

CACORE COMMON SECURITY MODULE (CSM)

Version 4.1 – Programmer's Guide



**NATIONAL[®]
CANCER
INSTITUTE**

Center for Biomedical Informatics
and Information Technology

This is a U.S. Government work.

November 18, 2008

Revision History

The most current version of this document is located on the CSM website:
<http://ncicb.nci.nih.gov/core/CSM>.

Revision History

Revision Date	Author	Summary of Changes
09/15/07	Vijay Parmar, Kunal Modi	Initial Table of Contents
10/22/07	Vijay Parmar	Added new chapters
11/05/2007	Vijay Parmar	Incorporate updates.
10/29/2008	Vijay Parmar	Version 4.1 updates
11/07/2008	Bronwyn Gagne	Doc converted to current CBIIT template, and edited as necessary.
11/14/2008	Vijay Parmar, Bronwyn Gagne	Final review and release of updated 4.1 guide.
11/18/2008	Vijay Parmar, Bronwyn Gagne	Correction made to guide – 'encryption-enable' changed to 'encryption-enabled'

Table of Contents

About This Guide	1
Purpose	1
Scope	1
Typical User	1
Topics Covered	2
Related Documentation	2
Text Conventions Used	3
Credits and Resources	3
Chapter 1 CSM Overview	5
Introduction	5
Security Concepts	7
Minimum System Requirements	8
Chapter 2 Using the CSM API	9
Workflow	9
API Services	10
Authentication Manager	10
Authorization Manager	10
Integrating with the CSM Authentication Service	10
Importing the CSM Authentication Manager Class	11
Using the CSM Authentication Manager Class	11
Installation and Deployment Configurations	12
RDBMS Credential Provider Properties and Login Module Configuration	13
LDAP Credential Provider Properties and Login Module Configuration	16
Activating CLM Audit Logging	19
Integrating With the CSM Authorization Service	19
Importing and Using the CSM Authorization Manager Class	20
Software Products and Scripts	21
Installation and Deployment Configurations	22
Audit Logging	24
JAR Placement	24
Integrating CLM APIs with CSM APIs	25
Deployment Steps	27
Chapter 3 User Provisioning Tool (UPT)	29
Workflow	29
Super Admin – To Register Applications and Admins	29
Admin – To Administer a Registered Application	29
Login	29
Common Basic Functions	30
Create New	31
Search for and Select Existing Elements	32
Update Elements	34
Deleting an Existing Element	34
Assignments and Associations	35
Assigning or Deassigning Elements - Individual to Group	35
Assigning or Deassigning Elements - Group to Individual	37
Super Admin Mode	39
Workflow for Super Admin	39
Application Administration – Super Admin Mode	40
User Administration– Super Admin Mode	42

Privileges and Standard Privileges	45
Admin Mode.....	47
Workflow for Admin	49
User Administration – Admin Mode	50
Protection Element Administration – Admin Mode	52
Privilege Administration – Admin Mode	54
Protection Group Administration – Admin Mode.....	55
Role Administration – Admin Mode.....	57
Group Administration – Admin Mode	59
Instance Level Security Administration – Admin Mode.....	62
UPT Installation and Deployment.....	66
UPT Release Contents	66
Installation Modes	66
UPT Deployment Checklist.....	69
UPT Deployment Steps.....	70
Chapter 4 CSM Security Web Services.....	73
Security Web Service WSDL.....	73
Security Web Service Operations.....	73
Login Operation.....	73
CheckPermission Operation	74
Workflow for CSM Security Web Service	75
Installation of CSM Security Web Service.....	75
Create and Prime Database	75
Configure Datasource	76
Configure JBoss JAAS Login Parameters	77
Deploy the Security WS .war File	78
Chapter 5 CSM Instance Level and Attribute Level Security.....	79
Instance Level Security	79
Requirements Addressed.....	79
Overall Design.....	81
Provisioning Instance Level Security	81
Using Instance Level Security.....	84
Known Issues.....	86
Attribute Level Security.....	86
Requirements Addressed.....	86
Overall Design.....	87
Strict Or Lenient Behavior	87
Provisioning Attribute Level Security	87
Using Attribute Level Security.....	88
Known Issues.....	89
Chapter 6 Acegi Adapter	91
Implementation	91
Method Level Security.....	92
Method Parameter Level Security.....	92
Workflow	92
Integrating and Configuring	93
Configure Acegi Security.....	93
Database Properties and Configuration.....	94
User provisioning via UPT.....	99

Chapter 7	CSM caGrid Authentication	101
	Authentication for caGrid	101
	CSM Configuration for IDP/Authentication Service.....	101
	Authorization for caGrid	103
	Using Grid Group Names for Check Permission	103
Appendix A	CSM/ACEGI Sample Configuration File	105
Appendix B	Migrating to CSM v4.1	109
	Migrating from CSM v3.2 to CSM 4.0.....	109
	MySQL Migration – CSM 3.2 to CSM 4.0	109
	Oracle Migration – CSM 3.2 to CSM 4.0.....	110
	Migrating from CSM v4.0 to CSM v4.1	110
	MySQL Migration – CSM 4.0 to CSM 4.1	110
	Oracle Migration – CSM 4.0 to CSM 4.1.....	111
	Glossary	113
	Index.....	115

About This Guide

This preface introduces you to the *caCORE Common Security Module (CSM) V 4.1 Programmer's Guide*.

Topics in this section include:

- [Purpose](#) on this page
- [Scope](#) on this page
- [Typical User](#) on page 1
- [Topics Covered](#) on page 2
- [Related Documentation](#) on page 2
- [Text Conventions Used](#) on page 3
- [Credits and Resources](#) on page 3

Purpose

This guide provides all the information application developers need to successfully integrate with NCICB's Common Security Module (CSM). The CSM was chartered to provide a comprehensive solution to common security objectives so not all development teams need to create their own security methodology. CSM is flexible enough to allow application developers to integrate security with minimal coding effort. This phase of the Common Security Module brings the NCICB team one step closer to the goal of application security management, single sign-on, and Health Insurance Portability and Accountability Act (HIPPA) compliance.

Scope

This document is a master document that covers all CSM modules and shows how to deploy and integrate the CSM services, including Authentication, Authorization, User Provisioning Tool, CSM Security Web Services, CSM Acegi Adapter, and CSM caGrid Integration. This document covers the User Guide and Application Developers Guide for all modules of CSM including CSM API, CSM UPT, CSM Security Web services, CSM Acegi Adapter, and CSM caGrid Integration. The CSM GAARDS Migration Module (CGMM) is out of scope. Refer the CGMM Guide for information

Typical User

The primary audience of this guide is the application developer who wants to integrate security. This guide assumes that you are familiar with the Java programming language and/or other programming languages, database concepts, and the Internet. If you intend to use caCORE CSM resources in software applications, it assumes that you have experience with building and using complex data systems.

Topics Covered

This brief overview explains what you will find in each section of this guide.

- [Chapter 1, CSM Overview](#) provides an overview of CSM and its capabilities.
- [Chapter 2, Using the CSM API](#) provides detailed information and a workflow for how to successfully integrate CSM into your applications.
- [Chapter 3, User Provisioning Tool \(UPT\)](#) provides information about installing the UPT, the workflow for its use, and details about the administrative tasks available in Super Admin and Admin modes.
- [Chapter 4, CSM Security Web Services](#) explains how to use the CSM Authentication and Authorization services exposed as a web service to the web service consumers.
- [Chapter 5, CSM Instance Level and Attribute Level Security](#) describes Instance and Attribute level security features available in CSM v4.1.
- [Chapter 6, Acegi Adapter](#) explains how method level and method parameter level security is implemented and available out of the box for applications that use or want to use Acegi and leverage CSM Authentication and Authorization features. This chapter also provides a workflow and steps necessary to integrate the CSM Acegi adapter into existing or new applications using the Acegi framework.
- [Chapter 7, CSM caGrid Authentication](#) describes how to leverage CSM in the caGrid environment.
- [Appendix A, CSM/ACEGI Sample Configuration File](#) provides the entire CSM Acegi Sample configuration file.
- [Appendix B Migrating to CSM v4.1](#) provides the steps necessary to migrate your existing CSM system (either 3.2 or 4.0) to CSM 4.1.

Related Documentation

More information can be found in the following related CSM documents:

- Software Architecture Document
- CSM Enterprise Architect Model
- Acegi Security CSM Adapter Design Document
- CLM Guide for Application Developers

These and other documents can be found on the CSM website:

http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/csm

Text Conventions Used

This section explains conventions used in this guide. The various typefaces represent interface components, keyboard shortcuts, toolbar buttons, dialog box options, and text that you type.

Convention	Description	Example
Bold	Highlights names of option buttons, check boxes, drop-down menus, menu commands, command buttons, or icons.	Click Search .
<u>URL</u>	Indicates a Web address.	http://domain.com
text in SMALL CAPS	Indicates a keyboard shortcut.	Press ENTER.
text in SMALL CAPS + text in SMALL CAPS	Indicates keys that are pressed simultaneously.	Press SHIFT + CTRL.
<i>Italics</i>	Highlights references to other documents, sections, figures, and tables.	See <i>Figure 4.5</i> .
<i>Italic boldface monospace type</i>	Represents text that you type.	In the New Subset text box, enter <i>Proprietary Proteins</i> .
Note:	Highlights information of particular importance.	Note: This concept is used throughout this document.
{ }	Surrounds replaceable items.	Replace {last name, first name} with the Principal Investigator's name.

Credits and Resources

caCORE CSM Development and Management Teams			
CSM Development Team	Other Development Teams	Documentation	Program Management
Vijay Parmar ¹	Satish Patel ¹	Vijay Parmar ¹	Avinash Shanbhag ³
Santhosh Garmilla ¹	Dan Dumitru ¹	Charles Griffin ¹	Charles Griffin ¹
Aynur Abdurazik ²		Bronwyn Gagne ⁴	
¹ Ekagra Software Technologies	² Science Applications International Corporation (SAIC)	³ National Cancer Institute Center for Bioinformatics	⁴ Lockheed Martin

CSM Resources	
Name	URL
Mailing List	security-csm-user@gforge.nci.nih.gov
Mailing List Archive	http://gforge.nci.nih.gov/pipermail/security-csm-user
GForge Project Home	http://gforge.nci.nih.gov/projects/security
CSM Support Tracker	http://gforge.nci.nih.gov/tracker/?atid=131&group_id=12&func=browse

Contacts and Support	
NCICB Application Support	http://ncicb.nci.nih.gov/NCICB/support Telephone: 301-451-4384 Toll free: 888-478-4423

Submitting a Support Issue

A GForge Support tracker group, which is actively monitored by CSM developers, has been created to track any support requests. If you believe there is a bug/issue in the CSM software itself, or have a technical issue that cannot be resolved by contacting the [NCICB Application Support](http://ncicb.nci.nih.gov/NCICB/support) group, please submit a new support tracker using the following link:

https://gforge.nci.nih.gov/tracker/?atid=131&group_id=12&func=browse.

Make sure to review any existing support request trackers prior to submitting a new one in order to help avoid duplicate submissions.

Release Schedule

This guide is updated for each caCORE CSM release. It may be updated between releases if errors or omissions are found. The current document refers to the 4.1 version of caCORE CSM, which was released in November 2008 by the NCI Center for Biomedical Informatics and Information Technology (CBIIT), formerly the National Cancer Institute Center for Bioinformatics (NCICB).

Chapter 1 CSM Overview

This chapter provides an overview of the Common Security Module (CSM). Topics in this chapter include:

- [Introduction](#) on this page.
- [Security Concepts](#) on page 7.
- [Minimum System Requirements](#) on this page 8.

Introduction

The CSM provides application developers with powerful security tools in a flexible delivery. CSM provides solutions for:

- **Authentication** - validating and verifying a user's credentials to allow access to an application. CSM, working with credential providers (Lightweight Directory Access Protocol (LDAP), Relational Database Management Systems (RDBMS), etc.), confirms that a user exists and that the password is valid for that application. It also provides a lockout manager which locks out unauthorized users for a pre-configured amount of time after the (also pre-configured) number of allowed attempts is reached.
- **Authorization** - granting access to data, methods, and objects. CSM incorporates an Authorization schema and database so that users can only perform the operations or access the data to which they have access rights.
- **Instance and Attribute Level Security** - allows users to perform instance level filtering of data. The User Provision Tool (UPT) allows administrators to provision security filters for instances of domain classes and the API filters the results of the queries based on the access policy. The filtering of data is done at the database level with minimum overheads. It also does attribute level filtering of data based on user permissions.
- **User Provisioning** - creating or modifying users and their associated access rights to your application and its data. CSM provides a web-based UPT that can easily be integrated with a single or multiple applications and authorization databases. The UPT provides functionality to create authorization data elements like Roles, Protection Elements, Users, etc., and also provides functionality to associate them with each other. The runtime API can then use this authorization data to authorize user actions. The UPT consists of two modes – Super Admin and Admin.
 - **Super Admin** – accessed by the UPT's overall administrator; used to register an application, assign administrators, and create or modify standard privileges.
 - **Admin** – used by application administrators to modify authorization data, such as roles, users, protection elements, etc
- **Audit Logging** - In an effort to make CSM compliant with CRF 21/ part 11, CSM provides auditing and logging functionality. CSM uses NCICB's Common Logging Module (CLM), which is another caCORE product, for the

purpose of event logging as well as automated object state change logging into a persistent database.

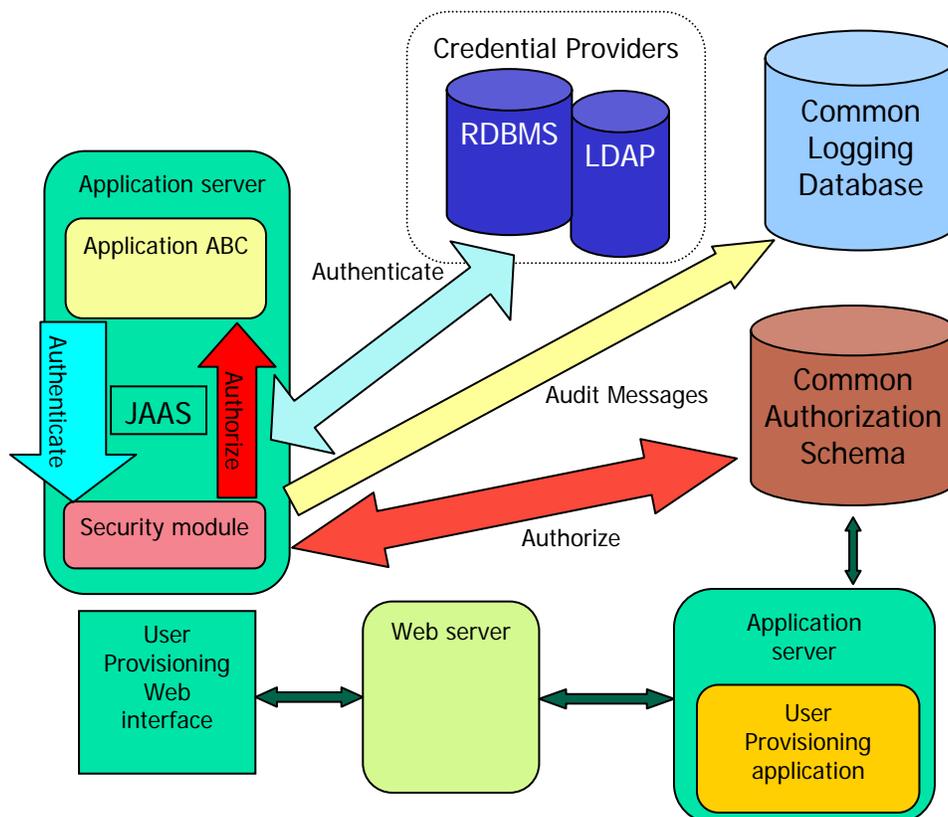


Figure 1-1 CSM Architecture

CSM works with Java Authentication and Authorization Service (JAAS) to authenticate and authorize for the application ABC. To authenticate, it references credential providers such as an LDAP or RDBMS. CSM can be configured to check multiple credential providers in a defined order. To authorize, CSM refers to the Authorization Schema. The Authorization Schema contains the Users, Roles, Protection Elements, etc., and their associations, so that the application knows whether or not to allow a user to access a particular object. The Authorization data can be stored on a variety of databases. It is created and modified by the application Administrator using the web-based UPT.

CSM uses NCICB's Common Logging Module (CLM) to perform all the audit and logging. CSM logs all of the events and object state changes (security objects stated below in Table 4-1). These logs will be stored in a separate Common Logging Database for backup and review. Since logging can be configured using log4j, client applications have control over the logging of audit trails. More details regarding audit logging by CSM can be found in the [Audit Logging](#) section, beginning on page 24.

Security Concepts

In order to successfully integrate CSM with an application, it is important to understand the definitions for the security concepts defined in the table below. Application Developers should understand these concepts and begin to understand how they apply to their particular application.

Security Concept	Definition
Application	Any software or set of software intended to achieve business or technical goals.
User	A User is someone that requires access to an application. Users can become part of a Group, and can have an associated Protection Group and Roles.
Group	A Group is a collection of application users. By combining users into a Group, it becomes easier to manage their collective roles and access rights in your application.
Protection Element	A Protection Element is any entity (typically data) that has controlled access. Examples include Social Security Number, City, and Salary. Protection Elements can also include operations, buttons, links, etc.
Protection Group	A Protection Group is a collection of application Protection Elements. By combining Protection Elements into a Protection Group, it becomes easier to associate Users and Groups with rights to a particular data set. Examples include Address and Personal Information.
Privilege	A Privilege refers to any operation performed upon data. CSM makes use of a standard set of privileges. This will help standardize authorization to comply with JAAS and Authorization Policy and allow for adoption of technology such as SAML in the future.
Role	A Role is a collection of application Privileges. Examples include Record Admin and HR Manager.

Figure 1-2 Security concept definitions

CSM users need to identify aspects of the application that should be labeled as Protection Elements. These elements are combined to Protection Groups, and then users are assigned Roles for that Protection Group.

Shown in Table 1-1 are definitions of related security terms.

Related Concept	Definition
Credential Provider	A credential is a data or set of data which represents an individual unique to a given application (username, password, etc.). Credential providers are trusted organizations that create secure directories or databases that store credentials. In an authentication transaction, organizations check with the credential providers to verify entered information is valid. For example, the NCI network uses a credential provider to verify that a user name and password match and are valid before allowing access.

Related Concept	Definition
JAAS	Set of Java packages that enable services to authenticate and enforce access controls upon users. JAAS implements a Java version of the standard Pluggable Authentication Module framework, and supports user- based authorization.
LDAP	Credential providers may choose to store credential information using a directory based on LDAP. An LDAP is simply a set of protocols for accessing information directories. Using LDAP, client programs can login to a server, access a directory, and verify credential entries.
RDBMS	Credential providers may choose to store credential information with a RDBMS. Unlike with LDAP, credential data is stored in the form of related tables.
Login Module	Responsible for authenticating users and for populating users and groups. A Login Module is a required component of an authentication provider, and can be a component of an identity assertion provider if you want to develop a separate LoginModule for perimeter authentication. LoginModules that are not used for perimeter authentication also verify the proof material submitted (for example, a user password).

Table 1-1 Related Security Concept Definitions

Minimum System Requirements

The following software is required and not included with CSM Software as listed in the table below. The software name, version, description, and URL hyperlinks are indicated in the table.

Software	Description	Version	URL
JDK	The J2SE Software Development Kit (SDK) supports creating J2SE applications	1.5.0_11 or higher	http://java.sun.com/j2se/1.5.0/download.html
Oracle	Database Server (only one is required)	9i	http://www.oracle.com/technology/products/oracle9i/index.html
MySQL		5.0.27	http://dev.mysql.com/downloads/mysql/5.0.html
JBoss	Application Server (only one is required)	4.0.5	http://labs.jboss.com/jbossas/downloads
Tomcat		5.5.20	http://tomcat.apache.org/download-55.cgi
Ant	Build Tool	1.6.5 or higher	http://ant.apache.org/bindownload.cgi

Table 1-2 Minimum Software Requirements

Chapter 2 Using the CSM API

This chapter provides an overview of the CSM API and how to use it. Topics in this chapter include:

- [Workflow](#) on this page.
- [API Services](#) on page 10.
- [Integrating with the CSM Authentication Service](#) on page 10.
- [Integrating With the CSM Authorization Service](#) on page 19.
- [Audit Logging](#) on page 24.

Workflow

This workflow section outlines the basic steps, both strategic and technical, for successful CSM integration.

1. Decide which services you would like to integrate with an application. If the application should authenticate users against an LDAP or other directory, select Authentication. If granular data protection is important, also integrate with the authorization and provisioning services. These options allow administrators to specify which users have access to particular components of the application.
2. Review the information in this document. It provides an overview, workflow, and specific deployment and integration steps. If using the provisioning service, pay special attention to [Chapter 3, User Provisioning Tool \(UPT\)](#) beginning on page 29.
3. Appoint a Security Schema Administrator who is familiar with the application and its user base. Using the User Provisioning Tool (UPT), these individuals input users, roles, etc., and ultimately gives privileges to users for certain application elements.
4. Determine a security authorization strategy. In this step, the Schema Administrator and the application team determines what data or links should be protected and what groups of people should have access to what.
5. Decide upon a deployment approach. As discussed in [Installation Modes](#) on page 66, authorization data can be stored on separate servers or as part of a common authorization schema. Similarly, the UPT can be hosted locally or commonly. Your decision may be made based on speed, security, user commonality, or other factors.
6. Deploy authentication, authorization, and user provisioning. These steps are listed in detail later in this chapter.
7. Decide if you want to enable audit logging for these services or not. If yes then configure as explained in the [Audit Logging](#) section of this chapter, beginning on page 24.
8. Input the authorization data using the UPT.

9. Integrate the application code using the integration steps for authentication, authorization, and user provisioning.
10. Test and refine CSM integration with your application. Confirm that your authorization policy and implementation meets requirements.

API Services

The Security API's consist of primary components – Authentication, Authorization, and User Provisioning. The following corresponding managers control these components:

- AuthenticationManager – for Authentication
- AuthorizationManager – for Authorization and User Provisioning.

Authentication Manager

The AuthenticationManager is an interface that authenticates a user against a credential provider. See [Integrating with the CSM Authentication Service](#) below to learn how to integrate with the AuthenticationManager. Developers will work primarily with the login method. Detailed descriptions about each method's functionality and its parameters are present in the CSM API Javadocs.

Authorization Manager

The AuthorizationManager is an interface which provides run-time methods with the purpose of checking access permissions. See [Integrating With the CSM Authorization Service](#) on page 19 to learn how to integrate with the AuthorizationManager. This manager also provides an interface where application developers can provision user access rights.

The user provisioning functionality is primarily used internally by the User Provisioning Tool (UPT) hence there is no integration shown in this document. Detailed descriptions about each method's functionality and its parameters are present in the CSM API Javadocs.

Integrating with the CSM Authentication Service

The CSM authentication service provides a simple and comprehensive solution for user authentication. Developers can easily incorporate the service into their applications with simple configuration and coding changes to their applications. Currently, the authentication service allows authentication using LDAP and RDBMS credential providers only.

The CSM authentication service is available for any application. It can be used exclusively and is effective on its own. However, if you want to use CSM, you do not have to replace the existing authentication in an application. CSM's authentication service can also be used to supplement an application's current authentication mechanism.

Importing the CSM Authentication Manager Class

To use the CSM authentication service, add the last two (highlighted) import statements shown in the sample below to the action classes that require authentication.

```
import gov.nih.nci.abcapp.UserCredentials;
import gov.nih.nci.abcapp.model.Form;
import gov.nih.nci.abcapp.util.Constants;
import gov.nih.nci.security.SecurityServiceProvider;
import gov.nih.nci.security.AuthenticationManager;
```

The class `SecurityServiceProvider` is the common interface class exposed by the CSM application. It contains methods to obtain the correct instance of the `AuthenticationManager` configured for that application.

The client application then uses the `AuthenticationManager` to perform the actual authentication using CSM.

NOTE: The above import statements identify the client application as “abcapp”; this same application is used for the examples provided throughout this document.

Using the CSM Authentication Manager Class

The example provided below illustrates how to use the CSM `AuthenticationManager` service class in the “abcapp” application.

```
UserCredentials credentials = new UserCredentials();
credentials.setPassword(Form.getPassword());
credentials.setUsername(Form.getUsername());
//Get the user credentials from the database and login
try{
    AuthenticationManager authenticationManager =
        SecurityServiceProvider.getAuthenticationManager("abcapp");
    boolean loginOK = authenticationManager.login(credentials.getUsername(),
        credentials.getPassword());
    if (loginOK)System.out.println("SUCESSFUL LOGIN");
    else System.out.println("ERROR IN LOGIN");
}catch (CSEException cse){
    System.out.println("ERROR IN LOGIN");
}
```

The client class obtains the default implementation of the `AuthenticationManager` by calling the static `getAuthenticationManager` method of the `SecurityServiceProvider` class and passing the Application Context name (“abcapp”). It then invokes the `login` method, passing the user’s ID and password.

Note that the application name should match the name used in the configuration files for JAAS to work correctly. If the credentials provided are correct, a Boolean

`true` is returned indicating that the user is authenticated. If there is an authentication error, a `CSEException` is thrown with the appropriate error message embedded.

Installation and Deployment Configurations

This section serves as a guide to help developers integrate applications with CSM's authentication service. It outlines the process required and provides the information developers need to know in order to successfully integrate CSM's authentication service. This includes:

- CSM API JAR placement
- LDAP/RDBMS properties and configuration
- Database properties and configuration
- If audit logging, CLM API JAR placement and configuration.

The CSM authentication service can be used on its own or to supplement an application's current authentication mechanism. Currently, only RDBMS-based and LDAP-based authenticated is supported.

JAR Placement

The CSM API's application is available as a JAR file, `csmapi.jar`, which must be placed in the class path of the application. Along with this JAR, there are many supporting JARs on which the CSM API depends. In case of web applications, these files should be added into the folder: `<application-web-root>\WEB-INF\lib`.

