NCICB

Ekagra

High Impact – High Value – Business Results

# Common Security Module

Reference Implementation Guide

Version No: 1.1

Last Modified: 7/29/05

| | | |
|---|---|---|
| Author | : | Vinay Kumar, Eric Copen, Brian Husted, Kunal Modi |
| Team | : | Common Security Module (CSM) |
| | | Purchase Order# 34552 |
| Client | : | National Cancer Institute - Center for Bioinformatics, |
| | | National Institutes of Health, |
| | | US Department of Health and Human Services |

# Document History

## Document Location

The most current version of this document is located in CVS under security/docs.

## Revision History

| Version Number | Revision Date | Author | Summary of Changes |
|---|---|---|---|
| 0.1 | 2/24/05 | Vinay Kumar, Eric Copen, Kalpesh Patel | Initial Table of Contents |
| 0.2 | 4/8/05 | Eric Copen, Brian Husted, Kunal Modi, Vinay Kumar | Initial Draft |
| 0.3 | 4/14/05 | Eric Copen, Brian Husted, Kunal Modi, Vinay Kumar | Complete Draft |
| 1.0 | 4/15/05 | Eric Copen, Brian Husted, Kunal Modi, Vinay Kumar | Initial Release |
| 1.1 | 7/29/05 | Eric Copen | Minor changes related to the 3.0.1 release |
| | | | |

## Review

| Name | Team/Role | Version | Date Reviewed | Reviewer Comments |
|---|---|---|---|---|
| Jill Hadfield | Technical Writer | 1.0 | 4/15/05 | Comments regarding mostly grammar and some structure. Most incorporated in version 1.0. |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

## Related Documents

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

| Document Name |
|---|
| CSM Guide for Application Developers |
| UPT User Guide |
| |
| |

These and other documents can be found on the CSM website: http://ncicb.nci.nih.gov/core/CSM

# Table of Contents

# Reference Implementation Guide

## 1. Introduction

This document provides the information application developers need to install, configure, and use the CSM Reference Implementation (RI). The RI demonstrates how to integrate CSM with an application to implement security requirements.

### 1.1 About CSM

CSM is an NCICB module chartered to provide a comprehensive solution to common security objectives. It is flexible enough so that development teams can use it for all or part of their security needs. CSM provides solutions for Authentication and Authorization, and also provides a web-based User Provisioning Tool. To download CSM and to learn more about the software go to http://ncicb.nci.nih.gov/core/CSM.

### 1.2 How to Use this Guide

Begin by installing the RI application and configuring the environment. Read this document and explore the RI application to understand the use cases and security requirements. Then analyze the code snippets provided to understand how your application might integrate with CSM. Download and install CSM and begin integration. See the *CSM Guide for Application Developers* for more information regarding installing and using CSM.

## 2. Executive Summary

The CSM team provides a Reference Implementation application ABC Toys Employee Records System. By installing this application and the CSM application, developers can come to understand how to integrate with CSM.

To install the RI, first download and install MySQL, the Java 2 Platform, and the JBoss application server. Download the CSM_RI.zip file. Create a database, configure the DataSource, and deploy the application WAR file. See the *CSM Guide for Application Developers* for instructions regarding installing and using CSM.

The RI Employee Records System holds vital employee information including name, address, salary, social security, business unit, and management status. ABC Toys is composed of three business units – HR, Business Development, and Information Technology, each with its own manager. Employees have certain access rights based on their individual needs and privileges. For example, only HR managers can create employees. Only the Business Development and IT managers can assign projects. Employees can view their own records only. Security examples like these highlight the use cases and security requirements for the RI application.

The RI provides easy navigation and a flexible workflow. A typical workflow would include a HR Manager first creating the record, an IT manager assigning a project, and an employee verifying his record information. This guide provides detailed instructions for using the application.

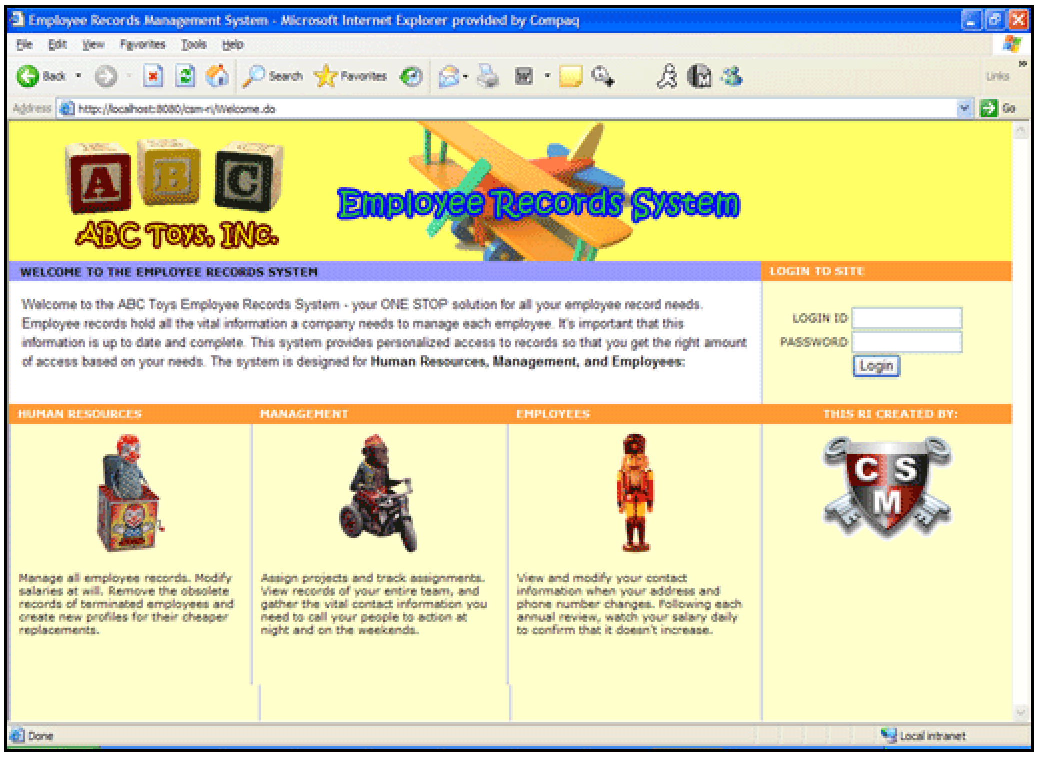
The RI utilizes the CSM database schema to protect actions and data from unauthorized access. This guide explains the data model implementation that enables fine grained authorization. It details how privileges, roles, etc. are associated with elements within the RI, and how these relationships enable security through CSM.

This guide also presents code snippets from the RI that show 1) how security is implemented 2) how the RI is integrated with CSM. It details method and class names, and method functions. For example the checkPermission method of the Authorization Manager in the SearchEmployeeAction class verifies whether the user can have access to a particular employee record.

Throughout creating the RI and CSM services, the CSM team has compiled a list of best practices and things to avoid that are listed in Sections 9 and 10. For example, suggestions include using DataSource for connection pooling and things to avoid include repeatedly instantiating methods.

## 3. The RI Application

The CSM team developed a simple web application that can be used to show how CSM integrates with an application. This Reference Implementation (RI), an Employee Records System for ABC Toys Incorporated, implements security on an individual basis. Each employee can gain access only to the information to which he is permitted. Installing this RI and reviewing the code and code explanations in this guide will help developers understand the specifics for CSM integration.


Employee Records Management System - Microsoft Internet Explorer provided by Compaq
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/csm-ri/Welcome.do
Go
ABC Toys, INc.
Employee Records System
WELCOME TO THE EMPLOYEE RECORDS SYSTEM
LOGIN TO SITE
LOGIN ID
PASSWORD
Login
HUMAN RESOURCES
MANAGEMENT
EMPLOYEES
THIS RI CREATED BY:
CSM
Done
Local intranet

Welcome to the ABC Toys Employee Records System - your ONE STOP solution for all your employee record needs. Employee records hold all the vital information a company needs to manage each employee. It's important that this information is up to date and complete. This system provides personalized access to records so that you get the right amount of access based on your needs. The system is designed for **Human Resources, Management, and Employees:**

Manage all employee records. Modify salaries at will. Remove the obsolete records of terminated employees and create new profiles for their cheaper replacements.

Assign projects and track assignments. View records of your entire team, and gather the vital contact information you need to call your people to action at night and on the weekends.

View and modify your contact information when your address and phone number changes. Following each annual review, watch your salary daily to confirm that it doesn't increase.

Figure 3.1 *The ABC Toys application shows how CSM can integrate with an application*

## 4. Installing the Application

This section describes the processes for installing and deploying the RI application. The steps are summarized as follows:

1. Downloading the RI
2. Downloading and installing the MySQL Database
3. Downloading and installing the Java 2 Platform
4. Downloading and installing the JBoss Application Server
5. Deploying the RI Application. This includes creating a database, configuring the DataSource, and deploying the application WAR file.

See the *CSM Guide for Application Developers* for instructions regarding installing and using CSM.

### 4.1 System Requirements

The instructions assume you will be deploying and executing the Reference Implementation application on a Windows platform. The instructions given in this document are also applicable, to a great extent, to a UNIX/Linux platform.

The core software requirements:

- MySQL Database (version 4.x recommended)

Installation

- J2SE 1.4.2
  Installation
- JBoss Application Server (version 4.0 recommended)
  Installation

## 4.2 Downloading the Reference Implementation

1. Download and save the CSM_RI_rel_3.0.zip file containing the Reference Implementation application files.
2. Extract the zip file into a folder of your own choosing, such as c:\CSM_RI.

***NOTE*** Instructions for deploying the Reference Implementation are provided in Section 7, Deploying the Reference Implementation Application.

## 4.3 Installing the MySQL Database

1. Download and save MySQL database http://www.mysql.com/downloads/mysql-4.0.html
2. (Optional) Download and save MySQL Control Center zip file for Windows from http://www.mysql.com/downloads/mysqlcc.html
3. (Optional) Install MySQL and the Control Center by following the instruction from http://www.mysql.com/doc/en/index.html.
4. See section 4.6.1 for the steps and the SQL script that will create the proper database and associated tables.

