

CA Embedded Entitlements Manager

Programming Guide

r8.4



This documentation and any related computer software help programs (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of the Documentation for their own internal use, and may make one copy of the related software as reasonably required for back-up and disaster recovery purposes, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the provisions of the license for the Product are permitted to have access to such copies.

The right to print copies of the Documentation and to make a copy of the related software is limited to the period during which the applicable license for the Product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

EXCEPT AS OTHERWISE STATED IN THE APPLICABLE LICENSE AGREEMENT, TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The use of any product referenced in the Documentation is governed by the end user's applicable license agreement.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Copyright © 2008 CA. All rights reserved.

CA Product References

This document references the following CA products:

- CA[®] Embedded Entitlements Manager (CA EEM)
- CA[®] Directory
- CA[®] SiteMinder[®] Web Access Manager (CA SiteMinder)
- CA[®] Identity Manager
- CA[®] Security Command Center
- CA[®] Integrated Threat Management
- CA[®] Enterprise Log Manager

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At <http://ca.com/support>, you can access:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, please complete our short [customer survey](#), which is also available on the CA support website, found at <http://ca.com/support>.

Contents

Chapter 1: Introduction	11
Who Should Read This Guide	11
Architecture	12
SDK Contents	13
Client Applications	14
Policy Server.....	14
Chapter 2: Application Instances	15
Overview	15
How to Register an Application.....	15
Attach to Backend Server.....	16
Create an Application Instance.....	17
Define User Attributes.....	18
Define Resource Classes	20
Define Obligations	21
Register Application.....	22
Modify an Application Instance	23
Unregister an Application Instance	24
Chapter 3: Users	25
Overview	25
Create Global Users.....	26
Create Application-Specific Users.....	27
Associate Global User with Application-Specific Details	28
Modify Membership	29
Search Users Using Attributes	30
Retrieve a Global User	31
Retrieve an Application-Specific User	32
Delete a User	33
Chapter 4: Groups	35
Overview	35
Create Global User Groups	36
Create Application-Specific User Groups	37
Search Groups Using Attributes.....	38

Retrieve a Global User Group	39
Retrieve a User Group	39
Delete a Group	40

Chapter 5: Access Management **41**

Policies	41
Overview	41
Types of Policies	42
Types of Authorization Checks	44
Create, Modify, and Verify Policies	44
Filters	49
Overview	49
Build Filters to Use in Searches	50
Build Filters to Use in Policies	54
Structure of a Filter	58
Authorization	62
SDK Cache	62
Session	63

Chapter 6: Authentication **65**

Pluggable Authentication Module	65
PassTicket	65
Prerequisites for Single Sign-On	66
Generate PassTicket	69
How Applications Communicate with Mainframe Systems	70
Kerberos	70
Using TGT	70
Using Kerberos Principal and Password	74

Chapter 7: Policy Evaluation **75**

Overview	75
How Policies Are Evaluated	76
Gathering Identity Attributes	77
Assembling Environment Information	77
Policy Matching	78
Evaluating Matching Algorithm	78
How the Best Match Algorithm is Evaluated	79
Best Match Handling for Regular Expression Policies	80
Policy Filter Evaluation	81
Delegated Authority Evaluation	82

How Obligations Are Calculated	84
Chapter 8: Exception Handling	85
Overview	85
Safe Exception.....	86
Safe Authorization Exception	88
Safe BackendServer Exception	88
Safe Password Exception.....	89
Chapter 9: Identity Management	91
Administration Methods	91
Administering Global Users, Groups, and Folders.....	92
Applying Password Policies.....	93
Identity Self Administration.....	95
Configure Externally Generated Certificates	96
Enable Trace on CA EEM Server	96
Enable Trace on CA EEM SDK.....	97
Dynamic user groups.....	97
How Offline Authentication Works.....	98
Chapter 10: Configure Directories	99
Overview	99
Configure External Directory.....	100
Restart iGateway (Linux)	101
Custom Mapped Directory	101
Example: Configure UPN Using Custom Mapped Directory.....	102
Test Configuration.....	103
CA SiteMinder.....	103
How You Integrate CA SiteMinder with CA EEM	104
CA SiteMinder Configuration Parameters	104
How Single Sign-on Works between CA SiteMinder and CA EEM.....	105
How Authentication Works Using CA SiteMinder Authentication Schemes	105
Chapter 11: Integrate Web Services with CA EEM	107
Web Services Architecture.....	108
Configure Web Services for CA EEM	110
Configuration File.....	112
Sample Configuration File.....	114
Tools.....	115

XACML Profile for CA EEM.....	116
XACML Integration.....	116
WSDL for CA EEM XACML.....	117
XACML Services for CA EEM	117
XACML Requests.....	118
XACML Responses.....	120
Export and Import Using XACML	121
Mapping CA EEM Operations to XACML Functions.....	126
Mapping CA EEM Data Types to XACML Data Types.....	127
XPath Expressions for CA EEM Filters.....	127
AttributeId Values for XACML AttributeDesignator Elements	128
Examples	128
SPML Profile for CA EEM	135
SPML Integration.....	135
Terminology	135
WSDL for CA EEM SPML.....	136
SPML Profile.....	137
Example SPML Requests and Responses.....	146

Chapter 12: Event Management **153**

Event Policies	153
How Event Policies are Evaluated	153
Controlling Event Delivery.....	154
Default Event Policy	155
Event Data Model.....	156
Administrative Events.....	157
Runtime Events.....	159
Coalesced Events.....	161
Reliable Event Delivery	161
Enable Reliable Event Delivery.....	162
Route Events.....	163

Chapter 13: Server Configuration **165**

Server Configuration	165
Using Java Authentication and Authorization Service	166

Chapter 14: Managing with CA Products **169**

Provisioning through CA Admin	169
Security Management with CA SCC.....	170
Reporting and Analysis	171

Work with Audit Events	171
Chapter 15: Sample WorkFlow	173
Overview	173
Defining Identity and Access Requirements.....	174
Designing Safe Objects to Implement	175
Defining the Application Instance	177
Defining Calendars.....	182
Defining Policies	185
Designing the User Interface.....	188
How to Design User Interface	188
Migrating	188
Identity.....	189
Modifying StoredObjects	190
Folders and Paths	192
Search Size	192
Appendix A: Safex Command Line Reference	193
Exit Codes	194
Appendix B: Example Safex XML Scripts	197
Register.....	198
Unregister.....	198
Export.....	199
Export Multiple	199
CreatedExportMultiple.....	200
Export Global Settings	201
Global Settings	201
Translations	201
Global User	202
User.....	203
UserGroups	203
GlobalUserGroup.....	203
Policy	204
Calendar.....	205
Extended User Attributes.....	206
Sample Application.....	207

Appendix C: Reference Matrix

219

Index

225

Chapter 1: Introduction

CA Embedded Entitlements Manager (CA EEM) allows applications to:

- Securely manage users
- Share common access policy management
- Authenticate and authorize from a data source

It provides a common web interface for policy definition, user management, and means to audit user's activities.

CA EEM allows several applications to share a single source to authenticate users and manage their access rights.

Who Should Read This Guide

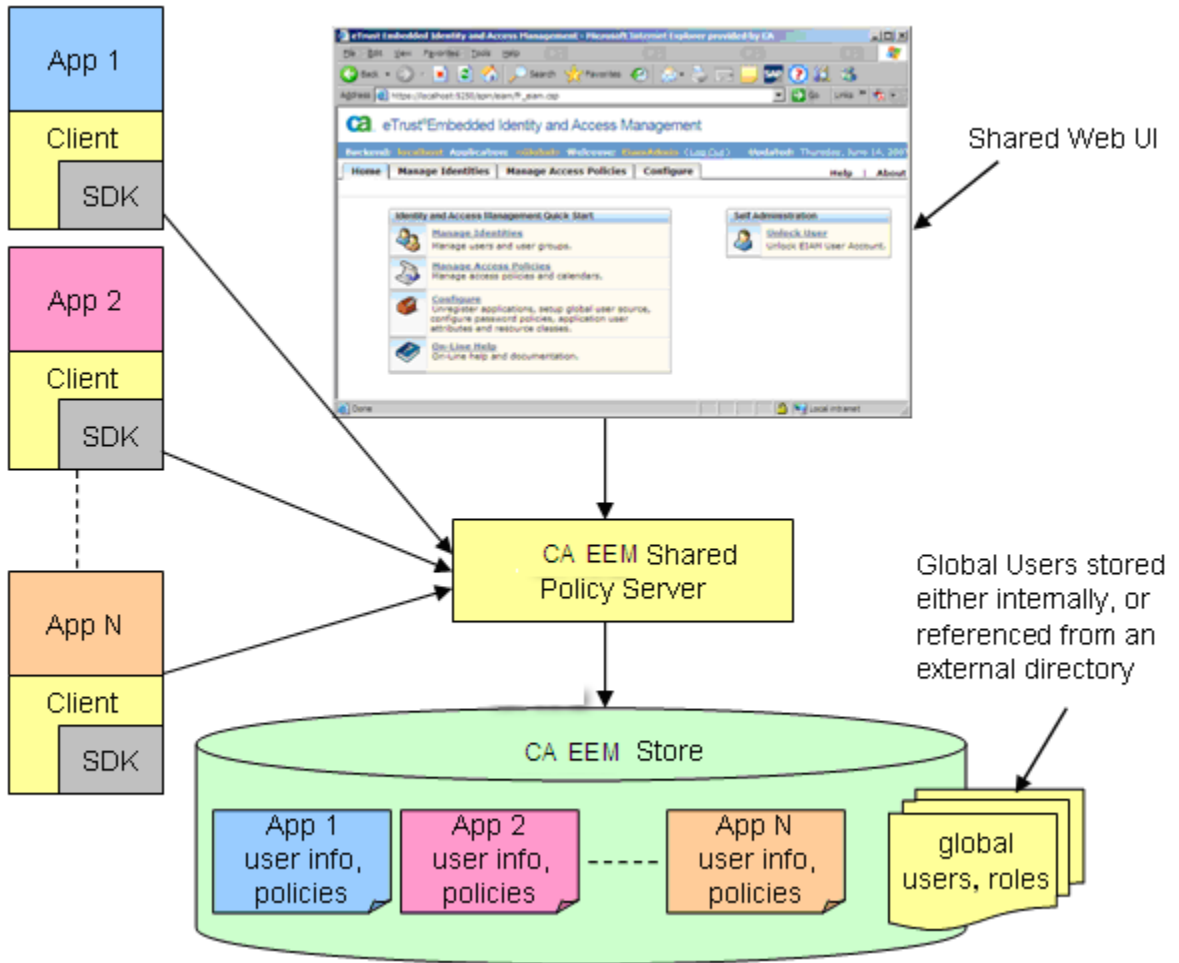
This guide is intended for system programmers, administrators, and integrators who are responsible for defining, monitoring, and managing CA EEM users and tasks. It describes how to integrate CA EEM with your application.

This guide assumes you have already installed the CA EEM and are familiar with the UNIX or Windows operating environment in which the tasks are performed. This guide also assumes you have knowledge on any of the following programming languages:

- C
- C#
- C++
- Java

Architecture

The following illustration depicts the relationship between CA EEM Server, applications, shared server, and database:



More Information:

[SDK Contents](#) (see page 13)

[Client Applications](#) (see page 14)

[Policy Server](#) (see page 14)

SDK Contents

The CA EEM SDK provides means to authenticate external user sources, develop access policies, and deliver security events. It provides a command-line interface that application administrators can use for silent administration.

The SDK APIs lets you manage application instances, resource classes, access policies, calendars, folders, and sessions. The APIs are in C, C#, C++, and Java languages.

The CA EEM SDK provides the following:

Content	Default Install Path
Sample application 'RBC_Hospital'	C:\Program Files\CA\Embedded IAM SDK\elsewhere\safex
Safetool source <ul style="list-style-type: none"> ■ C ■ C# ■ C++ ■ Java 	C:\Program Files\CA\Embedded IAM SDK\safetool
C# Unit Tests	<SDK_Installed_directory>/unittests/csharp Note: For more information on C# Unit Tests, see readme.txt under <SDK_Installed_directory>/unittests/csharp
C++ Unit Tests	<SDK_Installed_directory>/unittests/cpp Note: For more information on C# Unit Tests, see readme.txt under <SDK_Installed_directory>/unittests/cpp
JUnit Unit Tests	<SDK_Installed_directory>/unittests/java Note: For more information on C# Unit Tests, see readme.txt under <SDK_Installed_directory>/unittests/java

Documentation

The SDK documentation is installed along with the CA EEM SDK. The CA EEM SDK documentation is at: `<install_path>\CA\Embedded IAM SDK\Doc`

Default: C:\Program Files\CA\Embedded IAM SDK\Doc

It provides the following:

- Java reference
- C# reference
- C/C++ reference

Note: For information on installing CA EEM SDK, see *Getting Started*.

Client Applications

A business application that uses the CA EEM SDK is a client. When an application requires users to authenticate themselves, evaluate business policies, or to log a significant event, the application invokes one of the CA EEM SDK methods.

When an application instance is created with the CA EEM policy server, all application policies, calendars, and session-specific user groups are sent from the server, and are cached in the client for use during policy evaluation. The cache updates itself frequently.

More Information

[SDK Cache](#) (see page 62)

Policy Server

The CA EEM Policy Server is shared by all the applications and is used to:

- Authenticate users
- Deliver audit events to audit collection tools
- Store application-specific information
- Set application-level user attributes
- Configure external user store
- Set calendars

Chapter 2: Application Instances

This section contains the following topics:

[Overview](#) (see page 15)

[How to Register an Application](#) (see page 15)

[Attach to Backend Server](#) (see page 16)

[Create an Application Instance](#) (see page 17)

[Define User Attributes](#) (see page 18)

[Define Resource Classes](#) (see page 20)

[Define Obligations](#) (see page 21)

[Register Application](#) (see page 22)

Overview

CA EEM provides its services to the applications registered with it. When an application registers with CA EEM, an application instance is created. This application instance stores user details, access policies, calendars, and application-specific user groups and folders.

How to Register an Application

To register an application with CA EEM, perform the following tasks:

1. Attach to the backend server
2. Create an application instance
3. Define user attributes
4. Define resource classes
5. (Optional) Define obligations
6. Register the application

Note: You need administrative privileges to register an application with CA EEM. The Eiamadmin user has the administrative privileges.

Attach to Backend Server

You must attach to the CA EEM Policy Server to get the application-specific policies and resources to the client.

To attach an application to the backend server

1. Instantiate a SafeContext Class.
2. Set the backend server to the host where CA EEM Policy Server is running.
3. Call the authenticateWithpassword method to verify the authenticity of the user.

The method returns an instance of SafeSession, which is used during policy evaluation.

4. Attach to the global space using the session.

This session is valid for 24 hours, after which CA EEM refreshes the session automatically.

Example: Attach to backend server

The following example attaches an application to the backend server generating a session, and attaches to global space using the generated session:

```
//Instantiate a SafeContext Class.  
SafeContext safecontext = new SafeContext();  
//Set the backend to the host where Server is running.  
safecontext.setBackend("<hostname>");  
//Call the authenticateWithXXX to verify the authenticity.  
SafeSession safesession = safecontext.authenticateWithPassword("username", "password");  
//Attach to global space using the session.  
safecontext.attach(null,safesession);
```


Create an Application Instance

You must create an application instance to add application data. The application instance is used to store application-specific user details, access policies, calendars, and application-specific user groups and folders.

To create an application instance

1. Instantiate a `SafeApplicationInstance` class.
2. Assign a context to the application by calling the `setContext` method within the `SafeApplicationInstance` class.

The `SafeApplicationInstance` (`sai`) object is created, when a login is authenticated.

3. (Optional) Add additional information about the application, such as, major version, minor version, brand, application name, and description.

Example: Create an application instance

The following example creates an application instance:

```
//Instantiate a SafeApplicationInstance class.
SafeApplicationInstance sai = new SafeApplicationInstance();
//Assign a context to the application by calling the setContext method.
sai.setContext(safecontext);
//Add additional information about the application
//Set a label to the application.
sai.setLabel("xyzBank");
//Set a name for the application.
sai.setApplicationName("XYZ Bank Management");
//Set the major version.
sai.setMajorVersion("1");
//Set the minor version.
sai.setMinorVersion("0");
//Set the brand.
sai.setBrand("ABC");
//Set the description to the application.
sai.setDescription("For managing bank's resources/identities and define the access policies");
```

More Information:

- [Define User Attributes](#) (see page 18)
- [Define Resource Classes](#) (see page 20)
- [Define Obligations](#) (see page 21)
- [Register Application](#) (see page 22)

Define User Attributes

In CA EEM, every user of an application has application-specific attributes. You can define the application-specific user attributes while creating an application and refer to them in policies, or create application users by adding these attributes to a global user.

Note: You must attach to the backend server and create an application instance, before defining the user attributes.

To define user attributes, add the attributes by calling the `addUserAttribute` method, which has the following syntax:

```
addUserAttribute(attribute:field:value)
```

The `addUserAttribute` method supports the following types:

Text

Specifies a field that contains a text value.

Number

Specifies a field that contains a numeric value. You can use the following characters: 0-9, comma (,), or hyphen (-).

Password

Specifies a field that contains a masked value as you might see in a password field. Use the type to mask the content of the field from a user.

Boolean

Specifies a check box for user to select.

Select

Specifies a drop-down list from which a user can choose a value.

Multi-valued

Specifies the field can have multiple values. When assigning user attributes, the user is presented with multiple fields to enter values.