Configuring Lock-out In Authentication Manager

If desired, you can use the optional user lockout feature provided by CSM's default JAAS implementation of Authentication Manager.

There are three properties (listed below) that must be properly configured in order to use the lockout manager. If any of the three do not have a valid value, the lockout manager is disabled. To be valid, these values must be non-zero positive integers.

- `lockout-time`: This property specifies the time (in milliseconds) that the user will be locked out after the configured number of unsuccessful login attempts has been reached. **Default value = 180000 milliseconds.**
- `allowed-login-time`: This property specifies the time in milliseconds in which the configured number of unsuccessful login attempts must occur in order to lock the user out. **Default value = 60000 milliseconds**
- `allowed-attempts`: This property specifies the number of unsuccessful login attempts allowed before the user account is locked out. **Default value = 3.**

Alternatively, in the client application class, you can call and provide values for the lockout parameters by using the following method of `SecurityServiceProvider` Class:

```
public static AuthenticationManager getAuthenticationManager(String
applicationContextName, String lockoutTime, String allowedLoginTime, String
allowedAttempts) throws CSEException, CSCConfigurationException
```

RDBMS Credential Provider Properties and Login Module Configuration

In order to authenticate using an RDBMS database, developers must provide:

- The details about the database,
- The actual query which will make the database calls.

The CSM goal is to make authentication work with any compatible application or credential provider. To do this, CSM uses the same Login Modules to perform authentication, requiring these modules to possess a standard set of properties.

The properties needed to establish a connection to the database include:

- **Driver** - The database driver loaded in memory to perform database operations.
- **URL** - The URL used to locate and connect to the database.
- **User** - The user name used to connect to the database.
- **Password** - The password used to connect to the database.

The following property provides the database query to be used to retrieve the user:

- **Query** - The query that will be fired against the RDBMS tables to verify the user ID and the password passed for authentication.

The next sections provide instructions for configuring a login module using either JAAS or the JBoss *login-config.xml* file.

Configuring an RDBMS Login Module in JAAS

Developers can configure a login module for each application by making an entry in the JAAS configuration file for that application name or context.

The general format for making an entry into the configuration files is shown below:

```
Application 1 {
    ModuleClass Flag  ModuleOptions;
    ModuleClass Flag  ModuleOptions;
    ...
};
Application 2 {
    ModuleClass Flag  ModuleOptions;
    ...
};
Application 1 {
```

For “abcapp”, which uses *RDBMSLoginModule*, the JAAS configuration file entry would appear as follows:

```
abcapp
```

```
{
    gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule
    Required
    driver="oracle.jdbc.driver.OracleDriver"
    url="jdbc:oracle:thin:@oracle_db_server:1521:abcappdb"
    user="USERNAME"
    passwd="PASSWORD"
    query="SELECT * FROM users WHERE username=? and password=?"
}
```

The configuration file entry shown above contains the following:

- The **application** is *abcapp*.
- The **ModuleClass** is *gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule*.
- The **Required** flag indicates that authentication using this credential source is a must for overall authentication to be successful.
- The **ModuleOptions** are a set of parameters that are passed to the *ModuleClass* to perform its actions.
- The **database driver** is passed as:
driver="oracle.jdbc.driver.OracleDriver".
- The **URL** for the database is:
url="jdbc:oracle:thin:@oracle_db_server.nci.nih.gov:1521:abcappdb"
- The **User** (user ID) is passed as: *user="USERNAME"*
- The **Password** given is passed as: *passwd="PASSWORD"*
- The **Query** sent to the database to verify the user is:
*query="SELECT * FROM users WHERE username=? and password=?"*

The example shown indicates that since the “abcapp” application has only one credential provider, only one corresponding entry is made in the configuration file.

If the application uses multiple credential providers, the LoginModules can be stacked. In addition, as shown by the generic format for the configuration file (seen on the previous page), a single configuration file can contain entries for multiple applications.

Configuring an RDBMS Login Module in JBOSS

If the application uses a JBoss server, you can configure a login module by using the JBoss `login-config.xml` file, located at: `{jboss-home}\server\{server-name}\conf\login-config.xml`.

The example below shows how the entry for the “abcapp” application would appear:

```
<application-policy name = "abcapp">
    <authentication>
```

```

    <login-module code =
"gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule" flag =
"required" >
        <module-option name="driver"> oracle.jdbc.driver.OracleDriver</module-
option>
        <module-option
name="url">jdbc:oracle:thin:@oracle_db_server:1521:abcappdb</module-option>
        <module-option name="user">USERNAME</module-option>
        <module-option name="passwd">PASSWORD</module-option>
        <module-option name="query">SELECT * FROM users WHERE username=?
and password=?</module-option>
        <module-option name="encryption-enabled">YES</module-option>
    </login-module>
</authentication>
</application-policy>

```

As shown in this example:

- The **application-policy** specifies the application for which we are defining the authentication policy: *abcapp*.
- The **login-module** is the LoginModule class to be used to perform the authentication task:
gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule
- The **flag** provided is *required*.
- The **module-options** list the parameters passed to the LoginModule to perform the authentication task. In the above example they are:

```

<module-option name="driver">oracle.jdbc.driver.OracleDriver</module-
option>
<module-option name="url">jdbc:oracle:thin:@cbiodb2-
d.nci.nih.gov:1521:cbdev</module-option>
<module-option name="user">USERNAME</module-option>
<module-option name="passwd">PASSWORD</module-option>
<module-option name="query">SELECT * FROM users WHERE
username=? and password=?</module-option>
<module-option name="encryption-enabled">YES</module-option>

```

Enabling Encryption in the RDBMS Login Module

Encrypted passwords have been supported since CSM v3.2, and for version 4.0, encrypted passwords by default and stored them in the CSM database.

If an application is using the CSM User Table as the credential provider, you must use a *module-option* entry (shown below and in the *abcapp* example above) in the JBoss `login-config.xml` to tell the RDBMS Login Module to use encryption.

```
<module-option name="encryption-enabled">YES</module-option>
```

Adding the **encryption-enabled** option with a *YES* value uses the default CSM encryption to encrypt the user entered password before verifying it against CSM's User Table.

LDAP Credential Provider Properties and Login Module Configuration

The CSM default implementation provides an LDAP-based authentication module for use by client applications. In order to authenticate using LDAP, developers must provide:

- Details about the LDAP server, and
- The label for the user ID Common Name (CN) or User Identification (UID) in the LDAP server.

The properties needed to establish a connection to the LDAP include:

- **IdapHost** – The URL of the actual LDAP server.
- **IdapSearchableBase** – The base of the LDAP tree from where the search should begin.
- **IdapUserIdLabel** – The actual user ID label used for the CN entry in LDAP.

For LDAP credential providers that do not allow anonymous binding to verify user credentials, you will need to provide the common admin user name and password as additional properties to the LDAP Login Module configuration. Those properties are as follows:

- **IdapAdminUserName** – The fully qualified name of the common admin user, or the look up to be used to bind to the LDAP server in order to verify individual user ids and passwords.
- **IdapAdminPassword** – Password for the LDAP admin user mentioned above.

See [Configuring LDAP Login Module Using Anonymous Bind](#) on page 18 for details.

Configuring LDAP Login Module in JAAS

Developers can configure a login module for each application by making an entry in the JAAS configuration file for that application name or context.

The general format for making an entry into the configuration files is shown below:

```
Application 1 {  
    ModuleClass Flag  ModuleOptions;  
    ModuleClass Flag  ModuleOptions;  
    ...  
};  
Application 2 {  
    ModuleClass Flag  ModuleOptions;  
    ...
```

```
};
Application 1 {
```

The example below shows the JAAS config file entries needed for *abcapp* to use the LDAP Login Module.

```
abcapp
{
    gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule Required
    ldapHost= "ldaps://ncids2b.nci.nih.gov:636"
    ldapSearchableBase= "ou=nci,o=nih"
    ldapUserIdLabel="cn";
};
```

As shown in the above example:

- The **application** is *abcapp*.
- The **ModuleClass** is *gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule*.
- The **Required** flag indicates that authentication using this credential source is a must for overall authentication to be successful.
- The **LDAP details** are passed as follows:


```
ldapHost="ldaps://ncids2b.nci.nih.gov:636"
ldapSearchableBase= "ou=nci,o=nih"
ldapUserIdLabel="cn"
```

The example shown indicates that since the “abcapp” application has only one credential provider, only one corresponding entry is made in the configuration file.

If the application uses multiple credential providers, the LoginModules can be stacked. In addition, as shown by the generic format for the configuration file, a single configuration file can contain entries for multiple applications.

Configuring LDAP Login Module in JBOSS

If the application uses a JBoss server, you can configure a login module by using the JBoss `login-config.xml` file, located at: `{jboss-home}\server\{server-name}\conf\login-config.xml`.

The example below shows how the entry for the “abcapp” application would appear:

```
<application-policy name = "abcapp">
  <authentication>
    <login-module code =
"gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule" flag =
"required" >
```

```
<module-option name="ldapHost">ldaps://ncids2b.nci.nih.gov:636</module-  
option>  
<module-option name="ldapSearchableBase">ou=nci,o=nih</module-option>  
<module-option name="ldapUserIdLabel">cn</module-option>  
</login-module>  
</authentication>  
</application-policy>
```

As shown in this example:

- The **application-policy** specifies the application for which we are defining the authentication policy: *abcapp*.
- The **login-module** is the LoginModule class to be used to perform the authentication task:
gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule.
- The **flag** provided is *required*.
- The **module-options** list the parameters passed to the LoginModule to perform the authentication task. In the above example they are:

```
<module-option  
name="ldapHost">ldaps://ncids2b.nci.nih.gov:636</module-option>  
<module-option name="ldapSearchableBase">ou=nci,o=nih</module-  
option>  
<module-option name="ldapUserIdLabel">cn</module-option>
```

Configuring LDAP Login Module Using Anonymous Bind

If an application uses an LDAP server that does not support using anonymous binds to perform lookups, you will need to specify an Admin ID (or a lookup user) and password in order to bind to the LDAP server to verify user names and passwords. To do this you must provide additional parameters for the LDAP LoginModule entry in the JAAS Login Configuration file.

Below is an example of the JBoss `login-config.xml` file containing an anonymous bind configuration for the *abcapp* application:

```
<application-policy name = "OpenLDAP">  
<authentication>  
<login-module code =  
"gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule" flag =  
"required" >  
<module-option name="ldapHost">ldap://ncicbds-dev.nci.nih.gov:389</module-  
option>  
<module-option name="ldapSearchableBase">ou=csm,dc=ncicb-  
dev,dc=nci,dc=nih,dc=gov</module-option>  
<module-option name="ldapUserIdLabel">uid</module-option>
```

```

    <module-option
name="ldapAdminUserName">uid=csmAdmin,ou=csm,dc=ncicb-
dev,dc=nci,dc=nih,dc=gov</module-option>
    <module-option name="ldapAdminPassword">PASSWORD</module-option>
  </login-module>
</authentication>
</application-policy>

```

As shown in the example:

- The **application-policy** specifies the application for which we are defining the authentication policy: *abcapp*.
- The **login-module** is the LoginModule class to be used to perform the authentication task: *gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule*.
- The **flag** provided is *required*.
- The **module-options** list the parameters passed to the LoginModule to perform the authentication task. In this example they are:

```

  <module-option
name="ldapHost">ldaps://ncids2b.nci.nih.gov:636</module-option>
  <module-option name="ldapSearchableBase">ou=nci,o=nih</module-
option>
  <module-option name="ldapUserIdLabel">cn</module-option>
  <module-option
name="ldapAdminUserName">uid=csmAdmin,ou=csm,dc=ncicb-
dev,dc=nci,dc=nih,dc=gov</module-option>
  <module-option
name="ldapAdminPassword">PASSWORD</module-option>

```

Activating CLM Audit Logging

In order to activate the CLM's audit logging capabilities for Authorization, the user needs to follow the steps to deploy audit logging service as mentioned in the [Audit Logging](#) section on page 24.

Integrating With the CSM Authorization Service

The CSM security APIs can be used programmatically to facilitate security needs at runtime. The APIs have been written using Java, so it is assumed that developers using them are familiar with the Java language.

This section provides instructions for integrating the CSM APIs with JBoss. The integration is flexible enough to meet most users' needs, depending on the number of applications hosted on the JBoss server, and whether or not a common schema is used.

The following scenarios are supported for using CSM APIs with JBoss:

- JBOSS hosts a number of applications
 - using a common schema
 - using separate schemas
- BOSS hosts only one application
 - using a common schema
 - using separate schemas

Importing and Using the CSM Authorization Manager Class

To import the CSM authorization service, add the last two (highlighted) import statements shown in the sample below to the action classes that require authentication.

```
import gov.nih.nci.abcapp.UserCredentials;  
import gov.nih.nci.abcapp.model.Form;  
import gov.nih.nci.abcapp.util.Constants;  
import gov.nih.nci.security.SecurityServiceProvider;  
import gov.nih.nci.security.AuthorizationManager;
```

The class `SecurityServiceProvider` is the common interface class exposed by the CSM application. It contains methods to obtain the correct instance of the `AuthorizationManager` configured for that application.

NOTE: The above import statements identify the client application as “abcapp”; this same application is used for the examples provided throughout this document.

The client application then uses the `AuthorizationManager` to perform the actual authentication using CSM. The example below illustrates how to use the CSM `AuthorizationManager` service class in the “abcapp” application.

```
try {  
    AuthorizationManager authorizationManager =  
        SecurityServiceProvider.getAuthorizationManager("abcapp");  
    boolean hasPermission = authorizationManager.checkPermission("user  
name", "resource name", "operation");  
    if (hasPermission){ System.out.println("PERMISSION GRANTED.");  
    }else{ System.out.println("PERMISSION DENIED ");    }  
}catch ( CSEException cse){  
    System.out.println("ERROR IN AUTHORIZATION ");  
}
```

The client class obtains the default implementation of the `AuthorizationManager` by calling the static `getAuthorizationManager` method of the `SecurityServiceProvider` class and passing the Application Context name (“abcapp”). It then invokes the `checkPermission` method, passing the user’s ID, the resources it is trying to access, and the operation it wants to perform.

Note that the application name should match the name used in the configuration files as well as configured for the databases for authorization to work correctly. If the user has the required access permission, a Boolean `true` is returned, indicating that the user is authorized. If there is an authorization error, a `CSEException` is thrown with the appropriate error message embedded.

Software Products and Scripts

The tables below describe the products and files used for authorization.

Software Product	Description
JBoss Server	JBoss is the leading open source, standards-compliant, J2EE-based application server implemented in 100% Pure Java. Most caCORE applications use JBoss to host their applications.
MySQL Database	MySQL is an open source database. Its speed, scalability and reliability make it a popular choice for Web developers. CSM recommends storing authorization data in a MySQL database because it is a light database, easy to manage and maintain.
Oracle Database	Oracle's relational database was the first to support the SQL language, which has since become the industry standard. It is a proprietary database which requires licenses.
Hibernate	Hibernate is an object/relational persistence and query service for Java. CSM requires developers to modify a provided Hibernate configuration file (<code>hibernate.cfg.xml</code>) in order to connect to the appropriate application authorization schema.

Figure 2-1 Authorization software products

File	Description
<code>hibernate.cfg.xml</code>	The sample XML file which contains the hibernate-mapping and the database connection details.
<code>AuthSchemaMySQL.sql</code> OR <code>AuthSchemaOracle.sql</code> OR <code>AuthSchemaPostgres.sql</code>	This Structured Query Language (SQL) script is used to create an instance of the Authorization database schema which will be used for the purpose of authorization. In 3.0.1 and subsequent releases, this script populates the database with CSM Standard Privileges that can be used to authorize users. The same script can be used to create instances of authorization schema for a variety of applications.
<code>DataPrimingMySQL.sql</code> OR <code>DataPrimingOracle.sql</code> OR <code>DataPrimingPostgres.sql</code>	This SQL script is used for priming data in the authorization schema. Note that if the authorization database is going to host the UPT also then you need to use UPT Data Priming Scripts instead and add the application through the UPT.
<code>mysql-ds.xml</code> OR <code>oracle-ds.xml</code> OR <code>Postgres-ds.xml</code>	This file contains information for creating a datasource. One entry is required for each database connection. Place this file in the JBoss deploy directory.

Figure 2-2 Authorization Configuration and SQL Files

Installation and Deployment Configurations

This section serves as a guide to help developers integrate applications with CSM's Authorization Service. It outlines a step by step process that addresses what developers need to know in order to successfully integrate CSM's Authorization, which includes:

- CSM API jar placement
- Database properties and configuration
- If audit logging, CLM API jar placement and configuration.

JAR Placement

The CSM application is available as a JAR which needs to be placed in the classpath of the application. Along with this JAR, there are many supporting JARs on which the CSM API depends. These should be added into the following folder:

```
<application-web-root>\WEB-INF\lib.
```

Database Properties and Configuration

You can use either MySQL or Oracle as your database of choice to host the authorization data. Since the instructions in this section provide steps for both database types, be sure you follow the appropriate instructions based on your database selection.

When deploying Authorization, application developers may want to make use of a previously-installed common Authorization Schema. In this case, a database already exists so you do not need to create one (skip the procedures in Step 1 below).

NOTE: The Authorization Schema used by the run-time API and the UPT must be the same.

Step 1: Create and Prime the Database

Use the following steps (if necessary) to create a new database and prime it for authorization deployment. These procedures also install a new Authorization Schema. If you are using a previously-installed common Authorization Schema, skip these instructions as your database and schema already exist.

1. Log into the database using an account id that has permission to create new databases.
2. In the `AuthSchemaMySQL.sql` or `AuthSchemaOracle.sql` script, replace the `<<database_name>>` tag with the name of the authorization schema (e.g. "caArray").

NOTE: The Authorization Schema used by the run-time API and the UPT must be the same.

3. Save and then run the edited script from the database prompt. This should create a database with the `<<database_name>>` you provided in the script. The new database will also include CSM Standard Privileges.

4. In the `DataPrimingMySQL.sql` or `DataPrimingOracle.sql` file, do the following:
 - a. Replace the `<<application_context_name>>` with the name of application (e.g., “abcapp”). In later configuration steps, this will be called Application Context Name. This is the key to deriving security for the application.
 - b. Replace the `<<super_admin_login_id>>`, `<<super_admin_first_name>>` and `<<super_admin_last_name>>` entries with the super admin user’s login id, first name, and last name.

NOTE: The default password is always “changeme” and should be used for logging into the application’s UPT for the first time. After initial login, change this password immediately.
5. Save and then run the edited script from the database prompt. This should populate the database with the initial data. Verify this by querying the application table. It should include one record only.

Step 2: Configure the Datasource

Use the steps below to modify the database file that contains information for creating a datasource.

1. In the `mysql-ds.xml` or `oracle-ds.xml` file, edit the entries as follows: One entry is required for each database connection.
 - a. Replace the `<<application_context_name>>` tag with the name of the authorization schema. If you created a database using the above procedures, this is the name entered as the `<<database_name>>` in Step 2 of those procedures. Otherwise use the name of your existing authorization schema.
 - b. Replace the `<<database_user_id>>` and `<<database_user_password>>` with the user id and password of the user account that will be used to access the authorization schema.
 - c. Replace the `<<database_url>>` with the URL needed to access the authorization schema residing on the database server.
2. Save and then place the edited file into the JBoss deploy directory.

Below is an example of a configured `mysql-ds.xml` file. The name of the authorization schema in the example is “csmupt”.

```
<datasources>
  <local-tx-datasource>
    <jndi-name>csmupt</jndi-name>
    <connection-url>jdbc:mysql://mysql_db:3306/csmupt</connection-url>
    <driver-class>org.gjt.mm.mysql.Driver</driver-class>
    <user-name>name</user-name>
```

```
<password>password</password>
</local-tx-datasource>
<local-tx-datasource>
  <jndi-name>security</jndi-name>
  <connection-url>jdbc:mysql://mysql_db:3306/csd</connection-url>
  <driver-class>org.gjt.mm.mysql.Driver</driver-class>
  <user-name>name</user-name>
  <password>password</password>
</local-tx-datasource>
</datasources>
```

Audit Logging

This section serves as a guide to help developers integrate audit logging for CSM. It outlines a step-by-step process that addresses what developers need to know in order to successfully integrate the Common Logging Module (CLM), including:

- Jar placement,
- Configuring the JDBC Appender configuration file or the regular `log4j` configuration file.

In an effort to make CSM compliant with CRF 21/ part 11, CSM provides auditing and logging functionality. Currently CSM uses `log4j` for logging application logs. However, CRF21/ part 11 requires that certain messages are logged in a specific way. For example, all objects should be logged in a manner that allows them to be audited at later stage.

There are two types of audit logging: event logging and object state logging. These audit logging capabilities are provided through the CLM API that is available from `clm.jar`.

Audit logging is configurable by the client application developer via an application property configuration file. By placing the `clm.jar` file, along with the application property configuration file, in the same class path as the `csmapl.jar` file, the client application is able to utilize the built-in audit logging functionality. The logging results can be saved in a database or in a flat text file, depending on the configuration. In addition, logging can be enabled or disabled for any fully qualified class name.

JAR Placement

The audit logging application is available as a JAR file, called `clm.jar`. This JAR file along with the `csmapl.jar` must be placed in the classpath of the application.

If the client application is integrating the CSM APIs as part of a web application on JBoss, then `clmwebapp.jar` should be placed in the `lib` directory of the `WEB-INF` folder, and `clm.jar` should be placed in the common `lib` directory of JBoss.

Integrating CLM APIs with CSM APIs

The various services exposed by CSM have been enabled for the purpose of using CLM for audit and logging. If configured properly, client applications using the CSM APIs can enable the internal CLM-based audit and logging capabilities.

The sections that follow discuss the major components provided by the CLM APIs, and how they work to enable the audit logging capabilities provided by CSM.

NOTE: CSM is capable of performing both event and object state audit logging only for the operations and data pertaining to CSM. In order to use CLM features without using CSM, the client application can be downloaded and install CLM separately. In this case CLM can be used (without CSM) to provide event logging and automated object state logging capabilities using the special appender and schema. The log locator tool can also be used for the purpose of viewing the logs.

Event Logging

Both the Authentication and Authorization services have been modified to enable the logging of every event that the user performs.

For the authentication service, the CSM APIs log the login and logout events of the user. In addition, when a user lockout event occurs, a log is generated that records the username that was locked out.

For the authorization service, the CSM APIs track all *create*, *update*, and *delete* operations invoked by the client application. *Read* operations are not logged because they cause no changes to the data.

The UPT can perform all of the audit and logging services because it uses the CSM APIs (which use CLM APIs) to perform operations on the database.

Since the CLM APIs are based on `log4j`, the following logger names are used in the CSM APIs to perform the event logging:

Authentication Event Logger Name: *CSM.Audit.Logging.Event.Authentication*

Authorization Event Logger Name: *CSM.Audit.Logging.Event.Authorization*

The `log4j` log level used for all event logs is INFO.

In order to enable event logging, these loggers should be configured in the `log4j.xml` config file of Jboss as shown in the [JDBC Appender](#) section below.

Object State Logging

The Authorization Service of the CSM is enabled to log the object state changes using the automated object state logger available through the CLM APIs. This logger tracks all object state changes made using the CSM APIs. It also uses the `log4j`-based CLM APIs and the following logger name:

Authorization Object State Logger Name:
CSM.Audit.Logging.ObjectState.Authorization

The `log4j` log level used for all object state logs is INFO.

In order to enable object state logging, this logger should be configured in the `log4j.xml` config file of JBoss as shown in the [JDBC Appender](#) section below.

User Information

For the purpose of audit logging, in order to track which user is performing the specific operation being logged, CSM needs to obtain user information like user id and session id, and also the organization to which the user belongs. Since these values are only available with the client application, the client application needs to pass this information to the CSM APIs. To accomplish this, the client application must use the utility class *UserInfoHelper* provided by the underlying CLM APIs. This information must be set before calling any of the *create*, *update* or *delete* functions of the CSM APIs.

Common Logging Database

The Common Logging Database is the persistence storage used by the JDBC Appender to store the audit logs. The Log Locator application of CLM connects to this database to allow a user to browse the logs.

JDBC Appender

The CLM provides an asynchronous JDBC Appender to persist the audit logs. Thus, an application that wants to enable audit logging for the CSM APIs should also configure this Appender.

A sample `log4j` file entry is shown below:

```
<?xml version="1.0" encoding="UTF-8" ?><!DOCTYPE log4j:configuration SYSTEM
".\log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
  <appender name="CLM_APPENDER"
class="gov.nih.nci.logging.api.appender.jdbc.JDBCAppender">
    <param name="application" value="csm" />
    <param name="maxBufferSize" value="1" />
    <param name="dbDriverClass" value="org.gjt.mm.mysql.Driver" /> <param
name="dbUrl"
value="jdbc:mysql://<<SERVER_NAME>>:<<PORT>>/<<CLM_SCHEMA_NAME>>
" /> <param name="dbUser" value="<<DB_USER>>" />
    <param name="dbPwd" value="<<PASSWORD>>" />
    <param name="useFilter" value="true" />
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value=":: [%d{ISO8601}] %-5p
%c{1}.%M() %x - %m%n" />
    </layout>
  </appender>

  <category name="CSM.Audit.Logging.Event.Authentication">
    <level value="info" />
    <appender-ref
ref="CLM_APPENDER" />
  </category>

  <category name="CSM.Audit.Logging.Event.Authorization">
    <level value="info" />
    <appender-ref ref="CLM_APPENDER" />
  </category>
```

```

<category name="CSM.Audit.Logging.ObjectState.Authorization">
  <level value="info" />
  <appender-ref ref="CLM_APPENDER" />
</category>
</log4j:configuration>

```

Deployment Steps

The sections below provide the basic instructions necessary for enabling the audit logging capabilities provided by CSM (via CLM).

Step 1: Create and Prime the MySQL Logging Database

A database must be created that will persist the audit logs generated as a function of using of the CSM APIs. Refer to the *CLM Application Developer's Guide* for information on creating and priming the database for storing audit logs.

Step 2: Configure the log4j.xml file for JBoss

Use the sample `log4j` file provided in the CSM release to configure the `log4j.xml` file for JBoss. See the sample `log4j` file entry shown above.