## 4.4 Installing the Java 2 Platform, Standard Edition

1. Go to http://java.sun.com.
2. Click on the "J2SE (Core/Desktop)".
3. On the next page, click on the "J2SE 1.4.2" link.
4. On the download page, click on the "Java 2 Platform, Standard Edition version 1.4.2" link for Windows.
5. On the next page, click on SDK "Download" to the right of the "32-bit/64-bit for Windows/Linux/Solaris SPARC 32-bit for Solaris x86" field.
6. Go to the bottom of the next page and click on the "ACCEPT" radio button.
7. Download and save the .exe file. This file will take a bit longer to download due to its size.
8. Create a *java* directory on your C:\ drive (i.e., c:\java). You can also choose a location where you normally install your Java 2 Platform software.
9. Find and execute the file that was downloaded. Install j2sdk into the directory you have created/selected in step 9. The path to your installation directory is referred to as: "[JAVA_HOME]".
10. Create an Environment variable for Java from **Control Panel→System→Advanced** with "Variable Name" of JAVA_HOME and "Variable Value" of "c:\java\j2sdk1.4.2_05" (i.e., the path to the folder where you installed J2SE 1.4.2).

## 4.5 Installing JBoss Application Server

1. Go to JBoss.org and download version 4.0 of the application server.
2. Install JBoss according to the instructions. For documentation see this site.

## 4.6 Deploying the Reference Implementation Application

The Reference Implementation application is being provided as Web Archive (WAR) files. The release of the Reference Implementation consists of the following components:

1. security_ri.war
2. mysql-ds.xml
3. DatabaseLoad.sql
4. ApplicationSecurityConfig.xml
5. csm_ri.hibernate.cfg.xml
6. csmupt.hibernate.cfg.xml

Installation of the reference implementation is straight forward and involves deploying the following steps

### *4.6.1 Create the Database*

1. Log into MySQL database using an account id which has permission to create new databases.
2. Run this DatabaseLoad.sql on the mySQL prompt. This should create a database and load it with the initial sample data.
3. Note that for the purpose of simplicity of installation both the RI and the Authorization Schema have been co-hosted in the same database named csm_ri

### *4.6.2 Configure the Data Source*

1. Modify the mysql-ds.xml file which contains information for creating a datasource. One entry is required for each database connection. Edit this file to replace:
    a. The <<database_user_id>> with the user id and <<database_user_password>> with the password of the user account, which will be used to access the Reference Implementation Schema created in Step 1 above.
    b. The <<database_url>> with the URL needed to access the Reference Implementation residing on the MySQL database server.
2. Copy this file in the deploy folder of the JBoss.
   Here is an excerpt from the `mysql-ds.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>

<datasources>
  <local-tx-datasource>
    <jndi-name>csm_ri</jndi-name>
    <connection-url><<database_url>></connection-url>
    <driver-class>org.gjt.mm.mysql.Driver</driver-class>
    <user-name><<database_user_id>></user-name>
    <password><<database_user_password>></password>
   </local-tx-datasource>
</datasources>
```

### *4.6.3 Create Directory*

1. Create a directory on the server where all the configuration files pertaining to the RI as well as UPT will be kept. This directory can have any name and can reside anywhere on the server. However, it should be accessible to the JBoss id running the tool.

### 4.6.4 *Copy the Configuration Files*

1. Copy the ApplicationSecurityConfig.xml file to the directory created in section 4.6.3. Make sure that the JBoss id has access to it.
2. Edit the ApplicationSecurityConfig.xml file to replace the <<config_directory_base>> with the fully qualified path of the directory created in section 4.6.3.
3. Copy the csm_ri.hibernate.cfg.xml file to the directory created in the section 4.6.3. Make sure that the JBoss id has access to it
4. Copy the csmupt.hibernate.cfg.xml file to the directory created in the section 4.6.3. Make sure that the JBoss id has access to it

### 4.6.5 *Make an Addition to the JBoss Startup Properties File*

1. Edit the JBoss properties-service.xml to provide a startup parameter to the JBoss server. This file is located at the following path: {jboss-home}/server/ default/deploy/properties-service.xml where {jboss-home} is the base directory where JBoss is installed on the server. Add the following entry to the existing properties:

```
<attribute name="Properties"> <!-- could already exist -->
:
gov.nih.nci.security.configFile=/foo/bar/ApplicationSecurityConfig.xml
:
</attribute> <!-- could already exist -->
```

   a. The gov.nih.nci.security.configFile is the name of the property which points to the fully qualified path foo/bar/ApplicationSecurityConfig.xml where the ApplicationSecurityConfig.xml has been created in Section 4.6.4. The name of the property must be the gov.nih.nci.security.configFile and cannot be modified, as it is a system-wide property.
2. Save this file in a deploy folder (for example, {jboss-home}/server/default/deploy/).

### 4.6.6 *Configure Login Modules for Authentication*

1. CSM uses JAAS Login Modules for the purpose of authentication. In JBoss this is achieved by using the login`-config.xml` file which is located at `{jboss-home}\server\default\conf\login-config.xml.`
2. We will be using RDBMS based authentication for both UPT as well as RI. The sample entries for these are given below. The following configurations need to be updated before entries can be made in the login-config.xml file of JBoss:
   a. Replace the <<database_user_id>> with the user id and <<database_user_password>> with the password of the user account, which will be used to access the Reference Implementation as well as the UPT Schema created in Step 1 above.
   b. Replace the <<database_url>> with the URL needed to access the Reference Implementation as well as UPT residing on the MySQL database server.

```
<application-policy name = "csmupt">
  <authentication>
    <login-module code = "gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule"
flag = "required" >
      <module-option name="driver">org.gjt.mm.mysql.Driver</module-option>
      <module-option name="url"><<database_url>></module-option>
      <module-option name="user"><<database_user_id>></module-option>
      <module-option name="passwd"><<database_user_password>></module-option>
      <module-option name="query">SELECT * FROM USER WHERE LOGIN_NAME=? and
PASSWORD=?</module-option>
    </login-module>
  </authentication>
</application-policy>

