of Novell, Inc.

Virtual Loadable Module is a trademark of Novell, Inc.

VLM is a trademark of Novell, Inc.

Yes Certified is a trademark of Novell, Inc.

ZENworks is a registered trademark of Novell, Inc., in the United States and other countries.

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Contents

About This Guide

Novell SecretStore™ for Java (JSSO) provides Java classes that enable network applications to securely access user authentication information, such as user name, password, login methods, etc. This document describes the Java library that augments the Novell SecretStore for C (http://developer.novell.com/ndk/ssocomp.htm) API set. Refer to that document if you require additional detail about the authentication process, naming conventions, SecretStore, or single sign-on use scenarios.

This guide contains the following sections:

Audience

This guide is intended for Java developers who wish to enable Novell SecretStore functionality within their applications.

Feedback

We want to hear your comments and suggestions about this manual. Please use the User Comments feature at the bottom of each page of the online documentation and enter your comments there.

Documentation Updates

For the most recent version of this guide, see Novell SecretStore Developer Kit for Java (http://developer.novell.com/ndk/nssoj.htm).

Additional Documentation

For additional background about enabling your applications to use SecretStore, see:

- *A Technical Overview of Novell SecretStore 3.2* (http://support.novell.com/techcenter/articles/dnd20030503.html)
- *Understanding Novell's Single Sign-On* (http://support.novell.com/techcenter/articles/ana20000202.html)
- *SecretStore: Novell Single Sign-on Version 1.1* (http://support.novell.com/techcenter/articles/dnd20000402.html)
- *SecretStore Single Sign-on* (http://support.novell.com/techcenter/articles/dnd19991105.html)

You also might want to refer to the Novell SecretStore Administration Guide (http://www.novell.com/documentation/secretstore33/pdfdoc/nssadm/nssadm.pdf) to understand how security services are implemented by JSSO.

For SecretStore source code projects, visit Forge Project: Novell SecretStore Developer Kit for Java (http://forge.novell.com/modules/xfmod/project/?nssoj).

For SecretStore sample code, see Novell SecretStore Developer Kit for Java (http://forge.novell.com/modules/xfmod/sample/index.php?group_id=1074&sampleid=94).

For help with SecretStore problems or questions, visit the Novell Support Forum (http://forge.novell.com/modules/xfmod/newsportal/?group_id=1074).

Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux or UNIX, should use forward slashes as required by your software.

Getting Started

1

The Novell® SecretStore™ Developer Kit for Java* (JSSO) enables applications to automatically authenticate to the network using personal secrets stored in Novell eDirectory™.

Using SecretStore technology, users can store authentication information securely in eDirectory, then retrieve the information later and authenticate access without further manual interaction. This communication between your single sign-on-enabled applications and Novell SecretStore allows your network applications to authenticate securely and seamlessly, as illustrated in Figure 1-3 on page 14.

JSSO provides an object-oriented interface for applications that rely on either Novell Core Protocol (NCP) or Java Naming and Directory Interface (JNDI). The Java SecretStore library now comprises one jar file (jsso.jar) logically split into two sections—the API and the implementations.

Although only NCP and JNDI implementations are provided now, additional implementations may be added in the future. Regardless of which implementation of the JSSO library is used, users can run enabled applications seamlessly from any connected workstation.

This section consists of the following topics:

1.1 Novell eDirectory and SecretStore

Novell SecretStore is a service that leverages the security built into Novell eDirectory. Rather than storing user secrets (user name, password, etc.) on the client machine, Novell SecretStore uses a secure central location in eDirectory, which provides fault tolerance and secure management. Because the user is the only person authorized to access his or her secrets when a SecretStore-enabled application authenticates to eDirectory, access is tightly controlled. With SecretStore, user passwords and credentials are never stored or transmitted without being encrypted first. Applications relying on the Novell Core Protocol (NCP) implementation are encrypted using Novell International Cryptographic Infrastructure (NICI), while those implementing Java Naming and Directory Interface (JNDI) encrypt secret information using Secure Socket Layer (SSL) functionality. Encryption and storage of all user authentication information prevents unauthorized access to all SecretStore-enabled applications.