1. Open the `log4j.xml` file and do the following:
 - a. Replace the `<<SERVER_NAME>>`, `<<PORT>>` and `<<CLM_SCHEMA_NAME>>` entries with the appropriate values. These are the server name, port, and schema name for the database created and primed in the step noted above.
 - b. Replace the `<<DB_USER>>` and `<<PASSWORD>>` entries with the corresponding values for the user that has access to the schema.
2. Configure the appropriate loggers based on whether the application wants to enable event audit logging for authentication and authorization, or object state audit logging for authorization. See [Event Logging](#) or [Object State Logging](#) on page 25 as needed.

NOTE: The names of the loggers must *not* differ from the sample.

In the case of UPT the same `log4j` config file can be used.

Step 3: View the Logs

CLM provides a web-based locator tool that can be used to browse audit logs. The configuration steps for setting up the browser are defined in more detail in the CLM Log Locator Tool (LLT) User Guide located on GForge at:

https://gforge.nci.nih.gov/docman/index.php?group_id=49&selected_doc_group_id=2523&language_id=1.

Chapter 3 User Provisioning Tool (UPT)

The CSM User Provisioning Tool (UPT) is a web-based application used to provision an application's authorization data. The UPT provides the ability to create, edit, and delete authorization data elements such as roles, protection elements, users, groups, etc. Once the necessary elements have been created, the UPT also provides the ability to associate those elements with each other as needed. The runtime API can then use this authorization data to authorize user actions.

The intended audience for this section includes all users of the UPT including Super Administrators who may add applications and assign administrators for those applications, and Administrators who will perform provisioning for a particular application. This chapter provides an overview of the UPT, outlines a suggested workflow, and explains how to perform all UPT operations.

This chapter also explains how to deploy the UPT from start to finish – from uploading the Web application Archive (WAR) file, to editing configuration files, to synching the UPT with the application.

Workflow

The UPT includes two modes: Super Admin and Admin. The Super Admin operations are typically performed first, as they register the application and application administrators. After that, the primary mode operations, including authorization user provisioning, can occur.

Super Admin – To Register Applications and Admins

When first deploying the UPT for a particular application, the developer registers the application in Super Admin mode. Once the application is registered, the Super Admin can add the users who will serve as application administrators. The Super Admin can also register additional applications as they become available.

For details, see [Super Admin Mode](#) beginning on page 39.

Admin – To Administer a Registered Application

The primary or Admin mode of the UPT is used to perform user provisioning for a particular application. The Admin mode follows a simple workflow of creating elements, assigning them, and then associating them.

For details, see [Admin Mode](#) beginning on page 47.

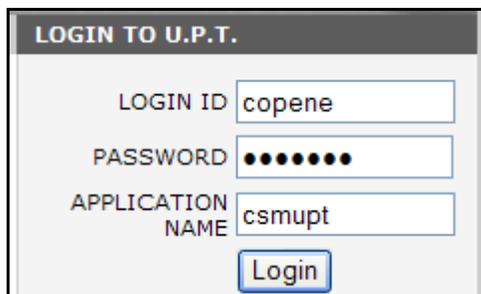
Login

The Login page of the UPT includes summary text, What's New, Did You Know, and most importantly the Login section itself. Logging into the UPT requires a Login ID, Password, and Application Name.

For a majority of UPT implementations, the NCICB LDAP serves as the authentication mechanism. This means that the user's Login ID is the same as the

user's NCICB username. In the figures below, the demonstration user is Eric Copen, whose NCICB username is "copene". Similarly, the password is the same as the NCICB password. The rules from the authentication system are applied to the username and password.

If logging on as a Super Admin, enter the application Name *csmupt* as shown in Figure 3-1 below. If logging in as an Admin, enter the appropriate application name as shown in Figure 3-2 below, where the application name is *security*.



LOGIN TO U.P.T.

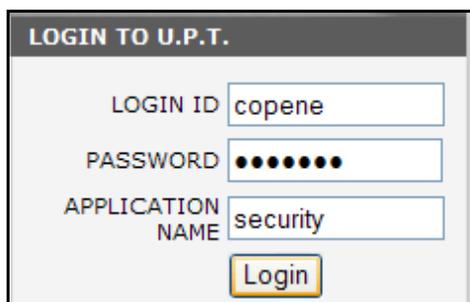
LOGIN ID copene

PASSWORD ●●●●●●

APPLICATION NAME csmupt

Login

Figure 3-1 Login as Super Admin



LOGIN TO U.P.T.

LOGIN ID copene

PASSWORD ●●●●●●

APPLICATION NAME security

Login

Figure 3-2 Login as Admin

Since UPT uses CSM's Authentication Manager, it can be configured to lock a user out if they try to make an unauthorized entry into the UPT. If configured appropriately, the UPT will the user out after a pre-configured number of unsuccessful attempts have been reached within the allowed login time frame. Once locked out, the user can log in only after the configured amount of lockout time has elapsed. This provides security from hacking attempts to break into the UPT.

Common Basic Functions

Within the UPT, there are several common operations that are repeated for most elements. These operations include Create New, Search, Update, Delete, and Assign/Associate. The sections that follow describe in more detail how these operations are performed.

Create New

The same basic steps can be followed to create any element. Use the procedure provided below as a guide for creating a new element. These particular instructions create a new User element.

To create a new element:

1. On the element Home page, select the **Create A New {element}** link. Figure 3-3 below shows the create new and select existing user links.

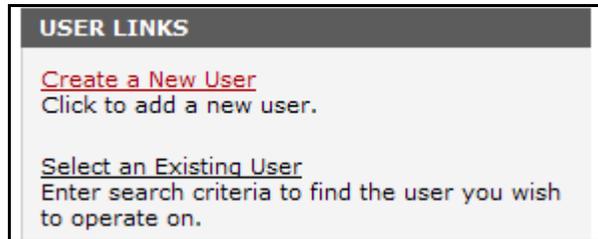


Figure 3-3 User Links section of the Home Page

2. Enter the details for the element you are creating. Figure 3-4 below shows the details for entering new user information.

ENTER THE NEW USER DETAILS	
* User Login Name	<input type="text" value="smithj"/>
User First Name	<input type="text" value="John"/>
User Last Name	<input type="text" value="Smith"/>
User Organization	<input type="text" value="NIH"/>

Figure 3-4 Enter New User Details form

3. Select **Add** to save the new element to the database. This save occurs immediately.

The **Back** button returns you to the previous page (in this instance, the Home page). The **Reset** button clears the data from the form.

NOTE: No data is saved until the **Add** button is selected.

Upon a successful save, the system displays **Add Successful** just below the menu and above the text.

After you have created the necessary element(s) in Super Admin mode, when you log into Admin mode, you will see an additional set of buttons below the details table. These buttons are used to administer the application and are discussed in more detail in the [Admin Mode](#) section on page 47.

Example Error Messages

When creating new elements, the UPT performs basic data validation, including field lengths and formats. If this validation fails, you will receive an error message explaining the problem.

Figure 3-5 shows the message that appears when a user enters an improperly formatted email address.

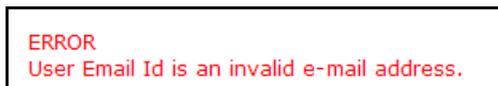


Figure 3-5 Error message for invalid e-mail address

Figure 3-6 shows the message that appears if you attempt to create a new user using a username already present in the system.



Figure 3-6 Error message for duplicate user entry

Search for and Select Existing Elements

In order to make changes to an existing element, you must find and select the element. Choosing to Select and Existing {element} opens a search page, allowing you to enter search criteria.

The same basic steps can be followed to find and select any element. Use the procedure provided below as a guide. These particular instructions search for and select an existing Role element.

To find and select an existing element:

1. On the element Home page, select the **Select an Existing {element}** link. Figure 3-7 below shows the create new and select existing role links.

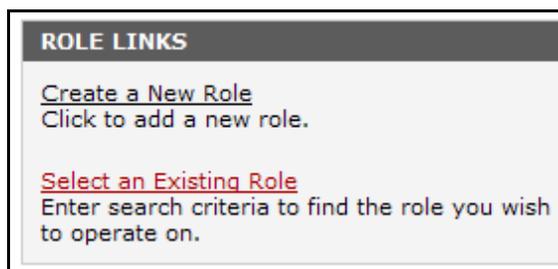


Figure 3-7 Role Links section of the Home page

2. In the Search Criteria box that appears, enter the necessary search criteria. Use the asterisk (*) as a wildcard character. Searches are not case-sensitive.

ENTER THE ROLE SEARCH CRITERIA	
Role Name	<input type="text" value="*1"/>
<input type="button" value="Back"/> <input type="button" value="Search"/> <input type="button" value="Reset"/>	

Figure 3-8 Search for Role text box

For example, searching for Role* returns *Role_name_1*, *Role_name_2*, or any other role beginning with “Role”. A search of *1 returns anything ending with “1” such as *Role_name_1*, *Role_name_101*, *Role_name_51*, etc.

3. Click **Search** to search for matching results.

The **Back** button returns you to the previous page (in this instance, the Home page). The **Reset** button clears the data from the search form.

When the search has completed, the system returns a list of elements matching the criteria you entered. The results are sorted alphabetically.

SEARCH RESULTS		
Select	Role Name	Role Description
<input type="radio"/>	Role_name_1	Role_Desc_1
<input type="radio"/>	Role_name_101	Role_Desc_10
<input type="radio"/>	Role_name_51	Role_Desc_5

Figure 3-9 Search Results for Role element search

4. If the element you want appears in the list, select the radio button of that element and then click **View Details** (located below the search results table). You can select only one element at a time to view.

<input checked="" type="radio"/>	Role_name_1	Role_Desc_1
----------------------------------	-------------	-------------

Figure 3-10 Element with radio button selected

If the element you want does NOT appear, click **Back** to return to the Search Criteria page and alter your search. If you cannot find the element you are looking for, it may not reside in the system.

Once the Details page of the selected element appears, you can use the fields to update that information, or select to delete the element entirely if appropriate. See the next section on updating elements or the following section on deleting elements for more information as needed.

If the search results in no matches, the system displays an error message indicating there are no matches. Modify the search criteria, and repeat until the intended results appear.

Update Elements

After you have searched for and selected to view the details of an element, if necessary you can make changes to the details and update the element's information in the database.

The same basic steps can be followed to update the details of any element. Use the procedure provided below as a guide. These particular instructions update an existing Protection element.

To update an existing element:

1. Using the search and select procedure provided above, open the details page for the element.

PROTECTION ELEMENT DETAILS	
* Protection Element Name	TestPE Name1103749550261
Protection Element Description	Test Desc

Figure 3-11 Protection element details form

2. Where necessary, replace the existing text in the details form and select **Update**.

PROTECTION ELEMENT DETAILS	
* Protection Element Name	TestPE Name1103749550261
Protection Element Description	This is my new text I want to update.

Figure 3-12 Details form with new description text

Upon a successful update, the system displays **Update Successful** just below the menu and above the text.

Whenever updates are made, the UPT performs basic data validation, including checks for field lengths and formats. The systems also checks for duplicates, preventing you from changing the element name to one that already exists. If any of the validation checks fail, you will receive an error message explaining the problem. See [Example Error Messages](#) above for examples of these messages.

Deleting an Existing Element

After you have searched for and selected to view the details of an element, if necessary you can select to delete the element from the database.

The same basic steps can be followed to delete any element. Use the procedure provided below as a guide. These particular instructions delete an existing Group element.

To update an existing element:

1. Using the search and select procedure provided above, open the details page for the element.
2. Click the **Delete** button located on the element details screen.
3. A confirmation dialog box appears asking if you are sure you want to delete the record. Click **OK** to confirm. Clicking **Cancel** negates the operation and returns you to the Details screen.

Upon confirming the deletion, the system returns you to the element home page and displays **Delete Successful** in blue text.

Assignments and Associations

The elements Role, Protection Group, and Group are simply collections of other elements – Privileges, Protection Elements, and Users respectively. Provisioning is the process of assigning elements to these “grouping” elements and deassigning (removing) elements from a grouping element.

For example, assigning Users to Groups greatly improves the ease with which an admin can provision access rights. An Admin can instantly assign a role and protection group to an entire group of users instead of repeating the same assignment for each individual user.

The process of assigning and deassigning elements differs, depending on whether you are working from the group element perspective or from the individual element perspective. Each of these processes are outlined briefly in the sections that follow. Specific instructions for assigning each type of element appears in the detail section for administering that element.

NOTE: The buttons that allow for provisioning of elements only appear in Admin Mode; Super Admins cannot do element provisioning. For more details on the available functionality for each mode, see [Super Admin Mode](#) on page 39, and [Admin Mode](#) on page 47.

Assigning or Deassigning Elements - Individual to Group

If you logged into Admin Mode and are working with an individual element (user, protection element, or privilege), you will notice a series of buttons on the Details page for the element that looks something like Figure 3-13 below:



Figure 3-13 User Element Details option buttons in Admin Mode

Depending on the type of element you have open, the button labels will vary but each functions in essentially the same way. They allow you to assign or associate other elements (group or individual) to the currently open element.

The left-most button contains the label **Associated {group element}** and is used to associate the open individual element to one or more group elements. For example, if the currently open element is a Protection Element, the leftmost button reads **Associated PG**, because a Protection Group is a grouping of Protection Elements.

If the open element is a user element, the button reads **Associated Groups** as shown above, because a Group is a collection of Users.

Clicking the Associated {element} button opens a screen that typically contains two lists: an Available {group element} list and an Assigned {group element} list, as shown in . These lists, as the labels indicate, show you which groups are available for assignment, and to which groups the element is already assigned.

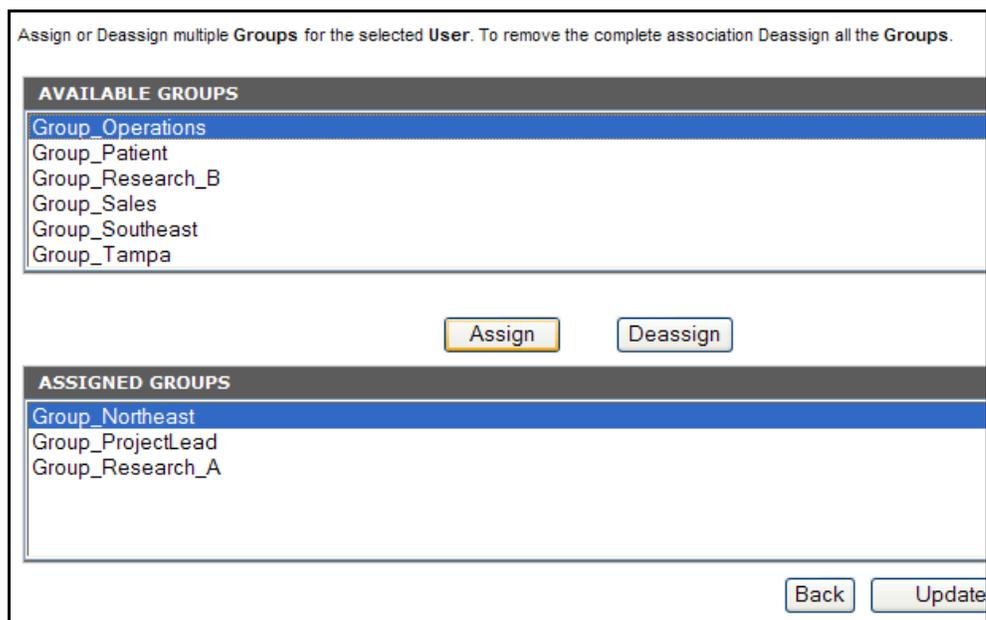


Figure 3-14 Groups lists for selected User element

In the figure above, the selected user (the one whose details page was open and from which the Associated Groups button was clicked) is a member of *Group_Northeast*, *Group_ProjectLead*, and *Groups_Research_A*.

In order to assign this user to another group, select the group from the top list box (Available Groups) and click Assign. This moves that group from the top to the bottom box, indicating assignment of the user to that group.

In order to remove the user from a group, select the group from the bottom list box (Assigned Groups) and click Deassign. This moves that group from the bottom list box to the top list box, indicating that the user is no longer a member of that group.

NOTE: You may select multiple items from each list using the standard CTRL+SELECT (non-sequential selection) or SHIFT+SELECT (sequential selection) methods.

When the groups you want to assign the element to appear in the correct lists (Available or Assigned), you must click **Update**. This saves your changes and returns you to the element Details screen.

While the example shown here is specific to User and Group elements, the same principal applies, regardless of the type of individual element you are working with.

Assigning or Deassigning Elements - Group to Individual

If you logged into Admin Mode and are working with a group element (group, protection group, or role), you will notice a series of buttons on the Details page for the element that looks something like Figure 3-15 below:



Figure 3-15 Group Element Details option buttons in Admin Mode

Depending on the type of element you have open, the button labels will vary but each functions in essentially the same way. They allow you to assign or associate other elements (group or individual) to the currently open element.

The left-most button contains the label **Associated {individual element}** and is used to associate the open group element with one or more individual elements. For example, if the currently open element is a Protection Group, the leftmost button reads **Associated PE**, because a Protection Group is a grouping of Protection Elements. If the open element is a User element, the button reads **Associated Users** as shown above, because a Group is a collection of Users.

Clicking the Associated {element} button opens a window that lists all of the individual elements currently associated with that group.

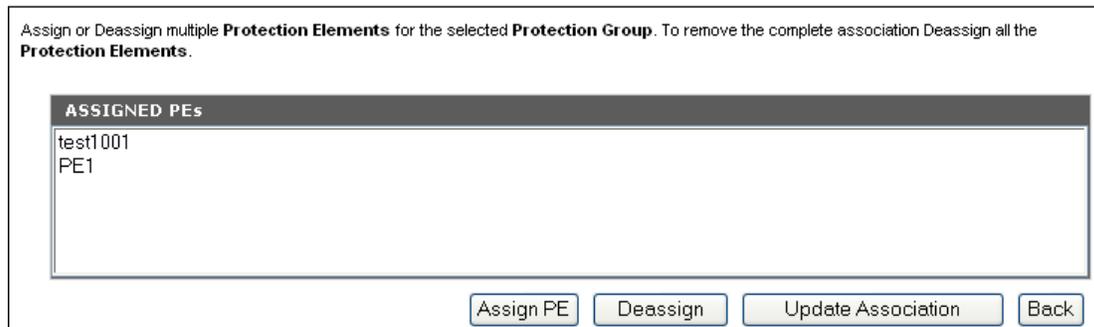


Figure 3-16 List of Protection Elements assigned to a Protection Group

In the figure above, the Associated PE button was clicked from the details page of a protection group. The resulting window lists the two protection elements currently assigned to that protection group: *test1001* and *PE1*.

To *remove* an assigned individual element from the list, highlight that element and click **Deassign**.

NOTE: You may select multiple items from the list using the standard CTRL+SELECT (non-sequential selection) or SHIFT+SELECT (sequential selection) methods.

To *add* an individual element to this group, click **Assign {element}**. The label of this button will vary depending on the type of element appropriate for the group type. In the example shown, you would click **Assign PE** to add protection elements to this protection group.

A search screen appears, allowing you search for the collection of elements you want to add to this group.

Figure 3-17 Search form for adding a Protection Element to a Protection Group

Use the search form to find the elements you want to add to the group. Since the result set allows you to select multiple elements to assign, we highly recommend performing a wildcard search using an asterisk (*) so that your result set can contain as many of the necessary elements as possible. This reduces the number of times you will need to perform these steps to complete the element assignments.

When finished, click **Search**. The result set appears, with check boxes located to the left of each item, allowing you to select multiple elements to add to the group.

Protection Element					
SEARCH RESULTS					
Select	Protection Element Name	Protection Element Description	Protection Element Type	Object Id	Attribute
<input checked="" type="checkbox"/>	PE1			PEID1	
<input type="checkbox"/>	PE1 - Name	PE1 - description a	PE1 - Type	PE1 - Object ID	PE1 - Attribute
<input checked="" type="checkbox"/>	PE2			PEID2	
<input type="checkbox"/>	PE4			PEID4	
<input type="checkbox"/>	PE5			PEID5	

Figure 3-18 Result Set from Search for elements to add to group

Notice in the figure above that the PE1 element is already selected. This is because that element is already assigned to the group from which we are working. The element PE2 has also been selected by the user.

When the **Assign PE** button is clicked, the checked elements are assigned to the group, and the list box of Assigned PEs re-appears, showing again all of the currently assigned elements for that group.

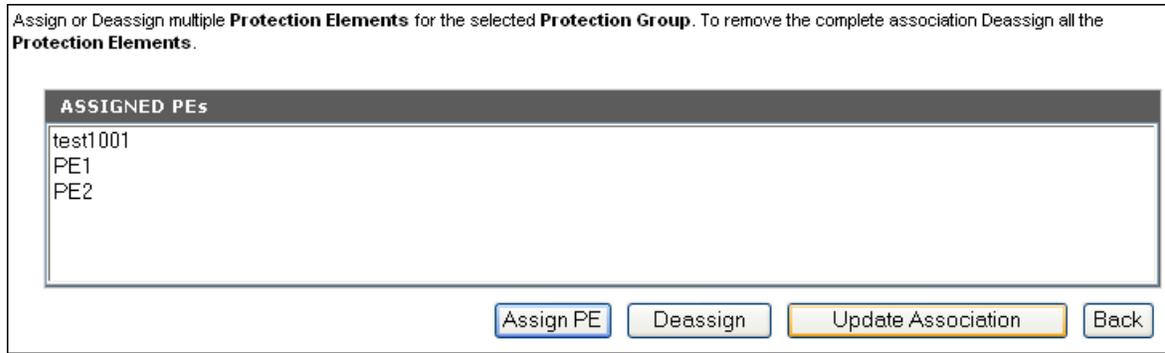


Figure 3-19 Updated Assigned PEs list with added PE elements

When the list of Assigned {elements} is complete (showing only those elements that are to be part of the group), you must click **Update Association**. This saves your changes and returns you to the group element Details screen.

While the example shown here is specific to Protection Elements and Protection Groups, the same principal applies, regardless of the type of group element you are working with.

Super Admin Mode

Super Admin Mode differs from Admin mode in some of the functionality provided to the logged in administrator. In some applications, Super Admin is the same as Admin except with additional capabilities. In the UPT however, the Super Admin responsibility is different than the Admin and the capabilities provided reflect that (although there is some overlap).

The job of the Super Admin is to add Applications to the UPT, add Users to the UPT and assign them as administrators for the created applications, and to create custom Privileges for the assigned administrators to provision later in Admin Mode.

The Super Admin is not able to provision any of the elements in UPT and is furthermore not able to create the grouping elements that allow for provisioning.

Finally, the Super Admin logs into the UPT directly whereas an Admin will log into a specific application. This is because the Super Admin's responsibilities are to set up the application and it's framework for the Admin to use, and an Admin's responsibilities are to administer that specific application.

The capability overlap for Super Admin to Admin occurs with respect to users, in that both an Admin and a Super Admin can create, modify, delete, and unlock users.

Workflow for Super Admin

The CSM team designed the UPT as a flexible tool with a flexible workflow. Any operation can be completed quickly, however it may be difficult at first to know where to start.

The main menu, located at the top of the page, contains navigation to the locations necessary for performing all Super Admin actions.



Figure 3-20 Super Admin main menu options (on Home page)

The menu option with a blue background identifies your current location. Roll over the other choices until they turn blue, and then click to navigate to that section. The Log Out selection logs the Super Admin out and returns the user to the Login page.

The following is a suggested workflow for getting started in Super Admin Mode. Each step identifies the location in which the actions are performed:

1. **Application** – Register the Application. When first deploying the UPT for a particular application, you must register or “create” the application in the Application section.
2. **Application** – Add and Update Application details.
3. **User** – Add and Update users who will serve as application Administrators.
4. **Application** – Assign users to the registered and configured Applications.
5. **Privilege** – If necessary, add or edit CSM Standard Privileges.

The sections that follow provide Information about the available functionality for each location in the Super Admin Interface.

Application Administration – Super Admin Mode

In the Application section of the Super Admin Mode, a Super Admin can add (register) an application to the UPT and add or modify the details of an application.

Create a New Application

Creating or adding an application in UPT registers that application and makes it available for the other administrative tasks that must be performed.

To add/create a new application:

1. Click **Application** from the main menu to open the Application home page.
2. Select **Create a New Application**. The Application Details form appears.
3. Enter data into the Application Details form. The form fields are defined as follows:
 - Application Name** – Uniquely identifies the application; is a required field.
 - Application Description** – A brief summary describing the application.
 - Declarative Flag** – Indicates whether application uses Declarative security.
 - Application Active Flag** – Indicates whether or not the application is currently active.
 - Database URL** – The JDBC Database URL for the given application.
 - Database User Name** – The Username for the application database.
 - Database Password** – The Password for the application database.
 - Database Dialect** – The Dialect for the application database.

Database Driver –The Driver for the application database.

4. Click **Add** to register the application in UPT.

NOTE: The Database fields must either be all completed together or left blank completely. They are all required fields if at least one of them is populated.

Select and View/Update an Existing Application

Once an application has been created, you can edit the details entered for the application any time necessary. In order to edit the application details, you must search for and identify the application to update and select to view the details.

To find and view/update an application:

1. Click **Application** from the main menu to open the Application home page.
2. Click on **Select an Existing Application**.
3. Enter search criteria into the Application Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Application Name of the application you want to view and/or edit, and then click **View Details**.
5. Review, and if necessary edit the data appearing on the Application Details form. The form fields are defined as follows:

Application Name – Uniquely identifies the application; is a required field.

Application Description – A brief summary describing the application.

Declarative Flag – Indicates whether application uses Declarative security.

Application Active Flag – Indicates whether or not the application is currently active.

Database URL – The JDBC Database URL for the given application.

Database User Name – The Username for the application database.

Database Password – The Password for the application database.

Database Dialect – The Dialect for the application database.

Database Driver –The Driver for the application database.

6. Click **Update** to save your changes.

NOTE: The Database fields must either be all completed together or left blank completely. They are all required fields if at least one of them is populated.