<application-policy name = "csm_ri">
  <authentication>
    <login-module code = "gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule"
flag = "required" >
      <module-option name="driver">org.gjt.mm.mysql.Driver</module-option>
      <module-option name="url"><<database_url>></module-option>
      <module-option name="user"><<database_user_id>></module-option>
      <module-option name="passwd"><<database_user_password>></module-option>
      <module-option name="query">SELECT * FROM EMPLOYEE WHERE LOGIN_NAME=? and
PASSWORD=?</module-option>
    </login-module>
  </authentication>
</application-policy>
```

#### *4.6.7 Deploy the Application War File*

1. Copy the RI security_ri.war in the deployment directory of JBoss which can be found at {jboss-home}/server/default/deploy/ where {jboss-home} is the base directory where JBoss is installed on the server.

#### *4.6.8 Deploy the UPT War File*

1. Obtained the upt.war from the CSM's download site and copy the upt.war file in the deployment directory of JBoss which can be found at {jboss-home}/server/default/deploy/ where {jboss-home} is the base directory where JBoss is installed on the server.
2. For detailed installation guide and how to use UPT refer to the *CSM Guide For Application Developers*.
3. Following are the login credentials for UPT

| Functionality | User Id | Password | Application Context Name |
|---|---|---|---|
| Super Admin | uptadmin | uptadmin | csmupt |
| RI Admin | riadmin | riadmin | csm_ri |

#### *4.6.9 Verify Installation*

1. Verify that the UPT is installed correctly by accessing it with the following URL.

   a. http://localhost:8080/upt
2. Verify that the reference implementation is installed correctly by accessing it with the following URL.
   b. http://localhost:8080/security_ri

## 5. Business Scenario (Introduction to the Application)

The Employee Records System allows employees to access and modify employee records that include demographic and personal information. The application limits access based on the user's relationship to the record – employing permission checks on the element or attribute level. Because the application requires fine-grain security it effectively exhibits the capabilities of the Common Security Module.

Functional Requirements

Record
1. Create
2. View
3. Modify
4. Search

Project
1. Create
2. Associate project with employee
3. Remove project/employee association

Employee Record

The employee record consists of the following:
1. User name
2. Password
3. Last name
4. First name
5. Middle initial
6. Street address
7. City
8. State code
9. Zip
10. Phone number
11. Email address
12. Social Security Number
13. Salary
14. Business Unit
15. Management indicator (yes or no)
16. Assigned Projects
17. Available Projects

## 5.1 Use Case Model

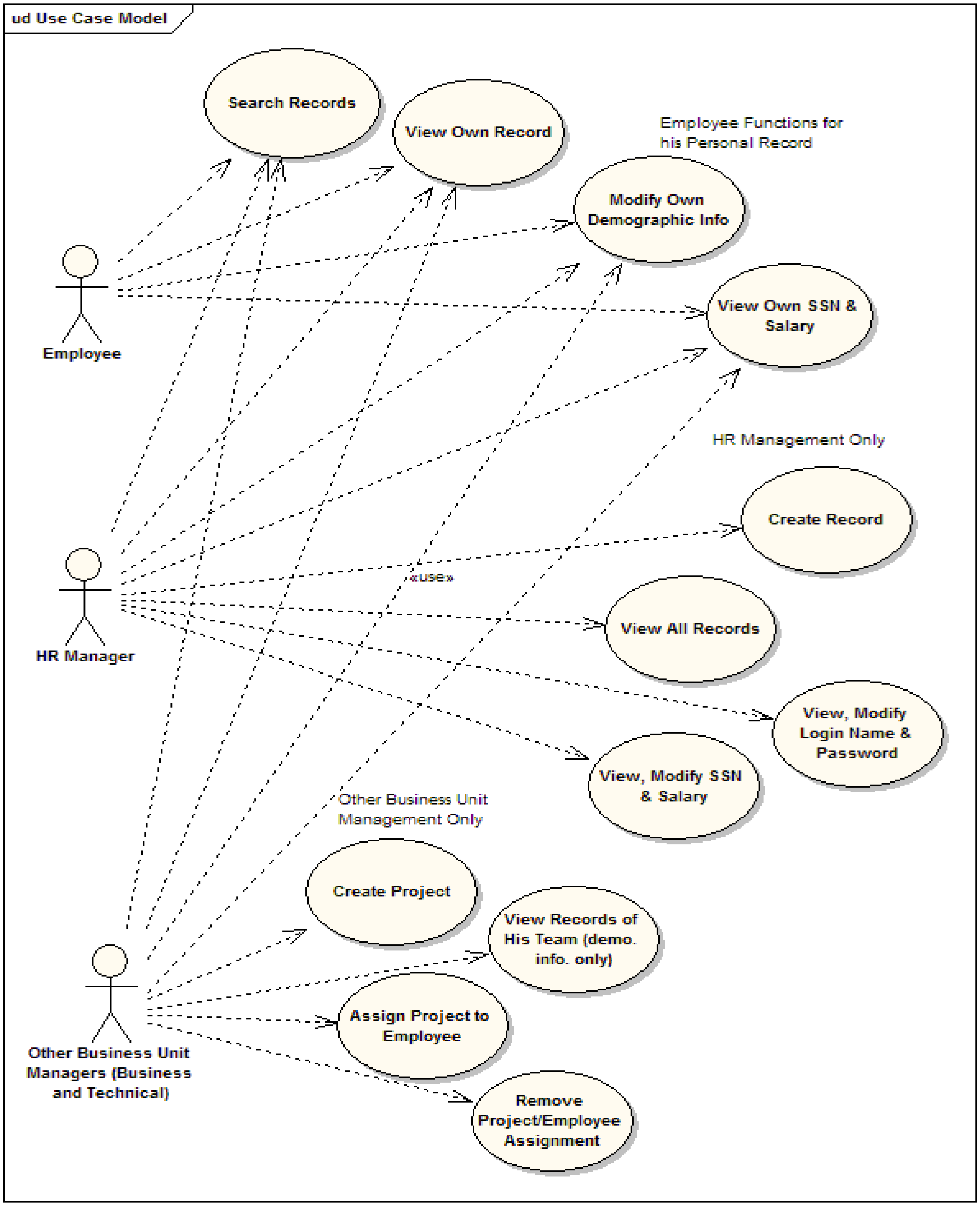

ud Use Case Model
Search Records
View Own Record
Employee Functions for his Personal Record
Modify Own Demographic Info
View Own SSN & Salary
Employee
HR Management Only
Create Record
«use»
View All Records
HR Manager
View, Modify Login Name & Password
View, Modify SSN & Salary
Other Business Unit Management Only
Create Project
View Records of His Team (demo. info. only)
Assign Project to Employee
Remove Project/Employee Assignment
Other Business Unit Managers (Business and Technical)

Figure 5.1 *Use case model for the ABC Toys Reference Implementation*

## 5.2 Description of Use Cases

Figure 5.1 displays a use case model for the ABC Toys Reference Implementation. ABC Toys is divided into three primary business units: Information Technology, Human Resources, and Business Development. Employees, whether they are managers or regular employees, can only access the information they need and can only perform the operations their function necessitates. For example, only an HR manager can create an employee record. Only the Business Unit managers for IT or Business Development can create and assign projects to the employees that report to him or her. The below use cases are grouped by function:

Regular Employee

1. Search records
2. View own record including SSN & Salary
3. Modify own demographic information

HR Manager

An HR Manager possesses the same rights as a regular employee, and can also:

1. Create a new record
2. View the records of all employees
3. View and modify a user's login name and password
4. Modify a user's Social Security Number and salary

Technical / Business Development Unit Managers

The ABC Toys has managers for Information Technology and Business Development. Along with the same rights as a regular employee, these managers can also perform the following team management functions:

1. Create a new project with a description
2. View employee records that are assigned to the managers
3. Assign projects to employees
4. Remove projects from employees (unassign)

## 5.3 Security Requirements

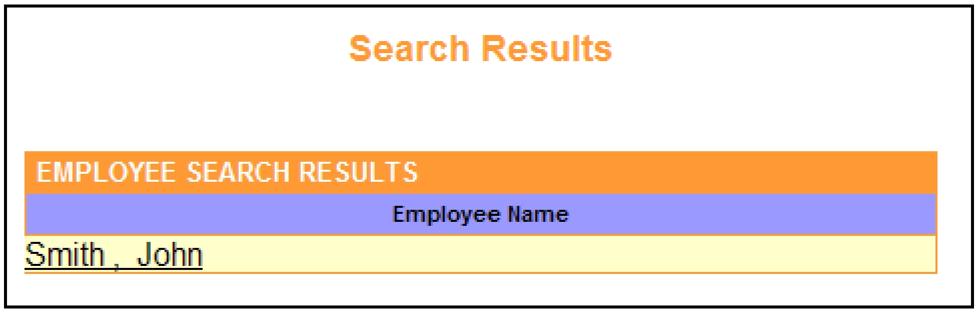
Within any company and ABC Toys in particular, demographic, personal, and compensation related information needs to be secure. Imagine the consequences if an employee could view or even change salaries. This could potentially cause chaos and have both immediate and long-term consequences including lost pay, bitterness, turnover, etc. Identity theft is also a concern with the Social Security information the Employee Records System holds. Keeping these concerns in mind, ABC has identified the following major security requirements:

Regular Employee

A regular employee **can NOT:**

1. Modify their SSN and Salary
2. Modify their project assignments
3. View or modify other employee records (see Figure 5.2)
4. Create an Employee
5. Create a Project

In the below example, employee John Smith logged in and performed a wildcard search for employee records. In accordance with ABC Toys security requirements, he can only select and view his own record.


Search Results

EMPLOYEE SEARCH RESULTS

| Employee Name |
|---|
| Smith , John |

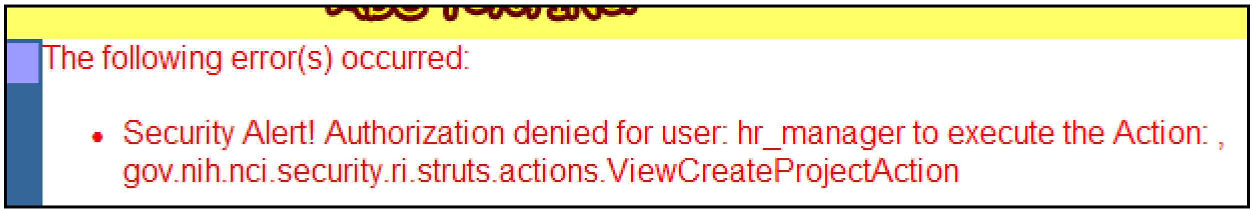
Figure 5.2 *A Regular Employee can retrieve only his own record*

HR Manager

HR Managers can view and update the information in every record except project assignments.