Example: Define user attributes

The following example defines the user attributes:

```
sai.addUserAttribute("text:memberID");  
sai.addUserAttribute("select:member:customer");  
sai.addUserAttribute("select:member:staff");  
sai.addUserAttribute("mvttext:memberID");
```

More Information:

[Attach to Backend Server](#) (see page 16)

[Create an Application Instance](#) (see page 17)

[Users](#) (see page 25)

Define Resource Classes

Resource classes are used in application instances to classify resources. You can identify the resource classes in an application and define access policies to restrict the access. Every resource class has a name, actions, and attributes associated.

Example: Consider a banking application in which you must restrict access to a resource called Loanrecords. In this scenario, LoanRecords is the resource class name, actions will be either read or write, and the resource class attributes will be amount, account ID, owner name, and so on.

Note: You must attach to the backend server and create an application instance, before adding the resource class.

To define resource classes

1. Identify all the resource classes in an application.
2. Add the resource classes to the safeApplicationInstance.
3. Protect the resources by defining access policies.

Example: Define resource class

The following example defines a LoanRecod resource class in a banking application.

```
//Instantiate a safe resource class object.  
SafeResourceClass res = new SafeResourceClass();  
//Set the name of resource class to loanrecord.  
res.setName("loanrecord");  
//Add an action 'read' to the resource class.  
res.addAction("read");  
//Add an action 'write' to the resource class.  
res.addAction("write");  
//Add a named attribute 'amount' to the resource class.  
res.addNamedAttr("amount");  
//Add a named attribute 'accountID' to the resource class.  
res.addNamedAttr("accountID");  
//Add a named attribute 'Ownername' to the resource class.  
res.addNamedAttr("ownerName");  
//Add the resource class to the safe application instance object.  
sai.addResourceClass(res);
```

More Information:

[Attach to Backend Server](#) (see page 16)

[Create an Application Instance](#) (see page 17)

[Modify an Application Instance](#) (see page 23)

Define Obligations

You can define obligations to policies that return actions to the application after an authorization check. Obligation policies are application-specific. A policy can contain one or more obligation names and attributes.

Using obligations, applications can control the actions it can perform when access is granted or denied. For example, the application might send an event or start a workflow process, or send an email.

Note: You must attach to the backend server and create an application instance, before adding obligations.

To define obligations

1. Define the obligation names in the `ObligationName` attribute of the `ApplicationInstance` object.
2. Create a new safe obligation policy.
3. Attach obligatory actions to the `SafeObligations` used in policies based on the authorization requests.

Example: Add an obligation

The following example adds an obligation to send email to the manager when a loan is approved. On successful loan approval, during policy evaluation, an obligatory action of sending email must happen. You can define an obligation named `email` and add it to the `safeApplicationInstance`.

```
//Create a safecontext (obj), attach to an application instance, and get the safe application instance (sai) object.
sai.addObligationName("email");
sai.somodify();
SafePolicy sp = new SafePolicy();
sp.setContext(obj);
sp.setResourceClassName("SafeObligation");
sp.setPath("New Oblig Policy");
sp.setDescription("New Oblig Policy");
sp.addIdentity("erdoctor");
sp.addResource("admit/patient/*");
sp.addAction("FulfillOnGrant");
```

More Information:

[How Obligations Are Calculated](#) (see page 84)

[Define User Attributes](#) (see page 18)

[Policies](#) (see page 41)

Register Application

You must register an application with CA EEM to use its services. When an application registers with CA EEM, an application instance is created.

Note: You must attach to the backend server and create an application instance, before you register an application.

To register an application

1. Create a safeapplicationinstance object.
2. Assign a context to the application by calling the setContext method and set the label, path, and name.
3. Add additional details like user attributes, resource classes, obligations, and translations.
4. Call the registerApplicationInstance method.

Example: Register an application

The following example registers an application:

```
//Instantiate a SafeApplicationInstance class.
SafeApplicationInstance sai = new SafeApplicationInstance();
//Assign a context to the application by calling the setContext method.
sai.setContext(safecontext);
//Add additional information about the application.
//Set a label to the application.
sai.setLabel("xyzBank");
//Add additional details like user attributes, resource classes, obligations, and translations.
//Set a name for the application.
sai.setApplicationName("XYZ Bank Management");
//Register the application
safecontext.registerApplicationInstance(sai,"samplecertfile.p12","samplepassword");
```

More Information:

[How to Register an Application](#) (see page 15)

[Attach to Backend Server](#) (see page 16)

[Create an Application Instance](#) (see page 17)

Modify an Application Instance

You can modify an application instance that is registered with CA EEM.

The following is a sample procedure to modify a resource class by adding a new attribute.

To modify an application instance

1. Attach to the application with administrative privileges.
2. Retrieve the application instance object.
3. Instantiate the safe resource class object.
4. Set the resource class name
5. Add a new attribute to the resource class.
6. Commit the changes by calling the soModify method.

The changes are applied to the application instance.

Example: Modify an application instance

The following example modifies a banking applications 'loan record' resource class by adding 'OwnerName' attribute:

```
//Attach to XYZ Bank Application
safecontext.attach("xyzBank",ss);
//Get Application Instance.
SafeApplicationInstance sai = safecontext.getApplicationInstanceObject();
//Instantiate the safe resource class object.
SafeResourceClass res = new SafeResourceClass();
//Set the name of resource class to loanrecord.
res.setName("loan record");
//Add a named attribute 'OwnerName' to the resource class.
res.addNamedAttr("OwnerName");
//Add the resource class to the safe application instance object.
sai.addResourceClass(res);
//Commit the modifications.
sai.soModify();
```

More Information:

[Create an Application Instance](#) (see page 17)

[Define Resource Classes](#) (see page 20)

Unregister an Application Instance

You can unregister an application instance that is registered with CA EEM from the global application space. Application instances must be unregistered before you uninstall CA EEM.

To unregister an application instance

1. Instantiate a SafeContext Class.
2. Set the backend server to the host where CA EEM Policy Server is running.
3. Call the authenticateWithpassword method to verify the authenticity of the user.
4. Log into the <global> application space by calling the attach method.
5. Call the UnregisterApplicationInstance method.

Example: Unregister an application instance

The following example unregisters an application instance:

```
//Instantiate a SafeContext Class.  
SafeContext safecontext = new SafeContext();  
//Set the backend to the host where Server is running.  
safecontext.setBackend("<hostname>");  
//Call the authenticateWithpassword to verify the authenticity.  
SafeSession safesession = safecontext.authenticateWithPassword("EiamAdmin", "EiamAdminpassword");  
//Attach to the global space.  
safecontext.attach(null, session);  
//Unregister the application instance.  
safecontext.unregisterApplicationInstance("Application name");
```

More Information

[Modify an Application Instance](#) (see page 23)

[Register Application](#) (see page 22)

Chapter 3: Users

This section contains the following topics:

[Overview](#) (see page 25)

[Create Global Users](#) (see page 26)

[Create Application-Specific Users](#) (see page 27)

[Search Users Using Attributes](#) (see page 30)

[Retrieve a Global User](#) (see page 31)

[Retrieve an Application-Specific User](#) (see page 32)

[Delete a User](#) (see page 33)

Overview

CA EEM lets you authenticate and authorize users. It lets you create policies that control the user access privileges to use applications. You can use CA EEM to support application-specific authentication and authorization.

Users in CA EEM are classified as follows:

Global users

Global users are users that are available for sharing across all application instances registered with CA EEM. Every user in CA EEM is a global user by default.

Application-specific users

Application users are specific to the application instance. The application-specific users are not shared across other application instances. Application-specific user attributes are defined when creating an application instance. You can add a global user to an application-specific group. Every global user can have application-specific user attributes.

Create Global Users

You can create global users that are available for all applications.

Note: You can create global users only if you store the global users and global groups in the CA Management Database (CA-MDB). If you reference from an external user source the global users are considered read-only.

To create global users

1. Create a SafeGlobalUser object.
2. Set the context by calling the `setContext` method.
3. Set the path where you want to create the user.

If you want users to be created under a folder, and if the folders are already created for users, you can call the `setPath` method to specify the folder name where the user must be stored, for example, `/foldername/username`.

4. Set a unique username and add additional information about the user, such as, `FirstName`, `LastName`, `DisplayName`, `Password`, `Description`, and `JobTitle`.
5. Call the `soInsert` method to add user to the database.

Example: Create global user

The following creates a global user:

```
//Create a SafeGlobalUser object.
SafeGlobalUser gu = new SafeGlobalUser();
//Set the context.
gu.setContext(safecontext);
//Set the path.
gu.setPath("/Asia/JohnDoe");
//Set the username.
gu.setUserName("JohnDoe");
//Add additional information.
gu.setFirstName("John");
gu.setLastName("Doe");
gu.setDisplayName("John Doe");
gu.setPassword("johndoe");
gu.setDescription("Application Administrator");
gu.setJobTitle("Captain");
//Call the soInsert method.
gu.soInsert();
```

Create Application-Specific Users

You can create application-specific users for an application.

To create application-specific users

1. Instantiate a SafeUser object.
2. Set the context by calling the setContext method.
3. Attach to the application using the session.
4. Set the required application-specific attributes.
5. Set the path and define the user by calling the setpath method.
6. Call the soInsert method.

Example: Create application-specific user

The following example creates an application-specific user:

```
//Instantiate a SafeContext Class.  
SafeContext obj = new SafeContext();  
SafeSession session = safecontext.authenticateWithPassword("EiamAdmin", "password")  
//Attach to the application using the session.  
obj.attach("RBC_Hospital", session);  
SafeUser obj2 = new SafeUser();  
obj2.setContext(obj);  
//Set the required application-specific attributes.  
obj2.insertXAttr("memberID","000369");  
//Set path and define the user you want to create.  
obj2.setPath("erdoctor");  
//Insert the Application-specific User.  
obj2.soInsert();
```

Associate Global User with Application-Specific Details

You can associate global user with application-specific details

To associate global user with application-specific details

1. Instantiate the SafeGlobalUser(user) class, which inherits from the SafeStoredObject class.
2. Set the context by calling the setContext method.
3. Set the path and retrieve the user by calling the setpath and soRetrieve methods.

Note: The path set for the application path must be same as the global user path.

4. Associate application-specific information.

Example: Associate application details to a global user

The following example associates application details to a global user:

```
//Instantiate the SafeContext Class.  
SafeGlobalUser obj = new SafeGlobalUser();  
//Set the context.  
obj.setContext(sc);  
//Path should be same as global user.  
obj.setpath("erdoctor");  
obj.soRetrieve()  
//Associate with a application-specific group name.  
obj.addgroup("Staff");  
obj.soModify();
```

Modify Membership

You can modify the user group for a user.

Note: You must have write access for the User or the GlobalUser object that you want to modify.

To modify user group

1. Instantiate the SafeContext Class.
2. Set the context.
3. Retrieve the user using the setpath and soRetrieve methods
4. Modify the user group by calling any of the methods:

addGroup

Adds a user to a particular user group.

delGroup

Removes a user from a particular user group.

getGroupQ

Displays the list of available groups.

clearGroupQ

Removes all the users from the associated group.

5. Call the soModify method to apply the changes.

Example: Modify membership

The following example modifies the user group by adding user to a 'Staff':

```
SafeUser u = new SafeUser();
u.setContext(sc);
//Path should be same as globaluser.UserName
u.setPath("JohnDoe");
//Add the user to a group
u.addGroup("Staff");
//Call the soModify method.
u.soModify();
```

Search Users Using Attributes

You can use filters to search users using attributes. To search for users using attributes, prefix the attribute names with 'A:'.

Note: You must prefix the 'A:' for standard LDAP attributes. For custom attributes, you can provide the attribute name without the prefix. For values to prefix before field type, see [Build Filters to Use in Policies](#) (see page 54).

Example: Search for global users

The following example searches all global users whose Job Title attribute is set to Captain:

```
List filterq = new ArrayList();
filterq.add(new SafeFilter(SafeEnum.Logic.NONE, 0, "A:JobTitle", SafeEnum.OpType.STRING,
    SafeEnum.Oper.EQUAL, "Captain", 0));
List globalUsers = safecontext.searchGlobalUsers(filterq);
Iterator itr = globalUsers.begin();
While(itr.hasNext()){
    SafeGlobalUser gu = (SafeGlobalUser) itr.next();
    System.out.println(gu.getName());
}
```

A list of matching global user objects based on the attributes is returned as a list.

Example: Search for application-specific users

The following example searches all the application-specific users by whose age is greater than 20:

```
List filterq = new ArrayList();
filterq.add(new SafeFilter(SafeEnum.Logic.NONE, 0, "A:Age", SafeEnum.OpType.INT32,
    SafeEnum.Oper.GREATER "20", 0));
List globalUserGroups = safecontext.searchUsers(filterq);
Iterator itr = users.begin();
While(itr.hasNext()){
    SafeUser user = (SafeUser) itr.next();
    System.out.println(user.getName());
}
```

A list of matching application-specific user objects whose age is greater than 20 is returned as a list.

Retrieve a Global User

You can retrieve a global user by the name.

To retrieve a global user

1. Instantiate a SafeContext Class.
2. Set the backend server to the host where CA EEM Policy Server is running.
3. Instantiate the SafeGlobalUser.
4. Set the context.
5. Call the soRetrieveByName method.

All the attributes of the user are populated with the data from server.

Note: You must use the soRetrieveByUserName method if the user's display name (cn) is not same as the User ID.

Example: Retrieve a global user

The following example retrieves a global user:

```
SafeContext safecontext = new SafeContext();
Safecontext.attach("<hostname>");
SafeGlobalUser ug = new SafeGlobalUser();
gu.setContext(safecontext);
gu.soRetrieveByName("JohnDoe");
System.out.println("Global User: " + gug.getJobTitle());
```

Retrieve an Application-Specific User

You can retrieve an application-specific user by name.

To retrieve an application-specific user

1. Instantiate a SafeContext Class.
2. Set the backend server to the host where CA EEM Policy Server is running.
3. Instantiate a SafeUser.
4. Set the context.
5. Call the soRetrieveByName method.

All the attributes of the user are populated with the data from server.

Example: Retrieve an application-specific user

The following example retrieves an application-specific user:

```
SafeContext safecontext = new SafeContext();
Safecontext.attach("<hostname>");
SafeUser u = new SafeUser();
u.setContext(safecontext);
u.soRetrieveByName("JohnDoe");
System.out.println("Application user: " + u.getGroupQ());
```


Delete a User

You can delete an existing user in CA EEM.

To delete any existing user, retrieve the user and call the soRemove method.

Example: Delete a global user

The following example deletes a global user:

```
SafeGlobalUser gu = new SafeGlobalUser();
gu.setContext(safecontext);
gu.soRetrieveByUserName("JohnDoe");
gu.soRemove();
```

Example: Delete an application-specific user

The following example deletes an application-specific user:

```
SafeUser u = new SafeUser();
u.setContext(safecontext);
u.soRetrieveByName("JohnDoe");
u.soRemove();
```

More Information:

[Retrieve a Global User](#) (see page 31)

[Retrieve an Application-Specific User](#) (see page 32)

Chapter 4: Groups

This section contains the following topics:

[Overview](#) (see page 35)

[Create Global User Groups](#) (see page 36)

[Create Application-Specific User Groups](#) (see page 37)

[Search Groups Using Attributes](#) (see page 38)

[Retrieve a Global User Group](#) (see page 39)

[Retrieve a User Group](#) (see page 39)

[Delete a Group](#) (see page 40)

Overview

CA EEM supports global user groups and application-specific user groups. Global User Groups are shared among all application instances. Application-specific groups are accessible only by their owning application instance, created based on the requirements of the application.

You can write access policies against attributes and group memberships of both global and application groups.

Groups in CA EEM are classified as follows:

SafeGlobalUserGroup

GlobalUserGroup are groups that are available for sharing across all application instances registered with CA EEM.

SafeUserGroup

UserGroups are specific to the application instance. The application-specific groups are not shared across other application instances.

Create Global User Groups

Global User Groups are groups that are available for sharing across all application instances registered with CA EEM.

Note: You can create GlobalUserGroup only in the CA Management Database (CA-MDB).

To create global user groups

1. Instantiate the SafeGlobalUserGroup(global user group) class, which inherits from the SafeStoredObject class.
2. Set the context by calling the setContext method.
3. Specify the path name of the Global User Group by calling the setPath method.

The name is set as the name of the group.

Note: You can also provide description to the group by calling the setDescription method.

4. Insert the safeglobalusergroup by calling the soInsert method.

Example: Create global user groups

The following example creates a global user group:

```
//Instantiate the SafeContext Class.  
SafeContext safecontext = new SafeContext();  
//Safecontext through which you are authenticated to the safebackend server.  
SafeGlobalUserGroup gug = new SafeGlobalUserGroup();  
//Set the context.  
gug.setContext(safecontext);  
//Set the path.  
gug.setPath("Engineers");  
//Provide the description.  
gug.setDescription("This global user group is for users who are engineers by profession");  
//Insert the safeglobalusergroup.  
gug.soInsert();
```

Create Application-Specific User Groups

Application-Specific User Groups are specific to the application instance. These groups are not shared across other application instances.

Note: You can create application-specific user groups only in the CA Management Database (CA-MDB).

To create application-specific user groups

1. Instantiate the SafeUserGroup(user group) that inherits the SafeStoredObject class.
2. Set the context by calling the setContext method.
3. Specify the path name of the User Group by calling the setPath method.
The name is set as the name of the user group.

Note: You can also provide description to the group by calling the setDescription method.