Delete an Existing Application

If necessary, you can delete an application from UPT. In order to delete the application, you must search for and select to view the details page of the application you want to delete.

To delete an application:

1. Click **Application** from the main menu to open the Application home page.
2. Click on **Select an Existing Application**.

3. Enter search criteria into the Application Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Application Name of the application you want to delete, and then click **View Details**.
5. From the details page, click **Delete**. A confirmation window appears.
6. Click **OK** to confirm the deletion.

Associating/Assigning an Administrator for an Application

Once the application is registered, you must assign one or more users as Administrators for the application so that they can perform the Admin Mode tasks. You may need to create new users before you can perform these procedures. See the [User Administration– Super Admin Mode](#) below for those instructions.

As an Administrator, the assigned users will have the rights to create and modify application elements such as Users, Roles and Groups. Administrators are also able to unlock any users who are locked out of the application because of exceeding the maximum allowed invalid login attempts.

To assign a user as an application Administrator:

1. Click **Application** from the main menu to open the Application home page.
2. Click on **Select an Existing Application**.
3. Enter search criteria into the Application Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Application Name of the application you want to view and/or edit, and then click **View Details**.
5. From the details page, click **Delete**. A confirmation window appears.
6. Select **Associate Admins**.
7. Find and associate the appropriate users as administrators for the application. Click on the **Assign** and **Deassign** buttons until the proper associations appear. For more information, see [Assignments and Associations](#) on page 35.
8. Click **Update Association** to save your changes. No associations or changes are saved until this button is selected.

User Administration– Super Admin Mode

In the User section of the Super Admin Mode, a Super Admin can create, edit, and delete UPT users. Super Admins are also able to unlock any users who are locked out of the application because of exceeding the maximum allowed invalid login attempts.

Create a New User

The User section allows you to create a new user for the UPT. Keep in mind that the user must be assigned to an application before they can log into UPT.

To create a new user:

1. Click **User** from the main menu to open the User home page.
2. Click on **Create a New User**.
3. Enter data into the User Details form. The form fields are defined as follows:
 - Name** – The Name acts like a username to uniquely identify the User; it is a required field and must be unique.
 - First Name** and **Last Name** – These details identify the User.
 - Organization** – The organization for which the User works. For example, The National Cancer Institute (NCI).
 - Department** – The department for which the User works. For example, caArray.
 - Title** – The job title or other functional identifier for the User.
 - Phone Number** – Provides telephone contact information for the user, typically the direct business phone number. The Phone Number field accepts the following formats: 0123456789, 012-345-6789, (012)3456789, (012)345-6789, (012)-345-6789.
 - Email ID** – Provides email contact information details for the User. An email ID must contain an '@' sign or you will receive an invalid email format error.
 - Password** – This is an optional field, used only if the schema for Authorization will also be used for Authentication. The only characters visible within this field are asterisks (*), meaning that even when entering the information, the password is not visible on the screen.
 - Confirm Password** – A field to verify the password entered in the Password field, ensuring the intended password was entered correctly. This field must match the password field exactly.
 - User Start Date** and **User End Date** – For informational purposes only.
4. When the desired fields are completed, click **Add**. The user is added to UPT.

Select and View/Update an Existing User

Once a user has been created for UPT, you can view or edit the user details any time necessary. In order to edit the user details, you must search for and identify the user to update and select to view the details.

To find and view/update a user:

1. Click **User** from the main menu to open the User home page.
2. Click on **Select an Existing User**.
3. Enter search criteria into the User Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the User Name of the user you want to view and/or edit, and then click **View Details**.
5. Review, and if necessary edit the data appearing on the User Details form. The form fields are defined as follows:

User Login Name – The User Login Name acts like a username to uniquely identify the User; it is a required field and must be unique.

First Name and Last Name – These details identify the User.

Organization – The organization for which the User works. For example, The National Cancer Institute (NCI).

Department – The department for which the User works. For example, caArray.

Title – The job title or other functional identifier for the User.

Phone Number – Provides telephone contact information for the user, typically the direct business phone number. The Phone Number field accepts the following formats: 0123456789, 012-345-6789, (012)3456789, (012)345-6789, (012)-345-6789.

Email ID – Provides email contact information details for the User. An email ID must contain an '@' sign or you will receive an invalid email format error.

Password – This is an optional field, used only if the schema for Authorization will also be used for Authentication. The only characters visible within this field are asterisks (*), meaning that even when entering the information, the password is not visible on the screen.

Confirm Password – A field to verify the password entered in the Password field, ensuring the intended password was entered correctly. This field must match the password field exactly.

User Start Date and User End Date – For informational purpose only.

6. When finished, click **Update**.

Delete an Existing User

If necessary, you can delete a user from UPT. In order to delete the user, you must search for and select to view the details page of the user you want to delete.

To delete a user:

1. Click **User** from the main menu to open the User home page.
2. Click on **Select an Existing User**.
3. Enter search criteria into the User Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the User Name of the user you want to delete, and then click **View Details**.
5. From the details page, click **Delete**. A confirmation window appears.
6. Click **OK** to confirm the deletion.

Unlock a User

This feature allows a Super Admin to unlock a locked-out user. A user can be locked out of the system if multiple login attempts are made using an invalid username/password combination, and those unsuccessful login attempts exceed the configured number of allowed attempts. The default is three invalid attempts.

NOTE: The application configuration includes a setting that identifies an amount of time, after which a locked-out user is automatically unlocked. This means that a user could wait the specified period of time rather than requiring manual unlock of their account.

To unlock a locked-out user:

1. Click **User** from the main menu to open the User home page.
2. Click on **Select an Existing User**.
3. Enter search criteria into the User Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the User Name of the user you want to delete, and then click **View Details**.
5. From the details page, click **Delete**. A confirmation window appears.
6. Click **Unlock** to unlock the user.

Privileges and Standard Privileges

A Privilege refers to any operation that can be performed upon data. Creating Privileges allows you to configure the granularity of the operations that can be performed on your data.

Assigning Privileges to users helps control access to important components of an application (Protection Elements). Within CSM, users may possess one or more of privileges for a particular protection element.

By default, the UPT installs with CSM Standard Privileges. These Privileges were agreed upon by the Security Working Group. The CSM Standard Privileges are listed and defined in the following table:

<i>Privilege Name</i>	<i>Privilege Definition</i>	<i>Applying the Privilege (Example)</i>
CREATE	This privilege grants permission to a user to create an entity. This entity can be an object, a database entry, or a resource such as a network connection.	A user can create a database entry.
ACCESS	This privilege allows a user to access a particular resource. Examples of resources include a network connection, database connection, socket, module of the application, or even the application itself.	A user can gain access to a particular module in an application.
READ	This privilege permits the user to read data from a file, URL, socket, database, or an object. This can be used at an entity level signifying that the user is allowed to read data about a particular entry (which can be object or database row, etc.)	A user can view personal information such as a Social Security Number.

Privilege Name	Privilege Definition	Applying the Privilege (Example)
WRITE	This privilege allows a user to write data to a file, URL, socket, database, or object. This can also be used at an entity level signifying that the user is allowed to write data about a particular entity (which may include an object, database row, etc.)	A user can add text to a database entry.
UPDATE	This privilege grants permission at an entity level and signifies that the user is allowed to update and modify data for a particular entity. Entities may include an object, an attribute of the object, a database row, etc.	A user can modify an object's attribute data.
DELETE	This privilege permits a user to delete a logical entity. This entity can be an object, a database entry, a resource such as a network connection, etc.	A user can delete record.
EXECUTE	This privilege allows a user to execute a particular resource. The resource can be a method, function, behavior of the application, URL, button etc.	A user can click on a button to perform a method.

The sections that follow provide information on the functionality available for Privileges.

Create a New Privilege

If the CSM Standard Privileges are not sufficient, a Super Admin can create privileges to meet application needs.

To create a new privilege:

1. Click **Privilege** from the main menu to open the Privilege home page.
2. Click on **Create a New Privilege**.
3. Enter data into the Privilege Details form. The form fields are defined as follows:
 - Name** – Use to uniquely identify the Privilege; it is a required field.
 - Description** – A brief summary describing the Privilege.
4. Select **Add**.

Select and View/Update an Existing Privilege

Once a privilege has been created for UPT, you can view or edit the details any time necessary. In order to edit the privilege details, you must search for and identify the privilege to update and select to view the details.

To find and view/update a privilege:

1. Click **Privilege** from the main menu to open the Privilege home page.
2. Click on **Select an Existing Privilege**.

3. Enter search criteria into the Privilege Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the privilege you want to view and/or edit, and then click **View Details**.
5. Review, and if necessary edit the data appearing on the Privilege Details form. The form fields are defined as follows:
 - Name** – Use to uniquely identify the Privilege; it is a required field.
 - Description** – A brief summary describing the Privilege.
6. Click **Update**.

Delete an Existing Privilege

If necessary, you can delete a privilege from UPT. In order to delete the privilege, you must search for and select to view the details page of the privilege you want to delete.

To delete a privilege:

1. Click **Privilege** from the main menu to open the Privilege home page.
2. Click on **Select an Existing Privilege**.
3. Enter search criteria into the Privilege Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the privilege you want to view and/or edit, and then click **View Details**.
5. From the details page, click **Delete**. A confirmation window appears.
6. Click **OK** to confirm the deletion.

Admin Mode

The difference between the functionality in the Super Admin Mode and Admin Mode is that the Super Admin can create and edit all of the objects necessary for use with an application, including creation of the application itself in UPT. The Admin mode provides the ability to associate those created objects in a logical manner, and to modify or remove those associations as needed.

Furthermore, a Super Admin logs into the UPT, whereas an Admin login requires the identification of a specific application. This means that all items created or modified in Admin Mode are done so for the specific application identified at login.

The Admin Mode of the UPT is divided into six major “locations” or Home pages: Group, Privilege, Protection Group, Role, and User. In these locations, an Admin can perform functions such as create, modify, or delete the elements, and manage or delete the associations between the objects. For example, you may create users and groups, or assign Privileges to a Role or Users to a Group.

Figure 3-21 below helps to illustrate how all elements (also sometimes referred to as objects) are related in the Authorization Schema.

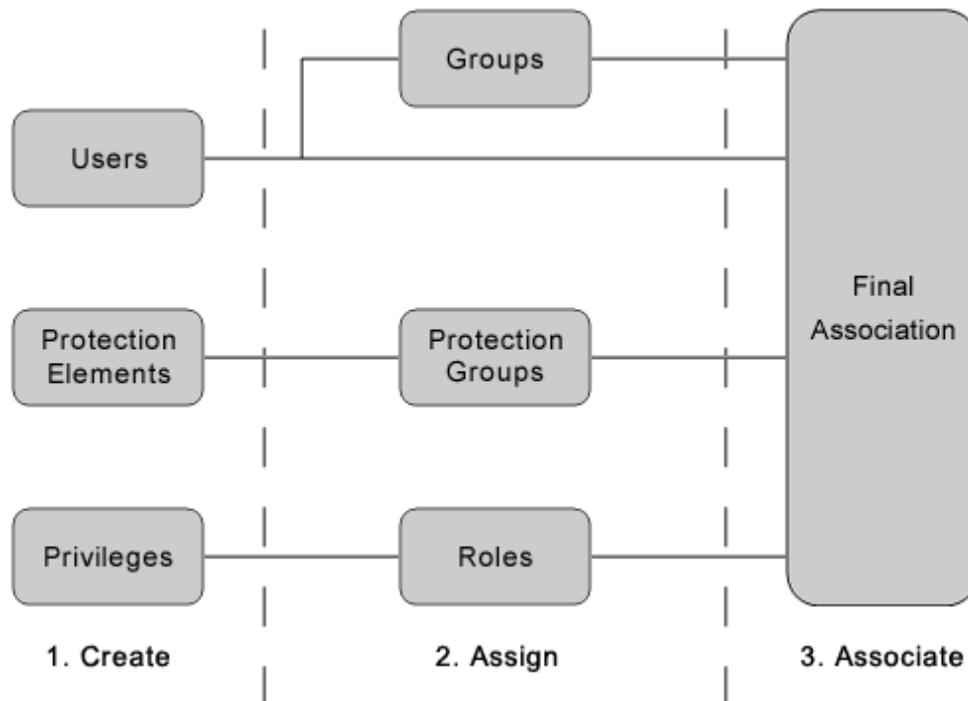


Figure 3-21 Authorization Schema element relationships

Figure 2-1 below provides definitions for each Authorization Schema element shown in the above diagram. The elements are listed by how they are associated, with the individual element defined and then the related group element defined.

Definitions for Authorization Status	
User	A User is someone who requires access to your application. Users can become part of a Group, and can have an associated Protection Group and Roles.
Group	A Group is a collection of application users. By combining users into a Group, it becomes easier to manage their collective roles and access rights in your application.
Protection Element	A Protection Element is any entity (typically data) that has controlled access. Examples include Social Security Number, City, and Salary.
Protection Group	A Protection Group is a collection of application Protection Elements. By combining Protection Elements into a Protection Group, it becomes easier to associate Users and Groups with rights to a particular data set. Examples include Address and Personal Information.
Privilege	A Privilege refers to any operation performed upon data. CSM makes use of a standard set of privileges. This will help standardize authorization to comply with JAAS and Authorization Policy and allow for adoption of technology such as SAML in the future.
Role	A Role is a collection of application Privileges. Examples include Record Admin and EmployeeModify.
Final Association	The final association is the correlation between a User or Group, and their assigned Roles for a particular Protection Group.

Figure 3-22 Authorization Status element definitions

Each User (and Group) assumes Roles (rights) for a Protection Group (protected entities like data fields). For example, User *John* has a Role *EmployeeModify* for all elements in the *Address* Protection Group.

Assigning Protection Groups and Roles is done from the **User** or **Group** sections of the Admin Mode, depending on how they are being assigned.

Workflow for Admin

The CSM team designed the UPT as a flexible tool with a flexible workflow. The general concept of the workflow is to create the base elements first and then create the groupings and associations.

Any operation can be completed quickly, however it may be difficult at first to know where to start. The main menu, located at the top of the page, contains navigation to the locations necessary for performing all Admin Mode actions.



Figure 3-23 Admin Mode main menu options (on Home page)

The menu option with a blue background identifies your current location. Roll over the other choices until they turn blue, and then click to navigate to that section. The Log Out selection logs the Admin out and returns the user to the Login page.

The following is a suggested workflow for getting started in Admin Mode:

1. **Create base objects:** Users and Protection Elements. CSM Standard Privileges are already provided, along with any custom Privileges created by the Super Admin.
2. **Create collection objects** and assign base objects to the collections. These can be performed in any order.

Groups:

- Create Groups
- Assign Users to Groups.

Protection Groups:

- Create Protection Groups
- Assign Protection Elements to Protection Groups.

Roles:

- Create Roles.
- Assign Privileges to Roles.

3. **Associate rights** with Users and Groups (in any order).
 - Assign a Protection Group and Roles to Users.
 - Assign a Protection Group and Roles to Groups.

The sections that follow provide instructions for performing these tasks in Admin Mode.

User Administration – Admin Mode

A User is simply someone that requires access to an application. The User location of the Admin Mode allows you to create users, update user details, and delete users. The instructions for performing these basic functions are the same in Admin Mode as they are in Super Admin mode. See [User Administration– Super Admin Mode](#) on page 42 for the procedures to create users, view or update user details, and delete users.

Because the functionality is specific to Admin Mode, this section provides procedures on how to associate or disassociate Users with a Group, Protection Group, and Roles. These tasks are performed from the User Details page, which is accessed as follows:

1. Click **User** from the main menu to open the User home page.
2. Click on **Select an Existing User**.
3. Enter search criteria into the User Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the User Name of the user you want to associate/disassociate, and then click **View Details**.

NOTE: You can also perform these functions from the details page while creating a new user. However the workflow listed above suggests creating the base objects first and then assigning them appropriately. Therefore the instructions in this section assume you have selected an existing user to administer, though the steps apply either way.

The User Details page in Admin Mode only displays four buttons as shown in Figure 3-24 below.



Figure 3-24 User Details page option buttons

The functionality of each of these buttons is detailed in the sections that follow.

Assign a User to a Group or Groups (Associated Groups button)

While users do not have to be assigned to a group, it is easier to manage user rights if groups of users are provided with specific types of access rather than having to manage these privileges for each individual user. Users can be assigned to one or more groups as appropriate.

To assign a user to a group/groups:

1. Search for and select to View Details for the user you want to assign.
2. From the User Details screen, select **Associated Groups**. A window containing a list of Available Groups and Assigned Groups appears.

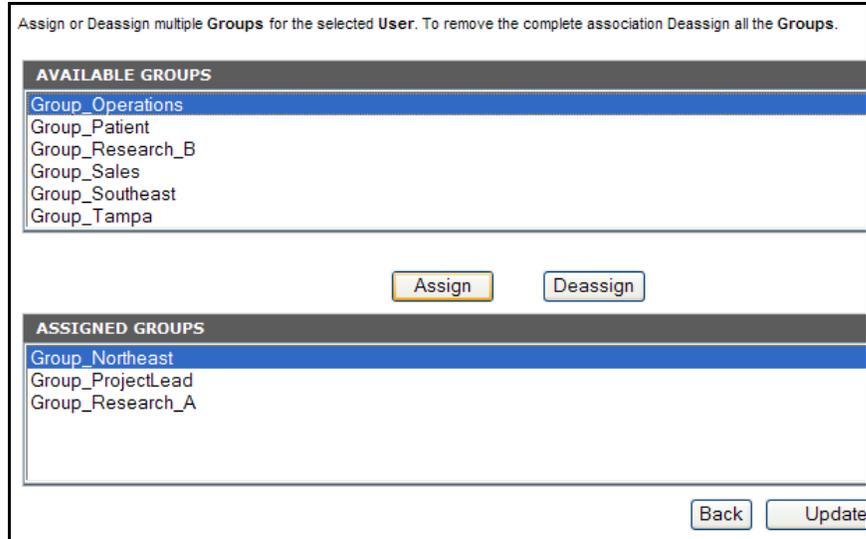


Figure 3-25 Figure 6.14 Available and Assigned Groups lists

The bottom list, **Assigned Groups**, shows the groups in which the current user is already a member. The **Available Groups** list shows the groups to which the user is not assigned.

3. Determine to which Groups the User should be assigned. Select these Groups by highlighting them in each list and clicking **Assign** or **Deassign** as appropriate.
4. Save your changes by clicking on **Update Association**. Note that association changes are not saved the update button is selected.

Assign a Protection Group and Roles to a User (Assign PG & Roles button)

1. Search for and select to View Details for the user you want to assign.
2. From the User Details screen, select **Assign PG & Roles**. A window containing a list of Protection Groups and Roles appears.
3. Determine to which Protection Groups and Roles you want to assign to the selected user and highlight them in each list. Use the CTRL+SELECT or SHIFT+SELECT method to make multiple selections from the lists.
4. Click **Assign** or **Deassign** as appropriate until the proper associations appear.
5. Click **Update Association** to save your changes. Note that changes are not saved the update button is selected.

Update Roles Associated with Assigned Protection Groups (Associated PG & Roles button)

1. Search for and select to View Details for the user you want to update.
2. From the User Details screen, select **Associated PG & Roles**. A window containing a list of all associated Protection Groups and Roles appears.
3. Select the radio button that corresponds with the intended Protection Group.

4. Determine which Roles you would like to assign to the User and highlight the Role name(s). Use the CTRL+SELECT or SHIFT+SELECT method to make multiple selections from the list.
5. Click **Assign** and **Deassign** buttons until the proper associations appear.
6. Click **Update Association** to save your changes. Note that changes are not saved the update button is selected.

View User Report (Associated PE & Privileges button)

The user reporting functionality available through the **Associated PE & Privileges** button shows a user's privileges for all of their assigned protection elements.

1. Search for and select to **View Details** for the user you want to view.
2. From the User Details screen, select **Associated PE & Privileges**. A report window appears containing a list of the selected user's privileges for each protection element.

Protection Element Administration – Admin Mode

A Protection Element is any entity (typically data) that is subject to controlled access. CSM allows for a broad definition of Protection Element. Nearly everything in an application can be protected – data, table, buttons, menu items, etc. Identifying individual Protection Elements makes it easier to control access to important data.

The sections that follow provide information for creating, updating, deleting, and assigning Protection Elements to Protection Groups.

Create a New Protection Element

1. Click **Protection Element** from the main menu to open the Protection Element home page.
2. Click on **Create a New Protection Element**.
3. Enter data into the Protection Element Details form. The form fields are defined as follows:
 - Name** – Use to uniquely identify the Protection Element; it is a required field.
 - Object ID** – A string that the application team assigns to the Protection Element.
 - Attribute Name** – Helps to further identify the Protection Element
 - Description** – A brief summary describing the Protection Element.
 - Update Date** – Indicates the date when the Protection Element's details were last updated
 - Type** – A string that the application team can assign that indicates what type of Protection Element this is.
4. When finished, click **Add** to add the new Protection Element.

Select and View/Update an Existing Protection Element

Once a Protection Element has been created, you can view or edit the details any time necessary. In order to edit the Protection Element details, you must search for and identify the Protection Element to update and select to view the details.

To find and view/update a Protection Element:

1. Click **Protection Element** from the main menu to open the Protection Element home page.
2. Click on **Select an Existing Protection Element**.
3. Enter search criteria into the Protection Element Search Criteria form. The fields available include **Name**, **Object ID** and **Attribute** name (these fields are defined below). You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Protection Element you want to view and/or edit, and then click **View Details**.
5. Review, and if necessary edit the data appearing on the Protection Element Details form. The form fields are defined as follows:

Name – Use to uniquely identify the Protection Element; it is a required field.

Object ID – A string that the application team assigns to the Protection Element.

Attribute Name – Helps to further identify the Protection Element

Description – A brief summary describing the Protection Element.

Update Date – Indicates the date when the Protection Element's details were last updated

Type – A string that the application team can assign that indicates what type of Protection Element this is.

6. When finished, click **Update**.

Delete an Existing Protection Element

If necessary, you can delete a Protection Element. In order to delete the Protection Element, you must search for and select to view the details page of the Protection Element you want to delete.

To delete a Protection Element:

1. Click **Protection Element** from the main menu to open the Protection Element home page.
2. Click on **Select an Existing Protection Element**.
3. Enter search criteria into the Protection Element Search Criteria form. The fields available include **Name**, **Object ID** and **Attribute** name (these fields are defined above). You may use an asterisk (*) as a wildcard character if necessary.

4. From the results list, select the radio button next to the Name of the Protection Element you want to delete, and then click **View Details**.
5. From the details page, click **Delete**. A confirmation window appears.
6. Click **OK** to confirm the deletion.

Assign a Protection Element to Protection Group(s)

1. Click **Protection Element** from the main menu to open the Protection Element home page.
2. Click on **Select an Existing Protection Element**.
3. Enter search criteria into the Protection Element Search Criteria form. The fields available include **Name**, **Object ID** and **Attribute** name (these fields are defined above). You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Protection Element you want to delete, and then click **View Details**.
5. From the details page, click **Associated PGs**.
6. Determine to which of the available Protection Group(s) the Protection Element should be assigned and highlight them from the list. Use the CTRL+SELECT or SHIFT+SELECT method to make multiple selections from the list if necessary.
7. Click **Assign** or **Deassign** as appropriate until the group assignments appear in the proper lists.
8. Click **Update Association** to save your changes. No associations are saved until the update button is selected.

Privilege Administration – Admin Mode

A Privilege refers to any operation performed upon data. Assigning privileges helps control access to important components of an application (Protection Elements). CSM provides a standard set of privileges that populate automatically when creating the authorization schema. See [Privileges and Standard Privileges](#) on page 45 for more details.

Because Standard Privileges are already provided, and because only a Super Admin can create custom Privileges, the Privilege location in Admin Mode only allows for viewing the details of existing privileges. The Role location allows Admins to assign privileges to roles.

To find and view a privilege:

1. Click **Privilege** from the main menu to open the Privilege home page.
2. Click on **Select an Existing Privilege**.
3. Enter search criteria into the Privilege Search Criteria form. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the privilege you want to view and/or edit, and then click **View Details**.

5. Review the data appearing on the Privilege Details form. The form fields are defined as follows:
 - Name** – Use to uniquely identify the Privilege; it is a required field.
 - Description** – A brief summary describing the Privilege.
6. Click **Back** when finished.

Protection Group Administration – Admin Mode

A Protection Group is a collection of application's Protection Elements. By combining Protection Elements into a Protection Group, it becomes easier to associate Users and Groups with rights to a particular data set.

The Protection Group is the only element that can have a Parent. Using Parents is a way to group Protection Groups within Protection Groups. This makes organizing users and their authorization rights easier.

The sections that follow provide instructions for creating new Protection Groups, modifying existing Protection Group details, assigning Protection Elements to Protection Groups, and assigning a parent for a Protection Group.

Create a New Protection Group

1. Click **Protection Group** from the main menu to open the Protection Group home page.
2. Click on **Create a New Protection Group**.
3. Enter data into the Protection Group Details form. The form fields are defined as follows:
 - Name** – Used to uniquely identify the Protection Group; it is a required field.
 - Description** – A brief summary describing the Protection Group.
 - Large Count Flag** – Used to indicate if the Protection Group has a large number of associated Protection Elements.
 - Update Date** – Indicates the date when this Protection Group's Details were last updated.
4. Click **Add**.

Select and View/Update an Existing Protection Group

Once a Protection Group has been created, you can view or edit the details any time necessary. In order to edit the Protection Group details, you must search for and identify the Protection Group to update and select to view the details.

To find and view/update a Protection Group:

1. Click **Protection Group** from the main menu to open the Protection Group home page.
2. Click on **Select an Existing Protection Group**.
3. Enter search criteria into the **Protection Group** Search Criteria form. Search by Protection Group Name. You may use an asterisk (*) as a wildcard character if necessary.

4. From the results list, select the radio button next to the Name of the Protection Group you want to view and/or edit, and then click **View Details**.
5. Review, and if necessary edit the data appearing on the Protection Group Details form. The form fields are defined as follows:
 - Name** – Used to uniquely identify the Protection Group; it is a required field.
 - Description** – A brief summary describing the Protection Group.
 - Large Count Flag** – Used to indicate if the Protection Group has a large number of associated Protection Elements.
 - Update Date** – Indicates the date when this Protection Group's Details were last updated.
6. Click **Update**.