HR Managers **can NOT**:

1. Create a project (see Figure 5.3)
2. Modify project assignments

In the below example, the HR Manager clicks on the **Create a Project** menu option. The system returns the following message indicating the HR Manager does not have security rights to perform this operation.


The following error(s) occurred:

- Security Alert! Authorization denied for user: hr_manager to execute the Action: , gov.nih.nci.security.ri.struts.actions.ViewCreateProjectAction

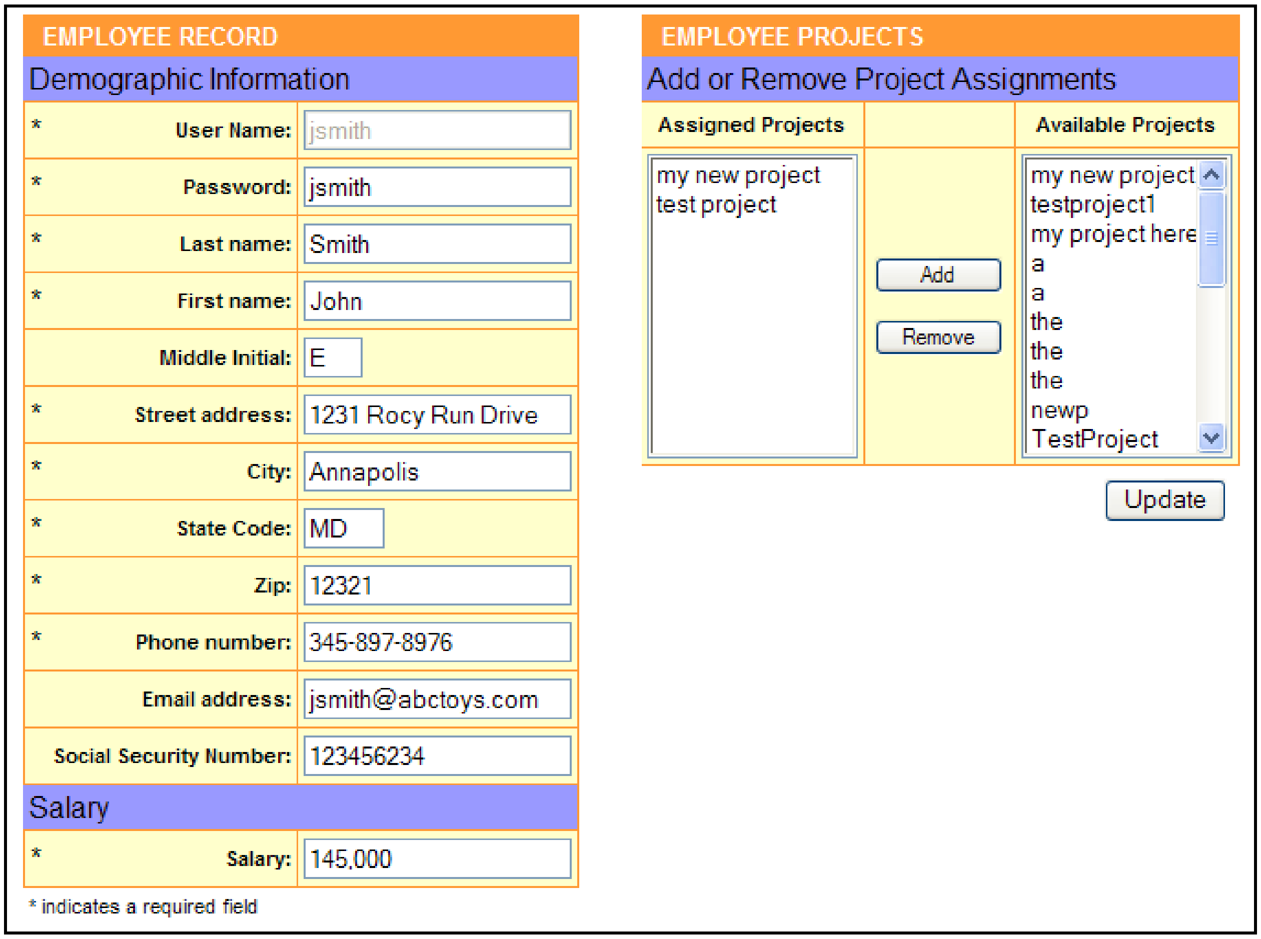
Figure 5.3 *A HR Manager can not create a project*

Technical / Business Development Unit Managers

Managers **can NOT**:

1. Create a record
2. View or modify records of those employees not on his team
3. Modify team record information other than project assignments (see below Figures)

In the following three step illustration, (Figures 4.4.1, 4.4.2, 4.4.3), the Business Development Unit Manager can not edit employee John Smith's secure information, specifically address, SSN, or Salary:


EMPLOYEE RECORD

Demographic Information

| | | |
|---|---|---|
| * | User Name: | jsmith |
| * | Password: | jsmith |
| * | Last name: | Smith |
| * | First name: | John |
| | Middle Initial: | E |
| * | Street address: | 1231 Rocy Run Drive |
| * | City: | Annapolis |
| * | State Code: | MD |
| * | Zip: | 12321 |
| * | Phone number: | 345-897-8976 |
| | Email address: | jsmith@abctoys.com |
| | Social Security Number: | 123456234 |

Salary

| | | |
|---|---|---|
| * | Salary: | 145,000 |

* indicates a required field

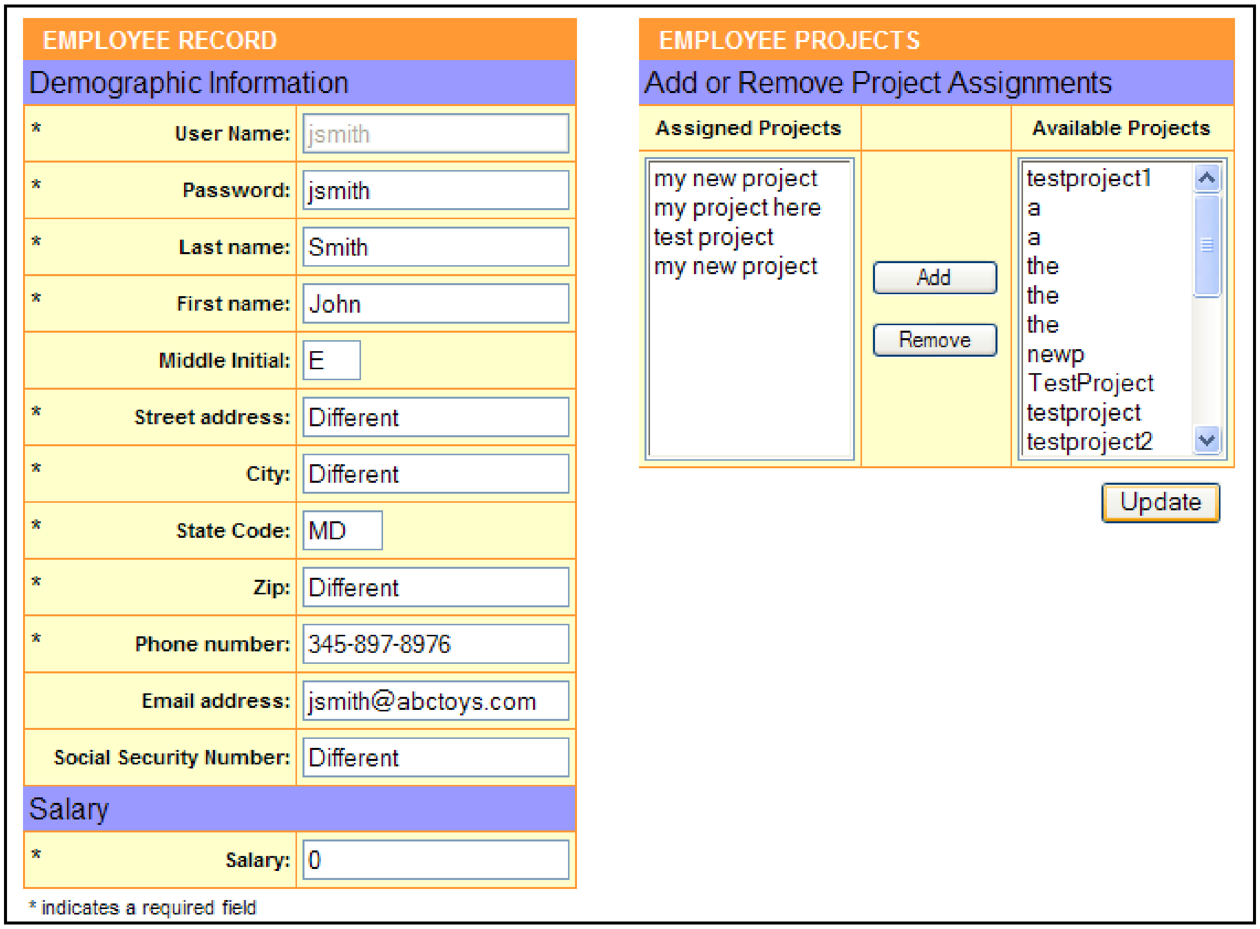
EMPLOYEE PROJECTS

Add or Remove Project Assignments

Assigned Projects:
my new project
test project

Available Projects:
my new project
testproject1
my project here
a
a
the
the
the
newp
TestProject

Figure 5.4.1 *The Business Development Manager opens the record of John Smith*



EMPLOYEE RECORD

Demographic Information

| | | |
|---|---|---|
| * | User Name: | jsmith |
| * | Password: | jsmith |
| * | Last name: | Smith |
| * | First name: | John |
| | Middle Initial: | E |
| * | Street address: | Different |
| * | City: | Different |
| * | State Code: | MD |
| * | Zip: | Different |
| * | Phone number: | 345-897-8976 |
| | Email address: | jsmith@abctoys.com |
| | Social Security Number: | Different |

Salary

| | | |
|---|---|---|
| * | Salary: | 0 |

* indicates a required field

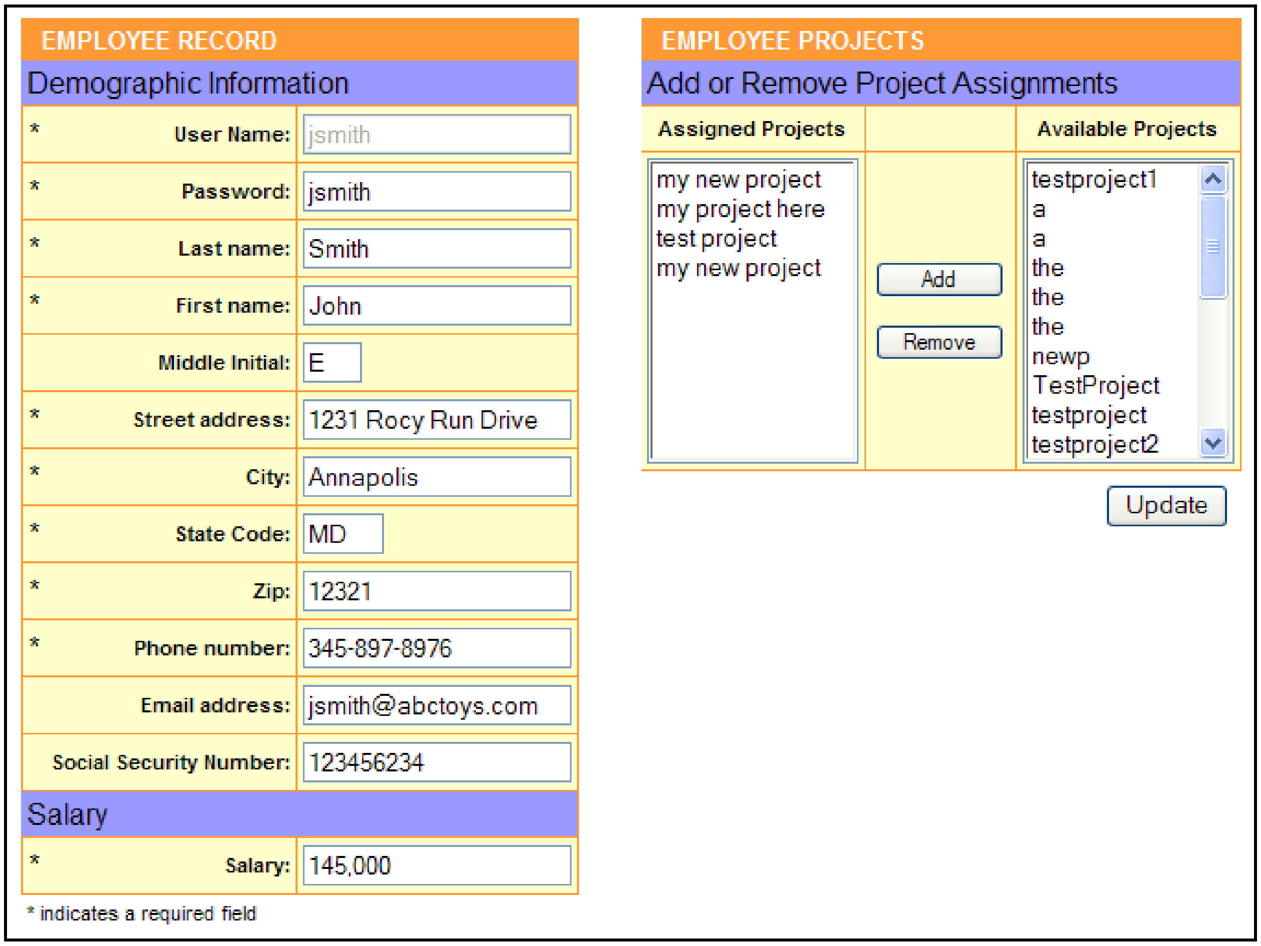
EMPLOYEE PROJECTS

Add or Remove Project Assignments

| Assigned Projects | | Available Projects |
|---|---|---|
| my new project<br>my project here<br>test project<br>my new project | Add<br>Remove | testproject1<br>a<br>a<br>the<br>the<br>the<br>newp<br>TestProject<br>testproject<br>testproject2 |

Update

Figure 5.4.2 *The Business Development Manager attempts to edit the address, SSN, salary, and project assignments of John Smith*



EMPLOYEE RECORD

Demographic Information

| | | |
|---|---|---|
| * | User Name: | jsmith |
| * | Password: | jsmith |
| * | Last name: | Smith |
| * | First name: | John |
| | Middle Initial: | E |
| * | Street address: | 1231 Rocy Run Drive |
| * | City: | Annapolis |
| * | State Code: | MD |
| * | Zip: | 12321 |
| * | Phone number: | 345-897-8976 |
| | Email address: | jsmith@abctoys.com |
| | Social Security Number: | 123456234 |

Salary

| | | |
|---|---|---|
| * | Salary: | 145,000 |

* indicates a required field

EMPLOYEE PROJECTS

Add or Remove Project Assignments

| Assigned Projects | | Available Projects |
|---|---|---|
| my new project<br>my project here<br>test project<br>my new project | Add<br>Remove | testproject1<br>a<br>a<br>the<br>the<br>the<br>newp<br>TestProject<br>testproject<br>testproject2 |

Figure 5.4.3 After t*he Business Development Manager selects* ***Update****, only the project assignments change. The other protected information remains unchanged..*

## 6. Using the RI application

### 6.1 Workflow

As demonstrated in the use cases, the Employee Records System can be used by different types of employees for different purposes. Therefore the workflow is very flexible depending on the person accessing the system.

The most common use of the Employee Records System is when an employee is first added to the system. The workflow involves the creation and modification of an employee record. The below workflow details this process that involves the HR manager, a new employee, and his manager:

New Employee Workflow

1. The HR Manager logs in and creates a record for the new ABC Toys employee
   - Enters login name and password
   - Enters demographic information
   - Enters SSN and salary
   - Designates which business unit to which the employee belongs
2. The employee's business unit manager logs in
   - Create a new project (if necessary)
   - Searches for and selects new employee record
   - Assign project(s) to the employee

3. The employee logs in
    Searches for and selects his employee record
    Views demographic information to confirm it is correct
    Modifies information if required (and permissible)
    Views assigned projects in order to get started

Once these three steps are completed, steps 2 and 3 can each be repeated in any particular order. Parts a-c of step 1 can also be repeated at any given time.

Sections 9.2 through 9.6 below explain how to perform each function of the Employee Records System RI.

## 6.2 Login

The Login page includes the banner and logos, summary text, and most importantly the Login section itself: **Login ID**, and **Password**. Simply enter the appropriate login ID and password, and select the **Login** button.

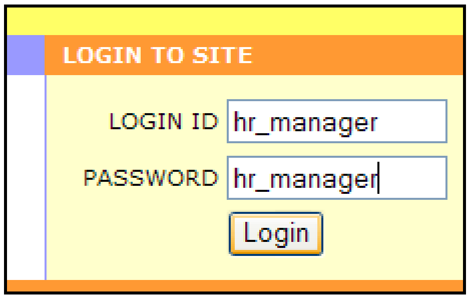

LOGIN TO SITE
LOGIN ID: hr_manager
PASSWORD: hr_manager

Figure 6.1 *Login as HR Manager*

When using the RI to explore how the CSM works, we suggest logging in multiple times as different users. The different users possess a variety of access rights. CSM has previously created the following users to help demonstrate the differences:

| ***Login*** | ***Password*** |
|---|---|
| hr_manager | hr_manager |
| technical_manager | technical_manager |
| business_manager | business_manager |
| employee_1 | employee_1 |