4. Insert the application-specific SafeUserGroup by calling the soInsert method.

Example: Create application-specific user group

The following example creates an application-specific user group:

```
//Instantiate the SafeContext Class.  
SafeContext safecontext = new SafeContext();  
//Attach to global space using the session.  
safecontext.attach(null,safesession);  
//Safecontext through which you are authenticated to the safebackend server.  
SafeUserGroup ug = new SafeUserGroup();  
ug.setContext(safecontext);  
//Set the path.  
ug.setPath("Staff");  
//Provide the description.  
ug.setDescription("Staff");  
// Insert the application-specific safeusergroup.  
ug.soInsert();
```

More Information

[Modify Membership](#) (see page 29)

Search Groups Using Attributes

You can use filters to search groups using attributes. To search for groups using attributes, prefix the attribute names with 'A:'

Note: You must prefix the 'A:' for standard LDAP attributes. For custom attributes, you can provide the attribute name without the prefix. For values to prefix before field type, see [Build Filters to Use in Policies](#) (see page 54).

Example: Search for a global users

The following example searches for global users by description:

```
List filterq = new ArrayList();
filterq.add(new SafeFilter(SafeEnum.Logic.NONE, 0, "A:Description", SafeEnum.OpType.STRING,
    SafeEnum.Oper.LIKE, "*engineers*", 0));
List globalUserGroups = safecontext.searchGlobalUserGroups(filterq);
```

A list of all matching global users based on the attributes is returned as a list.

Example: Search for a application-specific user groups

The following example searches for all application-specific users groups by description:

```
List filterq = new ArrayList();
filterq.add(new SafeFilter(SafeEnum.Logic.NONE, 0, "A:Description", SafeEnum.OpType.STRING,
    SafeEnum.Oper.EQUAL, ""engineers"", 0));
List UserGroups = safecontext.searchUserGroups(filterq);
```

A list of all matching application-specific user groups based on the attributes is returned as a list.

Retrieve a Global User Group

You can retrieve a global user group with the name by calling the `soRetrieveByName` method.

To retrieve a global user group

1. Instantiate the `SafeGlobalUserGroup`.
2. Set the context.
3. Call the `soRetrieveByName` method.

All the attributes of the global user groups are populated with the data from server.

Example: Retrieve a global user group

The following example retrieves a global user group:

```
SafeGlobalUserGroup gug = new SafeGlobalUserGroup();
gug.setContext(safecontext);
gug.soRetrieveByName("Engineers");
System.out.println("Description : " + gug.getDescription());
```

Retrieve a User Group

You can retrieve a application-specific user group with the name by calling the `soRetrieveByName` method.

To retrieve a user group

1. Instantiate the `SafeUserGroup`.
2. Set the context.
3. Call the `soRetrieveByName` method.

All the attributes of the user group are populated with the data from server.

Example: Retrieve a user group

The following example retrieves a user group:

```
SafeUserGroup ug = new SafeUserGroup();
ug.setContext(safecontext);
ug.soRetrieveByName("Staff");
System.out.println("Group Membership : " + ug.getGroupQ());
```

Delete a Group

You must retrieve a group to delete any existing group and use the `soRemove` method.

Note: Before deleting a group, ensure it is not referenced by any user or group.

Example: Delete a global user group

The following example deletes a global user group:

```
SafeGlobalUserGroup gug = new SafeGlobalUserGroup();
gug.setContext(safecontext);
gug.soRetrieveByName("Engineers");
gug.soRemove();
```

Example: Delete a user group

The following example deletes a user group:

```
SafeUserGroup ug = new SafeUserGroup();
ug.setContext(safecontext);
ug.soRetrieveByName("Staff");
ug.soRemove();
```

More Information:

[Retrieve a Global User Group](#) (see page 39)

[Retrieve a User Group](#) (see page 39)

Chapter 5: Access Management

This section contains the following topics:

[Policies](#) (see page 41)

[Filters](#) (see page 49)

[Authorization](#) (see page 62)

[SDK Cache](#) (see page 62)

Policies

Overview

CA EEM access policies are rules associated with users to define access to a particular resource of an application or a group. CA EEM determines whether policies apply to the particular user by matching identities, resources, resource classes, and evaluating the filters.

Access policies are divided into six parts:

Identities

Specifies the identities to which policies are applicable.

Calendar

Specifies a time for which the policies will be applicable.

Resources

Specifies the resources on which the policy will apply.

Actions

Specifies the type of access to resources (depends on kind of policy).

Filters

Specifies additional conditions for restricting the policy better.

ResourceClassName

Specifies the type of resource, which will come under a resource class.

More Information

[Filters](#) (see page 49)

[Policy Evaluation](#) (see page 75)

[How Policies Are Evaluated](#) (see page 76)

Types of Policies

Policies are broadly divided into the following six categories based on resource class names:

- Access Policies
- Delegation Policies
- Dynamic User Group Policies
- Obligation Policies
- Event Policies
- Scoping Policies

To classify a policy into any of these categories, you must mention the appropriate resource class name by calling the `setResourceClassName` method of the `SafePolicy` class.

The following table provides details on the actions that can be performed on a policy:

Type of Policy	SafeResourceClass	Actions	Description
Access Policy	User-defined safe resource class	User-defined actions for the resource class	Defines access rules for application-specific resources
Delegation Policy	SafeDelegation	inherit	Allows users to delegate their authority
Dynamic User Group Policy	SafeDynamicUserGroup	belong	Defines application-specific groups and their memberships based on rules
Event Policy	SafeEvent	submit, view	Defines who can submit and view events
Obligation Policy	SafeObligation	FulfillOnGrant, FulfillOnDeny	Defines the obligation that can be carried out
Scoping Policy	SafeObject	read, write	Define who has access to which objects

The following are the available resources for the SafeObject resource class:

- ApplicationInstance
- Calendar
- Policy
- User
- UserGroup
- GlobalUser
- GlobalUserGroup
- Folder
- GlobalFolder
- AppObject
- iPoz
- Notify

Types of Authorization Checks

The following table provides a list of methods to perform authorization check against user-defined policies:

Method	Description
■ SafeContext.authorizeWithSession	Performs authorization check against a single resource, accepting a character string as the identity/session
■ SafeContext.authorizeWithIdentity	
■ SafeContext.authorizeWithSessionDebug	Performs debug authorization check against a single resource, accepting a character string as the identity/session
■ SafeContext.authorizeWithIdentityDebug	
■ SafeContext.authorizeQWithSession	Performs authorization check against a list of resources, using a valid SafeSession as the identity/session
■ SafeContext.authorizeQWithIdentity	
■ SafeContext.processAuthorizationQ	Performs authorization checks for a queue of authorization objects
■ SafeContext.processAuthorizationMatrix	

Example: Perform an authorization check

The following example performs an authorization check to determine whether erdoctor can admit John:

```
SafeContext sc;  
sc.synchronize();  
SafeAuthorizationResult sar = sc.authorizeWithIdentity( "erdoctor", "admit", "patient", "John", null, null );  
System.out.println("Result :" + sar.getResult());  
System.out.println("PolicyName: " + sar.getPolicyName());
```

The results are displayed for the permission check and return a value.

Create, Modify, and Verify Policies

You can create, modify, and verify policies. The following examples describe the process for detail:

- [Anybody can admit John](#) (see page 45)
- [Anybody can admit John or John*](#) (see page 46)
- [Staff can admit anyone except Sam](#) (see page 47)
- [Nobody can admit Sam](#) (see page 48)

How You Create Anybody Can Admit John Policy

The following example describes the process for creating a policy 'AnyBody can admit John'.

Note: To write this policy, you must instantiate a SafePolicy class and set its context.

The policy consists of the following parts:

- Name of the policy: AnyBody can admit John
- Identities: AnyBody (all identities)
 - Note:** If you do not set any identity, the policy applies to all identities.
- Calendar: Any time
 - Note:** If you do not set a time, the policy applies to all times.
 - If you do not set any time, it is applicable at all times.
- ResourceClassName: Patient (since the resource "john" is a "patient")
- Resources: John
- Action: Admit
- Filters: No restrictions (No filters)

Example: Create a policy

The following example creates 'AnyBody Can Admit John' policy:

```
SafeContext sc = new SafeContext();
sc.setBackend("hostname");
SafeSession ss = sc.authenticateWithPassword("username","password");
sc.attach("elsewhere",ss); // you are attaching to elsewhere application
SafePolicy sp = new SafePolicy();
sp.setContext(sc);
sp.setPath("Anybody can admit john"); // name
sp.setResourceClassName("patient"); // resource class name
sp.addResource("John"); // resource
sp.addAction("admit"); // action
sp.solInsert();
```

The authorization check for the resource that starts with John will return 'true'.

More Information:

[Types of Authorization Checks](#) (see page 44)

[How You Create Anybody Can Admit John or John* Policy](#) (see page 46)

How You Create Anybody Can Admit John or John* Policy

The following example describes the process for creating a policy 'Anybody can admit John or John*'.

The policy consists of the following parts:

- Name of the policy: AnyBody can admit John or John*
- Identities: AnyBody (all identities)
Note: If you do not set any identity, the policy applies to all identities.
- Calendar: Any time
Note: If you do not set a time, the policy applies to all times.
- ResourceClassName: Patient (since the resource "john" is a "patient")
- Resources: John, John*
- Action: Admit
- Filters: No restrictions (No Filters)

Example: Create this policy

The following example creates 'Anybody Can Admit John or John*' policy:

```
safeContext.sc
SafePolicy sp = new SafePolicy();
sp.setContext(sc);
sp.soRetrieveByName("AnyBody can admit John or John*");
sp.addResource("John*");
sp.soModify();
```

The authorization check for the resource 'Johnathan' or 'Johnxyz' or any resource that starts with John will return 'true'.

More Information:

[Types of Authorization Checks](#) (see page 44)

[How You Create Anybody Can Admit John Policy](#) (see page 45)

How You Create Staff Can Admit Anyone Except Sam

The following example describes the process for creating a policy 'Staff can admit anyone except Sam'.

The policy consists of the following parts:

- Name of the policy: Staff can admit anyone except Sam
- Identities: Staff Group
- Calendar: Any time

Note: If you do not set a time, the policy applies to all times.
- ResourceClassName: Patient
- Resources: Not mentioned

Note: If you do not set any resource, the policy applies to all resources.
- Action: Admit
- Filters: "requested resource" should "not be equal" to "value Sam"

You will have to restrict the policy using filters. For more information about filters see, [Filters](#) (see page 49).

Example: Create policy using filters

The following example creates 'Staff Can Admit Anyone Except Sam' policy:

```
SafePolicy sp = new SafePolicy();
sp.setContext(safecontext);
sp.setPath("Staff can admit anyone except Sam"); //name
sp.setResourceClassName("patient"); //resource class name
sp.addIdentity(ug:Staff);
sp.addAction("admit"); //action
SafeFilter safefilter0 = new SafeFilter(SafeEnum.Logic.NONE, 0, "req:resource", SafeEnum.OpType.STRING,
    SafeEnum.Oper.NOTEQUAL, "val:Sam", 0);
sp.addFilter(safefilter0); //add the above built filter to safepolicy
sp.solInsert();
```

The results are displayed for the permission check and return a value.

More Information:

[Filters](#) (see page 49)

[Types of Authorization Checks](#) (see page 44)

How You Create Nobody Can Admit Sam Policy

The following example describes the process to create a deny policy for 'Nobody can admit Sam'.

Set the resources in the policy to explicitly deny and flag the 'setExplicitDeny' to 'true', to create a deny policy.

Note: By default, CA EEM denies permission unless a granting access policy is written.

CA EEM authorization evaluation gives priority to deny policies. If an explicit deny policy is set, CA EEM will deny the permission even if a granting policy is available. For more information on Policy Evaluation, see [Policy Evaluation](#) (see page 75).

The policy consists of the following parts:

- Name of the policy: Nobody can admit Sam
- Identities: AnyBody (all identities)
Note: If you do not set any identity, the policy applies to all identities.
- Calendar: Any time
Note: If you do not set a time, the policy applies to all times.
- ResourceClassName: Patient
- Resources: Sam
- Action: Admit
- Filters: None
- Policy: Explicit Deny

Example: Create policy using explicit deny

The following example creates an explicit deny for Nobody Can Admit Sam Policy:

```
SafePolicy sp = new SafePolicy();
sp.setContext(safecontext);
sp.setExplicitDeny(true);
sp.setPath("Nobody can admit Sam"); //name
sp.setResourceClassName("patient"); //resource class name
sp.addResource("Sam");
sp.addAction("admit"); //action
sp.solInsert();
```

The results are displayed for the permission check and return a value.

More Information:

[Create, Modify, and Verify Policies](#) (see page 44)

[Policy Evaluation](#) (see page 75)

[Types of Authorization Checks](#) (see page 44)

Filters

Overview

Filters are attached to the policies to limit the scope of a policy. CA EEM uses filters during the evaluation phase of the policy evaluation process.

Each filter consists of the following components:

- The connector to the previous filter (logic)
- The number of left parentheses before the expression
- A sub-expression, consisting of a left hand side value, operator, and a right hand side value
- The number of right parentheses after the expression

More Information

[Policies](#) (see page 41)

[Policy Evaluation](#) (see page 75)

Build Filters to Use in Searches

You can use filters to manage searches. Filters are similar to the 'where' clause in the databases when used in searches. You can define multiple filters by combining groups using the AND and OR operators.

In the SafeContext class, you can use search methods to locate stored objects.

The following objects are stored objects:

- Application Instances
- Calendars
- AppObjects
- Policies
- User Groups
- Global User Groups
- Global Users
- Users

By default, all the objects inherit from the SafeStoredObject class.

When searching for stored objects, use 'cn' to refer the name of the object.

Example: Search Stored Objects

The following example searches for stored objects:

```
List filterq = new ArrayList();
filterq.add(new SafeFilter(SafeEnum.Logic.NONE, 0, "cn", SafeEnum.OpType.STRING,
    SafeEnum.Oper.LIKE, "app*", 0));
// Build the filter and just call the appropriate search method.
List globalUserGroups = safecontext.searchApplicationInstances(filterq);
```

Note: During searches, in column field, you can prefix attributes names with 'A:' and specify the value in value field.

Following are the available attributes for each one of the stored objects:

Stored Object	Method to Invoke	MappedAttributes
Application Instance	searchApplicationInstances	<ul style="list-style-type: none"> ■ string ApplicationName ■ string Label ■ string Brand ■ string MajorVersion ■ string MinorVersion ■ Date InstallDate ■ string InstallIdentity ■ string InstallHost ■ string InstallHostAddress ■ string InstallHostInfo ■ string History ■ string Translations ■ portableobject ResourceClass ■ string UserAttribute ■ string Description ■ int CacheUpdateTime ■ string ObligationName
App Object	searchAppObjects	
Calendars	searchCalendars	<ul style="list-style-type: none"> ■ Date EffectiveStart ■ Date EffectiveStop ■ portableobject IncludeTimeBlock ■ portableobject ExcludeTimeBlock ■ string Description
Global User Groups	searchGlobalUserGroups	<ul style="list-style-type: none"> ■ string GroupMembership ■ string Description

Stored Object	Method to Invoke	MappedAttributes
Global Users	searchGlobalUsers	<ul style="list-style-type: none">■ string GroupMembership■ boolean Suspended■ string UserName■ string PasswordDigest■ string OldPasswordDigest■ Date PasswordChangeDate■ int IncorrectLoginCount■ Date SuspendedDate■ Date DisableDate■ Date EnableDate■ string Description■ string Comments■ string JobTitle■ string MailStop■ string FirstName■ string MiddleName■ string LastName

Stored Object	Method to Invoke	MappedAttributes
Global Users	searchGlobalUsers	<ul style="list-style-type: none">■ string Alias■ string Department■ string DisplayName■ string HomePhoneNumber■ string WorkPhoneNumber■ string MobilePhoneNumber■ string FaxPhoneNumber■ string EmailAddress■ string Address■ string City■ string State■ string PostalCode■ string Country■ string Office■ string Company■ boolean ChangePasswordNextLogin■ boolean PasswordTimeToWarn■ Date PasswordExpireTime■ boolean■ OverridePasswordPolicy

Stored Object	Method to Invoke	MappedAttributes
Policies	searchPolicies	<ul style="list-style-type: none">■ string Resource■ string Action■ string Identity■ string Calendar■ portableobject Filter■ string ResourceClassName■ string Description■ int PolicyType■ boolean Disabled■ string Delegator■ boolean PreDeployment■ boolean ExplicitDeny■ boolean RegexCompare■ string Label■ portableobject Obligation
User Groups	searchUserGroups	<ul style="list-style-type: none">■ string GroupMembership■ string Description
Users	searchUsers	<ul style="list-style-type: none">■ string GroupMembership■ boolean Suspended

Build Filters to Use in Policies

You can use filters in policies by specifying conditions and providing access policy. You must prefix the specific row values with the column field, different stored objects use with different values.