Delete an Existing Protection Group

If necessary, you can delete a Protection Group. In order to delete the Protection Group, you must search for and select to view the details page of the Protection Group you want to delete.

To delete a Protection Group:

1. Click **Protection Group** from the main menu to open the Protection Group home page.
2. Click on **Select an Existing Protection Group**.
3. Enter search criteria into the Protection Group Search Criteria form. Search by Protection Group name. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Protection Group you want to delete, and then click **View Details**.
5. From the details page, click **Delete**. A confirmation window appears.
6. Click **OK** to confirm the deletion.

Assign Protection Elements to the Protection Group

Once a Protection Group has been created, you can assign Protection Elements to it, allowing you to group those elements and more easily manage access to those elements.

1. Click **Protection Group** from the main menu to open the Protection Group home page.
2. Click on **Select an Existing Protection Group**.
3. Enter search criteria into the Protection Group Search Criteria form. Search by Protection Group name. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Protection Group you want to modify, and then click **View Details**.
5. From the details page, select **Associated PE's**.

6. An Assigned PE's list appears. You have the following options:
 To remove a PE from the group, highlight the PE and click **Deassign**.
 To add a PE to this group, click **Assign PE**.
 - c. In the Search form that appears, enter search criteria in the fields (using an asterisk as a wildcard if necessary) and click **Search**.
 - d. Use the checkboxes in the results list to select which PEs you want to assign to the Protection Group, and then click **Assign PE**.
7. The Assigned PE's list re-appears. Repeat the above steps as necessary until only the PEs you want to have assigned to this Protection Group appear.
8. When finished, click **Update Association** to save your changes. No associations are saved until the update button is selected.

Assign a Parent for the Protection Group

Assigning a Parent for a Protection Group is a way of adding another layer of grouping for your Protection Groups, to assist you in managing access to the Protection Elements contained within the groups.

NOTE: You must create the Parent group before you can assign Protection Groups to it.

1. Click **Protection Group** from the main menu to open the Protection Group home page.
2. Click on **Select an Existing Protection Group**.
3. Enter search criteria into the Protection Group Search Criteria form. Search by Protection Group name. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Protection Group you want to assign, and then click **View Details**.
5. From the details page, click **Associated Parent PG**.
6. From the list that appears, determine to which Parent Group the Protection Group should be assigned. Protection Groups can be assigned to only one Parent.
7. Highlight the appropriate Parent group and click **Assign**.
8. If necessary, you may need to remove the Protection Group from an existing Parent assignment. In this case, select the existing Parent and click **Deassign**.
9. Click Update Association to save your changes. No association is saved until this button is selected.

Role Administration – Admin Mode

A Role is a collection of Privileges. By combining Privileges into a Role, it becomes easier to associate Users and Groups with rights to a particular data set.

The sections that follow provide instructions for creating new Roles, modifying existing roles, deleting Roles, and assigning or deassigning Privileges to a given Role.

Create a New Role

1. Click **Role** from the main menu to open the Role home page.
2. Click on **Create a New Role**.
3. Enter data into the Role Details form. The form fields are defined as follows:
 - Name** – Use to uniquely identify the Role; it is a required field.
 - Description** – A brief summary describing the Role.
 - Active Flag** – Indicates whether or not the Role is currently active.
4. Select **Add**.

Select and View/Update an Existing Role

Once a Role has been created, you can view or edit the details any time necessary. In order to edit the Role details, you must search for and identify the Role to update and select to view the details. s

To find and view/update a Role:

1. Click **Role** from the main menu to open the Role home page.
2. Click on **Select an Existing Role**.
3. Enter search criteria into the Role Search Criteria form. Search by Role Name. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Role you want to view and/or edit, and then click **View Details**.
5. Review, and if necessary edit the data appearing on the Role Details form. The form fields are defined as follows:
 - Name** – Use to uniquely identify the Role; it is a required field.
 - Description** – A brief summary describing the Role.
 - Active Flag** – Indicates whether or not the Role is currently active.
6. Click **Update**.

Delete an Existing Role

If necessary, you can delete a Role. In order to delete the Role, you must search for and select to view the details page of the Role you want to delete. s

To delete a Role:

1. Click **Role** from the main menu to open the Role home page.
2. Click on **Select an Existing Role**.
3. Enter search criteria into the Role Search Criteria form. Search by Role Name You may use an asterisk (*) as a wildcard character if necessary.

4. From the results list, select the radio button next to the Name of the Role you want to view and/or edit, and then click **View Details**.
5. From the details page, click **Delete**. A confirmation window appears.
6. Click **OK** to confirm the deletion.

Assign Privileges to the Role

1. Click **Role** from the main menu to open the Role home page.
2. Click on **Select an Existing Role**.
3. Enter search criteria into the Role Search Criteria form. Search by Role Name You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Role you want to view and/or edit, and then click **View Details**.
5. From the details page, click **Associated Privileges**.
6. An Associated Privileges list appears. You have the following options:
To remove a privilege from the Role, highlight the privilege and click **Deassign**.
To add a privilege to this Role, click **Assign Privilege**.
 - a. In the Search form that appears, enter search criteria in the fields (using an asterisk as a wildcard if necessary) and click **Search**.
 - b. Use the checkboxes in the results list to select which privileges you want to assign to the Role, and then click **Assign Privilege**.
7. The Assigned Privilege list re-appears. Repeat the above steps as necessary until only the privileges you want to have assigned to this Role appear.
8. When finished, click **Update Association**. No associations are saved until this button is selected.

Group Administration – Admin Mode

A Group is a collection of application users. By combining users into a Group, it becomes easier to manage their collective roles and access rights in your application. When you select an existing group and associate a new Protection Group and Roles with that group, all users in that particular Group have the same rights.

Like the User Details page, the Group Details page also contains buttons that are only available in Admin Mode.



Figure 3-26 Group Details page option buttons

The sections that follow provide information on how to create, modify, and delete Groups as well as instructions for using each of the Group option buttons to associate or disassociate Groups' Protection Elements, Privileges and Roles.

NOTE: The association/assignment instructions in the below sections assume you have selected an existing group to administer, however you can also perform these functions from the details page while creating a new group.

Create a New Group

1. Click **Group** from the main menu to open the Group home page. s
2. Click on **Create a New Group**.
3. Enter data into the Group Details form. The form fields are defined as follows:
 - Name** – Use to uniquely identify the Group; it is a required field.
 - Description** – A brief summary describing the Group.
4. Select **Add**.

Select and View/Update an Existing Group

Once a Group has been created, you can view or edit the details any time necessary. In order to edit the Group details, you must search for and identify the Group to update and select to view the details.

To find and view/update a Group:

1. Click **Group** from the main menu to open the Group home page.
2. Click on **Select an Existing Group**.
3. Enter search criteria into the Group Search Criteria form. Search by Group Name. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Group you want to view and/or edit, and then click **View Details**.
5. Review, and if necessary edit the data appearing on the Group Details form. The form fields are defined as follows:
 - Name** – Use to uniquely identify the Group; it is a required field.
 - Description** – A brief summary describing the Group.
6. Click **Update**.

Delete an Existing Group

If necessary, you can delete a Group. In order to delete the Group, you must search for and select to view the details page of the Group you want to delete.

To delete a Group:

1. Click **Group** from the main menu to open the Group home page.
2. Click on **Select an Existing Group**.
3. Enter search criteria into the Group Search Criteria form. Search by Group Name. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Group you want to delete, and then click **View Details**.

5. From the details page, click **Delete**. A confirmation window appears.
6. Click **OK** to confirm the deletion.

Add Users to the Group (Associated Users)

1. Click **Group** from the main menu to open the Group home page.
2. Click on **Select an Existing Group**.
3. Enter search criteria into the Group Search Criteria form. Search by Group Name. You may use an asterisk (*) as a wildcard character if necessary.
4. From the results list, select the radio button next to the Name of the Group you want to modify, and then click **View Details**.
5. Click **Associated Users**.
6. A Group and User Association screen appears. You have the following options:
 - To remove a user from the Group, highlight the user and click **Deassign**.
 - To add a user to this Group, click **Assign User**.
 - c. In the Search form that appears, enter user search criteria (using an asterisk as a wildcard if necessary) and click **Search**.
 - d. Use the checkboxes in the results list to select which users you want to assign to the Group, and then click **Assign User**.
7. The Group and User Association list re-appears. Repeat the above steps as necessary until only the users you want to have assigned to this Group appear.
8. When finished, click **Update Association** to save your changes.

Click **Back** to return to the Group details screen.

Users can also be assigned to Groups in the User location of the Admin Mode. See [Assign a User to a Group or Groups \(Associated Groups button\)](#) on page 50 for more information.

Assign a Protection Group and Roles to a Group (Assign PG & Roles)

1. Search for and select to View Details for the Group you want to assign.
2. From the User Details screen, select **Assign PG & Roles**. A window containing a list of Protection Groups and Roles.
3. Determine which Protection Groups and Roles you want to assign to the selected Group and highlight them in each list. Use the CTRL+SELECT or SHIFT+SELECT method to make multiple selections from the lists.
4. Click **Assign** or **Deassign** as appropriate until the proper associations appear.
5. Click **Update Association** to save your changes. Note that changes are not saved the update button is selected.

Update Roles Associated with Assigned Protection Groups (Associated PG & Roles button)

1. Search for and select to View Details for the Group you want to update.
2. From the Group Details screen, select **Associated PG & Roles**. A window appears containing a list of all associated Protection Groups and their Roles.
3. Select the radio button that corresponds with the intended Protection Group.
4. Determine which Roles you would like to assign to the Group and highlight the Role name(s). Use the CTRL+SELECT or SHIFT+SELECT method to make multiple selections from the list.
5. Click **Assign** and **Deassign** buttons until the proper associations appear.
6. Click **Update Association** to save your changes. Note that changes are not saved the update button is selected.

View Group Report (Associated PE & Privileges button)

The Group reporting functionality available through the **Associated PE & Privileges** button shows a group's privileges for all of its assigned protection elements.

1. Search for and select to View Details for the Group for which you want information.
2. From the Group Details screen, select **Associated PE & Privileges**. A report window appears containing a list of the selected Group's privileges for each protection element.

Instance Level Security Administration – Admin Mode

Instance Level Security is a feature provided by CSM to allow filtering of the instances of data directly at the database level by creating filter criteria and linking those criteria with allowed values from CSM tables.

The sections that follow provide instructions for uploading the .jar file containing the Hibernate file and Domain Objects, creating a new filter clause, and searching for existing filter clauses.

To access the Instance Level functionality, click **Instance Level** from the Admin Mode main menu.



Figure 3-27 Admin Mode main menu options

Each of the available features is accessed by clicking the appropriate link located in the Instance Level Links section of the Instance Level Home page.

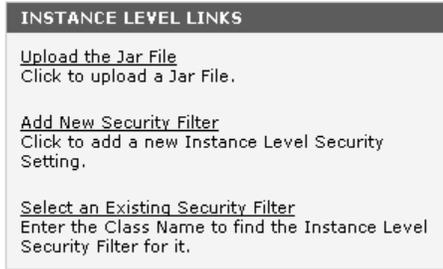


Figure 3-28 Instance Level Links on Admin Mode Home page

Uploading a File

1. From the Instance Level home page, click the **Upload the Jar File** link. The upload form appears.

* indicates a required field

UPLOAD THE APPLICATION JAR FILE	
* Application Jar File	L:\UPT\example-beans.jar <input type="button" value="Browse..."/>
Application Jar File	L:\UPT\example-orm.jar <input type="button" value="Browse..."/>
* Hibernate Configuration File Name	hibernate.cfg.xml

Figure 3-29 Jar File Upload Form

2. Provide the following information for upload:

Application Jar File – The path of the application jar file that contains the Hibernate configuration and mapping files, and domain objects.

Application Jar File – If the application is an SDK generated system, there are two jar files generated. Use this field to identify the location of the second jar file.

Hibernate Configuration File Name – The fully qualified name of the Hibernate configuration file located in the selected jar file.

3. Click **Upload** button.

Add New Security Filter

1. From the Instance Level home page, click the **Add New Security Filter** link. The New Filter Clause Details form appears.

* indicates a required field

ENTER THE NEW FILTER CLAUSE DETAILS	
* Class Name	gov.nih.nci.cacoresdk.domain.other.levelassociation.Deck
* Filter Chain	gov.nih.nci.cacoresdk.domain.other.levelassociation.Suit - suitCollection gov.nih.nci.cacoresdk.domain.other.levelassociation.Card - cardCollection Add Remove Done
* Target Class Attribute Name	image
Target Class Alias	
Target Class Attribute Alias	

Add Reset Back

Figure 3-30 New Filter Clause Details form

2. Enter the following information into the form:

Class Name - This the class for which you want to create a filter clause.

Filter Chain – This is a chain of the associated objects on which the security of the class depends. You have the following options:

- o In the case of the inherited security, you can follow the trail to the target class by selecting the associated class from the drop-down list and clicking the **Add** button *in this section of the form*.
- o You can remove the last associated class by clicking **Remove**.
- o If the security of the Class is dependant on itself, select the same Class (with the suffix “self”) from the Filter Chain drop-down list.

Once the proper filter chain appears, click **Done** in this section of the form.

Target Class Attribute Name – This field is populated with all of the attributes of the Final Target Class.

Target Class Alias – If you want to provide an alias for the Target Class Name, you can do so by providing a value in the Target Class Alias field.

Target Class Attribute Alias –You can provide an alias for the Target Class Attribute Name by providing a value in the Target Class Attribute Alias field.

3. When finished, click Add button at the bottom of the form to add the filter.

Select and View/Update an Existing Security Filter

1. From the Instance Level home page, click the **Select an Existing Security Filter** link. The Search form appears.

Use * to perform wildcard searches

ENTER THE FILTER CLAUSE SEARCH CRITERIA	
Class Name	<input type="text"/>
<input type="button" value="Search"/> <input type="button" value="Reset"/> <input type="button" value="Back"/>	

Figure 3-31 Filter Clause Search form

2. Enter search criteria into the Filter Clause Search Criteria form. Search by Class Name. You may use an asterisk (*) as a wildcard character if necessary.
3. Click **Search**.
4. From the results list, select the radio button next to the Class Name of the filter clause you want to view and/or edit, and then click **View Details**.

FILTER CLAUSE DETAILS	
Class Name	gov.nih.nci.cacoresdk.domain.manytomany.unidirectional.Author
Filter Chain	gov.nih.nci.cacoresdk.domain.manytomany.unidirectional.Author
Target Class Name	gov.nih.nci.cacoresdk.domain.manytomany.unidirectional.Author - self
Target Class Attribute Name	name
Target Class Attribute Type	java.lang.String
Target Class Alias	
Target Class Attribute Alias	
* Generated SQL For Group	<pre>ID in (select table_name_csm.ID from AUTHOR table_name_csm where table_name_csm.NAME in (SELECT Distinct pe.attribute_value FROM CSM_PROTECTION_GROUP pg, CSM_PROTECTION_ELEMENT pe, CSM_PG_PE pgpe, CSM_USER_GROUP_ROLE_PG ugrpg, CSM_GROUP g, CSM_ROLE_PRIVILEGE rp, CSM_ROLE r, CSM_PRIVILEGE p WHERE</pre>
* Generated SQL For User	<pre>ID in (select table_name_csm.ID from AUTHOR table_name_csm where table_name_csm.NAME in (select pe.attribute_value from csm_protection_group pg, csm_protection_element pe, csm_pg_pe pgpe, csm_user_group_role_pg ugrpg, csm_user u, csm_role_privilege rp, csm_role r, csm_privilege p where ugrpg.role_id = r.role_id and</pre>
Update Date	08/28/2008 (MMDD/YYYY)

Figure 3-32 Filter Clause Details form

5. The Filter Clause Details form contains only the following editable fields:
 - o **Generated SQL for Group** – This is the SQL filter generated by Hibernate based on the filter criteria selected for Group level security.
 - o **Generated SQL for User** – This is the SQL filter generated by Hibernate based on the filter criteria selected for User level security.

NOTE: Once you edit the SQL there is no way it can be regenerated without deleting and creating the filter clause again. Also, make sure you

follow the Hibernate Filter SQL specifications and have a valid working filtering SQL.

6. If you made changes, click **Upload** to update the record. You may also use the **Delete** button to delete the record

UPT Installation and Deployment

The UPT is released as a compressed web application in the form of a WAR (Web Archive) file. Along with the WAR, the release includes sample configuration files that help developers configure the UPT with their application(s).

UPT Release Contents

The UPT Release contents can be found in the `UPT.zip` file found on the NCICB download site: <http://ncicb.nci.nih.gov/download/index.jsp>. The UPT Release contents include the files in the table below.

<i>File</i>	<i>Description</i>
<code>upt.war</code>	The UPT Web Application
<code>Hibernate.cfg.xml</code>	The sample XML file which contains the hibernate-mapping and the database connection details.
<code>AuthSchemaMySQL.sql</code> OR <code>AuthSchemaOracle.sql</code>	This Structured Query Language (SQL) script is used to create an instance of the Authorization database schema which will be used for the purpose of authorization. In the 3.0.1 and subsequent releases, this script populates the database with CSM Standard Privileges that can be used to authorize users. The same script can be used to create instances of authorization schema for a variety of applications.
<code>DataPrimingMySQL.sql</code> OR <code>DataPrimingOracle.sql</code>	This SQL script is used for priming data in the UPT's authorization schema.
<code>mysql-ds.xml</code> OR <code>oracle-ds.xml</code>	This file contains information for creating a datasource. One entry is required for each database connection. Place this file in the JBoss deploy directory.

Table 3-1 UPT Release Contents

Installation Modes

UPT was developed as a flexible application that can be deployed in multiple ways depending on the need or scenario. The three primary modes to install the UPT include the following and are described in more detail in sections that follow:

- Single Installation, Single Schema
- Single Installation, Multiple Schemas
- Local installation, Local schema

Single Installation/Single Schema Deployment

In the single installation/single schema deployment scheme shown in Figure 3-33, there is only one instance of the UPT, hosted on a common JBoss server.

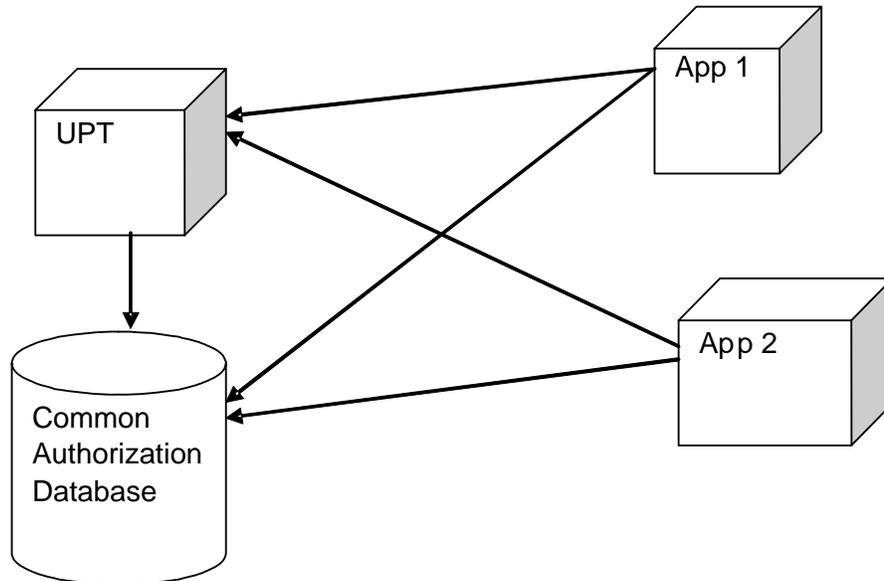


Figure 3-33 Single Installation/Single Schema deployment

In this deployment, a common installation is used to administer the authorization data for all applications, and this authorization data is stored in a common database. Therefore an application using UPT does not have to install its own authorization schema. Also, all applications can use the same hibernate-config file since they point to the same database.

Single Installation/Multiple Schema Deployment

As in the single schema deployment, the single installation/multiple schema deployment calls for the UPT to be hosted on a single JBoss Common Server as shown in Figure 3-34 below.

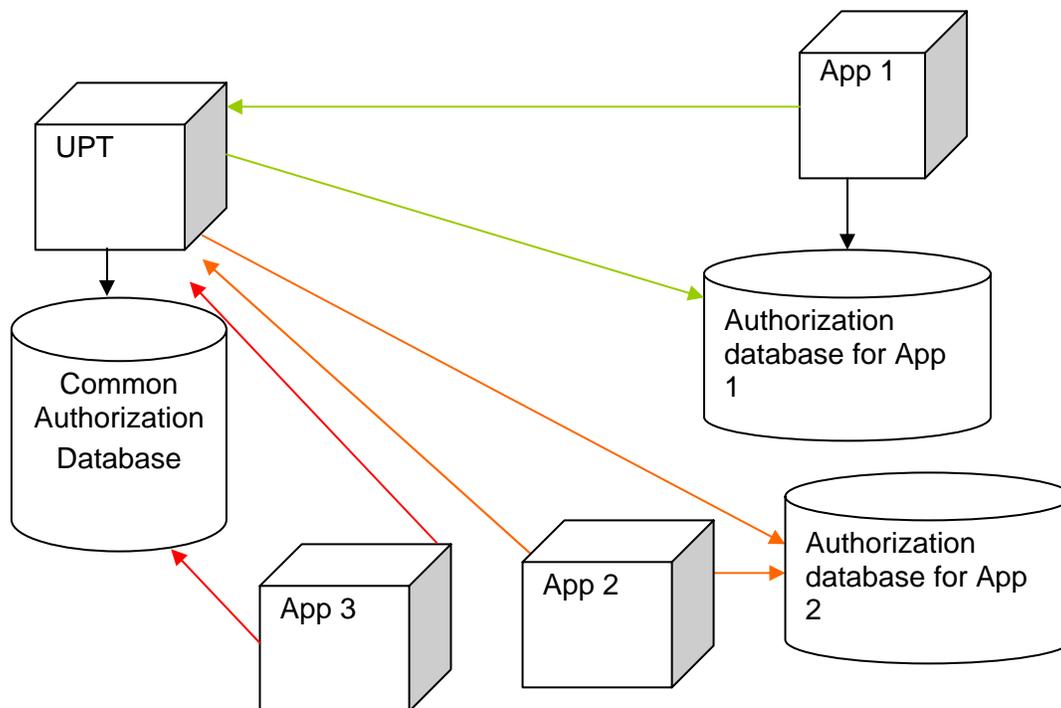


Figure 3-34 Single Installation/Multiple Schema deployment

In this deployment, like with the single schema deployment, a common installation is used to administer the authorization data for all applications. The difference is that an application *can* use its own authorization schema on a separate database if preferred. The authorization data can sit on individual databases, and at the same time some applications can still opt to use the Common Authorization Schema.

In the above figure, the three colors of arrows correspond to the three different applications shown. Notice that App 1 and App 2 have their own authorization databases, where as App 3 uses the Common Authorization Database.

Deploying UPT in this way requires each application to maintain its own hibernate-config file pointing to the database where its Authorization Schema is located. So when an application uses the UPT, the UPT communicates to the authorization schema for that application only.

Local Installation/Local Schema Deployment

The local installation/local schema deployment is the same as single installation/single schema deployment except that the UPT is hosted locally by the application as shown in Figure 3-35 below.

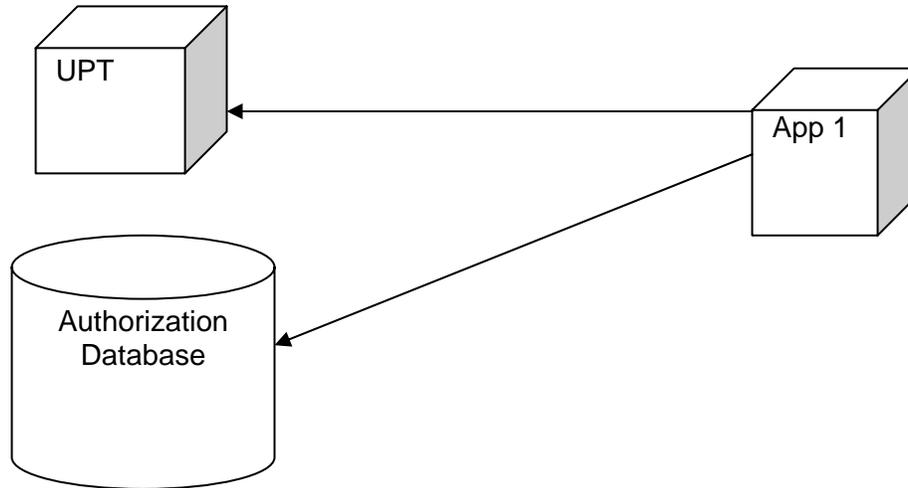


Figure 3-35 Local Installation/Local Schema deployment

This installation of UPT is not shared with other applications. This local installation is used to administer the authorization data for that particular application (or set of related applications) only. The authorization data for the application sits on its own database. In this scenario, the application requires its own hibernate-config file pointing to the database where its Authorization Schema is located.

UPT Deployment Checklist

Before deploying the UPT, verify the following environment and configuration conditions are met. These software and access credentials/parameters are required.

Environment

- JBoss 4.0 Application Server
- MySQL 4.0 OR Oracle 9i Database Server (with an account that can create databases)

UPT Release Components

- upt.war
- AuthSchemaMySQL.sql | AuthSchemaOracle.sql
- DataPrimingMySQL.sql | DataPrimingOracle.sql

UPT Deployment Steps

The sections below provide the necessary steps for deploying UPT along with instructions for completing those steps. As you follow the deployment steps, use the files containing the name corresponding with your database.

Create and Prime MySQL Database

Before beginning, you must log into the database using an account that has permission to create new databases. Also, make sure that the database you are about to create doesn't already exist. If it does, drop it and create new one.