Table 6.1 *Previously created Logins and Passwords*

## 6.3 Create Employee

An HR Manager can create a new employee record. Here are the steps:

1. Login as an HR Manager (hr_manager, hr_manager).
2. From the Home page, select **Create Employee** from the left menu.
3. Enter data into the Employee Record form.
    a. **User Name** – uniquely identifies the employee, required field.
    b. **Password** – used for Authentication, required field.
    c. **Last name, First name, and Middle initial** – attributes that help identify the employee.

   d. **Street address, City, State code, and Zip** – attributes that identify where an employee lives.
   e. **Phone Number** – provides contact information, typically the home phone number for the employee. The phone number field accepts the following formats: 0123456789, 012-345-6789, (012)3456789, (012)345-6789, (012)-345-6789
   f. **Email address** – provides the email contact details for the Employee. An email must contain a @.
   g. **Social Security Number** – a number issued by the US Government for uniquely identifying individuals.
   h. **Salary** – the base monetary compensation the employee receives on an annual basis.
   i. **Business Unit** – each employee of ABC Toys belongs to one of three business units including **Information Technology**, **Business Development**, and **Human Resources**. The HR Manager will select a radio button corresponding with the employee's unit.
   j. **Management indicator (yes or no)** – the HR Manager will select a radio button corresponding with the employee's management status – **YES** if a manager, **NO** if not. Managers have certain rights as described in section 8.2, *Description of Use Cases*.
4. Select **Submit** to save the data. Selecting **Reset** will clear all of the fields. No data is saved until the **Submit** button is selected.
5. Upon a successful save, the system displays **Created a new Employee Successfully** in green text just below the header. In addition the Employee Projects table appears.

## 6.4 Search Employee

All employees can search for an employee. Depending on the logged in employee, a different set of results can return. Regular employees will find only their own record, where Business Unit Managers can search for all records within their unit, and HR Managers can search for all employees. Here are the steps:

1. From the Home page, select **Search Employees** from the left menu.
2. To search for a record enter the employee's last name or use the * character to perform wildcard searches (see Figure 6.2). For example, searching for emp* returns employee_1, employee_2, or any other employee beginning with emp. A search of *1 returns anything ending with 1, such as the fictional last name employee_1. Select **Search** to receive results. **Reset** clears the data.

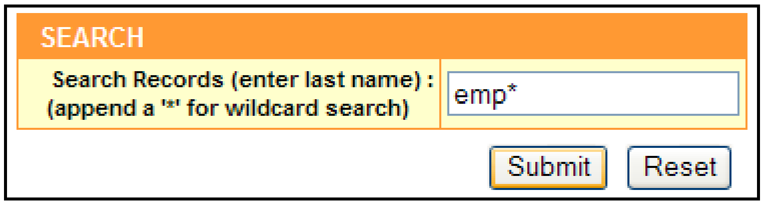

SEARCH
Search Records (enter last name) : (append a '*' for wildcard search) emp*

Figure 6.2 *Enter search criteria*

3. The system returns a list of matching employees.
4. Select the employee by clicking on the employee name.
5. The system then displays this record's details.

## 6.5 Create Project

Managers of the IT or Business Development business units may create and assign projects. Follow these simple steps to create a project:

1. From the Home page, select **Create Project** from the left menu.

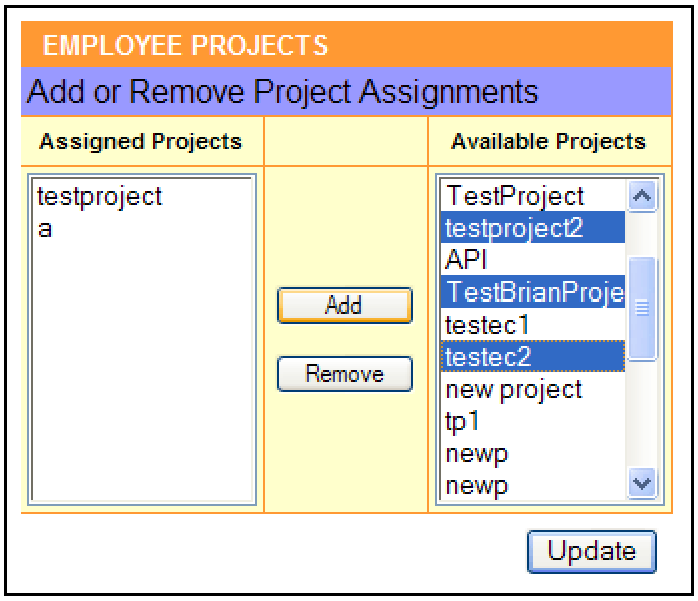
2. Enter data into the Employee Record form.
    a. **Project Name** –identifies the project, required field.
    b. **Project Description** – an overview of the project.
3. Select **Submit** to save the data. Selecting **Reset** will clear all of the fields. No data is saved until the **Submit** button is selected.
4. Upon a successful save, the system displays Created a new Project Successfully in green text just below the header.

## 6.6 Assign Projects

1. To assign a project, first login as the manager of the IT or Business Development business unit.
2. From the Home page, search for and select the appropriate employee record. See Section 9.4, *Search Employee* above.
3. In the Employee Projects table (see Figure 5.3 below), project assignments can be established or removed by simply selecting projects and moving them from one field to another. The field on the right lists the Available Projects (unassigned) while the field on the left lists the projects assigned to the employee – testproject and a. Simply highlight an Available Project and select **Add** to move it to the Assigned Projects field. Select **Remove** to move a project back to the Available Projects .

There are multiple ways to highlight the projects within the field:

1. Select one by clicking on the project name.
2. Select multiple users by holding down control while selecting and/or deselecting.
3. Select multiple by holding down the shift button while selecting the first and then last of a collection.


EMPLOYEE PROJECTS

Add or Remove Project Assignments

| Assigned Projects | | Available Projects |
|---|---|---|
| testproject<br>a | Add<br>Remove | TestProject<br>testproject2<br>API<br>TestBrianProje<br>testec1<br>testec2<br>new project<br>tp1<br>newp<br>newp |

Figure 6.3 *Add or remove project assignments using this interface*

## 6.7 Update Record

1. From the Home page, search for and select the appropriate employee record. See Section 9.4, *Search Employee* above.
2. Update the record by simply overwriting the current text in the field(s).
3. Select **Update** to save the data. Selecting **Reset** will clear all of the fields. No data is saved until the **Update** button is selected.
4. Upon a successful save, the system displays **Updated Employee Successfully** in green text just below

the header.

## 7. Data Model Implementation for Fine Grained Authorization

The RI utilizes the common security database schema to protect actions and data from unauthorized access. This section explains the data model implementation that enables fine grained authorization in the RI. Two database schemas used in this implementation have been combined to create the csm_ri data model. The first schema is the RI specific schema that contains data specific to the business logic of the RI (e.g., Employee and Project tables). The second schema is the CSM authorization schema that is used to store the authorization data.