The following table displays the field name and the value that must be prefixed along with the filters evaluated during policy are presented below:

Column Field Type	Value to Prefix	Filter Evaluation	Attributes
GlobalUserGroup	gug:	gug:{name} evaluates to the value(s) of the SafeGlobalUserGroup object attributes matching {name}	<ul style="list-style-type: none"> ■ string Name ■ string Parent ■ string Path ■ string[] GroupMembership ■ string Description
UserGroup	gu:	gu:{name} evaluates to the value(s) of the SafeGlobalUser object attributes matching {name}	<ul style="list-style-type: none"> ■ string Name ■ string Parent ■ string Path ■ string[] GroupMembership ■ boolean Suspended
User	u:	u:{name} evaluates to the value(s) of the SafeUser object attributes matching {name}	<ul style="list-style-type: none"> ■ string Name ■ string Parent ■ string Path ■ string[] GroupMembership ■ boolean Suspended
Named Attributes	name:	name:{name} evaluates to the value(s) of the named attributeq (namedattrq) (sessionattrq) matching {name}	String value from namedattrq
Session	ses:	ses:{name} evaluates to the value(s) of the session's attributeq (sessionattrq) matching {name}	String value from sessionattrq
Environment	env:	env:{name} evaluates to the value(s) of the environment attributeq (envattrq) matching {name}	String value from envattrq

Column Field Type	Value to Prefix	Filter Evaluation	Attributes
Request	req:	req:{identity action resource when delegator} evaluates to the corresponding values from the permission check request	String data
Value	val:	val:{data} evaluates to the single value of {data}	String data
Dynamic User Group	dug:	dug:Name evaluates to the name(s) of the Dynamic UserGroups the identity belongs to	String Name
Request time	when:	when:{offset} evaluates to req:when offset by the {offset} ({offset} is specified in minutes). Sets the offset in minutes from the current request time. For example, "-360" means 10 hours before the current request, and "60" means one hour after the current request.	String data
Custom variable	var:	var:{name} evaluates to the value(s) of the custom variableq matching {name}	String data
Calculation	calc:	calc:{calculation} evaluates to result of the {calculation}	String data

Column	Field Type	Value to Prefix	Filter Evaluation	Attributes
Global user		gu:	gu:{name} evaluates to the value(s) of the SafeGlobalUser object attributes matching {name}	<ul style="list-style-type: none"> ■ string Name ■ string Parent ■ string Path ■ string[] GroupMembership ■ boolean Suspended ■ string UserName ■ string PasswordDigest ■ string[] OldPasswordDigest ■ Date PasswordChangeDate ■ int IncorrectLoginCount ■ Date SuspendedDate ■ Date DisableDate ■ Date EnableDate ■ string Description ■ string[] Comments ■ string JobTitle ■ string MailStop ■ string FirstName ■ string MiddleName ■ string LastName ■ string Alias ■ string Department ■ string DisplayName ■ string HomePhoneNumber ■ string WorkPhoneNumber ■ string MobilePhoneNumber ■ string FaxPhoneNumber ■ string EmailAddress ■ string[] Address

Column Field Type	Value to Prefix	Filter Evaluation	Attributes
Global user	gu:	gu:{name} evaluates to the value(s) of the SafeGlobalUser object attributes matching {name}	<ul style="list-style-type: none"> ■ string City ■ string State ■ string PostalCode ■ string Country ■ string Office ■ string Company ■ boolean ChangePasswordNextLogin ■ boolean PasswordTimeToWarn ■ Date PasswordExpireTime ■ boolean OverridePasswordPolicy

Structure of a Filter

The following table displays the SafeFilter constructor parameters and their description in order:

Parameter	Description	Overview
logic	Constant integer value from SafeEnum.Logic	Represents the logic between each ordered filter. The available SafeEnum.Logic values are AND, OR, LAST, NONE.
lparens	Number of left parenthesis	Represents the logical grouping of filters. Works together with right parenthesis.
col	The column field	Represents the left side value of the condition.
optype	Constant integer value from SafeEnum.OpType	Operator Type (OpType) is used to set the operator's data type. This data type is used for evaluation of filters.
oper	Constant integer value from SafeEnum.Operator	Operators (oper) are used to compare values. Available operators are like, notlike, equal, notequal, match, notmatch, withinset, notinset, startswith, endswith, greater, greaterequal, less, lessequal, and contains.
val	The value field	Represents the left side value of the condition.

Parameter	Description	Overview
rparens	Number of right parenthesis	Represents the count, number of closing braces to end a group of conditions.

The following are the samples to create filters based on attributes:

- [How to Search on First Name](#) (see page 59)
- [How to Search on First Name and Designation](#) (see page 60)
- [How to Search on First Name, Designation, and Department](#) (see page 61)

Note: For information on how to create Filters using CA EEM web interface, see *Online Help*.

Example: How to Search on First Name

The following is an example to create a filter to search for condition where FirstName equal to 'ABC'.

Example: Search for condition where FirstName equal to 'ABC'

The following example searches on a condition where FirstName is equal to 'ABC':

```
new SafeFilter(SafeEnum.Logic.NONE, 0, "A:FirstName", SafeEnum.OpType.STRING,
SafeEnum.Oper.EQUAL, "ABC", 0);
```

More Information:

[Structure of a Filter](#) (see page 58)

[Example: How to Search on First Name and Designation](#) (see page 60)

[Example: How to Search on First Name, Designation, and Department](#) (see page 61)

Example: How to Search on First Name and Designation

The process to build a filter to search for users based on job title involves three steps:

- Build a filter to search on first name field, see [Search on First Name](#) (see page 59)
- Build a filter to search on designation field
- Combine the filters using AND logic.

Example: Search on designation

The following example searches based on designation:

```
new SafeFilter(SafeEnum.Logic.NONE, 0, "A:JobTitle", SafeEnum.OpType.STRING, SafeEnum.Oper.EQUAL, "Manager", 0);
```

Note: To combine two filters you can use the 'AND' logic instead of 'NONE' logic for the designation filter.

Example: Combine filters and search for users based on first name and designation

The following example searches for users based on first name and designation:

```
List filterq = new ArrayList();
filterq.add(new SafeFilter(SafeEnum.Logic.NONE, 0, "A:FirstName", SafeEnum.OpType.STRING,
    SafeEnum.Oper.EQUAL, "ABC", 0));
filterq.add(new SafeFilter(SafeEnum.Logic.AND, 0, "A:JobTitle", SafeEnum.OpType.STRING,
    SafeEnum.Oper.EQUAL, "Manager", 0));
List globalUsers = safecontext.searchGlobalUser(filterq);
```

A list of users with matching designation is displayed.

More Information:

[Structure of a Filter](#) (see page 58)

[Example: How to Search on First Name](#) (see page 59)

[Example: How to Search on First Name, Designation, and Department](#) (see page 61)

Example: How to Search on First Name, Designation, and Department

To search for users based on first name, designation, and department involves three steps:

- Build a filter to search on first name field, see [Search on First Name](#) (see page 59).
- Build a filter to search on designation field, see [Search on First Name and Designation](#) (see page 60).
- Build a filter to search on department field.

Example: Search for department

The following example searches for department:

```
new SafeFilter(SafeEnum.Logic.OR, 0, "A:Department", SafeEnum.OpType.STRING, SafeEnum.Oper.EQUAL, "Finance", 1 );
```

Note: Ensure the opening and closing braces in count are set properly.

Example: Search users based on first name, designation, and department

The following example searches for users based on first name, designation, and department:

```
List filterq = new ArrayList();
filterq.add(new SafeFilter(SafeEnum.Logic.NONE, 0, "A:FirstName", SafeEnum.OpType.STRING,
    SafeEnum.Oper.like, "Sam*", 0));
filterq.add(new SafeFilter(SafeEnum.Logic.AND, 0, "A:Designation", SafeEnum.OpType.STRING,
    SafeEnum.Oper.EQUAL, "Manager", 0));
filterq.add(new SafeFilter(SafeEnum.Logic.AND, 0, "A:Department", SafeEnum.OpType.STRING,
    SafeEnum.Oper.EQUAL, "Finance", 0));
List globalUsers = safecontext.searchGlobalUser(filterq);
```

A list of users with matching designation and department are displayed.

More Information:

[Structure of a Filter](#) (see page 58)

[Example: How to Search on First Name](#) (see page 59)

[Example: How to Search on First Name and Designation](#) (see page 60)

Authorization

You can check access rights for user-defined policies by calling the following methods:

- `Safe::Context::authorizeWithSession`
- `Safe::Context::authorizeWithIdentity`
- `Safe::Context::authorizeWithSessionDebug`
- `Safe::Context::authorizationWithIdentityDebug`
- `Safe::Context::authorizeQWithSession`
- `Safe::Context::authorizeQWithIdentity`
- `Safe::Context::processAuthorizationQ`
- `Safe::Context::processAuthorizationMatrix`

Additionally, you can check access for CA EEM stored objects by calling the following methods:

- `Safe::StoredObject::canContextRead`
- `Safe::StoredObject::canContextWrite`
- `Safe::StoredObject::canIdentityRead`
- `Safe::StoredObject::canIdentityWrite`

SDK Cache

CA EEM will cache all the policies, calendars, sessions, and user groups for every 30 seconds, by default. If you want to change the default cache update time, call the `setCacheUpdateTime` method.

You can manually update the cache by calling the `Safe::Context::synchronize` method.

Note: The cache update does not include sessions by default. To set cache update to perform a full synchronization, you must call the `setCacheUpdateSessionsAtSync` method.

Session

CA EEM associates every user with a safesession. It has two kinds of sessions:

Authenticated sessions

Session maintains the user details such as name and groups, along with the user authentication information. You can generate authenticated sessions by calling any of the following methods:

- `authenticateWithPassword`
- `authenticateWithCertificate`
- `authenticateWithArtifact`
- `authenticateWithNative`
- `authenticateWithDigest`
- `authenticateWithCredentials`

Note: Each authentication call returns a session object. You can use the session object to determine the authenticated users.

Unauthenticated sessions

Sessions maintains only the user details such as name and groups information. These sessions are used only during authorization checks. Unauthenticated sessions are generated by calling any of the following methods:

- `authorizeWithIdentity`
- `authorizeQWithIdentity`
- `authorizeWithIdentityDebug`

Note: The default size of unauthenticated session's cache is ten. You can modify the default size by calling the `setCacheUnauthenticatedQSize` method.

Chapter 6: Authentication

This section contains the following topics:

[Pluggable Authentication Module](#) (see page 65)

[PassTicket](#) (see page 65)

[Kerberos](#) (see page 70)

Pluggable Authentication Module

CA EEM allows client applications to verify the user authentication from a UNIX system. The CA EEM SDK calls communicate with the Pluggable Authentication Module (PAM) APIs to verify the authenticity.

Client applications can call the `authenticateWithPam` method to verify the authentication. The PAM APIs will communicate with the PAM service running on the UNIX system and return a boolean value.

Note: The authentication verification calls are directed to the UNIX system and not to the CA EEM Server.

Example: Authentication verification with PAM

The following example performs the authentication verification using PAM.

```
boolean Safe::Context::authenticateWithPam(const char *username, const char *password, const char *service,
Safe::Error &ee);
```

PassTicket

CA EEM supports single sign-on between client applications and mainframe applications by generating a PassTicket. A PassTicket is a system-generated password used to authenticate the client applications to access a Mainframe application without re-authentication.

More Information:

[Client Applications](#) (see page 14)

Prerequisites for Single Sign-On

Before you enable single sign-on, you must perform the following tasks:

- Configure the mainframe system
- Configure the client application in CA EEM Server

Configure the Mainframe System

You must configure the mainframe application profile to define the record ID (recid) and the Secure Sign-on (SSKEY) fields.

Record ID

Defines the 1 - 26 character profile name of the corresponding application. The record ID is determined by four distinct combinations of an application name, group name, and user ID. The combinations are as follows:

- application.group.userid
- application.userid
- application.group
- application

Secure Sign-on Key

Defines the 16-character hexadecimal representation of the eight-byte encryption key for the mainframe application.

Configure the Client Application in CA EEM

You must perform the following tasks to configure the client application in CA EEM for single sign-on:

- Configure PassTicket
- Create a Scoping Policy

Configure PassTicket

You must configure the recid and SSKEY values, defined in the mainframe application, in the CA EEM before the client applications can communicate with the Mainframe application.

Note: You can also configure the recid and SSKEY values in CA EEM Server PassTicket Configuration window of the user interface. For more information, see the *Online Help*.

To configure a PassTicket, you must call the configurePassTicket method.

Example: Configure PassTicket

The following example configures a PassTicket:

```
//Instantiate the SafeContext Class.
Safe::Context sc;
//Set the backend to the host where Server is running.
sc.setBackend("localhost");
//Provide the authentication details
std::string username = "EiamAdmin";
std::string password = "Eiampassword";
//Attach to an application space.
std::string applicationname = "ABC Application";
Safe::Session *session = sc.authenticateWithPassword(username.c_str(), password.c_str(), ee);
if(!sc.attach(applicationname.c_str(), session, ee) ) {
    showerror(ee);
    return false;
}
//Provide any additional information, if required
//Configure PassTicket
std::string profile = "CICS";
std::string key = "0123456789ABCDEF";
if ( !sc.configurePassTicket(profile.c_str(),key.c_str(),session,ee) ) {
    printf("unable to configure passticket");
    showerror(ee);
}
else {
    printf("passticket configured successfully\n");
}
}
```

You will receive a confirmation message if the PassTicket is successfully generated.

Create Scoping Policy

You must create a scoping policy for the client application in CA EEM server to configure or generate a PassTicket.


To configure scoping policy

1. Log into the CA EEM Server as EiamAdmin for the application you want to generate PassTicket.

The CA EEM home page appears.

2. Click Manage Access Policies, Explicit Grant, New Scoping Policy .

The New Scoping Policy page appears.

3. In Access Policy Configuration pane, select PassTicket in the Resources drop-down, and click Add Resource .

The PassTicket resource is added.

4. Set the required actions.

Read

Specifies the user can generate PassTicket for the attached application.

Write

Specifies the user can configure CA EEM for PassTicket.

Click Save.

The policy creation confirmation message appears.

Generate PassTicket

You can generate a Passticket when the client applications want to communicate with the mainframe application.

Note: Every client application registered with CA EEM must generate a separate PassTicket.

To generate a PassTicket, you must call the generatePassTicket method.

Example: Generate PassTicket

The following example generates a PassTicket:

```
//Instantiate the SafeContext Class.
Safe::Context sc;
//Set the backend to the host where Server is running.
sc.setBackend("localhost");
//Provide authentication details
std::string username = "EiamAdmin";
std::string password = "Eiampassword";
//Attach to an application space.
std::string applicationname = "ABC Application";
//Call the authenticateWithXXX to verify the authenticity.
Safe::Session *session = sc.authenticateWithPassword(username.c_str(), password.c_str(), ee);
if(!sc.attach(applicationname.c_str(), session, ee)) {
showerror(ee);
return false;
}
//Provide any additional information, if required
//Generate PassTicket
std::string identity = "bacan01";
char passticket[9] = {0};
if ( !sc.generatePassTicket(passticket,(char *) identity.c_str(),session,ee) ) {
printf("unable to generate passticket\n");
showerror(ee);
}
else {
printf("PassTicket: %s\n",passticket);
}
```

You will receive a confirmation message if the PassTicket is successfully generated.

How Applications Communicate with Mainframe Systems

The following process describes how a client application authorized by CA EEM communicates with a mainframe system:

1. Client application sends a request to CA EEM Server through CA EEM SDK to generate a PassTicket.
2. CA EEM Server generates a PassTicket based on the user ID, application profile, secret key, and timestamp values of the mainframe application.
3. CA EEM Server sends the PassTicket to the client application through CA EEM SDK.
4. Client application uses the PassTicket to authenticate with the mainframe application.

Kerberos

CA EEM lets client applications perform authentication for users stored in a UNIX-based Key Distribution Center (KDC) using Kerberos authentication. Client applications can use Kerberos authenticate with a UNIX KDC in two ways:

- [Ticket Granting Ticket \(TGT\)](#) (see page 70)
- [Kerberos Principal Name](#) (see page 74)

Using TGT

The Ticket Granting Ticket (TGT) is an encrypted identification file. The TGT file that contains the session key, expiration date, and user's IP address.

To use Kerberos authentication using TGT, you must perform the following tasks:

- Setup CA EEM Server
- Setup client system

Prerequisites

Following are the client and server requirements for Kerberos authentication using TGT.

Server

Following are the operating system requirements for Kerberos authentication on the server for using TGT:

- Red Hat Enterprise Linux 4 and 5 with `krb5-server<version>.rpm` package
- AIX 5.2 or 5.3 with `krb5.server`
- Solaris 10

Client

Following are the operating system requirements for Kerberos authentication on the client for using TGT:

- AIX 5.2 or 5.3 with `krb5.client` package
- Solaris 10

Setup CA EEM Server

To setup the CA EEM Server for Kerberos authentication, you must perform the following tasks:

- Add the service principal to Kerberos KDC
- Create a keytab file and add it

Add Service Principal name

You should add the service principal to the Kerberos KDC.

The service principal will have the following format:

```
<service name>/<hostname>@<realm>
```

Note: CA EEM has eiam as its service name and is in lowercase.

Example: If the CA EEM Server is running on a host server.ca.com with realm CA.COM, then the service principal will be:

```
eiam/server.ca.com@CA.COM
```

Note: The host name must be in lowercase and the realm name must be in uppercase.

To add the service principal to the Kerberos KDC:

1. Login as administrator.
2. Add the Service Principal to the Kerberos KDC.

Example: Add Service Principal Name

The following example adds a Service Principal Name to the Linux Kerberos KDC.

```
[root@materkdc root]# kadmin.local
kadmin.local: addprinc eiam/eiambuild-l-v4.ca.com@CA.COM
Enter password for principal "eiam/EEMserver.ca.com@CA.COM": <Enter Password>
Re-enter password for principal "eiam/EEMserver.ca.com@CA.COM": <Enter Password>
Principal "eiam/EEMserver.ca.com@CA.COM" created.
```

You receive a service principal creation confirmation message.

Create Keytab File

After you add the service principal to the Kerberos database, you should create a keytab file for the service principal.

To create a keytab file on Linux, enter the following command:

```
kadmin.local: ktadd -k EEMserver.ktab eiam/EEMserver.ca.com@CA.COM
```

A keytab creation confirmation message appears.