See “Novell Core Protocol (NCP) Implementation” on page 19 and “JNDI Implementation” on page 21 for more detailed information.

1.2 Java Library Design

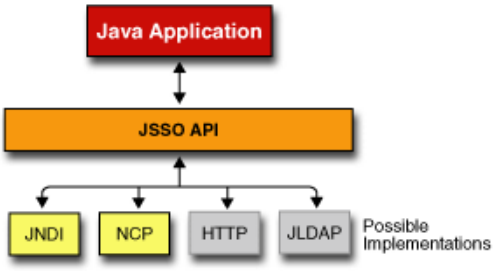
The JSSO library provides methods to create a common Java interface to SecretStore functionality that is independent of the communication pathway between the library and SecretStore in the directory. The library transforms the function-based C client API into an object-oriented Java client

API. Because the library is Java based, it is portable across multiple platforms. This extensible, modular design enables new library features to be added without impact to applications that use its functions.

1.2.1 Library Architecture

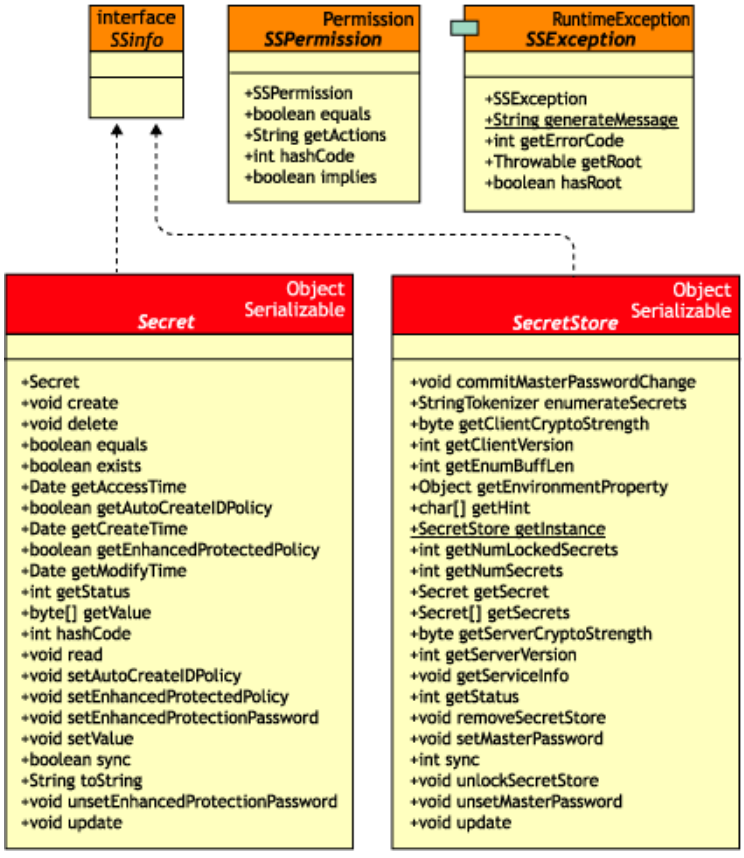
As shown in Figure 1-1 on page 12, the JSSO library architecture is split into two parts: the Java application programming interface and its possible implementations. Note that the NCP implementation requires a NetWare client and functions only on Windows. However, the LDAP implementation is platform independent.

Figure 1-1 *Java JNDI and NCP Implementations*



The JSSO library provides a common, object-oriented interface to SecretStore. Additional implementations can be added to the library without impacting the Java applications.Java applications use the JSSO API to obtain SecretStore functionality, primarily by using the Secret and SecretStore Java* classes, as shown in Figure 1-2 on page 13.