1. In the `AuthSchemaMySQL.sql` script file, replace the `<<database_name>>` tag with the name of the UPT Authorization schema, which is ***csmupt***.
2. Save and then run the edited from the database prompt. This should create a database with the given name.
3. In the `DataPrimingMySQL.sql` script file perform the following:
 - o Replace `<<super_admin_login_id>>` with the login id of the user who is going to act as the Super Admin for that particular installation.
 - o Provide the first name and last name of the Super Admin user by replacing `<<super_admin_first_name>>` and `<<super_admin_last_name >>` with the appropriate values.
 - o Replace `<<application_context_name>>` with the application name of the application for which UPT is being hosted.
4. Save and then run the edited script from the database prompt. This should populate the database with the initial data.
5. Verify population of the data by querying the `csm_application`, `csm_user`, `csm_protection_element`, and `csm_user_protection_element` tables. They should have one record each.

The database will also include CSM Standard Privileges, and the `csm_privilege` table should have 7 entries.

Configure Datasource

Once the database has been created and primed, you must modify the `mysql-ds.xml` file, which contains information for creating a datasource, and then copy it to the appropriate location.

NOTE: One entry for each database connection is required for the `mysql-ds.xml` file.

1. Edit the `mysql-ds.xml` file as follows:
 - o Replace the `<<application_context_name>>` tag with the name of the authorization schema, which is ***csmupt***.
 - o Replace the `<<database_user_id>>` with the user id, ***ncisecurity***. And `<<database_user_password>>` with the password of the user account.

- o Replace the `<<database_url>>` with the URL needed to access the Authorization Schema residing on the database server:
`jdbc:mysql://<<prod_database_server_name>>:3306/csmupt`

Shown below is an example `mysql-ds.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
  <local-tx-datasource>
    <jndi-name>csmupt</jndi-name>
    <connection-url>jdbc:mysql://Prod_DB.nci.nih.gov:3306/csmupt</connection-url>
    <driver-class>org.gjt.mm.mysql.Driver</driver-class>
    <user-name>name</user-name>
    <password>password</password>
  </local-tx-datasource>
</datasources>
```

2. Place the edited `mysql-ds.xml` file in the JBoss deployment directory:
`{jboss-home}/server/default/deploy/.`

Configure the JBoss JAAS Login Parameters

In order to configure the UPT to verify against the LDAP, create an entry in the `login-config.xml` of JBoss as shown in the sample below. This entry configures a login-module against the UPT application context. The location of this file is `{jboss-home}/server/default/conf/login-config.xml`.

```
<application-policy name = "csmupt">
  <authentication>
    <login-module code =
      "gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule" flag =
      "required" >
    <module-option name="ldapHost">ldaps://ncids4a.nci.nih.gov:636</module-option>
    <module-option name="ldapSearchableBase">ou=nci,o=nih</module-option>
    <module-option name="ldapUserIdLabel">cn</module-option>
  </login-module>
</authentication>
```

In the example above:

- The `application-policy` name is the name of the application for defining the authentication policy; in this case, `csmupt`.
- The `login-module` code is the LoginModule class used to perform the authentication task; in this case, it is `gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule`.
- The `flag` provided is `required`.

- The `module-options` list the parameters that are passed to the LoginModule to perform the authentication task. In this case, they point to the NCICB LDAP Server:

```
<module-option name="ldapHost">ldaps://ncids4a.nci.nih.gov:636</module-option>
```

```
<module-option name="ldapSearchableBase">ou=nci,o=nih</module-option>
```

```
<module-option name="ldapUserIdLabel">cn</module-option>
```

Deploy the UPT .war File

Copy the `upt.war` file to the JBoss deployment directory found at: `{jboss-home}/server/default/deploy/`.

Enable Audit Logging

In order to activate CLM's audit logging capabilities for UPT, you must follow the steps to deploy the audit logging service as mentioned in the [Audit Logging](#) section beginning on page 24.

In addition, you must place the `clm.jar` file in the common lib directory of the JBoss server.

Start JBoss

Once the deployment steps have been completed, start JBoss. Check the logs to confirm there are no errors while the UPT application is deployed on the server.

Once the JBoss server has completed deployment, open a browser to access the UPT. The URL is: `http://<jboss-server>/upt` where `<jboss-server>` is the IP Address or the DNS name of JBoss server.

After the UPT Login page appears, enter the UPT application using the login ID and password for the Super Admin identified when the database was created. Also use the UPT application Name specified for the application name.

Upon successful login, the UPT Application Home Page appears.

NOTE: In case of any errors, follow a debugging and troubleshooting procedure to diagnose and solve the issues.

Chapter 4 CSM Security Web Services

The CSM Security Web Services are introduced to expose the CSM authentication and authorization service features. The Security Web Services currently provide only two operations: Login and CheckPermission. The operations are exposed versions available in CSM API's.

Security Web Service WSDL

The CSM Security Web Service WSDL is diagrammed in Figure 4-1 below.

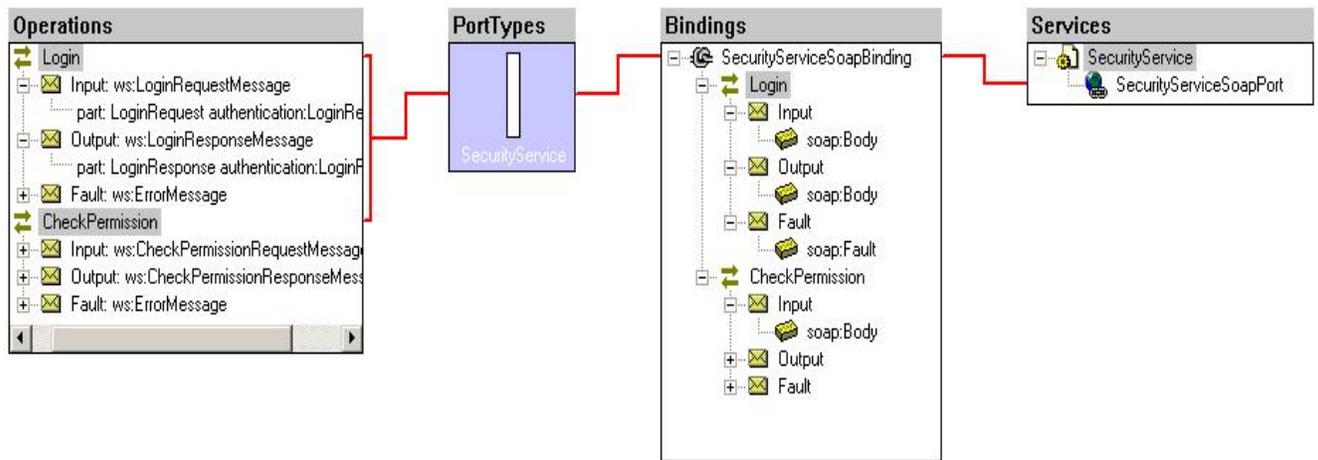


Figure 4-1 Security Web Service WSDL

The name of the exposed web service is 'SecurityService' and provides two operations, which are discussed in more detail in the next section.

Security Web Service Operations

The CSM Security Web Service provides two operations: Login and Check Permission. They are detailed in the sections below.

Login Operation

The Login web service operation is a request/response operation. This operation receives a LoginRequestMessage, performs authentication and responds with LoginResponseMessage to the web service consumer. If there are any problems with the processing the LoginRequestMessage and/or performing authentication on the user credentials then the web service operation will return a SOAP Fault response error message indicating an error code and the error details.

The below example shows the Schema (XSD) for Authentication:

```
<xs:schema targetNamespace="http://security.nci.nih.gov/ws/authentication"
  xmlns:authentication="http://security.nci.nih.gov/ws/authentication"
  elementFormDefault="qualified"
  attributeFormDefault="qualified"
```

```
version=".1">
<xs:element name="LoginRequest" type="authentication:LoginRequest"/>
  <xs:complexType name="LoginRequest">
    <xs:sequence>
      <xs:element name="UserName" type="xs:string"/>
      <xs:element name="Password" type="xs:string"/>
      <xs:element name="ApplicationContext" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

As displayed in the example above, the LoginRequest message consists of three parameters: UserName, Password and ApplicationContext. The Apache AXIS framework validates all request and response messages against the Schema specified in the Security Web Services WSDL.

When the LoginRequest message is received by the web service operation, the User credentials from the LoginRequest message are used by the CSM API to authenticate the user against privileges for the 'ApplicationContext'. If the User is authenticated and has privilege to access the ApplicationContext then a LoginResponse is returned with result value of 'true'. If the user is not authenticated and does not have access privilege for the 'ApplicationContext' then a LoginResponse is returned with the result value of 'false'.

CheckPermission Operation

The CheckPermission web service operation is a request/response operation. This operation receives a CheckPermissionRequestMessage, performs a permission check and responds with CheckPermissionResponseMessage. If there are any problems, the web service operation will return a SOAP Fault response error message indicating an error code and the error details.

The example below shows the Schema for Authorization.

```
<xs:schema targetNamespace="http://security.nci.nih.gov/ws/authorization"
  xmlns:authorization="http://security.nci.nih.gov/ws/authorization"
  elementFormDefault="qualified"
  attributeFormDefault="qualified" version=".1">
  <xs:element name="CheckPermissionRequest"
    type="authorization:CheckPermissionRequest"/>
  <xs:complexType name="CheckPermissionRequest">
    <xs:sequence>
      <xs:choice>
        <xs:element name="UserName" type="xs:string"/>
        <xs:element name="GroupName" type="xs:string"/>
      </xs:choice>
      <xs:element name="ObjectId" type="xs:string"/>
      <xs:element name="Attribute" type="xs:string" nillable="true"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

As displayed in the example above, the CheckPermission request message consists of UserName or GroupName, ObjectId, Attribute, Privilege and ApplicationContext. The Apache AXIS framework validates all request and response messages against the Schema specified in the Security WS WSDL.

When the CheckPermission request message is received by the web service operation, the CSM API's checkpermission method is invoked to check permission. If the User or Group has permission then a CheckPermissionResponse is returned with result value 'true' otherwise result value is 'false'.

Workflow for CSM Security Web Service

This workflow section outlines the basic steps, both strategic and technical, for successful CSM Security Web Services integration.

1. Read the deployment steps from this chapter as well as the rest of this guide. It provides an overview, workflow, and specific deployment and integration steps.
2. Determine the security requirements and provision security with CSM's UPT.
3. After the Security Web Service is deployed and user security provisioned with UPT, the Security Web Service is ready for operation and consumption.
4. Using the CSM Web Services Interface use the authentication and authorization operation exposed.
5. Using the LoginRequestMessage invoke and consume the Login Web Service Operation.
6. Using the CheckPermissionRequestMessage invoke and consume the CheckPermission Web Service operation.

Installation of CSM Security Web Service

The sections below provide the basic steps for installing the CSM Security Web Service as well as instructions for completing each step. As you follow the deployment steps, use the files containing the name corresponding with your database.

Create and Prime Database

Before beginning, you must log into the database using an account that has permission to create new databases. Make sure that the database you are about to create does not already exist. If it does, drop it to create the new one.

1. In the AuthSchemaMySQL.sql file replace the <<database_name>> tag with the target application scheme: *csmupt*.
2. Save and then run this script from the database prompt. This should create a database with the given name.
3. In the DataPrimingMySQL.sql file, perform the following:

- Replace the <<super_admin_login_id>> with the login id of the user who is going to act as the Super Admin for that particular installation. For example "doej" for John Doe admin.
 - Provide the first name and last name of the Super Admin user by replacing <<super_admin_first_name>> and <<super_admin_last_name >> with the appropriate values.
 - Replace the <<application_context_name>> with a test application entry such as **abc_app**. For example: the application name is 'abc_app' and application schema name is 'abc_app'. For the sake of this document we will use schema 'abc_app' and the application as 'abc_app'.
4. Save and then run the edited script from the database prompt. This should populate the database with the initial data.
 5. Verify the population of the data by querying the *application*, *user*, *protection_element*, and *user_protection_element* tables. They should have one record each.

The database will also include CSM Standard Privileges and the *privilege* table should have 7 entries.

Configure Datasource

Once the database has been created and primed, you must modify the `mysql-ds.xml` file, which contains information for creating a datasource, and then copy it to the appropriate location.

1. Edit the `mysql-ds.xml` file as follows:
 - Replace the <<database_user_id>> and <<database_user_password>> with the user id and password of the user account.
 - Replace the <<database_url>> with the URL needed to access the Authorization Schema residing on the database server:
`jdbc:mysql://<<stage_database_server_name>>:<<port>>/<<database_name>>`

Shown below is an example `mysql-ds.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
  <local-tx-datasource>
    <jndi-name>abc_app_ds</jndi-name>
    <connection-url>
jdbc:mysql://<<database_server_name>>:<<port>>/<<database_name>></connecti
on-url>
    <driver-class>org.gjt.mm.mysql.Driver</driver-class>
    <user-name>name</user-name>
    <password>password</password>
```

```

    </local-tx-datasource>
  </datasources>

```

2. Place the `mysql-ds.xml` file in the JBoss deployment directory located at: `{jboss-home}/server/default/deploy/`

Configure JBoss JAAS Login Parameters

In order to configure the CSM Web Service to verify against the LDAP, create an entry in the `login-config.xml` of JBoss as shown in the sample below. This entry configures a `login-module` against the application context. The location of this file is `{jboss-home}/server/default/conf/login-config.xml`.

Shown below is an example `login-config.xml` entry for “`abc_app`”.

```

<application-policy name = "abc_app">
  <authentication>
    <login-module code =
"gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule" flag =
"required" >
    <module-option name="ldapHost">ldaps://ncids4a.nci.nih.gov:636</module-option>
    <module-option name="ldapSearchableBase">ou=nci,o=nih</module-option>
      <module-option name="ldapUserIdLabel">cn</module-option>
    </login-module>
  </authentication>
</application-policy>

```

As shown in the example above:

- The `application-policy` name is the name of the application for defining the authentication policy. In this case, `abc_app`.
- The `login-module` code is the `LoginModule` class that is used to perform the authentication task. In this case it is: `gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule`.
- The `flag` provided is `required`.
- The `module-options` list the parameters that are passed to the `LoginModule` to perform the authentication task. In this case, they point to the NCICB LDAP Server:

```

<module-option name="ldapHost">ldaps://ncids4a.nci.nih.gov:636</module-
option>
<module-option name="ldapSearchableBase">ou=nci,o=nih</module-
option>
<module-option name="ldapUserIdLabel">cn</module-option>

```

You can also point to a RDBMS database using the username and password.

Deploy the Security WS .war File

Copy the `securityws.war` into the deployment directory of JBoss located at:
`{jboss-home}/server/default/deploy/`.

Chapter 5 CSM Instance Level and Attribute Level Security

Previously CSM APIs provided instance level and attribute level security. However this security is provided in the java tier.

The typical flow of events in the case of instance level security would be something like this:

- The user fires a business query on the database to obtain a result set.
- The entire result set is iterated through in Java, and for each and every record in it, a call is made to the CSM APIs to check whether or not the user or groups have access to that particular instance.

In the case of attribute filtering, for each of the accessible objects in the result set you would need to invoke the CSM APIs to check which attributes the user or groups can see.

In the both the situations mentioned above, there are several issues:

1. The entire result set is to be returned from the database to the application resulting in network traffic and latency.
2. Once the result set is obtained, it needs to be iterated through in Java, adding to processing time.
3. For each record there is a database call to CSM to determine if the user or groups has access or not.

The design of CSM v4.0 and higher address all the performance issues mentioned above. Details on how CSM provides Instance Level and Attribute Level security are discussed in the sections that follow.

Instance Level Security

Requirements Addressed

The following functional requirements are addressed and provided as part of CSM's instance level security solution.

Direct Instance Level Security

Direct Instance Level Security can be defined as where security for a particular instance is dependent on itself. A user or group(s) has access to a particular object based on the value of one of its attributes. There is no relation or association with another object. This type of instance level security is ad hoc and dependant on the associations done between that instance and the user or group(s) by security administrators.

For example, out of 456 patients in the patient table, user ABC has access to a specifically assigned 28 patients, based on the patient id.

Here out of the total patients in that database, the security administrator has assigned 28 patient ids to user ABC. Based on this, the solution should filter any query fired on the patient table such that for user ABC only those 28 records are accessible. This example also applies for group(s) where groups G1, G2, and G3 have access to 28 patients based on patient id. The results would be the same for those groups as they are for the example user.

Cross Dependant Instance Level Security

Cross Dependant Instance Level Security can be defined as where security for a particular instance is dependent on some other object. A user or group(s) has access to a particular object based on its association to some other higher level object, to which the user has been granted access. There is an association with another object which is generally higher up in the data graph. This type of instance level security is based on the relationship between the queried tables to the table to which the security is assigned. This type of security is used generally where it is much easier to assign and manage security at a the higher level of data

For example, a user or group(s) has access to only those lab results that are associated to a particular study to which the user or groups(s) have access.

Here in this example there can be thousands of lab results whereas the number of studies could be less than 100. Also as per the business rule, if user or group(s) is assigned access to the study then the user or group(s) can access everything associated to with that study. In this case it is presumed that the assignment and management of security is much easier for the studies as they are less in number.

Instance Level Security at User or Group Level

In CSM v4.0 the Instance Level Security was supported at User level only. As of v4.1, CSM supports Instance Level Security at group level, which allows groups to be considered when performing instance level security. An administrator can provision groups using UPT, and associate roles with those groups. This way, administrators can design the application's instance level security at the group level out of the box. This simplifies and reduces the effort involved for respective applications.

Another goal of this requirement is to ensure caGrid compatibility for instance level security. Instance level security within CSM should work with groups as defined in the Grid Grouper application.

Support Non-CSM Custom Domain Object Filters

CSM v4.1 supports the non-CSM custom filters defined for respective domain objects. The non-CSM custom filters can be defined using HBM or via @Filter annotations.

Integration of Instance Level Security for an SDK Generated System

CSM's instance level solution is integrated with the caCORE SDK so that it can be provided as an out of the box solution for SDK generated systems.

Instance Level Security Support for Non-SDK Systems

CSM is adaptable for Non-SDK systems with minor modifications if required. The general principle should be same as for an SDK generated system. It can be assumed that users will need to configure the solution and adapt it for their application.

Overall Design

In order to provide instance level security, CSM utilizes the filter capability provided by Hibernate. These filters contain filtering queries that are injected to the actual business queries, which are fired by the user. These filters are applied at the class level. So that whenever the class is queried, the attached filter is appended to the actual business query directly by Hibernate.

CSM provides capabilities for creating these filters through its UPT tool. It allows you to configure these filters for either the Direct or Cross Dependant type of instance level security. These filters contain queries which join with the CSM tables to obtain the instances of data on which the user or group has access. These filters are stored in the CSM database.

The custom (non-CSM) filters defined via HBM files or by using @filter annotations are also considered. At runtime, the client application calls CSM's helper methods, which retrieve these filters from the CSM Database. They also inject these filters into hibernate configuration for the appropriate classes.

Now since these filters are to be applied for a particular user or groups, the user name or group names are passed as parameters to these filters. So at runtime, filter queries are injected into the actual user or group queries. This combined query is fired at the database and the resulting data is filtered based on the instances on which the user or group has access.

Provisioning Instance Level Security

A new menu tab has been added to UPT for the purpose of provisioning Instance Level Security. This tab lets you configure the filter clauses for various classes in your application. Once the filtering clauses are configured, an Admin can create Protection Elements for the Instances of Objects on which the users have access, and assign them access. The following activity diagram (Figure 5-1) shows how the filter clause functionality under the new Instance Level menu tab works

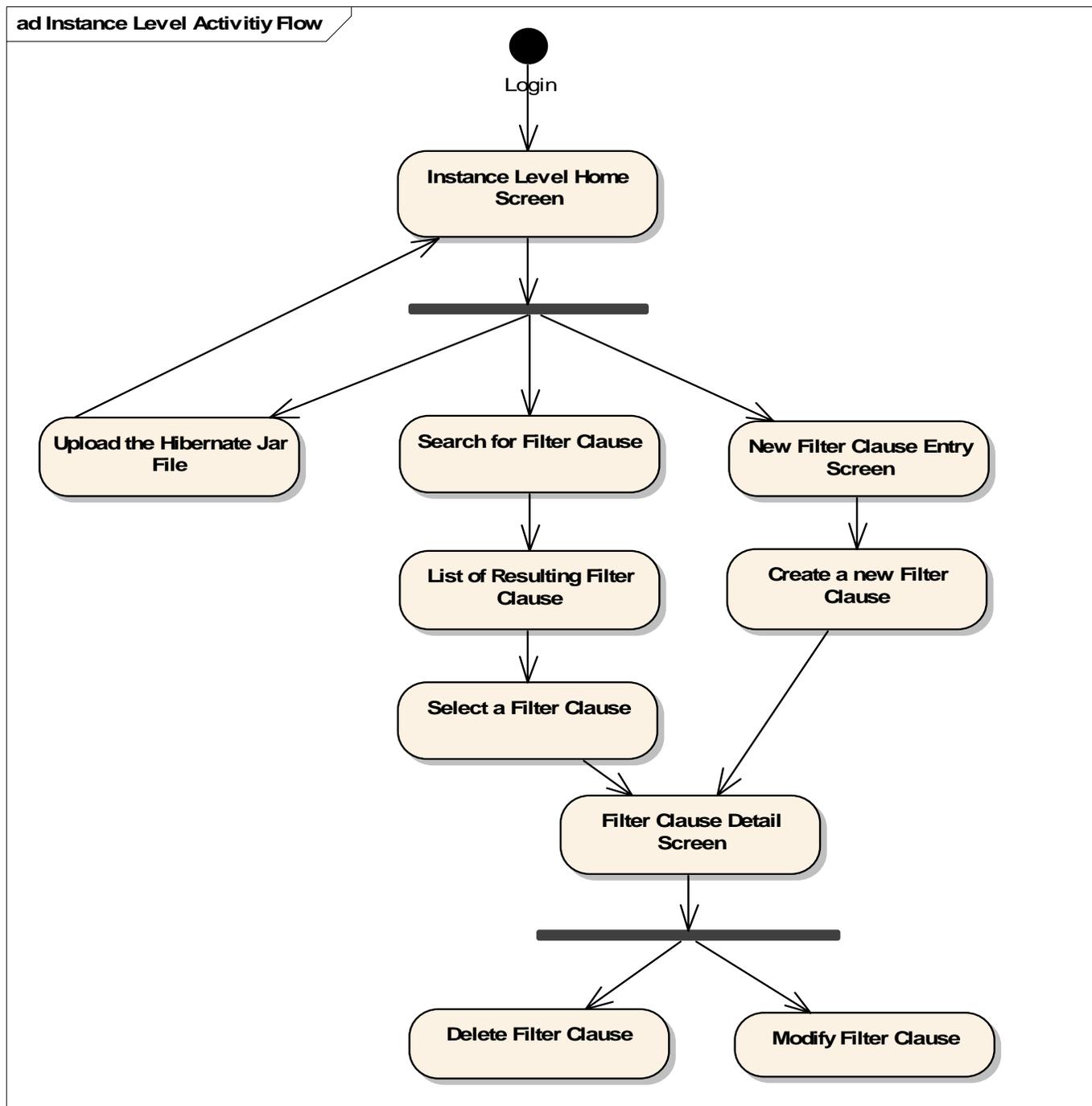


Figure 5-1 Instance Level Activity Flow

The details for performing these operations are provided in the UPT chapter of this document, in [Instance Level Security](#) on page 62.

In addition, the Protection Element has been enhanced to now include a new value field, which the admins can use to provide values for the instances on which users have access.

The sections that follow provide a workflow for provisioning instance level security.

Uploading an Application File

The first step is to upload a file which contains the Hibernate files along with the domain objects. This file should be a valid java archive and contain the following.

- Hibernate Configuration File – with database connection information.
- Hibernate Mapping files if not using annotations.
- Domain Objects Or Domain Objects with Hibernate Annotations.

NOTE: As of CSM v.4.1, the Hibernate annotations and @Filter annotations are supported with the Instance Level Security feature.

In the case of an SDK generated system, there are two jar files generated containing the Hibernate and Domain Objects separately. In this case, both of these files have to be uploaded.

In addition, a fully qualify hibernate configuration file name should be provided along with the files. Once the file is successfully uploaded a success message is given to the user.

Creating a Filter Clause

Once the file containing the Hibernate information is uploaded, we can use it to create filter clauses for different objects.

On the filter clause screen, the user first has to select the class for which they want to add the filter. Once the class is selected, the Filter Chain drop-down list is automatically populated with the associated classes.

NOTE: There is an entry for the master class itself in the list. This is to allow for direct instance level security.

If you want to provision a Direct Instance Level Security, select the class itself from the Filter Chain drop-down list and click **Done**.

In case of Cross Dependant Security, select the associated class from the Filter Chain drop-down list. Note that you can drill down the class hierarchy by clicking the Add button located in the Filter Chain section of the Filter Clause Details form. This brings the associated child classes. Once you have reached the final class on which the security for the class is dependant, you can click **Done**.

On pressing Done, the attribute list is populated with the attributes from the last class in the filter chain. Select the attribute on whose value the user will be granted access.

Once selected you can also provide an alias for the target class name and attribute. This is in cases where the attribute selected holds value for some other class. For example you have a Patient Object which has an attribute Security Key on whose value you want to filter the instances. However from a business perspective, the actual value in the Security Key is the value of the Study Id to which the patient belongs. In this case even though the security filter is set for the Patient based on the security key attribute, in business sense the filtering is happening at the Study Id Level. Hence you can provide an alias that will be used to determine to which protection elements the user has been granted access.

Once everything is selected, clicking **Add** at the bottom of the filter form creates the filter. Once the filter is created, the filtering SQL statements are generated and displayed back to the user.

The generated SQL statements are for User and Group level filtering. Note that these fields are kept editable to allow users to modify the SQL in case if they want to optimize it further. The User SQL generated will be used if instance level security is being done at the user level. The SQL generated for Group will be used when instance level security is being doing for group level. Both SQL statements are different and special attention must be paid while editing any of the SQL filter query.

Creating Protection Elements

Once the security filters have been created, you need to provision the actual instances on which the user has access. This is done by creating protection elements for these instances and providing access to the users.

The following table provides descriptions of the Protection Element fields that an Admin must populate when creating a Protection Element.