Add Keytab File

You must add the Kerberos keytab file to the default operating system keytab file of the CA EEM Server as follows:

Note: If the key tab file is not in the default location, you must export the KRB5_KTNAME environment variable. For example, export KRB5_KTNAME=FILE:/root/eiam.keytab.

```
[root@eiambuild-l-v4 root]# ktutil
ktutil: rkt eiam_CACOM.ktab
ktutil: wkt /etc/krb5.keytab
```

The Kerberos keytabs are added to the default keytab file on the operating system and the CA EEM server is ready to validate the users using Kerberos tickets.

Note: Install CA EEM Server, if it is not already installed. If it is already installed and iGateway is running then restart iGateway service.

Setup Client

To authenticate the client with Kerberos, configure the krb5.conf file existing on the computer where CA EEM Server is installed and generate a TGT for the user principal in realm.

Note: For information on how to configure krb5.conf file, see the Kerberos documentation.

Generate TGT

TGT is generated when the user is authenticated by the KDC subsystem of Kerberos.

To generate a TGT for a userprincipal in realm, provide the user principal details with password as follows:

```
#kinit -p userprincipal@realm
Password for userprincipal@realm:<Password>
```

After you generate a TGT, use the authNative method to authenticate with CA EEM Server.

Note: For an existing TGT of a userprincipal, view the TGT using the klist command.

Using Kerberos Principal and Password

You can authenticate CA EEM using Kerberos principal and password.

Prerequisites

Following are the operating system requirements for Kerberos authentication on the server for using the principal name and password:

- Red Hat Enterprise Linux 4 and 5
- AIX 5.2 or 5.3

Note: The computer on which CA EEM server must be installed must have Kerberos.client package installed. This package exists in AIX expansion pack.

- Solaris 10 with kernel patch level 120011-14 or higher

Configure Server

To authenticate CA EEM using Kerberos principal and password, you must modify the krb5.conf file existing on the computer where CA EEM Server is installed.

Note: For information on how to edit krb5.conf file, see the Kerberos documentation.

Chapter 7: Policy Evaluation

This section contains the following topics:

[Overview](#) (see page 75)

[How Policies Are Evaluated](#) (see page 76)

[Gathering Identity Attributes](#) (see page 77)

[Policy Matching](#) (see page 78)

Overview

CA EEM performs policy evaluation by calling the `Safe::Context::authorize` method. This method is invoked with the following parameters:

- Identity to check
- Resource class name
- Resource name
- Action requested
- Queue of named attributes

How Policies Are Evaluated

Policies are evaluated in the following process:

1. Check for explicit denies:
 - a. Match for explicit denies.
 - b. Evaluate matched policy filters.
 - c. In case of explicit deny, stop checking, and return a denied recommendation specifying the policy.
2. Check for explicit grants:
 - a. Match for explicit grants policies.
 - b. Evaluate matched policy filters.
 - c. In case of explicit grants, stop checking, and return a granted recommendation specifying the policy.
3. Check for delegated authority:
 - a. Match/evaluate the delegated authority. For each delegator, find a grant with no explicit deny.
 - b. For each delegator, repeat step 1 and search for explicit grants.
 - c. If a grant was returned by delegation, return a granted recommendation specifying the policy and the delegator chain.
4. Calculate obligations for this access check:
 - a. Add the following attributes to the ones passed in the authorization call:
 - PolicyName, the name of the obligation policy that caused the response
 - DelegationChain, the name of the delegation chain returned
 - b. Match and evaluate each SafeObligation as follows:
 - ResourceClass set to SafeObligation.
 - Resource name set to {action} + "/" + {original resource class} + "/" {original resource name}.
 - Action set to FulfillOnGrant (if the authorization results in a grant), or FulfillOnDeny (if the authorization results in a deny).

- c. Do the following for each matching or evaluating SafeObligation policy:
 - Append each obligation to the authorization results.
 - Calculate the values of the obligation attributes and append them to the authorization results.

Note: Applications must handle the obligations returned from an authorization check. The application should not grant or deny access until and unless the obligations could not be performed.

5. Return a denied recommendation, in case of no matches.

Note: All the policies are not evaluated for every request. Since CA EEM supports explicit policies (explicit grant or explicit deny), policy evaluation is performed only at the first instance.

Gathering Identity Attributes

Identity attributes are collected and stored in the identity's session as a local cache. Identity attributes are used during evaluation.

Identity attributes are collected as a cache for the following actions:

- Authentication (Safe::Context::authenticateXXX)
- First invocation of Safe::Context::authorizeWithIdentity (an unauthenticated session)
- Attach application, all identity attributes contained in the cached sessions rebuilt/re-synchronized
- Detach application, all identity attributes contained in the cached sessions are destroyed

Assembling Environment Information

Environment attributes are used during evaluation. Environment attributes can be modified by invoking the following methods:

- Safe::Context::insertEnvAttr
- Safe::Context::removeEnvAttr

Policy Matching

Evaluating Matching Algorithm

CA EEM uses an algorithm to match policies against a request. A policy match can be against an identity, groups, action, resourceclassname, resource, and time. A policy is matched and returns a value 'true' only if all of the following conditions are satisfied:

- Policy must not be disabled.
- Policy must not be pre-deployed (If it is pre-deployed, one of the policy's labels must match a label in `Safe::Context::getPreDeploymentLabels`).
- Policy must match the resource name.
- Policy must match the action or must contain no actions.
- Policy must match the identity (user/group) or must contain no identities.
- Policy must be within the calendar's scope or must contain no calendars.
- Policy must match the resource ("like" expression) or must contain no resources.

Note: Matching does not check the policy's filters. Filters are matched during evaluation.

How the Best Match Algorithm is Evaluated

The best match algorithm is evaluated in the following process:

- Determine the characters matched (total number of non-asterisk characters) in the policy's resource name mask
- Determine the number of asterisks in the policy's resource name mask
- Policies with the matching characters and least asterisks are retained
- Empty masks ("", "*", and "**") all evaluate to 0 matching characters, 0 asterisks

The following table displays how four policies with resource name masks of "PAY", "PAY*", "*PAY", "*PAY*", "P*", and "*" match:

Resource Name	Policies Matched	Matching Characters	Asterisks
PAY123	PAY*	3	1
PAY	PAY	3	0
1PAY1	*PAY*	3	2
PAYPAY	PAY*, *PAY	3	1
P1AY	P*	1	0
QAY	*	0	1
123PAY	*PAY	3	1

Best Match Handling for Regular Expression Policies

Regular expression (Regex) policies are policies that are treated as regular expressions. The regex policies have 'regexcompare' flag enabled.

The best match algorithm for regular expression policies are evaluated in the following process:

- Start with matching character count set to the length of the resource name mask, and reset the asterisk count zero
- If the last character of the resource name mask is '\$', decrement matching character count, else increment asterisk count
- If the first character of the resource name mask is '^', decrement matching character count, else increment asterisk count
- For each "." found in the resource name mask, decrement matching character count by two, and increment asterisk count
- For each "?" found in the resource name mask, decrement matching character count by two, and increment asterisk count
- For each "+" found in the resource name mask, decrement matching character count by two, and increment asterisk count
- For each non-escaped backslash ('\'), decrement matching character count

The following table displays how regex policies with resource name masks of "PAY", "PAY*", "*PAY", "*PAY*", "P*", and "*" match:

Resource Name	Policies Matched	Matching characters	Asterisks
PAY123	^PAY	3	1
PAY	^PAY\$	3	0
1PAY1	PAY	3	2
PAYPAY	^PAY, PAY\$	3	1
P1AY	^P	1	1
QAY	.*	0	0
123PAY	PAY\$	3	1

Policy Filter Evaluation

CA EEM evaluates filters only if no 'empty filter' policies matched.

Policies are evaluated against the identity, session, environment, and named attributes. Each filter in the policy is evaluated based on order/parentheses/logic as follows:

- calculate list of "col" values
- calculate list of "val" values
- for each col/val pair, apply "oper" based on "optype"

Note: If any policies containing no filters match the authorization request, the evaluation step is not invoked.

Calculating lists of values:

- val:{data} evaluates to the single value of {data}
- env:{name} evaluates to the value(s) of the environment attributeq (envattrq) matching {name}
- u:{name} evaluates to the value(s) of the Safe::User object attributes matching {name}
- gu:{name} evaluates to the value(s) of the Safe::GlobalUser object attributes matching {name}
- ug:{name} evaluates to the value(s) of the Safe::UserGroup object attributes matching {name}
- gug:{name} evaluates to the value(s) of the Safe::GlobalUserGroup object attributes matching {name}
- dug:Name evaluates to the name(s) of the Dynamic UserGroups the identity belongs to
- ses:{name} evaluates to the value(s) of the session's attributeq (sessionattrq) matching {name}
- name:{name} evaluates to the value(s) of the named attributeq (namedattrq) (sessionattrq) matching {name}
- req:{identity|action|resource|when|delegator} evaluates to the corresponding values from the permission check request
- when:{offset} evaluates to req:when offset by the {offset} ({offset} is specified in minutes)
- calc:{calculation} evaluates to result of the {calculation}

Delegated Authority Evaluation

CA EEM evaluates delegated authority if no grants are found during policy match, evaluation, and if the request was not already for the SafeDelegation resource class name.

CA EEM invokes `Safe::Context::authorizeWithSession` with the following attributes:

- Session set to the request session
- Resourceclassname set to `SafeDelegation`
- Resource set to the original action + "/" + resourceclassname + "/" + original resource

Example:

```
[read|write]/SafeObject/[Calendar|Policy|User|UserGroup|GlobalUser|GlobalUserGroup|ApplicationInstance|AppObject|iPoz]action/application-resource-class-name/resourceName
```

- Action set to `inherit`
- The original named attribute queue, with an additional named attribute added: `"DelegationLevel"`, set to the depth of the delegation (starting at 1)

If access is granted, the 'Delegator' identity of the policy is retrieved. If an authorization request is not submitted against this identity (prevents infinite recursion), the original authorization request is re-submitted using the new identity.

If the (possibly recursive) authorization request comes with a GRANT, then access to the object will be allowed.

How Delegated Policies Are Evaluated

CA EEM performs a policy evaluation using delegated authority on an identity's authorization request for a specific resourceclass, action, and resource, in the following process:

1. CA EEM evaluates permissions for the identity based on the specified resourceclass, resource, and action.
2. If a grant is not found, CA EEM searches for a Delegated Authorization by issuing an authorization request for the following:
 - identity "{original identity}"
 - resource class "SafeDelegation",
 - resource name "{original action}/{original resource class}/{original resource name}"
 - action "inherit"
 - the original named attribute queue, with an additional named attribute:
 - "DelegationLevel" set to the depth of the delegation level (starting at 1)
3. For each matching Delegated Authorization retrieved, CA EEM issues an authorization request for:
 - identity "{Delegator}" (from the Delegated Authority Policy)
 - resource class "{original resource class}"
 - resource "{original resource}"
 - action "{original action}".
 - the original named attribute queue
4. If a grant is found, access is granted to the original identity.

How Obligations Are Calculated

After an authorization check is complete, CA EEM determines if any obligations must be attached to the authorization result in the following process:

1. Add two named attributes to the ones passed in the authorization call:
 - a. PolicyName, set to the name of the policy that caused the response (may be empty on a default deny)
 - b. DelegationChain, set to the delegation chain returned (may be empty)
2. For each SafeObligation policy, match and evaluate as follows:
 - a. ResourceClass set to SafeObligation
 - b. Resource name set to {action} + "/" + {original resource class} + "/" + {original resource name}
 - c. Action set to "FulfillOnGrant" if the authorization check resulted in a Grant, or "FulfillOnDeny" if the authorization check resulted in a Deny.
3. For each matching/evaluating SafeObligation Policy:
 - a. Append each attached obligation to the authorization results
 - b. Calculate the values of the obligation attributes, and append these to the authorization results (if unable to calculate, attach an empty result attribute).

Chapter 8: Exception Handling

This section contains the following topics:

[Overview](#) (see page 85)

[Safe Exception](#) (see page 86)

[Safe Authorization Exception](#) (see page 88)

[Safe BackendServer Exception](#) (see page 88)

[Safe Password Exception](#) (see page 89)

Overview

Exceptions indicate unusual error conditions that occur during the execution of an application. When you call an object method, and an 'exceptional' event occurs (such as being unable to access a file or network resource), the method can stop execution, and 'throws' an exception. When exception is thrown, it passes an object (the exception), back to the calling code. The code can then handle the event, and deal with the condition.

CA EEM supports four types of exception handling:

- Safe Exception
- Safe Authorization Exception
- Safe BackendServer Exception
- Safe Password Exception

Safe Exception

The Safe Exception is the generic exception of CA EEM. Most of the exceptions thrown are using the safe exception. Exceptions are thrown when you encounter issues while performing the following tasks:

- Insert an object
- Retrieve an object
- Modify an object
- Delete an object
- Authentications
- Ping the backend server
- Add or remove the folder or global folders
- Synchronization
- Change password and so on

The hierarchy of the safe exception classes in Java is as follows:

```
java.lang.Object
+- java.lang.Throwable
+- java.lang.Exception
+- com.ca.eiam.SafeException
```

Methods such as `getException` and `getExceptionString` provide details on the kinds of exceptions that are tagged in the Safe API layer. All these exceptions are available as `SafeEnum.ErrorCode`.

The following table displays all the possible Safe API exceptions with the safe enumeration return values and their descriptions:

Return Value	Description
<code>int SUCCESS = 0;</code>	<code>EE_SUCCESS</code> Success
<code>int EXCEPTION = 1;</code>	<code>EE_EXCEPTION</code> Exception
<code>int NOCREDS = 2;</code>	<code>EE_NOCREDS</code> No Credentials
<code>int NOBACKEND = 3;</code>	<code>EE_NOEIAMNODES</code> No SafeNodes Defined
<code>int SPONSORERROR = 4;</code>	<code>EE_SPONSORERROR</code> iSponsor Error
<code>int NOTATTACHED = 5;</code>	<code>EE_NOTATTACHED</code> Not Attached to Repository
<code>int NOTFOUND = 6;</code>	<code>EE_NOTFOUND</code> Object Not Found

Return Value	Description
int EXISTS = 7;	EE_EXISTS Object Already Exists
int BADOBJECT = 8;	EE_BADOBJECT Bad Object
int AUTHFAILED = 9;	EE_AUTHFAILED Authentication Failed
int EIAMUNREACHABLE = 10;	EE_EIAMUNREACHABLE Backend Unreachable
int POZERROR = 11;	EE_POZERROR Repository Error
int SESSIONEXPIRED = 12;	EE_SESSIONEXPIRED Session Expired
int ALREADYATTACHED = 13;	EE_ALREADYATTACHED Already Attached
int MAXSIZEEXCEEDED = 14;	EE_MAXSIZEEXCEEDED Max Search Size Exceeded
int CHANGEPASSWORD = 15;	EE_CHANGEPASSWORD User needs password changed
int TRYAGAIN = 16;	EE_TRYAGAIN Try again
int MAINTENANCE = 17;	EE_MAINTENANCE Backend down for maintenance
int NOTALLOWED = 18;	EE_NOTALLOWED Operation not allowed
int PW_TOOSHORT = 19;	EE_PW_TOOSHORT Password too short
int PW_TOOLONG = 20;	EE_PW_TOOLONG Password too long
int PW_BADMIX = 21;	EE_PW_BADMIX Password doesn't contain enough special characters
int PW_MATCHESID = 22;	EE_PW_MATCHESID Password matches account name
int PW_TOOSOON = 23;	EE_PW_TOOSOON Password cannot be changed yet
int PW_REUSED = 24;	EE_PW_REUSED Password already used
int PW_USERLOCKED = 25;	EE_PW_USERLOCKED Account locked
int PW_REPETITION = 26;	EE_PW_REPETITION Password has too many repeating chars
int PW_EXPIRED = 27;	EE_PW_EXPIRED Password has expired

Return Value	Description
int REFERENCED = 28;	EE_REFERENCED Object still referenced
int LAST = REFERENCED;	To obtain the highest error code value

Safe Authorization Exception

The Safe Authorization Exception is inherited from the SafeException. Exceptions caused during the authorization calls are thrown as SafeAuthorizationExceptions. These exceptions occurs when you try to authorize against a null session or when an identity passed for authorization is either null or empty.

The hierarchy of the safe authorization exception classes in Java is as follows:

```
java.lang.Object
+- java.lang.Throwable
+- java.lang.Exception
+- com.ca.iam.SafeException
+- com.ca.iam.SafeAuthorizationException
```

Safe BackendServer Exception

The Safe BackendServer Exception is inherited from the SafeException. This exception is thrown when you try to make calls to CA EEM without setting the backend server.

The hierarchy of the safe backendserver exception classes in Java is as follows:

```
java.lang.Object
+- java.lang.Throwable
+- java.lang.Exception
+- com.ca.iam.SafeException
+- com.ca.iam.SafeBackendServerException
```


Safe Password Exception

The Safe Password exception is inherited from `SafeException`. You may receive this exception during authentication calls, change password, change password for an identity, and `unlockUser` method calls. You will receive the Safe Password exception even when the user is locked, an incorrect password is provided, or if the password is expired.

The hierarchy of the safe password exception classes in Java is as follows:

```
java.lang.Object
+- java.lang.Throwable
+- java.lang.Exception
+- com.ca.eiam.SafeException
+- com.ca.eiam.SafePasswordException
```


Chapter 9: Identity Management

This section contains the following topics:

[Administration Methods](#) (see page 91)

[Configure Externally Generated Certificates](#) (see page 96)

[Enable Trace on CA EEM Server](#) (see page 96)

[Enable Trace on CA EEM SDK](#) (see page 97)

[Dynamic user groups](#) (see page 97)

[How Offline Authentication Works](#) (see page 98)

Administration Methods

CA EEM Server enables the following features when you configure CA EEM to store global users and global groups in the CA Management Database (CA-MDB):

- [Administering Global Users, Groups, and Folders](#) (see page 92)
- [Applying password policies](#) (see page 93)
- [Identity self-administration](#) (see page 95)

Administering Global Users, Groups, and Folders

Note: If CA EEM is configured to reference global users from an external directory, the Global Users and Global User Groups are read-only. You cannot perform insert, modify, or delete operations on the Global Users and Global User Groups when referenced from an external directory.