Figure 1-2 *Secret and SecretStore Java Classes*



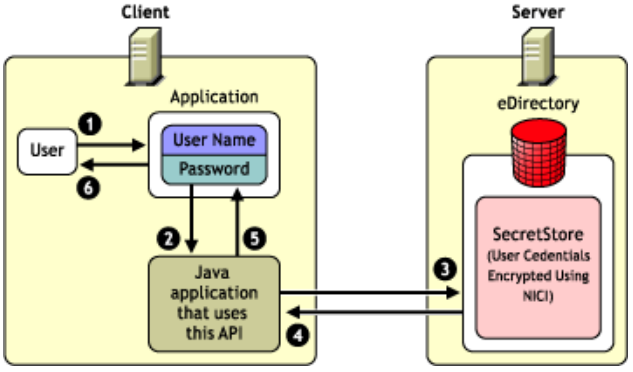
As shown in the figure, the Secret and SecretStore classes make up most of the API. The Secret class manages functionality that involves a secret, while the SecretStore class deals with functionality that involves the SecretStore as a whole.

Applications obtain an instance of the SecretStore by calling the getInstance (hashtable environment) method. Applications can provide information to the library in the form of environment properties.Implementations plug into the library by providing two classes that extend the Secret and SecretStore classes. The API classes provided in this document handle all of the business logic that concerns SecretStore functionality. This enables implementations to contain only code that deals with how the library communicates with SecretStore.

1.3 SecretStore Functionality

The API in this document describes the Get and Set methods required to store user or application secrets (e.g., user name, password, identification number, pin, token or biometric authentication information, etc.) in a persistent storage location on eDirectory, as shown in Figure 1-3 on page 14.

Figure 1-3 *Processes in User Applications to Implement SecretStore*



In essence, SecretStore consists of encrypted hidden user attributes that are contained on the user object. Encryption of these attributes in eDirectory using the Novell International Cryptographic Infrastructure (NICI) ensures that authentication information remains safe and secure from unauthorized access. Depending on your geographic location, credentials can be encrypted using either DES (64-bit) or Triple DES (128 bit) encryption strength.

NOTE: See the Novell SecretStore Administration Guide (http://www.novell.com/documentation/secretstore33/pdfdoc/nssadm/nssadm.pdf) for more information about implementation of NICI encryption features.

Instead of making direct calls to eDirectory to obtain stored user credentials, the SecretStore API enables your applications' connectors to make the necessary Get and Set calls to facilitate passing of credentials between SecretStore and the application.

1.4 Related Novell Security and Authentication Solutions

In addition to the SecretStore APIs, Novell offers a number of solutions to help integrate NDS with various authentication services. For more information, check the following links:

- Novell SecretStore Administration Guide (http://www.novell.com/documentation/secretstore33/pdfdoc/nssadm/nssadm.pdf)—enables strong encryption technology in network applications.
- Novell Modular Authentication Service (http://www.novell.com/products/nmas)—allows implementation of different authentication methods. Also see the NDK: Novell Modular Authentication Services NDK information.
- Novell eDirectory (http://www.novell.com/products/edirectory)—stores millions of identities, including detailed information about each user's roles and business relationships.

1.5 Novell Single Sign-on Documentation History

Prior to the February 2002 release of the Novell Development Kit (NDK), this document was entitled “Novell Single Sign-on™ for Java”. Although content of the current document is very similar to the former version, the name was changed to reflect Novell's product shift away from “Novell Single Sign-on” to other products that rely on SecretStore functionality, which facilitates the single sign-on process.

Indeed, Novell Single Sign-on was the first Novell product that used Novell SecretStore technology. Now, in addition to Novell Single Sign-on, a growing number of other products consume the SecretStore methods described in this document: Novell SecureLogin, Novell iChain, Novell Portal Services (NPS), Novell DirXML, virtual CDs (VCD), and others. Novell is now labelling and branding SecretStore components separately from the products that consume them.