<i>Field Name</i>	<i>Description</i>
Protection Element Name	Distinct name that uniquely identifies the Protection Element
Protection Element Description	Description of the Protection Element
Protection Element Type	Can be left blank
Protection Element Object Id	The target class name on which the security of the master class depends. If an alias class name is used, then the alias should be entered here.
Protection Element Attribute	The name of the attribute of the target class on which the security of the master class depends. If an alias attribute name is used then the alias should be entered here
Protection Element Value	The actual value of the attribute on which user has access.
Update Date	Date when the protection element was last updated

Figure 5-2 Protection Element Fields

Using Instance Level Security

In order for the client application to inject Instance Level Security, CSM provides a helper class which assists them. This class contains methods which allow the user to add these filters to the Hibernate Configuration at the time of loading of the system, and also initialize and parameterize these filters at runtime for the particular user or group(s) firing the query.

To add filters when using instance level security for user:

```
public static void addFilters( AuthorizationManager authorizationManager,
Configuration configuration)
```

```
public static void addFilters(AuthorizationManager
authorizationManager, Configuration configuration, List<String>
definedFilterNamesList)
```

To add filters when using instance level security for groups:

```
public static void addFiltersForGroups( AuthorizationManager authorizationManager,
Configuration configuration)
```

```
public static void addFiltersForGroups(AuthorizationManager
authorizationManager, Configuration configuration, List<String>
definedFilterNamesList)
```

One of these methods should be called only once for an application just after the Hibernate Configuration object is created (by reading the configuration file) and before the Session Factory Method is created. The methods inject the security filters that have been created for this application. CSM retrieves a list of all the filters that have been defined for the application from the CSM Database.

Since the non-CSM custom filters defined in HBM or via `@Filter` annotations are supported, if the `definedFilterNamesList` parameter is passed while adding filters, then the named filters will be added as well to the persistent classes.

After that, for each filter in the list, CSM creates a new `FilterDefinition` (Hibernate) object. CSM then retrieves the persistent class from the passed `Configuration Object` using the class name for which the filter is defined. CSM then adds the filter to the persistent class by setting the filtering query.

To initialize filters when instance level security for user:

```
public static void initializeFilters (String userName, Session session,
AuthorizationManager authorizationManager)
```

```
public static void initializeFilters(String groupNames, Session session,
AuthorizationManager authorizationManager, Map<String,String>
definedFilterNamesMap)
```

To initialize filters when instance level security for groups:

```
public static void initializeFiltersForGroups(String[] groupNames, Session session,
AuthorizationManager authorizationManager)
```

```
public static void initializeFiltersForGroups(String[] groupNames, Session session,
AuthorizationManager authorizationManager, Map<String,String>
definedFilterNamesMap)
```

One of these methods should be invoked after obtaining the `Session` from the `SessionFactory` and just before executing the user query. This method initializes the filters that are already added to the `SessionFactory`. This method first obtains the list of all the defined filters from the `SessionFactory` in the passed `Session` object. It then iterates through the filter list and sets the *user name* or *group names* and the *application name* parameters. It retrieves the *application name* from the passed `Authorization Manager`.

It is important to note that the instance level security filters can be added or initialized for a user or groups exclusively. Meaning that if the `addFilter` method is invoked for groups, then the `Group(s)` based instance level security filter queries are added. As mentioned above, the decision to utilize user or group based instance level security must be made before adding the filters to the Hibernate session and should not be used interchangeably.

Known Issues

In case of eager loading, filtering of the child object doesn't work.

Hibernate, by default, injects only the filter for the parent object. In the case where you have the eager loading mode set to "true", the child objects (the associated objects which are being eagerly loaded) filter is not injected. SDK by default comes with eager loading set to "false", leaving up to the users to explicitly turn it on.

Multiple filters on a single object will be always AND-ed

If you have multiple filters defined for a single domain object, Hibernate will inject all of them with "AND" conditions between them. This is the default behavior of Hibernate and would require programmatic enhancements to handle the OR-ing of filters.

Filtering in the case of inheritance needs to be further investigated

Hibernate DTD has a limitation of not allowing a user to add a filter for the inherited classes. The DTD allows filters only to be added to the super class. However Hibernate API allows the addition of these filters. This issue will be investigated in detail during implementation, and the results will be posted accordingly.

Attribute Level Security

Requirements Addressed

The following functional requirements have been addressed and are provided as part of CSM's attribute level security solution.

Attribute Level Security

CSM provides Attribute level security at object level. Attribute level security can be defined as security where you can control access to the attributes of an object. A user can be granted and revoked access to these attributes. Based on the user's access level, those attributes should be visible to the user or not.

For example, a Patient object has the following five attributes: Name, Address, Social Security Number, Phone Number, and Disease. A researcher who has access to all of the attributes except Social Security Number should be able to see the Patient object with all attributes filled with data except the Social Security Number attribute.

Both Single or Many Object Retrieval

CSM provides Attribute level security both for queries that result in a single object being returned from the database as well as a list of objects being returned from the database. In case of the list, each object in the list should be filtered based on the attributes to which the user has access.

Runtime Decision of Strict or Lenient Behavior

By default, the CSM Instance and Attribute level security denies access to all attributes of an object instance unless the user/group is provisioned to gain access on particular attributes. This is the default strict behavior of the attribute level security feature of CSM. This new feature allows applications to configure attribute level security to enforce either strict behavior or lenient behavior. Lenient behavior

allows access to all associations within Parent and Child objects while securing access to the rest of the attributes of an object based on the user/group security provisioning done via UPT.

Automatically Provide Attribute Level Security for SDK Generated Systems

Attribute Level Security is integrated with the caCORE SDK so that it can be provided as an out of the box solution for SDK generated systems.

Provide Attribute Level Security Support for Non-SDK Generated Systems

CSM is adaptable for non SDK generated systems with minor modifications if required. The general principle is the same as for an SDK generated system. It can be assumed that users will need to configure the solution and adapt it for their application.

Overall Design

CSM utilizes the SessionInterceptor feature provided by Hibernate to inject attribute level security. It traps a user session during the loading of an object from the underlying database. During the load process it intercepts the incoming stream of result data and checks to see which attributes the user has access to. If the user does not have access to an attribute, it nullifies the attribute value such that the resulting data contains values for only those attributes to which the user has access.

Since it would need to access the CSM table to check if user has access to an attribute every time an object is loaded, the solution implements a cache that holds the users attribute access map. The interceptor looks up against this cache to inject attribute level security, speeding up the overall filtering process.

Strict Or Lenient Behavior

This is a new feature added in the CSM v4.1 version. This feature for attribute level security was requested and provided for caCORE SDK, though it can also be used by non-SDK generated systems. The default behavior is the strict behavior that was introduced in CSM v4.0. In the strict behavior, the attribute level security feature restricts access to all attributes of an object to which a user/groups does not have access.

The new lenient behavior allows access to those attributes of an object that are of an association type. This feature is, as mentioned above, implemented by the caCORE SDK to support the new Writable API feature of SDK. The lenient behavior ignores attributes of association type and hence leaves intact any associations between parent and child objects.

Provisioning Attribute Level Security

There are new special changes in the UPT for provisioning of Attribute Level Security. If attribute level security is turned on, by default all object attributes are secured. If you want to grant a user access to an attribute, you must create a protection element for that attribute and then grant the user access to it like any other protection element.

The following table provides descriptions of the Protection Element fields that an Admin must complete when creating a new Protection Element.

Field Name	Description
Protection Element Name	Distinct name which can identify the Protection Element
Protection Element Description	Description for the Protection Element
Protection Element Type	Can be left blank
Protection Element Object Id	The class name on whose attribute the user is to be granted access
Protection Element Attribute	The attribute name on which the user is to be granted access
Protection Element Value	Can be left blank
Update Date	Date when the protection element was last updated

Figure 5-3 Protection Element Fields

Using Attribute Level Security

In order to use Attribute Level Security, the client application must attach the attribute level Session interceptor to its session. This can be done when obtaining the Hibernate Session from the SessionFactory object, as shown in the examples below. Once the session interceptor is in place, it will inject Attribute level security every time an object is loaded from the database for a query.

The default strict behavior is implemented as follows:

```
// default strict behavior
Session session = sessionFactory.openSession(new
AttributeSecuritySessionInterceptor());

Session session = sessionFactory.openSession(new
AttributeSecuritySessionInterceptor(false));
```

The lenient behavior is implemented as follows:

```
// lenient behavior
Session session = sessionFactory.openSession(new
AttributeSecuritySessionInterceptor(true));
```

To inject custom interceptors along with CSM's attribute level security interceptor:

```
// inject custom Interceptors
List<Interceptor> interceptors = new ArrayList<Interceptor>();
Interceptors.add(new AttributeSecurityInterceptor(true));
Interceptors.add(new ObjectStateInterceptor());
Session session = sessionFactory.openSession(new new
GenericSecurityInterceptor(interceptors));
```

Known Issues

In the case of eager loading, attribute filtering happens only for the parent object.

The *onLoad* method is invoked for each record returned from the database. However this works only for the parent object. If you have eager loading set to “true”, the child object’s attributes (the associated objects which are eagerly loaded) aren’t filtered. SDK, by default, comes with eager loading set to “false”, leaving up to the users to explicitly turn it on.

Primitive attribute type filtering is not possible.

Since a primitive data type cannot be set to null, the current attribute solution does not work if the domain objects contain primitive data types as attributes. The default values for primitive data types (0 for int, false for a boolean) can be valid values, meaning that setting primitive attributes to their default values is also not an option.

Filtering on queries with projection on certain attributes will not work.

For queries where the user has set a project on certain attributes of the object rather than returning the whole object back, the CSM attribute level security will not work. This is because in the case of projections, Hibernate returns the attribute values directly from the database as Java data types. As a result, the *onLoad* method of the session interceptor is not invoked, thereby not injecting the attribute level security.

Chapter 6 Acegi Adapter

The Acegi Framework is quickly becoming the preferred framework for many Spring framework powered applications to implement security. Acegi Security is the de-facto standard for security in Spring Framework. Existing applications and new applications wanting to leverage CSM can now do so using the CSM Acegi Adapter.

The CSM Acegi Adapter allows applications to use CSM's Authentication and Authorization under the Acegi Security Framework. CSM Acegi Adapter implementation provides Authentication, Authorization - Method Level Security and Object Parameter level security.

Implementation

Acegi Security is widely used within the Spring community for comprehensive security services to Spring-powered applications. It comprises a set of interfaces and classes that are configured through a Spring IoC container. The design of Acegi Security allows many applications to implement the common enterprise application security requirements via declarative configuration settings in the IoC container. Acegi Security is heavily interface-driven, providing significant room for customization and extension. Important Acegi Security, like Spring, emphasizes pluggability.

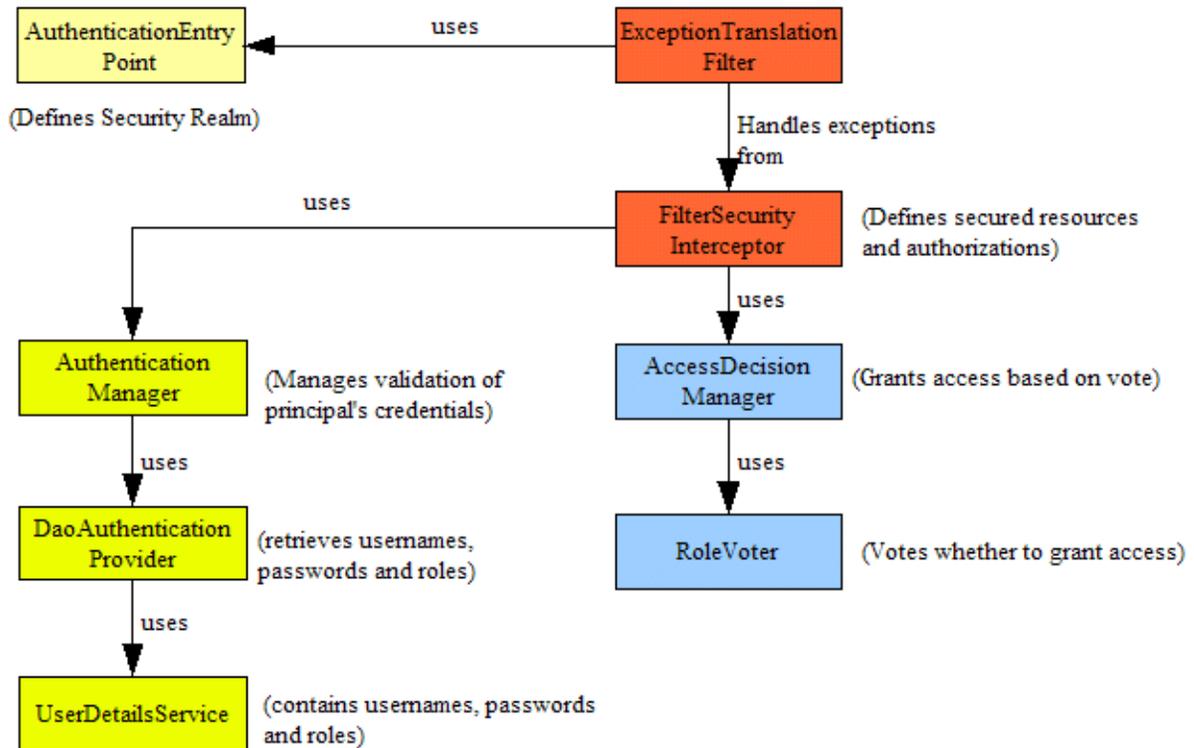


Figure 6-1 Authentication and Authorization in Acegi framework

Figure 6-1 demonstrates the control flow by Acegi for authentication and authorization. The CSM Acegi Adapter uses this approach to provide CSM security.

Authentication is implemented by extending this design. Acegi provides interceptors which can be configured through Acegi Security Configurations in Spring. For a detailed understanding of the Acegi Frameworks Authentication and Authorization implementation by CSM, see the sections that follow.

NOTE: Detailed explanations of Acegi interfaces implemented by CSM Acegi Adapter are beyond the scope of this guide. Refer the Acegi Security CSM Adapter Design document for details, and check out Acegi security reference documentation, available at: <http://www.acegisecurity.org/guide/springsecurity.html>.

Currently the CSM Acegi Adapter implementation provides Method Level and Method Parameter Level security.

Method Level Security

The current out of box implementation of the CSM Acegi Adapter provides method level security. The Adapter implements Acegi's MethodInterceptor. The CSMMethodSecurityInterceptor, CSM's custom implementation of the MethodInterceptor, enables security at method level by intercepting method calls on the secured bean specified in the MethodDefinitionSource. All the methods are intercepted for each secured bean. Please see the Workflow and Integrating and Configuring sections below for more details.

Method Parameter Level Security

In this implementation, the CSM Acegi Adapter provides method parameter level security. Applications that need method parameter level security have to implement CSM's SecurityHelper. The SecurityHelper interface provided by CSM, allows the application to control authorization. Refer the CSM API source for more details.

Workflow

Below is a basic list of steps needed to implement the CSM Acegi Adapter.

1. Determine the level of security required for your application: Method Level, Object Parameter Level, etc.
2. Define the beans that need to be protected.
3. Define appropriate Security Interceptors.
4. Define Security Interceptors for various beans that need protection.
5. Configure the Acegi security configuration file: `csm-acegi-security.xml`.
6. Configure a JAAS LoginModule for the Application Context.
7. Configure database properties.
8. Configure user provisioning using CSM UPT.

Integrating and Configuring

This section serves as a guide to help developers integrate applications with CSM Acegi Adapter. It outlines a step by step process that addresses what developers need to know in order to successfully integrate CSM's Acegi Adapter into their applications. This includes:

- Configure Acegi Security in `csm-acegi-security.xml`.
- Database properties and configuration.
 - Configure Datasource OR
 - Configure Hibernate configuration file.
- LDAP properties and configuration.
- Provision user access authorization policy.

Configure Acegi Security

This section provides instructions for configuring Acegi Security. The examples used in this section are taken from a sample configuration file, which has been provided in full in [Appendix A, CSM/ACEGI Sample Configuration File](#) on page 105.

1) Define the beans that need to be protected.

Example from Appendix A:

```
<bean id='applicationService'
class='test.gov.nih.nci.security.acegi.sdk.ApplicationServiceImpl' /> .
```

This configuration secures the `ApplicationServiceImpl` class and intercepts all of its method calls.

2) Define the SecurityHelper Impl Class.

This class needs to be implemented by the developers that want to integrate CSM Adapter into their new or existing application with Acegi Security Framework. In this implementation it is a custom `CSMMethodSecurityInterceptor` that intercepts any method calls on the 'applicationService' bean.

Example:

```
<bean id='securityHelper'
class='test.gov.nih.nci.security.acegi.sdk.SecurityHelperImpl' />
```

3) List the beans that need to be protected by the 'securityInterceptor' for the 'autoProxyCreator'.

Example:

```
<bean id='autoProxyCreator'  
class='org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator'  
>  
<property name='interceptorNames'>  
    <list>  
        <value>securityInterceptor</value>  
    </list>  
</property>  
<property name='beanNames'>  
    <list>  
        <value>applicationService</value>  
    </list>  
</property>  
</bean>
```

4) Specify the Application Context that will be used for CSM's Authentication and Authorization service.

Example:

```
<bean id="userDetailsService"  
    class="gov.nih.nci.security.acegi.authentication.CSMUserDetailsService"  
>  
<!-- Specify the Application Context required by CSM -->  
    <property name="csmApplicationContext">  
        <value>acegittest</value>  
    </property>  
</bean>
```

Database Properties and Configuration

You can use either MySQL or Oracle as your database of choice to host the authorization data. Since the instructions in this section provide steps for both database types, be sure you follow the appropriate instructions based on your database selection.

When deploying Authorization, application developers may want to make use of a previously-installed common Authorization Schema. In this case, a database already exists so you do not need to create one (skip the procedures immediately below).

NOTE: The Authorization Schema used by the run-time API and the UPT must be the same. The sections that follow provide the instructions needed.

Create and Prime Database

Use the following steps (if necessary) to create a new database and prime it for authorization deployment. These procedures also install a new Authorization Schema. If you are using a previously-installed common Authorization Schema, skip these instructions as your database and schema already exist.

1. Log into the database using an account id that has permission to create new databases.
2. In the `AuthSchemaMySQL.sql` or `AuthSchemaOracle.sql` script, replace the `<<database_name>>` tag with the name of the authorization schema (e.g., *acegitest*).
3. Save and then run this script from the database prompt. This should create a database with the given name. The database will include CSM Standard Privileges.
4. In the `DataPrimingMySQL.sql` or `DataPrimingOracle.sql` file, perform the following:
 - o Replace the `<<application_context_name>>` with the name of application. This is the key to derive security for the application. This will be called application context name.
 - o Replace the `<<super_admin_login_id>>` with the login id of the Super Admin.
 - o Replace the `<<super_admin_login_id>>`, the `<<super_admin_first_name>>` and `<<super_admin_last_name>>` entries with the super admin user's login id, first name, and last name.

NOTE: The default password is always “changeme” and should be used for logging into the application's UPT for the first time. After initial login, change this password immediately.

5. Save and then run this script from the database prompt. This should populate the database with the initial data.
6. Verify this by querying the application table. It should include one record only.

Configure Datasource

Use the steps below to modify the database file that contains information for creating a datasource.

NOTE: One entry is required for each database connection.

1. Modify the provided `mysql-ds.xml` or `oracle-ds.xml` file as follows:
 - a. Replace the `<<application_context_name>>` tag with the name of the authorization schema (for example, *acegitest*).
 - b. Replace the `<<database_user_id>>` and `<<database_user_password>>` with the user id and with the password of the user account that will be used to access the Authorization Schema created in the section above.
 - c. Replace the `<<database_url>>` with the URL needed to access the Authorization Schema residing on the database server.

2. Place the edited `mysql-ds.xml` or `oracle-ds.xml` file in the JBoss deployment directory located at:
`{jboss-home}/server/default/deploy/.`

Shown below is an example of the `mysql-ds.xml` file.

```
<datasources>
  <local-tx-datasource>
    <jndi-name>csmupt</jndi-name>
    <connection-url>jdbc:mysql://mysql_db:3306/csmupt</connection-url>
    <driver-class>org.gjt.mm.mysql.Driver</driver-class>
    <user-name>name</user-name>
    <password>password</password>
  </local-tx-datasource>
  <local-tx-datasource>
    <jndi-name>acegitest</jndi-name>
    <connection-url>jdbc:mysql://mysql_db:3306/csd</connection-url>
    <driver-class>org.gjt.mm.mysql.Driver</driver-class>
    <user-name>name</user-name>
    <password>password</password>
  </local-tx-datasource>
</datasources>
```

Configure Hibernate Configuration File

If the integrating application does not want to use `datasources`, the Hibernate configuration file can be used. Use the steps below to modify the Hibernate configuration file that contains information for creating a `datasource`.

NOTE: One entry is required for each database connection.

Modify the provided `acegitest.csm.new.hibernate.xml` as follows:

1. Replace the `connection.url` property name with URL needed to access the Authorization Schema residing on the database server, followed by the name of the authorization schema.
2. Replace the `connection.username` and `connection.password` property names with the user id and password of the user account that will be used to access the Authorization Schema created in the section above.

Below is an example of an `acegitest.csm.new.hibernate.xml` file configured for application context name `acegitest`:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 2.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-2.0.dtd">
<hibernate-configuration>
```

```

<session-factory>
  <property
name="connection.url">jdbc:mysql://<<server>>:<<port>>/acegitest</property>
  <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="connection.username">USERNAME</property>
  <property name="connection.password">PASSWORD</property>
  <property name="connection.driver_class">org.gjt.mm.mysql.Driver</property>
  <property name="hibernate.show_sql">>false</property>
  <property name="connection.zeroDateTImeBehavior">convertToNull</property>
  <property name="hibernate.cache.use_query_cache">>false</property>
  <property name="hibernate.cache.use_second_level_cache">>false</property>

  <mapping
resource="gov/nih/nci/security/authorization/domainobjects/Privilege.hbm.xml"/>
  <mapping
resource="gov/nih/nci/security/authorization/domainobjects/Application.hbm.xml"/>
    <mapping resource="gov/nih/nci/security/authorization/domainobjects/Role.hbm.xml"/>
    <mapping resource="gov/nih/nci/security/dao/hibernate/RolePrivilege.hbm.xml"/>
    <mapping resource="gov/nih/nci/security/dao/hibernate/UserGroup.hbm.xml"/>
  <mapping
resource="gov/nih/nci/security/dao/hibernate/ProtectionGroupProtectionElement.hbm.xml"/>
  <mapping
resource="gov/nih/nci/security/authorization/domainobjects/Group.hbm.xml"/>
    <mapping resource="gov/nih/nci/security/authorization/domainobjects/User.hbm.xml"/>
  <mapping
resource="gov/nih/nci/security/authorization/domainobjects/ProtectionGroup.hbm.xml"/>
  <mapping
resource="gov/nih/nci/security/authorization/domainobjects/ProtectionElement.hbm.xml"/>
  <mapping
resource="gov/nih/nci/security/authorization/domainobjects/UserGroupRoleProtectionGroup.
hbm.xml"/>
  <mapping
resource="gov/nih/nci/security/authorization/domainobjects/UserProtectionElement.hbm.xml
"/>

</session-factory>

</hibernate-configuration>

```

Configure JAAS LoginModule

Developers can configure a login module for each application by making an entry in the JAAS configuration file for that application name or context.

The general format for making an entry into the configuration files is shown in the following example:

```
Application 1 {
    ModuleClass Flag  ModuleOptions;
    ModuleClass Flag  ModuleOptions;
    ...
};
Application 2 {
    ModuleClass Flag  ModuleOptions;
    ...
};
```

For *acegittest*, which uses *RDBMSLoginModule*, the JAAS configuration file entry is as the below example shows:

```
acegittest
{
    gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule
    Required
    driver=" org.gjt.mm.mysql.Driver"
    url=" jdbc:mysql://<<server>>:<<port>>/acegittest "
    user="USERNAME"
    passwd="PASSWORD"
    query="SELECT * FROM users WHERE username=? and password=?"
    encryption-enabled="YES";
}
```

The configuration file entry shown above contains the following information

- The **application** is *acegittest*.
- The **ModuleClass** is *gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule*
- The *Required* flag indicates that authentication using this credential source is a must for overall authentication to be successful.
- The **ModuleOptions** are a set of parameters that are passed to the ModuleClass to perform its actions.

In the prototype, the database details as well as the query are passed as parameters:

- `driver=" org.gjt.mm.mysql.Driver "`
- `url=" jdbc:mysql://<<server>>:<<port>>/acegittest "`
- `user="USERNAME"`
- `passwd="PASSWORD"`
- `query="SELECT * FROM users WHERE username=? and password=?"`
- `encryption-enabled="YES"`

As shown in the example, since the *acegittest* application has only one credential provider, only one corresponding entry is made in the configuration file. If the application uses multiple credential providers, the LoginModules can be stacked. In addition, a single configuration file can contain entries for multiple applications.

User provisioning via UPT

The following steps need to be completed in order to provide User Provisioning through UPT:

- Create Protection Elements for objects that need to be secured.
- Create Protection Group for the Protection Elements.
- Create a Role with Privilege assigned to it.
- Create a User.
- Assign Protection Group and Role to the Users that are allowed access.

Chapter 7 CSM caGrid Authentication

caGrid is a core infrastructure project of the cancer Bioinformatics Grid. It consists of architectural components and tools that enable any application to be deployed on the grid as a node. It also provides tools for discovering existing grid services and invoking them.

In order to be able to securely invoke grid services, the caGrid architecture needs to authenticate and authorize the user trying to make the service call. This requires both an authoring mechanism to provide appropriate permissions to the user, and a runtime mechanism to verify the granted permissions.

The CSM capabilities for this solution described in this chapter provide information on how CSM can be leveraged in the caGrid environment.

Authentication for caGrid

CSM is designed to return a subject for a user upon authentication. The returned subject contains user attributes like Last Name, First Name, and Email Id, which are required to prepare the SAML which is to be sent to Dorian.

CSM Configuration for IDP/Authentication Service

CSM has been integrated into the caGrid IDP module to facilitate local authentication. In order to support creation of SAML assertions by the IDP, CSM needs to retrieve user attributes from the Credential Providers and supply them back to the caGrid component. In order to be able to retrieve these attributes, CSM provides configuration settings that can be used to map the attributes to individual credential providers. These attributes are returned as CSM currently returns Principles in a JAAS Subject as part of the following new method added to the AuthenticationManager.