CA EEM lets you assign user privileges to perform insert, modify, and delete actions on global users, groups and folders. You can also perform the following actions on global users:

Suspend

Specifies the user is suspended, and cannot login.

Reset Password

Prompts to reset the user's password.

Override Password Policies

Specifies whether to permit the user to have passwords that do not meet the password policy.

Modify Global Group Membership

You can control the Global user group membership by using the GroupQ in the Global User Object and Global User Group objects by calling the following methods:

- `getGroupQ`
- `addGroup`
- `delGroup`
- `clearGroupQ`

Note: If you want to modify Global User's group membership, you must have write access to the Global User object and Global User Group.

Applying Password Policies

You can set the password policies for internal global users. Password policies are applied only when the users change their passwords.

Note: Password policies can administered only by the administrator.

To set the password policies, call the `SafeContext.pozConfigure` method using the following `pozConfigure` parameters:

nameq

Specifies an `istringQ` that specifies the name of the field.

valueq

Specifies an `istringQ` that specifies the value of the corresponding field.

ee

Specifies error object, if any.

Note: This parameter is specific to C++ environment.

Example: Change maximum password length

The following example changes the maximum password of length.

```
obj = new SafeContext();
obj.setBackend(backend);
SafeSession session = obj.authenticateWithPassword("EiamAdmin", adminPwd);
obj.attach("RBC_Hospital", session);
List nameq = new ArrayList();
List valueq = new ArrayList();
nameq.add("PwMaxLength");
valueq.add("5");
obj.pozConfigure(nameq, valueq);
```

The maximum password length is now set to five.

Password Policy Attributes

You can use the password policy attributes if the global users and global usergroups are stored internally.

Note: The password policy attributes cannot be applied if global users and global user groups are stored in an external directory.

PwUnlockAllowed

Specifies if the users can unlock their accounts.

PwMinLength

Defines the minimum length in characters for a password. Zero indicates there is no minimum length.

PwMaxLength

Defines the maximum length in characters for a password. Zero indicates there is no maximum length.

PwMinNumeric

Defines the minimum number of numeric characters in a password. Zero indicates there is no minimum length.

PwAllowId

Specifies if the users can use their 'User Name' as their password.

PwMinAge

Defines the minimum number of days before a password can be changed. Zero indicates there is no minimum number of days.

PwMaxAge

Defines the maximum number of days before a password must be changed. Zero indicates there is no maximum number of days.

PwReuseCount

Defines the number of new passwords before a password can be re-used. Zero indicates this rule is not applicable.

PwFailureCount

Defines the number of bad authentication attempts before the Global User account is locked. Zero indicates this rule is not applicable.

PwWarningAge

Defines the number of days before the user is prompted to change their password. Zero indicates no warning.

PwMaxRepeatChar

Defines the number of characters that can be sequentially repeated in a password. Zero indicates no limit.

Identity Self Administration

You can self-administer the accounts of global users stored in the CA-MDB and perform the following tasks:

- Reset EiamAdmin Password
- Change passwords
- Unlock accounts

For more information to Changing passwords and Unlocking accounts, see *Online Help*.

Reset EiamAdmin Password

CA EEM lets you reset the password for EiamAdmin user, if the password is lost.

To reset EiamAdmin password

1. In the command prompt, goto the iTechnology folder and run the safex command.

```
safex.exe -munge <newpassword>
```

The password is displayed in encrypted format.

2. Stop the iGateway service.

```
/S99gateway stop
```

3. Open the iPoz.conf file and add the encrypted password that is generated in step 2 to the following tag:

```
<EiamAdminPassword><Newpassword></EiamAdminPassword>
```

Save the iPoz.conf file.

4. Start the iGateway service.

```
/S99gateway start
```

Use the new password to login as EiamAdmin user.

Configure Externally Generated Certificates

CA EEM lets you use an externally generated certificate for user authentication. To use an externally generated certificate, you must configure iGateway to trust the root certification authority.

To configure iGateway

1. Enter the URL `http://<hostname>:5250/spin`.
Where `hostname` is the name of the host where iGateway is installed.
2. Select iTech Administrator.
3. Log in as root or administrator by selecting Host or as eiamadmin by selecting iAuthority.
4. Click the iAuthority tab, in the Add Trusted Root section, add the root certification authority as trusted.
5. In the client application, call the `AuthenticateWithCertificate` method passing the personal certificate (pkcs12) issued by the root certification authority and the password of the certificate.

Enable Trace on CA EEM Server

CA EEM lets you enable tracing for the CA EEM Server to log events.

To enable trace on CA EEM Server

1. Stop the iGateway service.

```
/S99igateway stop
```
2. Open the `iPoz.conf` file and add the following tag:

```
<DebugLevel>ISP_FILE</DebugLevel>
```
3. (Optional) Specify the level of logging by adding one of the following tags:
 - To log all the error messages, add the following tag:

```
<LogLevel>ERRORS</LogLevel>
```
 - To log all the warnings and error messages, add the following tag:

```
<LogLevel>WARNING</LogLevel>
```
 - To log all the messages, add the following tag:

```
<LogLevel>INFO</LogLevel>
```
4. Start the iGateway service.

```
/S99igateway start
```

CA EEM Server stores the log file in the iTechnology installation folder.

Enable Trace on CA EEM SDK

CA EEM lets you enable tracing for the CA EEM SDK to log events. To enable tracing, you must call the `enableDebug` method.

Example: Enable Trace on CA EEM SDK

The following example enables trace on CA EEM SDK.

```
SafeUtil.enableDebug();  
// The logging level to be used  
SafeUtil.setDebugLevel(SafeEnum.DebugLevel.WARNING);  
// If you want to dump log to a file  
SafeUtil.setDebugFile(new java.io.File("c:\\eiam-sdk.log"));
```

Note: To disable trace, you can call the `SafeUtil.disableDebug` method.

Dynamic user groups

You can use dynamic user groups to specify membership policies instead of explicitly adding each user or group into a user group. Dynamic user groups are created using dynamic user group policies. Dynamic user groups attributes include, name of the dynamic user group (name of the resource in the policy) and filters.

Example: If you have a dynamic group with the country name as United States, adding a new user with country name United States will automatically be included in the dynamic user group and the group membership policies will be implied to the user.

Dynamic user groups are useful to an application in the following ways:

- To use the same set of filters for several policies by separating them using the dynamic user groups
- To achieve application-specific user groups by writing dynamic user group policies against global user attributes without creating a user object for an application
- To increase the execution speed of authorizations, as dynamic user groups are validated when the user session is built

How Offline Authentication Works

CA EEM supports offline authentication for users authenticating with passwords. The authentication process works in the follows process:

1. For each SafeContext, the last successful session object is saved for each authenticated user, along with a hash of the user's password.
2. `authenticateWithPassword` verifies with the hashed password, if the backend server is not reachable.
3. If the password is valid, CA EEM authenticates the previously saved session and the object is returned.

Chapter 10: Configure Directories

This section contains the following topics:

[Overview](#) (see page 99)

[Configure External Directory](#) (see page 100)

[Custom Mapped Directory](#) (see page 101)

[Test Configuration](#) (see page 103)

[CA SiteMinder](#) (see page 103)

Overview

CA EEM supports referencing external LDAP directories as its authoritative source of global users and groups. Pointing to an external LDAP directory for the global user and group information allows applications to share and reuse the global user information and passwords already configured.

Note: You must select the source of global users before registering the application instances, as changing the source later affects the applications sharing the same backend server and leads to unmapped user information in applications.

CA EEM supports the following external directories:

- CA Admin
- Microsoft Active Directory
- Novell eDirectory
- Novell eDirectory-CN
- Sun One Directory
- Custom Mapped Directory

Configure External Directory

You can configure CA EEM to any of the supported external directory as its authoritative source of global users and groups.

To configure an external directory

1. Log into CA EEM Server as the EiamAdmin user using the Application <Global>.

The CA EEM home page appears. For information on how to log on, see *Getting Started* guide.

2. Select Configure, Embedded IAM Server, Global Users/Global Groups.

The Global Users/Global Groups pane appears.

3. Select Reference from an external directory.

The external directory page appears.

4. Select the type of external directory and set the values.

For information on values for the selected external directory fields, see Online Help.

5. (Optional) Select Use Transport Layer Security (TLS).

Note: On Linux, if enabling TLS is unsuccessful, restart iGateway service to start TLS connection. To restart iGateway, see [Restart iGateway](#) (see page 101).

6. Click Save.

The configuration is saved for CA EEM to use the external directory as its authoritative source.

The status icons on the screen will indicate whether the configuration is successful. If the status does not update automatically, click Refresh status.

More Information:

[Custom Mapped Directory](#) (see page 101)

[Test Configuration](#) (see page 103)

Restart iGateway (Linux)

To restart igateway

1. Go to the iTechnology installation folder.

Default location: /opt/CA/SharedComponents/iTechnology

2. Type the following to stop iGateway:

```
./S99gateway stop
```

3. Type the following to start iGateway:

```
./S99gateway start
```

More Information:

[Example: Configure UPN Using Custom Mapped Directory](#) (see page 102)
[Test Configuration](#) (see page 103)

Custom Mapped Directory

You can configure CA EEM to communicate with an external directory server using LDAP v3 by providing the required mapping to object classes, user attributes, group attributes, and search filters.

Every LDAP directory refers the attributes in a unique way. CA EEM provides mapping for default LDAP directories, for other directories you must provide the custom mapping and search filters. The mapping for custom mapped directory is stored in the iPoz.map file, which is located at C:\Program Files\CA\SharedComponents\iTechnology, by default.

For information on fields to configure a Custom Mapped Directory, see *Online Help*.

Example: Configure UPN Using Custom Mapped Directory

You can log into CA EEM using the User Principal Name (UPN) by mapping the User Name and setting the User Filter. The User Filter obtains the user distinguished name (DN). The user filter has two component pre-filter and post filter.

The user look up is performed against Active Directory as {Pre-user filter}{UserName}{Post-user filter}.

Example: If the UserName is mapped to UPN, the pre-filter would be "(&(userPrincipalName=" and pos-filter will be ")". A user with UPN "john@foo.com" will be searched as "(&(userPrincipalName=john@foo.com))" against the Active Directory.

To configure CA EEM to use UPN

1. Log into CA EEM Server as the EiamAdmin user using the Application <Global>.

The CA EEM home page appears. For information on how to log on, see *Getting Started* guide.

2. Select Configure, Embedded IAM Server.

The Embedded IAM Server Configuration pane appears.

3. Select Reference from an external directory.

The external directory page appears.

4. Select Type as Custom Mapped Directory and click Label.

A list of available tags are displayed in the Custom Directory Mapping pane.

Note: You can define your own tag or customize the existing tags.

5. Change the User Name mapping by modifying the User Name attribute to 'userPrincipalName'.

6. Change the User Filter.

7. Click Save Label, Save.

A confirmation message appears.

Test Configuration

After configuring directories, you can test the CA EEM configuration.

To test the configuration

1. Log into CA EEM Server as the EiamAdmin user using the Application <Global>.

The CA EEM home page appears. For information on how to log on, see *Getting Started* guide.

2. Click Manage Identities, Users.
3. Search for a user.

All the global users are displayed under the 'Users' folder, based on the hierarchy of the external directory.

Note: If CA EEM is configured to use UPN, the User Name will display the UPN based on the search.

CA SiteMinder

You can configure CA EEM to use the existing CA SiteMinder data store for retrieving user and group information. You can use CA EEM with CA SiteMinder to perform the following:

Retrieve user and group information from CA SiteMinder data store

Applications that embed CA EEM can access the user and group information from CA SiteMinder data store and define policies using CA EEM interface.

Authenticate users against CA SiteMinder user store

User with existing CA SiteMinder login credentials can log into any CA EEM enabled application.

Use the CA SiteMinder session information for user authentication

CA EEM supports single sign-on for CA SiteMinder users. If you have an existing CA SiteMinder session, you can access a CA EEM protected application without being prompted for authentication.

More Information:

[How You Integrate CA SiteMinder with CA EEM](#) (see page 104)

[CA SiteMinder Configuration Parameters](#) (see page 104)

[How Single Sign-on Works between CA SiteMinder and CA EEM](#) (see page 105)

[How Authentication Works Using CA SiteMinder Authentication Schemes](#) (see page 105)

How You Integrate CA SiteMinder with CA EEM

To integrate CA SiteMinder with CA EEM, perform the following in CA SiteMinder Administrator:

- Create an agent in CA SiteMinder for communication between CA EEM and CA SiteMinder policy server. Ensure the agent supports 4.x agents.
- Create an administrator or use the existing default administrator "SiteMinder" with system level scope.
- Create a CA SiteMinder User Directory for authorization, which is used by CA EEM to retrieve LDAP attributes. Ensure the UniversalID field uniquely identifies a user in the directory on the User attributes tab.
- Create a CA SiteMinder data store for authentication, which is used by CA EEM to authenticate users.
Note: If the authentication and authorization user store is same, use the existing user store created for authorization.
- Create a Realm with the Resource Filter as "/iamt.html".
- Create a CA SiteMinder domain and add the User Directories, administrator, and Realm to the domain.

For more information about CA SiteMinder, see the CA SiteMinder documentation.

CA SiteMinder Configuration Parameters

CA SiteMinder consists of two components, policy server and web agents.

Policy server

Policy server provides policy management, authentication, authorization and accounting.

Web agents

Web agents assist CA SiteMinder to access to Web applications and content according to defined security policies.

To enable the protocol between the agent and server, the agent must have a unique name and a shared secret key along with information to define a connection between the client application and the policy server.

For information on parameters to define between the client application and the policy server, see *Online Help*.

How Single Sign-on Works between CA SiteMinder and CA EEM

If you use an application that has an existing CA SiteMinder session to access an CA EEM enabled application, CA EEM recognizes the CA SiteMinder session ticket and creates an CA EEM session without re-authentication.

The following is the basic flow of events for application created using CA EEM with CA SiteMinder integration:

Example: Protecting a web application using CA SiteMinder

A web application using CA EEM with web server pages protected by CA SiteMinder is considered.

1. A user accesses a web application.
2. CA SiteMinder prompts for user authentication and the user submits credentials and is authenticated.
3. The user tries to access the original web application created using CA EEM.
4. Servlet code accesses the HttpServletRequest context and sends the CA SiteMinder session token to the CA EEM using `authenticateWithArtifact`.
5. CA EEM Server validates the CA SiteMinder session against the CA SiteMinder Policy Server.
6. An CA EEM session is created and the user identity is loaded, if validation succeeds.

How Authentication Works Using CA SiteMinder Authentication Schemes

The following process describes how authentication is performed using CA SiteMinder APIs:

- A user calls the `authenticateWithPassword` method by providing the username and password.
- CA EEM sends this information to the CA EEM Server.
- Based on the information, the authentication is performed by calling the CA SiteMinder APIs.
- The group and user information is loaded for the authenticated user.

Note: When CA EEM is connected to a CA SiteMinder user directory, the search calls use the CA SiteMinder APIs instead of the CA EEM search calls.