Single Sign-on Evolution

Novell Single Sign-on (NSSO) Version 1.0 provided single sign-on access for a limited number of key applications, primarily used in Intranet environments. Version 2.*x*, offered in a bundle with Passlogix v-GO*, expanded the functionality to most web sites and Windows-based applications, with limited support for terminal emulators.

In June 2001, Novell released Novell SecureLogin 2.5 (NSL), an interim single sign-on solution that provided enhanced features of NSSO 2.x but lacked integration with several key Novell technologies (SecretStore, NMAS, NICI, etc.). Consequently, Novell introduced the Novell SecureLogin 3.0 snap in in late 2001, which combined features of both NSSO and NSL and integration with Novell security technologies. The new NSL 3.0 is now fully integrated with SecretStore.

JSSO Core Library Installation

2

The jsso.jar file and libraries listed in Prerequisites must be added to the classpath. The JSSO library installation depends on which implementation of JSSO is used. Currently, there is only one jsso.jar file for all implementations right now because of its small size.

2.1 Dependencies

To SSO-enable a Java application, you need the libraries included in the Novell SecretStore for C Developer Kit (http://developer.novell.com/ndk/ssocomp.htm) and Novell SecretStore for Java (http://developer.novell.com/ndk/nssoj.htm) downloads.

The JSSO library contains one .jar file (jsso.jar) split into two logical sections—the API and the specified implementations. In addition to core prerequisites, other prerequisites depend on which implementation of JSSO is used. See the sections Section 2.3, “Server Requirements,” on page 17 and Section 2.1, “Dependencies,” on page 17 for the dependencies required to invoke a specific implementation.

To test and use a Novell SecretStore-enabled application, install the listed software on the client and server.

2.2 Client Requirements

JSSO operates on any system that supports a Java Virtual Machine (JVM) 1.2.2 or above.

NOTE: See the JDK documentation for system requirements and JDK-specific installation instructions at Sun Microsystems documentation for supported libraries (http://java.sun.com).

2.3 Server Requirements

- Novell SecretStore 3.0
- NetWare® 5.0 with Support Pack 1 or later versions
- NICI 2.01 on NetWare or NICI 2.02 on Windows
- Novell Directory Services (version installed with NetWare 5)

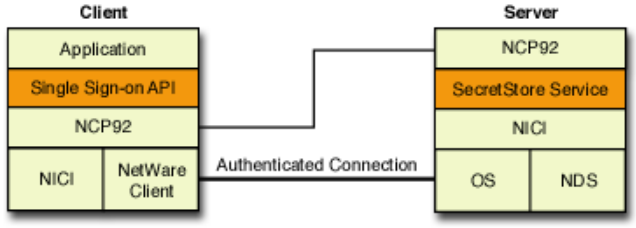
NOTE: If NDS 8 for NetWare is listed, verify that it is 8.12 or later. If you are using NDS 8, make sure you use NetWare 5.0 Support Pack 1.

Novell Core Protocol (NCP) Implementation

3

Applications enabled using the NCP implementation rely on Novell eDirectory and Novell International Cryptographic Infrastructure (NICI) to securely store, access, and retrieve user authentication information. The client application makes read and write calls to SecretStore on the server, which processes and executes authentication requests. User secrets (such as the username and password) are encrypted using NICI and stored as an attribute in the user's SecretStore on the user object in Novell eDirectory.

Figure 3-1 *NCP Implementation*



3.1 NCP Code Requirements

The sample test code demonstrates how to access basic SecretStore functionality by using the NCP implementation. To use this implementation, call the method *getInstance(Hashtable env)* in the *com.novell.security.sso.SecretStore* class. The environment property *com.novell.sso.secretstore* should be set to *com.novell.security.sso.ncp.NCPSecretStore*.

The property *com.novell.sso.handle* is not used. The properties *com.novell.sso.userdn* and *com.novell.sso.tree* are optional. If not specified this implementation automatically accesses the SecretStore on the primary tree and server for the currently logged on user to that tree and server.