### 7.1 User Groups

User provisioning is performed at runtime in the RI implementation. Once an employee is created via the RI, a User object is created an inserted into the User table. The user is then assigned to the appropriate user Group. The user is associated with either the HR Manager, Business/Technical Manager or Employee group.

By using user groups, the group association with a particular protection group and role only needs to occur one time; on the other hand, if user's must be associated directly then an association must be made for every single user for each protection group and role for which access is to be restricted or granted. As a result, the use of user groups is highly recommended for performance and maintainability of the authorization schema. See Figure 7.1 below.

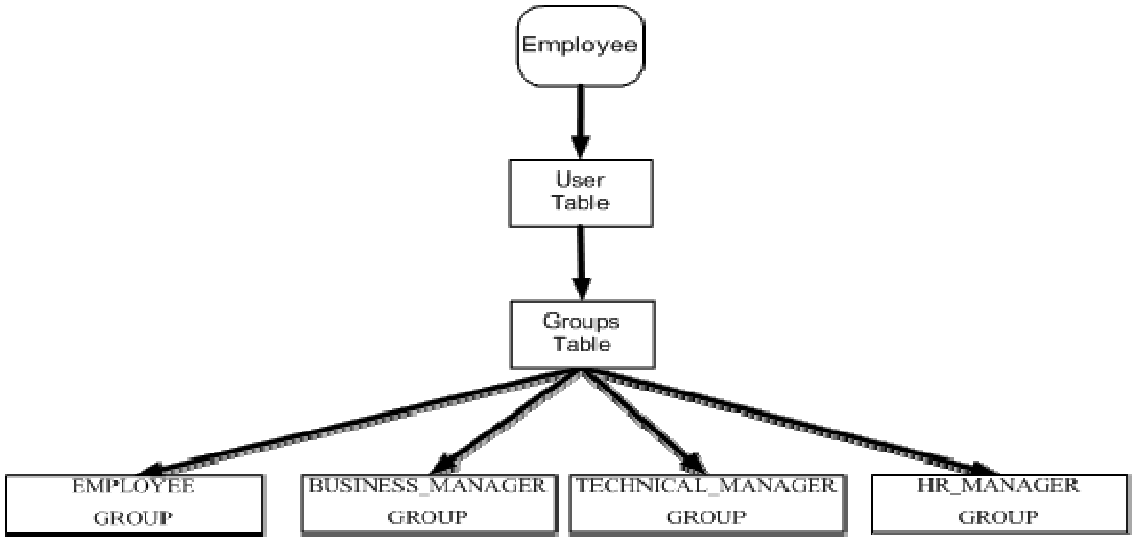


Figure 7.1 *User group diagram*

## 7.2 Roles and Privileges

The RI implementation utilizes several Roles and Privileges. There are three main roles based on the use cases of the RI: Employee, Employee Manager, and HR Manager. Each role is assigned the privileges to enable the appropriate access for that type of user. The Roles are then assigned to the respective user groups mentioned in the previous section.

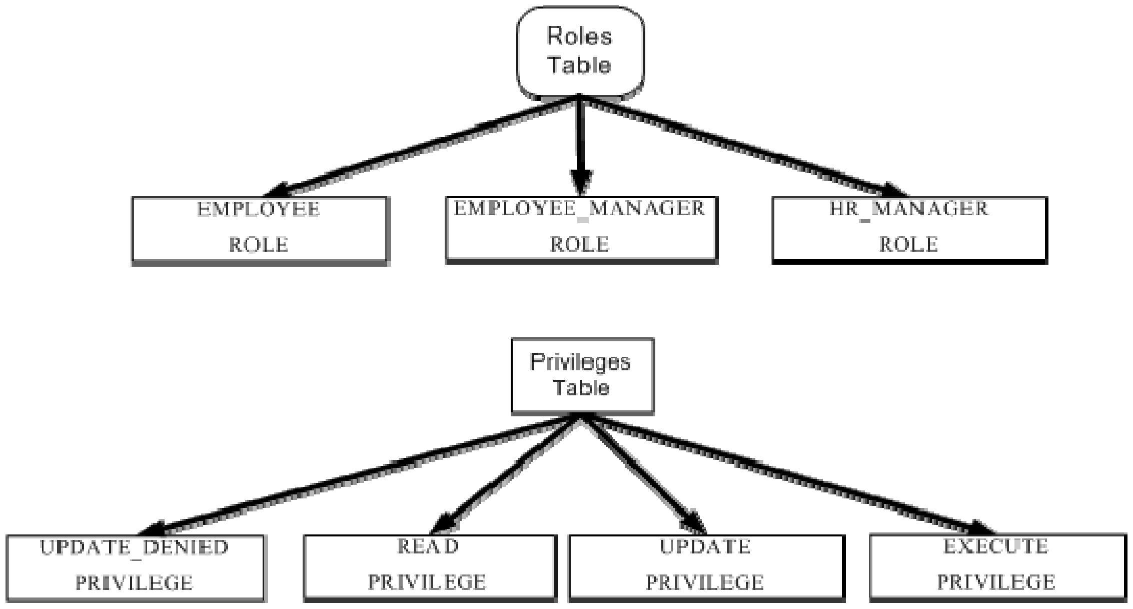


Figure 7.2 *Roles and Privileges Tables*

## 7.3 Protection Groups and Protection Elements

The use of Protection Groups is critical to the structure and foundation of the data model implementation. The RI utilized the projection groups to store collections of related protection elements. See table 7.1 below for a detailed description of protection groups utilized in the RI.

| **Protection Group Name** | **Description** |
|---|---|
| SECURED_MANAGER_ACTIONS | Collection of protection elements that actions that a Manager Role may perform. Elements contain class names of Struts actions. See the SecureAction in the java doc. |
| SECURED_HR_ACTIONS | Collection of protection elements that actions that a HR Role may perform. Elements contain class names of Struts actions. See the SecureAction in the java doc. |
| HR_DIVISION | Collection of protection elements that represent employee records that belong to the hr division. This allows the hr manager to view all employee records. |
| TECHNICAL_DIVISION | Collection of protection elements that represent employee records that belong to the technical division. This allows the technical manager to view their employee records. |
| BUSINESS_DIVISION | Collection of protection elements that represent employee records that belong to the business division. This allows the business manager to view their employee records. |
| EMPLOYEE_PROTECTED_ATTRIBUTES | Collection of protection elements that represent attributes of an employee (e.g., SSN, Salary) that may not be updated and/or viewed. |

Table 7.1 *Protection Groups*

## 7.4 HR Manager

The HR Manager user group, associated protection groups and roles are illustrated in figure 7.3 below. The HR Manager may EXECUTE secured HR Actions and may view all employee records contained in the HR Division protection group.

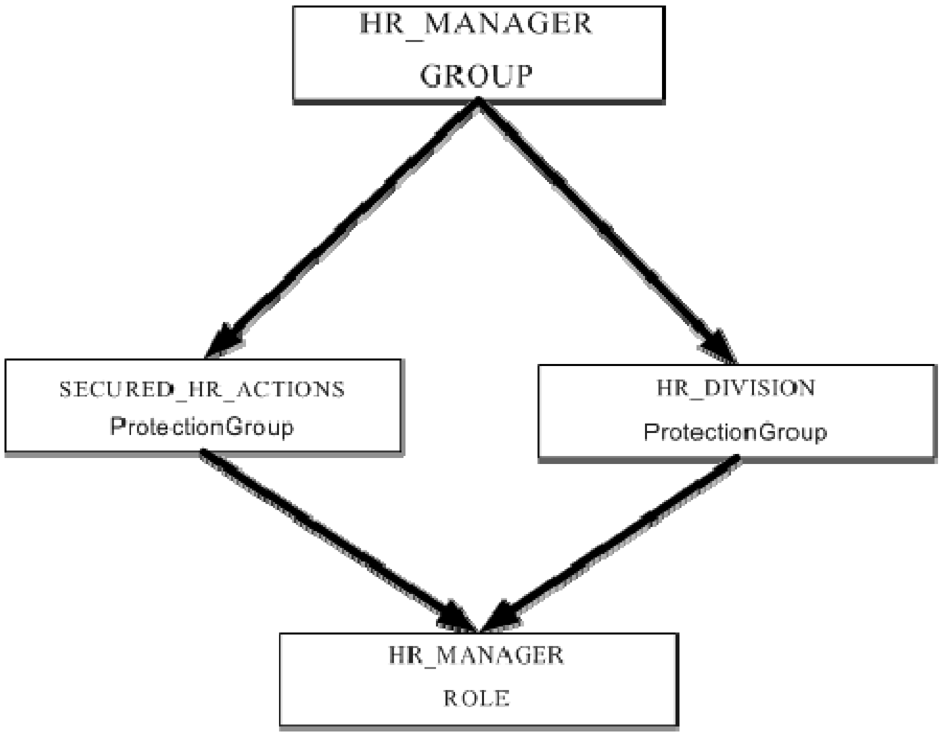

HR_MANAGER GROUP
SECURED_HR_ACTIONS ProtectionGroup
HR_DIVISION ProtectionGroup
HR_MANAGER ROLE

Figure 7.3 *The HR Manager user group, associated protection groups and roles*

## 7.5 Business Manager

The Business Manager user group, associated protection groups and roles are illustrated in figure 7.4 below. The Business Manager may EXECUTE secured Manager Actions and may view all employee records contained in the Business Division protection group The Business Manager role contains the UPDATE_DENIED privilege and is associated with the employee protected elements protection group which prevents the SSN and Salary elements from being updated.

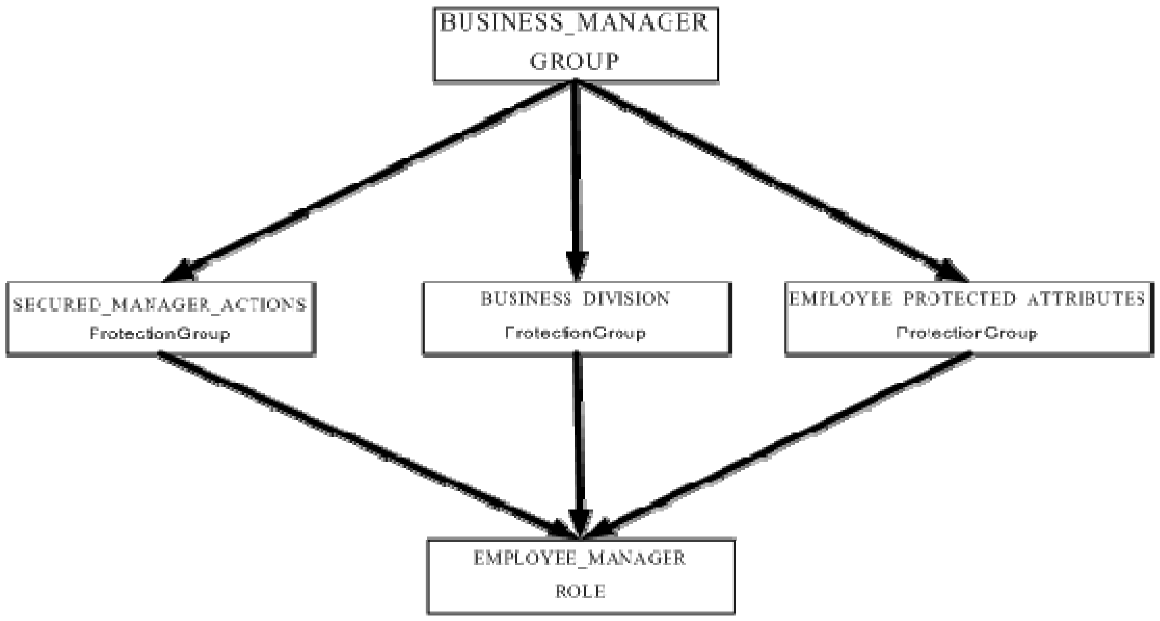

BUSINESS_MANAGER GROUP
SECURED_MANAGER_ACTIONS ProtectionGroup
BUSINESS DIVISION ProtectionGroup
EMPLOYEE PROTECTED ATTRIBUTES ProtectionGroup
EMPLOYEE_MANAGER ROLE

Figure 7.4 *The Business Manager user group, associated protection groups and roles*

## 7.6 Technical Manager

The Technical Manager user group, associated protection groups and roles are illustrated in figure 7.5 below. The Technical Manager may EXECUTE secured Manager Actions and may view all employee records contained in the Technical Division protection group. The Technical Manager role contains the UPDATE_DENIED privilege and is associated with the employee protected elements protection group which prevents the SSN and Salary elements from being updated.

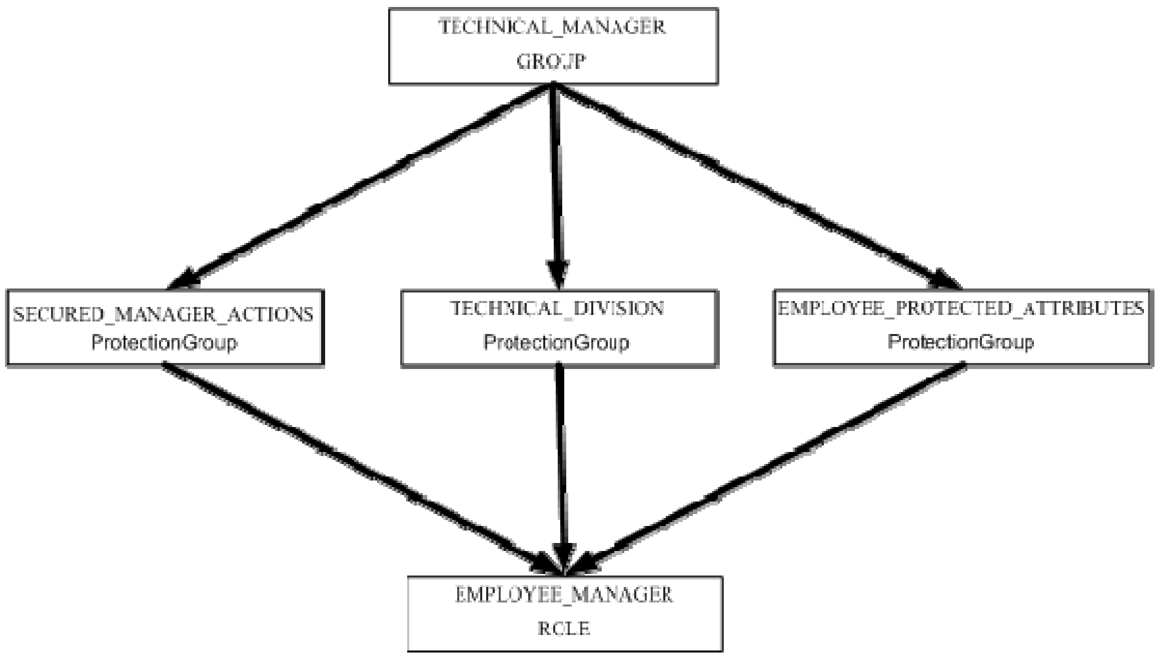

TECHNICAL_MANAGER GROUP
SECURED_MANAGER_ACTIONS ProtectionGroup
TECHNICAL_DIVISION ProtectionGroup
EMPLOYEE_PROTECTED_ATTRIBUTES ProtectionGroup
EMPLOYEE_MANAGER ROLE

Figure 7.5 *The Technical Manager user group, associated protection groups and roles*

## 7.7 Employee

The Employee user group, associated protection groups and roles are illustrated in figure 7.6 below. The employee may not EXECUTE secured Manager Actions or view other employee records. The Employee role contains the UPDATE_DENIED privilege and is associated with the employee protected elements protection group which prevents the SSN and Salary elements from being updated.

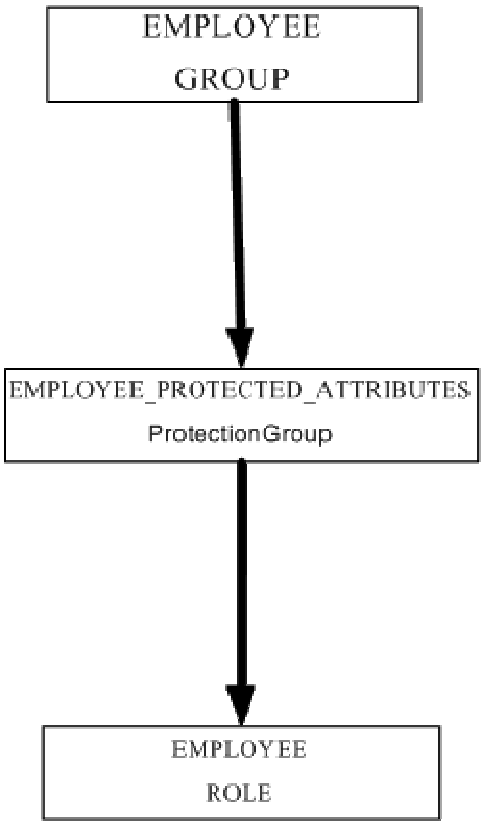


Figure 7.6 *The Employee user group, associated protection groups and roles*

## 8. Implementation of business scenarios using CSM

This section explains how CSM was used to implement the ABC Toys use cases and security requirements.

### 8.1 Login Functionality

In order to check the credentials provided by the user, RI uses the Authentication Manager Service exposed by the CSM APIs. It uses RDBMS to store the user credentials. The following is the configuration for the required to configure the RBMS based credential provider for RI in the login-config.xml file of JBoss. For details about each of the parameters refer to the *CSM Application Developers Guide*.