Chapter 11: Integrate Web Services with CA EEM

This section contains the following topics:

[Web Services Architecture](#) (see page 108)

[Configure Web Services for CA EEM](#) (see page 110)

[Configuration File](#) (see page 112)

[Sample Configuration File](#) (see page 114)

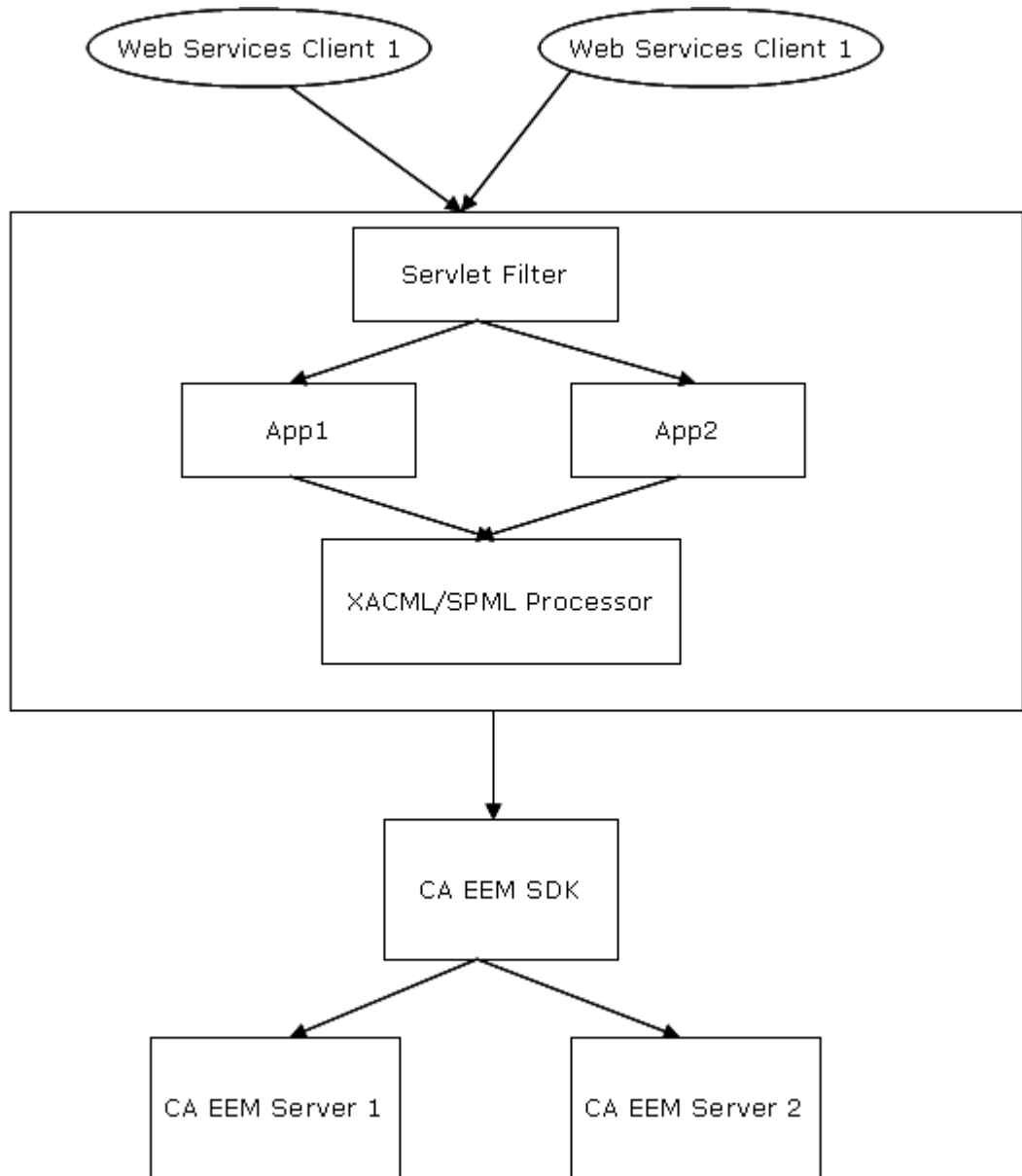
[Tools](#) (see page 115)

[XACML Profile for CA EEM](#) (see page 116)

[SPML Profile for CA EEM](#) (see page 135)

Web Services Architecture

The following illustration depicts one of the deployment architectures of implementing SPML integration with CA EEM:



The following process details how a web service request is implemented in CA EEM:

1. A client application sends an XACML or an SPML to the application's web service. CA EEM implements web services using oasis tools. The oasis tools are deployed on the web server for CA EEM server.
2. The servlet filter on the web server processes the XACML and SPML and extracts the application information. The web service processor deployed in the servlet filter maps the application to a CA EEM server using the `eiam-oasis.xml` configuration file.
3. The web service processor engine in the web server parses the client application's request and calls the appropriate CA EEM APIs.
4. The corresponding CA EEM server implements the request and sends a response to the web server, and the web server will forward the response to the client application.

Configure Web Services for CA EEM

You must register your client application with CA EEM server before you configure web services for SPML and XACML usage. CA EEM provides the following tools that you can use to access the web services.

Oasis.war

An archive file that provides the implementation of CA EEM web services.

Oasis-tools

A compressed file that provides the administrative tools required to configure the service.

To configure Web Services for CA EEM

1. Copy Oasis.war file from the following location to your Web server root directory:

`<CAEEMSDK_InstallDirectory>\oasis`

2. Copy tools.jar from jdk1.5/lib to the following directory:

`Tomcat\web Apps\Oasis\WEB-INF\lib`

3. Untar or unzip oasis-tools.zip file found at the following location:

`<CAEEMSDK_InstallDirectory>\oasis`

4. Generate certificates for your client applications using the following tools based on your operating system

Windows

`GenCert.bat`

UNIX/Linux

`GenCert.sh`

5. Copy the generated certificates to the following location:

`Tomcat\web Apps\Oasis\WEB-INF\certs`

6. Munge the passwords for the certificates generated in previously by using the following tools based on your operating system:

Windows

`MungePwd.bat`

UNIX/Linux

`MungePwd.sh`

7. Edit the eiam-oasis.xml file found at the following location to configure the oasis services:

`Tomcat\web Apps\Oasis\WEB-INF\classes`

Note: For more information on eiam-oasis.xml configuration file, see [Configuration File](#) (see page 112).

More Information:

[Configuration File](#) (see page 112)

[Tools](#) (see page 115)

Configuration File

To interact with the CA EEM web service, the web service relies on a configuration file `eiam-oasis.xml`. The `eiam-oasis.xml` file contains authentication details and CA EEM server connection details for each client application.

The following are the descriptions for the tags that are used the `eiam-oasis.xml` file:

<eiam-oasis>

Specifies the root tag of the `eiam-oasis.xml` file.

app

Specifies the application specific details. You can include one or more applications in the `eiam-oasis.xml` file. All details for a particular application must be mentioned between `<app>` and `</app>` tags.

name

Specifies the application name. The application name should be unique in the `eiam-oasis.xml` configuration file.

cert-file

Specifies the certificate name that should be used to authenticate the current application. The certificates for applications that intend to use CA EEM web services must be available at the following location:

`Tomcat\web Apps\Oasis\WEB-INF\certs`

password

Specifies the munged password to be used with a certificate.

backend-url

Specifies the IP address or hostname of CA EEM server.

locale

Specifies the language and country. The format is `<language code>_<country code>`. Language is in lowercase and is represented by a two-lettered ISO 639 code. Country is in uppercase and is represented by a two-lettered ISO-3166 code. For example if English is the language and the US as the country, locale will be `en_US`. This tag is optional and a default locale is used if left blank.

cacheUnauthenticatedSessionQSize

Specifies the maximum size of the cache's unauthenticated session queue.

Default: 10.

cacheUpdateTime

Specifies the interval between successful cache updates.

Default: 30 seconds.

eventCoalesceTime

Specifies the interval in seconds between successful coalesce event generation.

Default: 300 seconds.

eventDeliveryHost

Specifies host name of the event delivery host.

Default: Value mentioned in <backend-url>.

eventDrainTime

Specifies the event drain time in seconds.

Default: 10 seconds.

maxSearchSize

Specifies the maximum queue size that is returned from searches.

Default: 2000.

maxContextCount

Specifies the maximum number of SafeContext instances of the given application.

Sample Configuration File

The following code is an example configuration file:

```
<?xml version="1.0" standalone="no" ?>
<eiam-oasis>

  <app>
    <name>App1</name>
    <cert-file>App1.p12</cert-file>
    <password>EwASHkYCAg==</password>
    <backend-url>localhost</backend-url>
    <locale>en_US</locale>
    <cacheUnauthenticatedSessionQSize>1</cacheUnauthenticatedSessionQSize>
    <cacheUpdateTime>1</cacheUpdateTime>
    <eventCoalesceTime>1</eventCoalesceTime>
    <eventDeliveryHost>localhost</eventDeliveryHost>
    <eventDrainTime>1</eventDrainTime>
    <maxSearchSize>1</maxSearchSize>
  </app>

  <app>
    <name>RBC_Hospital</name>
    <cert-file>RBC_Hospital.p12</cert-file>
    <password>EwASHkYCAg==</password>
    <backend-url>localhost</backend-url>
    <locale>en_US</locale>
    <maxContextCount>2</maxContextCount>
  </app>

</eiam-oasis>
```

Tools

The following tools are packaged as part of oasis-tools.zip or oasis-tools.tar. You must use the following tools when you configure web services for CA EEM.

GenCert

GenCert.tools are used to generate certificates that are used to authenticate a client application with a CA EEM server. The tools must be run with the following arguments as inputs to generate a certificate:

Windows

```
GenCert.bat <CAEEMserver_hostname> <application name>  
<CAEEM_username> <password> <certificatename>  
<password_certificate>
```

For example:

```
GenCert.bat localhost RBC_Hospital eiamadmin a RBC_Hospital.p12 a
```

UNIX/Linux

```
GenCert.sh <CAEEMserver_hostname> <application name>  
<CAEEM_username> <password> <certificatename>  
<password_certificate>
```

For example:

```
GenCert.sh localhost RBC_Hospital eiamadmin a RBC_Hospital.p12 a
```

MungePwd

MungePwd tools are used to munge the certificate password. The tools take in a password in clear text as input argument. You must run this tool on the host that has the eiam-oasis.xml configuration file.

ExportEiam

ExportEiam is used to export a CA EEM application to an XACML Policy Decision Point (PDP). The tools must be run with the following arguments as inputs:

Windows

```
ExportEiam.bat <WebServices_hostname> <application name>  
<CAEEM_username> <password> <path_destinationfolder>  
<password_certificate>
```

UNIX/Linux

```
ExportEiam.sh <WebServices_hostname> <application name>  
<CAEEM_username> <password> <path_destinationfolder>  
<password_certificate>
```

ImportXacml

ImportXacml is used to export an XACML application, a set of resource classes and policies to a CA EEM application. to an XACML Policy Decision Point (PDP). The tools must be run with the following arguments as inputs:

Windows

```
ImportXacml.bat <WebServices_hostname> <application name>  
<CAEEM_username> <password> <path_XacmlPoliciesFile>
```

UNIX/Linux

```
ImportXacml.sh <WebServices_hostname> <application name>  
<CAEEM_username> <password> <path_XacmlPoliciesFile>
```

Note: If you are using an IPv6 environment, IPv6 address provided as an input argument to the preceding tools must be enclosed in [] (Square brackets).

XACML Profile for CA EEM

XACML Integration

CA EEM provides support for the OASIS eXtensible Access Control Markup Language 1.0 (XACML) as a web service. Using XACML, CA EEM supports the following operations:

- Authorization checks
- Import of external policies
- Export of policies

Note: To use XACML Web services, you must have JRE 1.5.1 and Tomcat 5.5 to be installed.

WSDL for CA EEM XACML

Web Services Description Language (WSDL) is an XML format that is used to describe the network services that are provided by an application. CA EEM publishes the XACML operations it supports using WSDL. After you have deployed the oasis tools, you can access web services for CA EEM using the following URL:

`http://<hostname>:8080/oasis/<app name>.jws.`

For example, if you have deployed your web server on your localhost and App1 is the application name, you can access web services for App1 by using the URL, `http://localhost:8080/oasis/App1.jws.`

You can access the WSDL provided by an application deployed on a Web server using the following URL:

`http://<hostname>:8080/oasis/<app name>.jws?wsdl`

The WSDL for CA EEM supports the following default XACML functions:

XACML Function Name	Input Parameters for XACML Request	Return Value in XACML Response
checkAccess	XACML request string	XACML response string
exportEiamPolicies		XACML response string
importXacmlPolicies	XACML policy set string	boolean

XACML Services for CA EEM

You must register your client application with CA EEM server before you configure XACML requests or responses. The following sections explain the following XACML services:

- XACML Requests
- XACML Responses
- Export or Import resource classes or policies

XACML Requests

The following section explains how you map CA EEM objects to XACML requests. You can create XACML requests for the following:

- Users and User Groups
- Actions
- Resource and Named Attributes
- Calendar and Environmental Variables

Users and User Groups

You can authorize only one identity, one action, and one resource at a time using XACML. The CA EEM users are mapped to XACML Subject element in a XACML request. You should follow the following instructions when creating XACML requests for authorizing users and user groups:

- A XACML request element must have at least one Subject element.
- If a XACML request has multiple subject elements, only the first Subject element is considered. Also, the request must have at least one attribute. All other attribute and subject elements are ignored.
- The AttributeId of the only attribute must be `urn:oasis:names:tc:xacml:1.0:subject:subject-id`.
- DataType attribute in Attribute element must be `http://www.w3.org/2001/XMLSchema#string`.
- AttributeValue element in Attribute element must be a valid identity.

Actions

You must map CA EEM actions to XACML Action element to force some action on any of the CA EEM objects. You must follow the following instructions when creating XACML requests:

- A XACML Request element must have at least one Action element.
- If a XACML request has multiple action elements, only the first action element is considered. Also, the request must have at least one attribute. All other attribute and action elements are ignored.
- AttributeId attribute in Attribute element must be `urn:oasis:names:tc:xacml:1.0:action:action-id`
- DataType attribute in Attribute element must be `http://www.w3.org/2001/XMLSchema#string`.
- AttributeValue element in Attribute element must be a valid action.

Resources and Named Attributes

You must map CA EEM resources to XACML Resource element. You must express any Named attributes as XPATH expressions. These expressions are evaluated against XACML ResourceContent element. You must follow the following instructions when creating XACML requests:

- A XACML Request element must have one Resource element.
- The Resource element must have at least one Attribute.
- AttributeId attribute in Attribute element must be urn:oasis:names:tc:xacml:1.0:resource:resource-id in case of resource class name and name.
- AttributeId attribute in Attribute element must be urn:oasis:names:tc:xacml:1.0:resource:xpath for named attributes
- DataType attribute in Attribute element must be http://www.w3.org/2001/XMLSchema#string for all attributes.
- AttributeValue element in Attribute element should be the resource class name, resource name, or named attribute. For resource class name and name the syntax is resource class name/resource name. For named attributes, the syntax must be a valid XPath expression in the form of named attribute name/named attribute value.

Calendars and Environment Variables

You must map CA EEM calendars and environment variables to XACML Environment. You must follow the following instructions when creating XACML requests:

- An environment element is optional in a XACML request element.
- The environment element can optionally have attributes.
- AttributeId attribute in an Attribute element can be anything for environment attributes. For example, the name of the environment Attribute.
- DataType attribute in Attribute element for calendar object must be http://www.w3.org/2001/XMLSchema#dateTime.
- DataType attribute in Attribute element for environment object must be and http://www.w3.org/2001/XMLSchema#string.
- AttributeValue element in Attribute element must be a valid date time object in case of calendars and can be anything for environment attributes.

XACML Responses

The following are the elements in a XACML response:

ResourceId

A XACML Response element may or may not have a ResourceId attribute. If access is granted to an object, the value for ResourceId will be set to the allowed resource name with the following syntax:

resource class name/resource name.

If access is denied, ResourceId attribute is ignored.

Decision

A XACML Response element should have at least one Decision element. The values for the Decision element will be set to the one of the following:

- PERMIT, if the result is true
- DENY, if the result is false
- INDETERMINATE, if an exception has occurred

Status

An XACML Response element should have at least one Status element. The Status element has the following syntax based on the Decision element and exceptions:

- StatusCode element will be set to urn:oasis:names:tc:xacml:1.0:status:ok if Decision element is set to PERMIT or DENY. The other elements StatusDetail and StatusMessage are ignored.
- StatusCode element will be set to urn:oasis:names:tc:xacml:1.0:status:missing-attribute if the attribute is either missing or invalid. The StatusMessage element will be set to the message arising out of the exception.
- StatusCode element will be set to urn:oasis:names:tc:xacml:1.0:status:processing-error if an exception has occurred other than a missing or invalid attribute. The StatusMessage element will be set to the message arising out of the exception.

Obligations

An XACML Response element may or may not have an Obligation element.

- An obligation element is added only if Decision element is set to PERMIT or DENY.
- ObligationId element will be of syntax <obligation name>/<obligation content>.

- The Obligation element will have at least one AttributeAssignment element. All AttributeId elements will be <obligation attribute name> and the content will be <obligation attribute value>. However, one AttributeAssignment will be same as ObligationId element.
- DataType of AttributeAssignment element will be <http://www.w3.org/2001/XMLSchema#string>.
- FulfillOn will be according to the Decision element.

Restrictions

- Does not support all XACML data types for subjects, actions and resources.
- All other immediate and descendant elements and XACML functions are ignored.

Export and Import Using XACML

You can export or import resource classes and policies to CA EEM using XACML. The following are the prerequisites for exporting or importing.

- Before exporting an application's resource classes and policies, you must register the application with CA EEM server.
- Before importing resource classes and policies to an application, you must register the application, without any resource classes and policies, with CA EEM server.

The following sections describe the mappings of a CA EEM server policies to an XACML policy set.

Policy Name

CA EEM policy name is mapped to PolicyId attribute of the XACML Policy element. Blank spaces in the policy name are replaced with a '_'.

Policy Description

CA EEM policy description is mapped to Description element of XACML Policy element. Empty descriptions are replaced with a blank space.

Users and User Groups

CA EEM users and user groups are mapped to XACML Subject elements. For each CA EEM user or user group mentioned in an CA EEM policy there is an XACML Subject element.

- Users are mapped to SubjectAttributeDesignator element.
- User groups are mapped to AttributeSelector element.
- MatchId attribute of SubjectMatch element is a `urn:oasis:names:tc:xacml:1.0:function:string-equal`.
- AttributeValue element should be valid a CA EEM identity in case of users.
- CA EEM AllIdentities is mapped to XACML AnySubject element. In case of user groups its syntax is CA EEM group type/group name.
- XACML SubjectAttributeDesignator has the data type as `http://www.w3.org/2001/XMLSchema#string`.
- AttributeId is set to `urn:oasis:names:tc:xacml:1.0:subject:subject-id`.
- XACML AttributeSelector has the data type as `http://www.w3.org/2001/XMLSchema#string` with the RequestContextPath set to `//CA EEM group type/group name`.

Actions

CA EEM actions are mapped to the XACML Action element. For each CA EEM action mentioned in an CA EEM policy there is an XACML Action element.

- The MatchId attribute of ActionMatch element is `urn:oasis:names:tc:xacml:1.0:function:string-equal`.
- AttributeValue element should be a valid CA EEM action. CA EEM AllActions is mapped to XACML AnyActions.
- XACML ActionAttributeDesignator has the data type as `http://www.w3.org/2001/XMLSchema#string`.
- The AttributeId is set to `urn:oasis:names:tc:xacml:1.0:action:action-id`.

Resource Class and Resources

CA EEM resources class and resources are mapped to the XACML Resources element. For each CA EEM resource mentioned in an CA EEM policy there is an XACML Resource element.