To specify a different user and/or tree, use the environment properties *com.novell.sso.userdn* and *com.novell.sso.tree*. *com.novell.sso.userdn* is the user's distinguished name (*java.lang.String*) and *com.novell.sso.tree* is the tree (*java.lang.String*).

3.2 NICI and SecretStore

Authentication secrets passed between Novell SecretStore on the client and SecretStore on the server are securely encrypted during transmission using NICI. The secrets also are protected with NICI within the eDirectory SecretStore repository.

When the client application retrieves the secret from eDirectory, the secret is decrypted at the client side and, upon successful completion of the application's authentication process, is promptly destroyed and removed from memory (similar to how the eDirectory private key is handled in the eDirectory authentication process).

To give users access to network services, eDirectory uses an authentication service based on the RSA public-key/private-key encryption/decryption algorithms. This authentication mechanism uses

a private key attribute and a digital signature to verify a user's identity. eDirectory authentication is session-oriented, and the client's signature is valid only for the duration of the current session.

The client doesn't have to be reauthenticated every time the user asks for additional services or applications, since reauthentication takes place automatically in the background. Therefore, the integrity of SecretStore-enabled applications is protected and secure, and the user can access resources globally without having to authenticate each time.

Users first log in to eDirectory where their enabled applications have access to the SecretStore service. The user of the application is then granted access to the enabled application service and resources without seeing a password dialog or other authentication screens. It appears to the user as if access were granted automatically.

NOTE: The API is designed for client enablement for SSO. For single sign-on to work, SecretStore 3.0 must be installed on a connected server.

3.3 Prerequisites

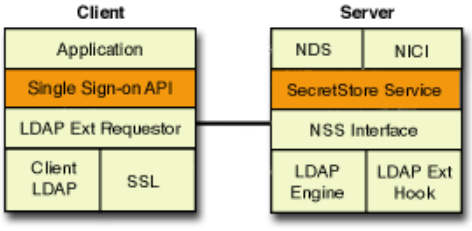
- Novell SecretStore 3.0 client software (available in the Novell SecureLogin 3.0 product) installed on a workstation running Novell Client™ for Windows NT version 4.5 or later, or Novell Client for Windows 95/98 version 3.0 or later.
- NICI Client Software 2.02

JNDI Implementation

4

Applications enabled using JNDI implementation connect to SecretStore via LDAP using LDAP v3 extended operations. The library does not provide any connection and/or session management. This is left up to the application.

Figure 4-1 *JNDI Implementation*



The sample test code demonstrates how to access basic SecretStore functionality by using the NCP implementation. To use this implementation, call the method *getInstance (Hashtable env)* in the *com.novell.security.sso.SecretStore* class. The environment property *com.novell.sso.secretstore* should be set to *com.novell.security.sso.ldap.jndi.JNDISecretStore*.

The application must provide a *java.naming.LdapContext* in the form of the *com.novell.sso.handle* environment property. When accessing SecretStore functionality via LDAP, the connection must be SSL based for security.

4.1 Prerequisites

Server

- eDirectory (NDS version 8.5)

Client

- Sun Microsystems JNDI 1.2.1.Sun Microsystems JSSE 1.0.2 (A SSL connection is required when communicating with SecretStore.)JNDI service provider that supports LDAP v3.0.JSSE service provider.

4.2 Installation

To establish a SSL connection with Novell eDirectory (the LDAP server where SecretStore resides), you must place the Trusted Root Certificate of the server in the truststore that can be used by the JSSE when it seeks to establish this connection. The server's Trusted Root Certificate may be exported by using the Novell ConsoleOne™ administration tool. The certificate can be imported into the truststore by using Keytool, a program shipped with Sun Microsystems' JDK. See Sun Microsystems' (http://www.sun.com) documentation on how to create a truststore and how to import a Trusted Root Certificate. (Use Keytool and refer to documentation in JSSE.) Also, see Novell's documentation on how to export a Trusted Root Certificate (http://www.novell.com/documentation/lg/crt203ad/crtadmin/data/a2ebopb.html#a2ebopd).