```
<application-policy name = "csm_ri">
   <authentication>
      <login-module code = "gov.nih.nci.security.authentication.loginmodules.RDBMSLoginModule" flag = "required" >
         <module-option name="driver">org.gjt.mm.mysql.Driver</module-option>
         <module-option name="url">jdbc:mysql://cbiodev104.nci.nih.gov:3306/csm_ri</module-option>
         <module-option name="user">ncisecurity</module-option>
         <module-option name="passwd">ncisecurity</module-option>
         <module-option name="query">SELECT * FROM EMPLOYEE WHERE LOGIN_NAME=? and PASSWORD=?</module-
option>
      </login-module>
   </authentication>
</application-policy>
```

In the LoginAction class the handle to Authentication Manager is obtained in the getAuthenticationManager method and stored as a static variable. The RI Application Context Name is passed to obtain the Authentication Manager and the related configuration specific to RI.

```
protected AuthenticationManager getAuthenticationManager()
      throws CSException {
   if (authMgr == null) {
      synchronized (LoginAction.class) {
         if (authMgr == null) {
            authMgr = SecurityServiceProvider
                  .getAuthenticationManager(CSM_RI_CONTEXT_NAME);
         }
      }
   }

   return authMgr;

}
```

In order to login, RI calls the login method of the Authentication Manager passing the user credentials. This method returns a Boolean true if the user credentials are valid indicating successful login. In case of login failure it throws an exception indicating the problem which could be related to the configuration not being correct or simply invalid credentials.

```
boolean loginSuccess = false;
try {
   loginSuccess = getAuthenticationManager().login(
         loginForm.getLoginID(), loginForm.getPassword());
} catch (CSException ex) {
   loginSuccess = false;
   log.debug("The user was denied access to the csm application.", ex);
}
```

## 8.2 Obtaining the Authorization Manager

In order to perform check access permissions for a user on any of the element, you first need to obtain the Authorization Manager. In RI the handle to authorization manager is obtained in the SecureAction Class however the method is implemented in the parent BaseAction class and stored as a static attribute. All the actions which require using the Authorization Manager should extend SecureAction class

Following is the implementation to obtain the Authorization Manager in the BaseAction class.

```
protected AuthorizationManager getAuthorizationManager() throws CSException {

   if (authMgr == null) {
      synchronized (BaseAction.class) {
         if (authMgr == null) {
            authMgr = SecurityServiceProvider
                  .getAuthorizationManager(CSM_RI_CONTEXT_NAME);
         }
      }
   }

   return authMgr;

}
```

## 8.3 Business Use Case: Creating an Employee

As per the business use case only the employees belonging to the HR Manager Group have access to create the employees. In RI the fully qualified class name is used as an object id which acts as the representative of the corresponding action. Using the user name of the logged in user, the class name and the “EXECUTE” privilege a call is made to the check permission method of the Authorization Manager. This method returns a Boolean true if the user is authorized. This functionality is always called when the execute method of all the classes extending the SecureAction is called. As a result, a check is always performed to check if the user has the “EXECUTE” privilege on that action or not.

Following is the isAuthorized method in the SecureAction class.