```
public Subject authenticate(String userName, String password) throws CSEException,  
    CSLoginException, CSInputException, CSConfigurationException,  
    CSInsufficientAttributesException;
```

Listed below are the attributes that are returned and their corresponding PrincipleNames:

- First Name - gov.nih.nci.security.authentication.principal.FirstNamePrincipal
- Last Name - gov.nih.nci.security.authentication.principal.LastNamePrincipal
- Email Id - gov.nih.nci.security.authentication.principal.EmailIdPrincipal
- First Name - gov.nih.nci.security.authentication.principal.LoginIdPrincipal

Both RDBMSLoginModule and LDAPLoginModule have been updated to return these attributes. The next two sections talk about how this is done.

Configuring RDBMS Login Module for CSM/caGrid IDP Integration

If an application uses an RDBMS Server from which the user attributes are to be retrieved, the attribute mapping described above should be added in the JAAS login-config file.

Below is a sample entry for the attribute mapping in the JAAS `login.conf` file:

```
RDBMSGRID{
    gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule Required
    driver="org.gjt.mm.mysql.Driver"
    url="jdbc:mysql://mysql_db_server:3620/CSMAuthSchema"
    user="USER "
    passwd="PASSWORD"
    TABLE_NAME="CSM_USER"
    USER_LOGIN_ID="LOGIN_NAME"
    USER_PASSWORD="PASSWORD"
    USER_FIRST_NAME="FIRST_NAME"
    USER_LAST_NAME="LAST_NAME"
    USER_EMAIL_ID="EMAIL_ID";
};
```

Where:

- TABLE_NAME is the name of the table where the attributes can be found
- USER_LOGIN_ID is the name of the column in the table storing the user's login id
- USER_PASSWORD is the name of the column in the table storing the user's password
- USER_FIRST_NAME= is the name of the column in the table storing the user's first name
- USER_LAST_NAME= is the name of the column in the table storing the user's last name
- USER_EMAIL_ID= is the name of the column in the table storing the user's email id

NOTE: In order to activate CLM's audit logging capabilities for the authentication service, the user needs to follow the steps to deploy audit logging service as mentioned in the [Audit Logging](#) section on page 24.

Configuring LDAP Login Module for CSM/caGrid IDP Integration

If an application uses an LDAP Server from which the user attributes are to be retrieved, the attribute mapping described above should be added to the JAAS login-config file.

Below is a sample entry for the attribute mapping in the JAAS `login.conf` file:

```
LDAPGRID{
    gov.nih.nci.security.authentication.loginmodules.LDAPLoginModule Required
    ldapHost="ldap://ncicbds-dev.nci.nih.gov:389"
    ldapSearchableBase="ou=csm,dc=ncicb-dev,dc=nci,dc=nih,dc=gov"
```

```

    ldapUserIdLabel="uid"
    ldapAdminUserName="uid=csmAdmin,ou=csm,dc=ncicb-
dev,dc=nci,dc=nih,dc=gov"
    ldapAdminPassword="PASSWORD"
    USER_FIRST_NAME="givenName"
    USER_LAST_NAME="sn"
    USER_EMAIL_ID="mail";
};

```

Where:

- USER_FIRST_NAME is the ldap attribute which stores the first name
- USER_LAST_NAME is the ldap attribute which stores the last name
- USER_EMAIL_ID is the ldap attribute which stores the email id

Authorization for caGrid

Using Grid Group Names for Check Permission

As part of the CSM caGrid Integration, CSM allows users to check permission using the Grid Grouper Group Name. In earlier versions of CSM, the check permission method took only the user name and checked permission for that particular user. However new methods have been introduced which can take in a group name and check permission against the group name.

Alternatively there are two other methods provided that return the list of all the groups which have the same noted privilege on a particular resource.

Below are the method definitions. More details are provided in the Javadocs.

```

public boolean checkPermissionForGroup(String groupName, String objectId, String
attributeName, String privilegeName) throws CSEException;

public boolean checkPermissionForGroup(String groupName, String objectId, String
privilegeName) throws CSEException;

public List getAccessibleGroups(String objectId, String privilegeName) throws
CSEException;

public List getAccessibleGroups(String objectId, String attributeName, String
privilegeName) throws CSEException;

```

NOTE: If you are using Group level security, at the time of provisioning you will need to make sure that the group name provided to the group (via UPT) is same as the Grid Grouper group name.

Appendix A CSM/ACEGI Sample Configuration File

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
'http://www.springframework.org/dtd/spring-beans.dtd'>
<beans>

    <!-- This is the bean that needs to be protected. -->
    <bean id='applicationService'
    ='test.gov.nih.nci.security.acegi.xyzApp.ApplicationServiceImpl' />
    <!--The application integrating CSM Acegi adapter needs to provide actual implementation
    for SecurityHelper. The class name to reflect the impl of SecurityHelper-->
    <bean id='securityHelper'
    ='test.gov.nih.nci.security.acegi.xyzApp.SecurityHelperImpl' />

    <!-- This bean defines a proxy for the protected bean. Notice that -->
    <!-- the id defined above is specified. When an application asks Spring -->
    <!-- for a applicationService it will get this proxy instead. -->
    <bean id='autoProxyCreator'
    ='org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator'>
        <property name='interceptorNames'>
            <list>
                <value>securityInterceptor</value>
            </list>
        </property>
        <property name='beanNames'>
            <list>
                <value>applicationService</value>
            </list>
        </property>
    </bean>

    <!-- This bean specifies which roles are authorized to execute which methods. -->
    <bean id='securityInterceptor'
    ='gov.nih.nci.security.acegi.CSMMethodSecurityInterceptor'>
        <property name='securityHelper' ref='securityHelper' />
        <property name='authenticationManager'
        ='authenticationManager' />
        <property name='accessDecisionManager'
        ='accessDecisionManager' />
        <property name='afterInvocationManager'
        ='afterInvocationManager' />
        <property name='objectDefinitionSource'
        ='csmMethodDefinitionSource' />
    </bean>
```

```
<bean id='csmMethodDefinitionSource'  
='gov.nih.nci.security.acegi.authorization.CSMMethodDefinitionSource'  
  <property name='methodMapCache'  
    ='ehCacheBasedMethodMapCache' />  
</bean>  
<bean id='ehCacheBasedMethodMapCache'  
='gov.nih.nci.security.acegi.authorization.EhCacheBasedMethodMapCache'  
  <property name="cache">  
    <bean  
      ='org.springframework.cache.ehcache.EhCacheFactoryBean'  
      <property name="cacheManager">  
        <bean  
          ='org.springframework.cache.ehcache.EhCacheManagerFactoryBean' />  
        </property>  
        <property name="cacheName" value="userCache" />  
      </bean>  
    </property>  
  </bean>  
</bean>  
  
<!-- This bean specifies which roles are assigned to each user. -->  
<bean id="userDetailsService"  
="gov.nih.nci.security.acegi.authentication.CSMUserDetailsService">  
  <!-- -->  
<!-- Specify the Application Context required by CSM -->  
  <!-- -->  
  <property name="csmApplicationContext">  
    <value>acegitest</value>  
  </property>  
</bean>  
  
<!-- This bean specifies that a user can access the protected methods -->  
<!-- if they have any one of the roles specified in the objectDefinitionSource above. -->  
<bean id='accessDecisionManager'  
='org.acegisecurity.vote.AffirmativeBased'  
  <property name='decisionVoters'  
    <list>  
      <ref bean='roleVoter' />  
    </list>  
  </property>  
</bean>  
  
<!-- The next three beans are boilerplate. They should be the same for nearly all applications.  
-->  
<bean id='authenticationManager'  
='org.acegisecurity.providers.ProviderManager'  
  <property name='providers'  
    <list>  
      <ref bean='authenticationProvider' />  
    </list>  
  </property>
```

```
</bean>

<bean id='authenticationProvider'
      ='gov.nih.nci.security.acegi.authentication.CSMAuthenticationProvider'>
  <property name='userService' ref='userService' />
</bean>

<bean id='roleVoter'
      ='gov.nih.nci.security.acegi.authorization.CSMRoleVoter' />

<bean id='afterInvocationManager'
      ='gov.nih.nci.security.acegi.CSMAfterInvocationProviderManager'>
  <property name='providers'>
    <list>
      <ref bean='afterInvocationProvider' />
    </list>
  </property>
</bean>

<bean id='afterInvocationProvider'
      ='gov.nih.nci.security.acegi.CSMAfterInvocationProvider' />

</beans>
```


Appendix B Migrating to CSM v4.1

In order to take advantage of the features added to CSM for version 4.1, if you are using an earlier version of CSM, you must migrate your existing authorization schema up to the new version.

If you are using CSM 3.2, you must first migrate the schema to v4.0 and then migrate again to 4.1. You cannot migrate an authorization schema for CSM 3.2 directly to CSM 4.1. If you are currently using CSM 4.0, you can migrate directly to CSM 4.1.

The sections in this appendix provide detailed instructions for each migration.

Migrating from CSM v3.2 to CSM 4.0

CSM allows you to choose whether to use a MySQL database or an Oracle Database for your Authorization Schema. The sections that follow provide instructions for migrating each type of database from CSM 3.2 to CSM 4.0.

MySQL Migration – CSM 3.2 to CSM 4.0

The following procedure provides the steps needed to update the MySQL database from an existing CSM 3.2 authorization schema to a CSM 4.0 authorization schema.

1. Obtain the CSM API v4.0 Release from NCICB Download Center located at: <http://ncicb.nci.nih.gov/download>.
2. In the `MigrationScript4.0MySQL.sql` in the CSM API v4.0 release, replace the `<<database_name>>` tag with the name of the database.
3. Go to the directory that contains the executables for MySQL and provide the following command:

```
mysql --user=[user_name] --password=[password] -h  
[hostname] [auth_schema] < MigrationScript4.0MySQL.sql
```

Where:

- o `[user_name]` is the user name used to connect the MySQL database.
 - o `[password]` is the password for the user name entered.
 - o `[hostname]` is the host URL where the MySQL database is hosted. If you are running this command from the same machine where MySQL is hosted, you do not need to provide this parameter.
 - o `[auth_schema]` is the name of the database created with the new authorization schema.
 - o `[MigrationScript4.0MySQL.sql]` is the file containing the data exported from the old schema, which needs to be loaded into the new schema
4. Verify that there are no errors in the SQL Script executed. Also make sure that the database has been appropriately updated.

Oracle Migration – CSM 3.2 to CSM 4.0

The following procedure provides the steps needed to update the Oracle database from an existing CSM 3.2 authorization schema to a CSM 4.0 authorization schema.

1. Obtain the CSM API v4.0 Release from NCICB Download Center located at: <http://ncicb.nci.nih.gov/download>.
2. Log onto the Oracle Server, into the Schema where the CSM Database is present. using SQL Plus or TOAD or any other tool.
3. Copy all the SQL commands from `MigrationScript4.0Oracle.sql` file in the CSM API v4.0 Release, and paste them onto the SQL Editor/Console.
4. Execute all of the copied commands in a batch.
5. Verify that there are no errors in the SQL Script executed. Also make sure that the database has been appropriately updated.

Migrating from CSM v4.0 to CSM v4.1

CSM allows you to choose whether to use a MySQL database or an Oracle Database for your Authorization Schema. The sections that follow provide instructions for migrating each type of database from CSM 4.0 to CSM 4.1.

MySQL Migration – CSM 4.0 to CSM 4.1

The following procedure provides the steps needed to update the MySQL database from an existing CSM 4.0 authorization schema to a CSM 4.1 authorization schema:

1. Obtain the CSM API v4.0 Release from NCICB Download Center located at: <http://ncicb.nci.nih.gov/download>.
2. In the `MigrationScript4.1MySQL.sql` in the CSM API v4.1 release, replace the `<<database_name>>` tag with the name of the database.
3. Go to the directory that contains the executables for MySQL and provide the following command:

```
mysql --user=[user_name] --password=[password] -h  
[hostname] [auth_schema] < MigrationScript4.1MySQL.sql
```

Where:

- o `[user_name]` is the user name used to connect the MySQL database.
- o `[password]` is the password for the user name entered.
- o `[hostname]` is the host URL where the MySQL database is hosted. If you are running this command from the same machine where MySQL is hosted, you do not need to provide this parameter.
- o `[auth_schema]` is the name of the database created with the new authorization schema.
- o `[MigrationScript4.0MySQL.sql]` is the file containing the data exported from the old schema, which needs to be loaded into the new schema

4. Verify that there are no errors in the SQL Script executed. Also make sure that the database has been appropriately updated.

Oracle Migration – CSM 4.0 to CSM 4.1

The following procedure provides the steps needed to update the Oracle database from an existing CSM 4.0 authorization schema to a CSM 4.1 authorization schema.

1. Obtain the CSM API v4.1 Release from NCICB Download Center located at: <http://ncicb.nci.nih.gov/download>.
2. Log onto the Oracle Server, into the Schema where the CSM Database is present. using SQL Plus or TOAD or any other tool.
3. Copy all the SQL commands from `MigrationScript4.1Oracle.sql` file in the CSM API v4.1 Release, and paste them onto the SQL Editor/Console.
4. Execute all of the copied commands in a batch.
5. Verify that there are no errors in the SQL Script executed. Also make sure that the database has been appropriately updated.

Glossary

The following table contains a list of terms used in this document along with their definitions.

Term	Definition
Acegi	Acegi is a security framework that provides a powerful, flexible security solution for enterprise software, with a particular emphasis on applications that use the Spring Framework. Acegi Security provides comprehensive authentication, authorization, instance-based access control, channel security and human user detection capabilities. See http://www.acegisecurity.org/ for more information.
Ant	Apache Ant is a Java-based build tool used to perform various build related tasks. For more information on how Ant is used within the SDK. See http://ant.apache.org/ for more information on Ant itself.
caGrid	The cancer Biomedical Informatics Grid, or caBIG [®] , is a voluntary virtual informatics infrastructure that connects data, research tools, scientists, and organizations to leverage their combined strengths and expertise in an open federated environment with widely accepted standards and shared tools. The underlying service oriented infrastructure that supports caBIG [®] is referred to as caGrid. See http://www.cagrid.org
Ehcache	Ehcache is a simple, fast and thread safe cache for Java that provides memory and disk stores and distributed operation for clusters. CSM uses ehcache in conjunction with Hibernate. See http://sourceforge.net/projects/ehcache for more information.
Hibernate	Hibernate is an object-relational mapping (ORM) solution for the Java language, and provides an easy to use framework for mapping an object-oriented domain model to a traditional relational database. Its purpose is to relieve the developer from a significant amount of relational data persistence-related programming tasks. See http://www.hibernate.org/ for more information.
JAR	JAR file is a file format based on the popular ZIP file format and is used for aggregating many files into one. A JAR file is essentially a zip file that contains an optional META-INF directory.
JAAS	The JAAS 1.0 API consists of a set of Java packages designed for user authentication and authorization. It implements a Java version of the standard Pluggable Authentication Module (PAM) framework and compatibly extends the Java 2 Platform's access control architecture to support user-based authorization.
SAML	Security Assertion Markup Language (SAML) is an XML standard for exchanging authentication and authorization data between security domains, that is, between an identity provider (a producer of assertions) and a service provider (a consumer of assertions). SAML is a product of the OASIS Security Services Technical Committee
Spring	Spring Framework is a leading full-stack Java/JEE application framework. Led and sustained by Interface21, Spring delivers significant benefits for many projects, increasing development productivity and runtime performance while improving test coverage and application quality. See http://www.springframework.org/ for more information.

Term	Definition
WSDD	An acronym for Web Service Deployment Descriptor, which can be used to specify resources that should be exposed as Web Services. See http://ws.apache.org/axis/java/user-guide.html#CustomDeploymentIntroducingWSDD for more information.
WSDL	An acronym for Web Services Definition Language, which is an XML-based language that provides a model for describing Web services. See http://www.w3.org/TR/wsdl.html or http://en.wikipedia.org/wiki/WSDL for more information.

Index

A

- Acegi configuration file, 105
- add security filter, 63
- Admin
 - defined, 5
- Admin Mode
 - group options, 59
 - workflow, 49
- Admin Mode provisioning buttons, 35
- administer protection elements, 52
- administrative rights, 42
- anonymous binds
 - LDAP login module, 18
- Ant, 8
- API
 - authorization JAR placement, 22
 - integrating with CLM, 25
 - integration workflow, 9
 - services, 10
 - with JBoss, 20
- application administration, 40
- application Administrator, 42
- assign application Admin, 42
- assign element to group, 35, 37
- assign PG and role to group, 61
- assign PG and role to user, 51
- assign privileges to a role, 59
- assign protection element to protection group, 54, 56
- assign protection group to parent group, 57
- assign provisioning element, 35
- assign user to group, 50, 61
- assigning elements, 49
- attribute level security
 - defined, 5
 - overall design, 87
 - provisioning, 87
- audit logging
 - CLM & CSM APIs, 25
 - common logging database, 26
 - configuration, 24
 - configure log file, 27
 - create database, 27
 - defined, 5
 - deploying, 27
 - enable for UPT, 72
 - event logging, 25
 - JAR file placement, 24
 - JBoss, 24
 - JDBC Appender, 26
 - object state logging, 25
 - overview, 24

- user information, 26
- view logs, 27
- workflow, 9

- authentication
 - caGrid, 101
 - defined, 5
 - LDAP, 9
- authentication manager, 10
 - configure lock-out, 12
 - import class, 11
 - using, 11
- authentication service
 - installation, 12
 - integration, 10
 - JAR placement, 12
 - LDAP, 12, 16, 17, 18
 - RDBMS, 12, 13, 14, 15
- authorization
 - defined, 5
 - input data, 9
 - workflow, 9
- authorization manager, 10
 - import class, 20
 - products and scripts, 21
- authorization schema, 6
 - create, 22
 - datasource, 23
 - security, 75
 - use existing, 22
- authorization schema objects, 47
- authorization service
 - database, 22
 - event logging, 25
 - installation, 22
 - integration, 19
 - JBoss, 19

B

- browsing logs, 26

C

- caGrid authentication, 101
- caGrid authorization, 103
- check permission operation, 74
- CLM, 6
- Common Logging Module. See CLM
- configure authorization datasource, 23
- configure authorization schema, 22
- configure log file, 27
- configure security datasource, 76
- configure UPT datasource, 70
- create application in UPT, 40

- create authorization database, 22
- create element, 31
- create filter clause, 83
- create group in UPT, 60
- create logging database, 27
- create privilege in UPT, 46
- create protection element, 84
- create protection element in UPT, 52
- create protection group in UPT, 55
- create provisioning element, 31
- create role in UPT, 58
- create security database, 75
- create UPT database, 70
- create user in UPT, 42, 50
- credential Provider, 7
- CRF 21, 5
- CSM
 - architecture, 6

D

- database
 - configuring for Acegi, 94
 - create authorization schema, 22
 - create for logging, 27
 - create security authorization schema, 75
 - create UPT schema, 70
- deassign element from group, 35, 37
- deassign provisioning element, 35
- delete application from UPT, 41
- delete element, 34
- delete group, 60
- delete privilege, 47
- delete protection element, 53
- delete protection group, 56
- delete provisioning element, 34
- delete role, 58
- delete user from UPT, 44
- deploy authorization database, 22
- deploy authorization datasource, 23
- deploy authorization service, 22
- deploy security datasource, 76
- deploy securityws.war file, 78
- deploy UPT datasource, 70
- deploying audit logging, 27
- deployment modes, 66

E

- edit/update application details, 41
- edit/update group details, 60
- edit/update privilege details, 46
- edit/update protection element details, 53
- edit/update protection group details, 55
- edit/update role details, 58
- edit/update user details, 43
- encryption in RDBMS, 15

- event logging, 25

F

- filter clauses, 62, 63, 65, 83

G

- Glossary, 113
- Grid Grouper, 103
- group level security for caGrid, 103
- group PEs and privileges, 62
- group PG and role, 61
- group reporting, 62
- groups
 - administration, 59, 61
 - create, 60
 - delete, 60
 - overview, 59
 - view/update, 60

H

- Hibernate configuration file, 21, 63, 83, 93, 96

I

- import authentication manager, 11
- import authorization manager, 20
- install authorization service, 22
- installation modes, 66
- instance level security
 - activity flow, 82
 - add filter clause, 63
 - administration, 62
 - defined, 5
 - file upload, 62, 63
 - modify filter clause, 65
 - overall design, 81
 - provisioning, 81
- instance level security in UPT
 - overview, 62
- integrate application, 10
- integrate authentication service, 10
- integrate authorization service, 19
- integrating CSM with CLM, 25

J

- JAAS, 6, 8
 - LDAP login module, 16
 - login module, 98
 - RDBMS login module, 13
- jar file
 - instance level security in UPT, 63
- JBoss
 - audit logging, 24
 - authorization service, 19
 - configure log file, 27

- deploy securityws.war file, 78
- enable audit logging, 72
- event logging, 25
- JAAS login, 71, 77
- LDAP login module, 17
- object state logging, 26
- RDBMS login module, 14
- UPT common server deployment, 68
- UPT deployment, 72
- UPT deployment modes, 67

JBOSS, 8

JDBC Appender, 26

JDK, 8

L

LDAP, 8, 9

- anonymous binds, 18
- authentication, 12, 16, 17
- configuring for caGrid integration, 102
- JBoss login, 71, 77
- login module, 16, 17, 18, 102

lenient behavior, 87

lock-out, 12

log locator, 27

log locator, 26

logging user information, 26

login module, 8

- JAAS, 98

login operation, 73

M

migrating to CSM 4.1, 109, 110

minimum requirements, 8

modify element, 34

MySQL, 8

MySQL migration, 109, 110

O

object state logging, 25

Oracle, 8

oracle migration, 110

Oracle migration, 111

overview

- Admin, 5
- Admin Mode, 47
- assigning/associating elements, 35
- attribute level security, 5
- audit logging, 5
- authentication, 5
- authorization, 5
- groups, 59
- instance level security, 5
- instance level security in UPT, 62
- integration workflow, 9
- privileges, 45, 54
- protection group, 55
- roles, 57

- security, 79
- security concepts, 7
- Super Admin, 5, 39
- UPT, 29
- user provisioning, 5

P

privilege administration, 46, 47

privileges

- create, 46
- deleting, 47
- grouping, 59
- overview, 45, 54
- view/update, 46
- viewing, 54

protection element

- administration, 52, 54
- create, 52, 84
- delete, 53
- grouping, 54, 56
- view/update, 53

protection group, 7

- administration, 55, 56, 57
- create, 55
- delete, 56
- overview, 55
- parent grouping, 57
- parents, 55
- view/update, 55

provisioning

- Admin Mode, 47
- attribute level security, 87
- element relationships, 47
- find element to assign, 38
- Super Admin, 39
- Super Admin tasks, 40

provisioning tool, 29

R

RDBMS, 8

- configuring for caGrid integration, 101
- encryption, 15
- login module, 13, 14, 98, 101

RDBMS authentication, 12, 13, 14, 15

register application in UPT, 40

related documents, 2

release schedule, 4

roles

- administration, 57, 59
- create, 58
- delete, 58
- overview, 57
- view/update, 58

S

sample error messages, 32

scripts for authorization, 21

- search for element, 32
- search for provisioning element, 32
- security
 - add filter, 63
 - attribute level, 5, 79, 86
 - authorization strategy, 9
 - check permission operation, 74
 - concepts defined, 7
 - create filter clause, 83
 - datasource, 76
 - deploy securityws.war file, 78
 - filter clause, 65
 - instance level, 5, 62, 79
 - integrating Acegi, 93
 - JAAS login parameters, 77
 - login operation, 73
 - modify filter, 65
 - overview, 79
 - protection element, 84
 - schema administrator, 9
 - web services, 73
 - workflow, 75
 - WSDL, 73
- select provisioning element, 32
- software for authorization, 21
- software requirements, 8
- standard privileges
 - overview, 45, 54
 - view/update, 46
- strict behavior, 87
- submit support issue, 4
- Super Admin
 - defined, 5
 - options, 39
 - overview, 39
 - tasks, 40
 - workflow, 39
- support
 - submitting issue, 4
- supported authentication, 12
- system requirements, 8

T

- test integration, 10
- Tomcat, 8

U

- unlock user, 44
- update PG and role for group, 62
- update PG and role for user, 51
- update provisioning element, 34
- UPT
 - add security filter, 63
 - Admin Mode, 47
 - assign element, 35, 37
 - basic functions, 30

- create element, 31
- create schema and database, 70
- deassign element, 35, 37
- delete element, 34
- deployment, 66, 69, 72
- edit/update security filter, 65
- event logging, 25
- find element to assign, 38
- installation, 66, 69
- instance level security, 62
- JAAS login parameters, 71
- JBoss common server, 68
- JBoss deployment, 67
- local install/local schema, 69
- login, 29
- modify element, 34
- multi-install/multi schema, 68
- overview, 29
- release contents, 66
- sample error messages, 32
- schema datasource, 70
- search for element, 32
- single install/single schema, 67
- standard privileges, 45, 54
- Super Admin, 39
- view element details, 32
- view security filter, 65
- workflow, 29

- user administration, 42, 43, 44, 50, 51
- user creation, 42, 50
- user grouping, 50, 61
- user PEs and privileges, 52
- user PGs and roles, 51
- user provisioning, 5, 99
- user reporting, 52
- using authentication manager class, 11

V

- view application details, 41
- view element details, 32
- view group details, 60
- view logs, 27
- view PE and privileges for group, 62
- view PE and privileges for user, 52
- view privilege details, 46, 54
- view protection element details, 53
- view protection group details, 55
- view role details, 58
- view user details, 43

W

- workflow
 - Acegi, 92
 - Admin Mode, 49
 - security web service, 75
 - Super Admin, 39