IMPORTANT: Installation of eDirectory should set up the LDAP server object and an associated SSL certificate object. View the LDAP server properties to make sure the correct certificate object was exported.1. Select the SSL Configuration tab. The SSLCertificate textbox should specify which certificate object will be used for SSL communications.

2. Ensure that the Trusted Root Certificate is exported from the Certificate object that the textbox specifies.

4.3 Sample Test Code

The sample test code demonstrates how to access basic SecretStore functionality by using the JNDI implementation. The sample code uses the Sun Microsystems LDAP service provider for the JNDI. It also uses the Sun Microsystems JSSE service provider. These libraries are available on the Sun Microsystems' (http://java.sun.com) website.

Javadoc References

5

See specific Javadoc reference information for the two Java wrapper classes for Novell SecretStore at Secret Class reference (../api/com/novell/security/sso/Secret.html) and at SecretStore Class reference (../api/com/novell/security/sso/SecretStore.html).

5.1 Enabling the AdminDemo.java Example

When running the AdminDemo.java (../../../samplecode/nssoj_sample/index.htm) example, a commonly returned error is:

```
javax.naming.CommunicationException: simple bind failed:
developer.mycompany.com.br:636. Root exception is
javax.net.ssl.SSLHandshakeException: Couldn't find trusted certificate
```

During the handshake between the client and the server, the server sends a digital certificate so that the client can authenticate the server. The client attempts to authenticate the server by verifying if the certificate was issued by a Certificate Authority (CA) that the client trusts. The SSLHandshakeException will be thrown if the server returns a certificate that was not issued by a CA that has a corresponding trusted root certificate in the truststore that Java Secure Socket Extension (JSSE) is using.

By default JSSE uses the *cacerts* file as the default truststore. This file is provided in each Java Runtime Environment (JRE). You can tell which trusted root certificates are in the *cacerts* truststore by running keytool (provided with each JRE) with the -list option on the *cacerts* file that is located in the <javahome>\lib\security folder (on Windows at least). This will list all the trusted root certificates in the *cacerts* truststore.

You can prevent the javax.naming.CommunicationException: by obtaining the trusted root certificate for the server's certificate and importing it into the truststore that the JSSE is using. You import certificates into a truststore by using keytool.ext with the -import command selected. By default, this is the *cacerts* file.

NOTE: Before you can perform any operations (except the list operation) on a keystore you must provide the password. The password for the *keystore* file is "changeit". After doing this make sure to restart the JSSE. The easiest way to do this is by restarting the JVM.

This should fix the JSSE issues unless you have Mutual Authentication enabled on the server. When this is enabled, the server must be able to authenticate the client. The client's certificate must be in the keystore that the JSSE is using or else the handshake will fail.

Revision History

A

March 5, 2005	• Transitioned to revised Novell documentation standards. • Fixed broken links.
March 2, 2004	Fixed broken links and revised documentation to facilitate management on the Novell Forge development site.
November 8, 2004 Midrelease	• Fixed broken links and made minor technical corrections. • Updated the organization structure of the document to conform more fully to the Novell Developer Kit style.
February 18, 2004	Made minor editorial revisions to fix trademark bug.
June 2003	• Changed flags from NSSO to NSSS • Refreshed software and sample code
March 2003	Added support for shared secrets in the NSSOJ software.
September 2002	Added two new sections: • How to enable the AdminDemo.java sample code • Listing of Novell solutions to help integrate NDS with various authentication services
February 2002	Changed documentation to reflect labelling and branding of SecretStore components separately from the products that consume them. As a result, deleted documentation related to v-GO secrets.
June 2001	Added a class that provides the ability to create/view/edit v-GO secrets.
February 2001	Added Java library concepts chapter and reference information.
July 2000	New Java wrapper classes provided for Novell Single Sign-on 1.1 APIs.