```
private boolean isAuthorized(HttpServletRequest request) throws Exception {
    Employee user = getUser(request);
    String objectId = SecurityUtils.getObjectIdForSecureMethodAccess(this
            .getClass());

    log.debug("The User name is: " + user.getUserName());
    log.debug("The Object ID is: " + objectId);

    boolean isAuthorized = false;
    try {
        isAuthorized = getAuthorizationManager().checkPermission(
                user.getUserName(), objectId, EXECUTE);

    } catch (CSException ex) {
        log.fatal("The Security Service encountered "
                + "a fatal exception.", ex);
        throw new Exception(
                "The Security Service encountered a fatal exception.", ex);
    }

    return isAuthorized;

}
```

This method is invoked from the executeWorkflow which is invoked from the execute method of the BaseAction. So for all the Action classes which extend the SecureAction class this method is automatically executed thereby controlling access to that class's actually action logic.

This mechanism is also used for the following Business Use Cases

1. A Manager can only create Projects
2. An Employee cannot create other employees

### 8.4 Business Use Case: Updating Salary and SSN

As per the business use case only the employees belonging to the HR Manager Group have access to update the salary of an employee. In RI this functionality is achieved by using achieved by using the secureUpdate method of the Authorization Manager in the UpdateEmployeeAction class. This method takes the original and the mutated object as parameters. It returns a new object containing the updated values for fields on which the user has access to update. So using this functionality if any other manager or employee tries to update the salary or social security number it will be reset to the original value.

Following is the implementation for the secureUpdate method in the UpdateEmployeeAction class.

```
public ActionForward executeWorkflow(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response) throws Exception {
    // TODO Auto-generated method stub

    //The modified employee object
    Employee mutatedObject = (Employee) form;

    //The original employee object
    Employee originalObject = EmployeeDAO
            .searchEmployeeByPrimaryKey(mutatedObject.getEmployeeId());

    log.debug("The original salary is: " + originalObject.getSalary());
    log.debug("The original ssn is: " + originalObject.getSsn());

    String[] associatedIds = mutatedObject.getAssociatedIds();
    String[] availableIds = mutatedObject.getAvailableIds();

    //secure the object by setting attributes to null where
    //the UPDATE_DENIED access is specified
    Employee securedObject = (Employee) getAuthorizationManager()
            .secureUpdate(getUser(request).getUserName(), originalObject,
                    mutatedObject);

    securedObject = EmployeeDAO.updateEmployee(securedObject);

    log.debug("The salary after secure update is: "
            + securedObject.getSalary());
    log.debug("The ssn after secure update is: " + securedObject.getSsn());

    ActionMessages messages = new ActionMessages();
    //ensure that the user is authorized
    //to update employee projects
    if (isAuthorized(request)) {
        securedObject.setAssociatedIds(associatedIds);
        securedObject.setAvailableIds(availableIds);
        securedObject = EmployeeDAO.updateEmployeeProjects(securedObject);

        messages.add(ActionMessages.GLOBAL_MESSAGE, new ActionMessage(
                Constants.MESSAGE_ID, "Updated Employee Successfully"));
    } else {
        //warn the user they don't have privilege to update employee
        // projects
        messages
            .add(
                ActionMessages.GLOBAL_MESSAGE,
                new ActionMessage(
                    "access.modify.employee_project.denied",
                    new String[] {
                        getUser(request).getUserName(),
                        ", "
                            + SecurityUtils
                                .getObjectIdForEmployeeProjecAccess() }));

    }

    saveMessages(request, messages);

    List l = new LinkedList();
    l.add(securedObject);

    request.getSession().setAttribute(EMPLOYEE_LIST, l);
    request.getSession().setAttribute(EMPLOYEE_ID,
        securedObject.getEmployeeId().toString());

    return mapping.findForward(ACTION_SUCCESS);

}
```

This mechanism is also used for the following Business Use Cases

1. An Employee can update their personal information except Salary, Social Security Number and Project.

## 8.5 Business Use Case: Assigning a Project

As per the business use case only the employees belonging to the Manager Group have access to assign a project to an employee. In RI this functionality is achieved by using achieved by using the checkPermission method of the Authorization Manager in the UpdateEmployeeAction class. This method verifies whether the user has access to assign a project to the employee. The call to this method is made from the executeWorkflow method as shown above in section 7.4

The checkPermission is implemented in the isAuthorized method as shown below

```
private boolean isAuthorized(HttpServletRequest request) throws Exception {

    String user = getUser(request).getUserName();
    boolean isAuthorized = false;
    try {
        isAuthorized = getAuthorizationManager()
                .checkPermission(user,
                        SecurityUtils.getObjectIdForEmployeeProjecAccess(),
                        EXECUTE);

    } catch (CSException ex) {
        log.fatal("The Security Service encountered "
                + "a fatal exception.", ex);
        throw new Exception(
                "The Security Service encountered a fatal exception.", ex);
    }
    return isAuthorized;
}
```

## 8.6 Business Use Case: Search Results

As per the business use case, the search results should return only the records to which the user has access.

- An employee has access to only his own record.
- The Manager has access to records of the employees which belong to their unit.
- The HR Manager has access to all the employee records.

In RI this functionality is achieved by using the checkPermission method of the Authorization Manager in the SearchEmployeeAction class. This method verifies whether the user has access to the employee record. Once the search result is obtained from the database based on the criteria entered by the user, a check is performed to verify if the user has access on each of the record or not. The checkPermission checks if the user is the owner of the record or not (As in case of an employee owing their own record) or if the user has access to the record via a protection group (As in the case of Manager and HR Manager having access to other employee records). If the user has access either way then a boolean true is returned allowing the user to access that particular record.

The call to checkPermission method is made from the doAuthorization method as shown below

```
private List doAuthorization(HttpServletRequest request, List employees)
        throws Exception {
    Iterator i = employees.iterator();
    String user = getUser(request).getUserName();
    List filterList = new LinkedList();
    try {
        while (i.hasNext()) {
            Employee empl = (Employee) i.next();
            log.debug("Checking permission for employee "
                    + empl.getLastName());
            if (getAuthorizationManager().checkPermission(user,
                    SecurityUtils.getEmployeeObjectId(empl), READ)) {
                //add only employees from the list where
                //access is granted for READ permission
                //or they must be the owner
                //Peformance will also be better
                //since creating a new list is faster
                //than deleteing from an existing list.
                filterList.add(empl);
            }

        }

    } catch (CSException ex) {
        log.fatal("The Security Service encountered "
                + "a fatal exception.", ex);
        throw new Exception(
                "The Security Service encountered a fatal exception.", ex);
    }

    return filterList;

}
}
```

## 8.7 Business Use Case: Setting Access Level

As per the business use case, whenever a new employee is added to the system, the access level should be set. The employee should be made the owner of the record so that he/she can access it. Also the employee should be added to their respective business group’s protection group, so that the managers from that group can have access to their record. Additionally, the employee should be added to the HR division so that they are visible to the HR Manager. This is achieved by using various methods of the Authorization Manager. Whenever a new Employee is created, a corresponding protection element is created, representing the user in the Authorization Database. For this use case, a handle to the User Provisioning Manager was also obtained to create a new user and to assign the user to the group. The employee is also assigned to the protection group which corresponds to its business unit so that the managers belonging to the unit can have access to the employee’s record. The employee entry also should be added to the HR division so that it is visible to HR Manager.

This implementation is found in the doAuthorization method of the CreateEmployeeAction class as shown below:

```
private void doAuthorization(HttpServletRequest request, Employee empl)
        throws Exception {

    try {

        //Create a protection element that represents the employee record
        ProtectionElement pe = new ProtectionElement();
        pe.setObjectId(SecurityUtils.getEmployeeObjectId(empl));
        pe.setProtectionElementName("EMPLOYEE_RECORD_"
                + empl.getEmployeeId());
        pe
                .setProtectionElementDescription("The gov.nih.nci.security.ri.valueObject.Employee Object");

        //create the employee protection element to protected the
        //employee's data
        getAuthorizationManager().createProtectionElement(pe);

        //Create the User for Authorization
        User user = createUser(empl);

        //Assign the User to the appropriate UserGroup
        getUserProvisioningManager().assignUserToGroup(user.getLoginName(),
                SecurityUtils.getEmployeeGroup(empl));

        //assign the employee as owner of record
        getAuthorizationManager().setOwnerForProtectionElement(
                user.getLoginName(), pe.getObjectId(), null);

        //assign the employee to his business unit
        getAuthorizationManager().assignProtectionElement(
                empl.getBusinessUnit(), pe.getObjectId());

        //If they are not part of HR division then add
        //employee record so that HR managers can view
        //all employee data
        if (!HR_DIVISION.equals(empl.getBusinessUnit())) {
            //assign access to the employee for HR Division
            getAuthorizationManager().assignProtectionElement(HR_DIVISION,
                    pe.getObjectId());
        }
    } catch (CSException ex) {
        log
                .fatal(
                        "The Security Service encountered a fatal exception.",
                        ex);
        throw new Exception(
                "The Security Service encountered a fatal exception.", ex);
    }
```

## 9. Best Practices

1. For an application in a container, you must use DataSource for connection pooling. If the application is using the local JVM, use your preferred connection pooling. If you do not use DataSource, known bugs may result.
2. Use the Tomcat or JBoss containers, as these have both been tested with CSM.
3. Instantiate AuthManager once per application (use a singleton method).
4. Assign negative privileges where possible. Examples include update_denied and read_denied. Using negative privileges for a few protected attributes is more efficient than assigning privileges to most.
5. Use common privileges such as create, read, update, and delete. In future releases CSM will move toward standardization.
6. For the Object ID, provide the fully qualified class name for check permission.
7. If using a small data set for the authorization data, cache the data.
8. Use declarative authorization in order to minimize programmatic security and minimize security maintenance. The RI provides an example – in this implementation objects choose whether or not to extend SecureObject or other classes.
9. When using CSM, assign users to a Group and then assign the roles and PGs to a group rather than a user. Advantages include:
    a. improved performance and maintainability
    b. retained relationships even if the user no longer exists
    c. ability to scale to high volumes of users

## 10. Things to avoid

1. Do not instantiate the CSM API Authorization Manager or other methods repeatedly. Also avoid calling CheckPermission in a loop.
2. Avoid using security for data aggregation. Although this is done in the RI to show the breadth of CSM, to conserve system resources use your own application to perform data aggregation.
3. Do not use connection pooling such as c3po or DBCP, rather use a DataSource that is configurable.