- The MatchId attribute of ResourceMatch element is a `urn:oasis:names:tc:xacml:1.0:function:regexp-string-match`.
- AttributeValue element uses the syntax resource class name/resource name. In case of CA EEM AllResources the resource name is '*' (asterisk).
- XACML ResourceAttributeDesignator has the data type as `http://www.w3.org/2001/XMLSchema#string`.
- The AttributeId is set to `urn:oasis:names:tc:xacml:1.0:resource:resource-id`.

Filters

CA EEM filters are represented using XACML Rule and Apply elements. If there is no CA EEM filter in a policy a default XACML Rule is added with the EffectType of the XACML Rule same as the CA EEM policy type, that is explicit grant/deny. Each CA EEM filter is wrapped in a XACML Apply element which has a XACML type any-of-any function.

- The XACML Condition element has the function as urn:oasis:names:tc:xacml:1.0:function:boolean-equal. This wraps the outer most XACML Apply element and an XACML AttributeValue element.
- The AttributeValue element has the data type as http://www.w3.org/2001/XMLSchema#boolean with content set to true.
- RuleId attribute is same as the policy name with a suffix of '-Rule'. All blank spaces are replaced with a '_'.
- CA EEM filter operation is mapped to an XACML function. NOT operations are expressed as a negation of the actual operation; eg. neq is not + equal i.e. equal operation wrapped inside a not operation.
- CA EEM filter value is mapped to XACML AttributeValue element in case of simple values, CA EEM identities, resources, and actions. The data type of the AttributeValue element is the same as the CA EEM filter operation type. The content of AttributeValue is a simple value is of syntax CA EEM filter column name/CA EEM filter value. For a set based CA EEM operation each element of the set is mapped to an XACML AttributeValue. All the AttributeValue elements are wrapped in an XACML Apply element whose function is an XACML of the CA EEM operation type.
- CA EEM filter value is mapped to XACML AttributeSelector element for CA EEM named attributes, custom attributes, session attributes, request time, users, and user groups. The data type of the AttributeSelector element is same as the CA EEM filter operation type. The RequestContextPath is set to an XPath //CA EEM filter column prefix/CA EEM column suffix.
- CA EEM filter column is mapped to XACML AttributeSelector for CA EEM named attributes, custom attributes, session attributes, request time, users and user groups. The data type of the AttributeSelector element is same as the CA EEM filter operation type. The value of the XPath expression should be of the syntax CA EEM filter column suffix/value.
- CA EEM filter column is mapped to a SubjectAttributeDesignator in case of simple identity; ActionAttributeDesignator in case of simple action; ResourceAttributeDesignator in case of simple resource. The data type of the AttributeDesignator is http://www.w3.org/2001/XMLSchema#string.

- CA EEM filter column with a prefix of req: mapping is based on the CA EEM filter column suffix. If the suffix is identity it is mapped to SubjectAttributeDesignator; action is mapped to ActionAttributeDesignator; resource is mapped to ResourceAttributeDesignator. In this case the data type of the designator is set to <http://www.w3.org/2001/XMLSchema#string>. All other suffixes of request are mapped to AttributeSelector with the RequestContextPath.
- CA EEM filter column with a prefix of env: is mapped to XACML EnvironmentAttributeDesignator. The data type of the AttributeDesignator is <http://www.w3.org/2001/XMLSchema#string>. The AttributeId is set to env/column suffix.

Miscellaneous

Other XACML elements have the following values.

- The XPathVersion element is <http://www.w3.org/TR/1999/Rec-xpath-19991116>.
- The rule combining algorithm is <urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides>.
- The policy combining algorithm is <urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides>.
- All the policies are wrapped in XACML PolicySet element where PolicySetId is the same as the CA EEM application name.
- The Target element in PolicySet element is set to AnySubject, AnyResource and AnyAction.

Restrictions for Exporting or Importing to XACML

An existing CA EEM application can be exported or imported with the following restrictions:

- Only access and dynamic user group CA EEM policies are exported.
- Policies are exported as simple access policies.
- Exported policies are not marked as 'Use best match algorithm'.
- Disabled polices are not exported.
- Pre-Deployment policies are not exported.
- Calendar objects are not exported.
- CA EEM calc objects are not supported.
- CA EEM regex and xpath as a data type are not supported.
- During an import, CA EEM resource class and action information is updated from filters. Hence the import may not be complete.
- During an import, CA EEM named attributes and user attributes available in the imported application are not created.
- During an import, XACML Target defined in XACML Rule elements is ignored.

Mapping CA EEM Operations to XACML Functions

The following table explains the mappings of CA EEM operations to XACML functions:

CA EEM Operation Name	XACML Function Name
Equal	equal
Neq	not + equal
Less	less-than
Greater	greater-than
Lessequal	less-than-or-equal
Greaterequal	greater-than-or-equal
like	regexp-string-match
notlike	not + regexp-string-match
withinset	is-in
notinset	not + is-in

CA EEM Operation Name	XACML Function Name
startswith	regexp-string-match + *
endswith	* + regexp-string-match
contains	* + regexp-string-match + *
match	regexp-string-match
notmatch	not + regexp-string-match

Mapping CA EEM Data Types to XACML Data Types

The following table explains the mappings of CA EEM operations to XACML functions:

CA EEM Data Type	XACML Data Type
string	http://www.w3.org/2001/XMLSchema#string
int32	http://www.w3.org/2001/XMLSchema#integer
int64	http://www.w3.org/2001/XMLSchema#integer
real32	http://www.w3.org/2001/XMLSchema#double
real64	http://www.w3.org/2001/XMLSchema#double
boolean	http://www.w3.org/2001/XMLSchema#boolean
timestamp	http://www.w3.org/2001/XMLSchema#dateTime

XPATH Expressions for CA EEM Filters

The following table explains the XPATH expressions for CA EEM filters:

CA EEM Filter Column	XPATH Expression
Users and user groups	<code>//CA EEM column prefix/CA EEM column suffix</code>
Named attributes	<code>//namedAttributes/namedAttribute/column suffix</code>
Custom attributes	<code>//customAttributes/customAttribute/column suffix</code>
Session attributes	<code>//sessionAttributes/sessionAttribute/column suffix</code>
Request (other than identity/action/resource)	<code>//requestAttributes/requestAttribute/column suffix</code>
Request time	<code>//when</code>

AttributeId Values for XACML AttributeDesignator Elements

The following table explains the AttributeId values you must use with AttributeDesignator elements in XACML policies:

CA EEM Column Suffix	AttributeId
Identity	urn:oasis:names:tc:xacml:1.0:subject:subject-id
Action	urn:oasis:names:tc:xacml:1.0:resource:action-id
Resource	urn:oasis:names:tc:xacml:1.0:resource:resource-id

Examples

The examples in this section describe some sample XACML requests and responses that are sent to a web service. The examples in this section use the RBC_Hospital application that is installed with CA EEM. For these examples it is assumed that the web services are running on localhost. Hence, the request URL for the RBC_Hospital application is:
`http://localhost:8080/oasis/RBC_Hospital.jws.`

XACML Request to Add Patient to ER Ward

The following is an example of an XACML request for a policy that states that only an Emergency Response nurse (ernurse) or erdoctor can admit a patient to the ER ward.

In the following example, a client application sends a request from a user ernurse to admit a patient 'John' to the ER ward. This XACML request is processed by CA EEM server and based on the admit policy, a response is sent to the client application.

The following table explains the mappings of CA EEM objects to XACML elements:

CA EEM Object	XACML Element
ernurse	Subject
John	Resource
ward/ER	ResourceContent

The following is the example:

```

<Request>
<Subject>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
  DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>emurse</AttributeValue>
  </Attribute>
</Subject>
<Resource>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
  DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>patient/john</AttributeValue>
  </Attribute>
<ResourceContent>
<namedAttributes>
<namedAttribute>
<ward>ward/ER</ward>
</namedAttribute>
</namedAttributes>
<u>
  <ward>ward/ER</ward>
</u>
<ug>
  <Nurses>ug/Nurses</Nurses>
</ug>
</ResourceContent>
<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
  DataType="http://www.w3.org/2001/XMLSchema#string">
  <AttributeValue>namedAttributes/namedAttribute[1]/ward</AttributeValue>
</Attribute>
</Resource>
<Action>
<Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
  DataType="http://www.w3.org/2001/XMLSchema#string">
<AttributeValue>admit</AttributeValue>
</Attribute>
</Action>
</Request>

```

CA EEM server returns the following XACML response for the preceding request:

```

<?xml version="1.0" encoding="UTF-8" ?>
<Response xmlns="urn:oasis:names:tc:xacml:1.0:context" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="Response">
  <Result ResourceId="patient/john">
<Decision>Permit</Decision>
<Status>

```

```
<StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />  
</Status>  
</Result>  
</Response>
```

XACML Request to Locate Patient During Visiting Hours

The example in this section is an XACML request that requests access based on the following policy:

A receptionist can locate a patient only during the visiting hours. The visiting hours are from 9:00 AM to 11:00 AM and 4:00 PM to 6:00 PM.

The following is the XACML request that details a receptionist trying to locate a patient John at 10:23 AM:

```
<Request>
  <Subject>
    <Attribute AttributeId="um:oasis:names:tc:xacml:1.0:subject:subject-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>receptionist</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="um:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>patient/John</AttributeValue>
    </Attribute>
    <ResourceContent>
      <gu>
        <UserName>UserName/receptionist</UserName>
      </gu>
      <ug>
        <Staff>ug/Staff</Staff>
      </ug>
    </ResourceContent>
  </Resource>
  <Action>
    <Attribute AttributeId="um:oasis:names:tc:xacml:1.0:action:action-id"
      DataType="http://www.w3.org/2001/XMLSchema#string">
      <AttributeValue>locate</AttributeValue>
    </Attribute>
  </Action>
  <Environment>
    <Attribute AttributeId="um:oasis:names:tc:xacml:1.0:environment:current-dateTime"
      DataType="http://www.w3.org/2001/XMLSchema#dateTime">
      <AttributeValue>2006-02-15T10:23:47-05:00</AttributeValue>
    </Attribute>
  </Environment>
</Request>
```

The CA EEM server evaluates XACML request based on the policy and grants a PERMIT action. Following is the XACML response for the preceding request:

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

```
<Response xmlns="urn:oasis:names:tc:xacml:1.0:context" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Response">
  <Result ResourceId="patient/John">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
    </Status>
  </Result>
</Response>
```

Admit a Patient to a Ward

The example in this section explains how to import an XACML request that describes the following policy:

The policy states that in RBC_Hospital, a patient can be admitted to the ER ward only by staff that are assigned to ER ward.policy according to which a patient can be admitted to ER ward only by any staff assigned to ER ward or by the patient's doctor.

Following is the XACML request for the preceding policy.

```
<Policy PolicyId="patient_er_admission" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:deny-overrides">
  <Description>patient can be admitted to the ER by any staff assigned to ER, or the patient's
doctor</Description>
  <PolicyDefaults>
    <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion>
  </PolicyDefaults>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">ug/Doctors</AttributeValue>
          <AttributeSelector RequestContextPath="//ug/Doctors"
DataType="http://www.w3.org/2001/XMLSchema#string" />
        </SubjectMatch>
      </Subject>
      <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">ug/Nurses</AttributeValue>
          <AttributeSelector RequestContextPath="//ug/Nurses"
DataType="http://www.w3.org/2001/XMLSchema#string" />
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">patient* </AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string" />
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Actions>
    <Action>
      <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
```

```

        <AttributeValue
Data Type="http://www.w3.org/2001/XMLSchema#string">admit</AttributeValue>
        <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        Data Type="http://www.w3.org/2001/XMLSchema#string" />
        </ActionMatch>
    </Action>
</Actions>
</Target>
<Rule RuleId="patient_er_admission-Rule" Effect="Permit">
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-equal">
    <AttributeValue Data Type="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
            <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
            <AttributeSelector
RequestContextPath="//namedAttributes/namedAttribute/ward"
Data Type="http://www.w3.org/2001/XMLSchema#string" />
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
                    <AttributeValue
Data Type="http://www.w3.org/2001/XMLSchema#string">ward/ER</AttributeValue>
                </Apply>
            </Apply>
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:or">
                <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
                    <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
                    <AttributeSelector RequestContextPath="//namedAttributes/namedAttribute/ward"
Data Type="http://www.w3.org/2001/XMLSchema#string" />
                        <AttributeSelector RequestContextPath="//u/ward"
Data Type="http://www.w3.org/2001/XMLSchema#string" />
                    </Apply>
                    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
                        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
                            <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
                            <AttributeSelector RequestContextPath="//namedAttributes/namedAttribute/doctor"
Data Type="http://www.w3.org/2001/XMLSchema#string" />
                                <AttributeSelector RequestContextPath="//gu/UserName"
Data Type="http://www.w3.org/2001/XMLSchema#string" />
                            </Apply>
                            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
                                <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
                                <AttributeSelector RequestContextPath="//ug/Name"
Data Type="http://www.w3.org/2001/XMLSchema#string" />
                                    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
                                        <AttributeValue
Data Type="http://www.w3.org/2001/XMLSchema#string">Name/Doctors</AttributeValue>
                                    </Apply>
                                </Apply>
                            </Apply>
                        </Apply>
                    </Apply>
                </Apply>
            </Apply>
        </Apply>
    </Condition>
</Rule>

```

```
</Apply>
</Apply>
</Condition>
</Rule>
</Policy>
```

SPML Profile for CA EEM

SPML 1.0 is an OASIS standard to automate provisioning requirements. CA EEM provides an SPML profile that you can use to implement your provisioning setup.

SPML Integration

CA EEM provides support for the OASIS Service Provisioning Markup Language 1.0 (SPML) as a web service. You can create, modify, delete, and search on the following CA EEM objects:

- Global Users and Application specific Users
- Global Groups and Application specific Groups
- Policies
- Application Instances
- Calendars

Note: To use SPML Web services, you must have JRE 1.5.1 and Tomcat 5.5 to be installed.

Terminology

The terminology used in SPML implementations is documented as part of OASIS specifications for SPML 1.0. For more information on terminology for SPML, see <http://www.oasis-open.org/committees/download.php/3032/cs-pstc-spml-core-1.0.pdf>.

WSDL for CA EEM SPML

Web Services Description Language (WSDL) is an XML format that is used to describe the network services that are provided by an application. CA EEM publishes the SPML operations it supports using WSDL. After you have deployed the oasis tools, you can access web services for CA EEM using the following URL:

`http://<hostname>:8080/oasis/<app name>.jws.`

For example, if you have deployed your web server on your localhost and App1 is the application name, you can access web services for App1 by using the URL, `http://localhost:8080/oasis/App1.jws.`

You can access the WSDL provided by an application deployed on a Web server using the following URL:

`http://<hostname>:8080/oasis/<app name>.jws?wsdl`

The WSDL for CA EEM supports the following SPML operations:

SPML Operation	Input Parameters for SPML Request	Return Value in SPML Response
add	SPML request string	SPML response string
modify	SPML request string	SPML response string
delete	SPML request string	SPML response string
search	SPML request string	SPML response string
schema	SPML request string	SPML response string
batch	SPML request string	SPML response string

SPML Profile

You must register your client application with CA EEM server before you configure SPML requests. CA EEM supports the following operations on SPML requests:

- Add
- Modify
- Delete
- Search
- Schema
- Batch

CA EEM has the following constraints when processing the preceding operations on SPML requests:

- CA EEM does not support parallel processing in an SPML batch request. Also, SPML extendedRequest is not supported for SPML batch request.
- A schema request should include one of the valid objectclass attribute values. The schemaid and providerid should be set to the following values:

schemaid

urn:oasis:names:tc:SPML:1:0#GenericString. and may have one of the valid objectclass attribute values and the

providerId

eiam.

The schema request may include one of the valid objectclass and its attribute values.

- A search request must include one of the valid objectclass attribute values. Also, the IdentifierType must be:
urn:oasis:names:tc:SPML:1:0#GenericString
- A modify request must include one of the valid objectclass attribute values. The valid objectclass attribute value must be mentioned as an attribute under IdentifierAttribute. The SPML attribute operation under modification element must be replace. The IdentifierType must be:
urn:oasis:names:tc:SPML:1:0#GenericString.
- A delete request must include one of the valid objectclass attribute values. The valid objectclass attribute value must be mentioned as an attribute under IdentifierAttribute. The IdentifierType must be:
urn:oasis:names:tc:SPML:1:0#GenericString.
- An add request must include a valid objectclass attribute value.

The following table maps the SPML functions to the corresponding CA EEM operations:

SPML Function	CA EEM Operation Name
approxmatch	match
equalitymatch	equal
greaterequal	greaterequal
lessequal	lessequal
initial-substring	startswith
finalsubstring	endswith
anysubstring	contains

Global Users and Application Users in SPML Profile

Add, Modify, and Delete operations are not supported when connected to an external directory as the user source.

The following are the mappings from CA EEM object attributes for global and application users to SPML profile attributes:

- All CA EEM Safe::GlobalUser and Safe::User object attributes mentioned in the CA EEM SDK are supported.
- CA EEM global users have the SPML objectclass attribute as SafeGlobalUser.
- CA EEM application users have the SPML objectclass attribute as SafeUser.
- All single valued CA EEM object attributes have only a single SPML value element for a SPML attr element.
- All multi valued CA EEM object attributes can have multiple SPML value elements for a single SPML attr element.
- Username and cn are the synonyms for CA EEM object attribute 'name'.
- CA EEM object attributes password and oldpassword are mapped to SPML attributes PasswordDigest and OldPasswordDigest respectively.
- A SPML request can have multiple attr elements with the same name. If there are multiple attr elements with the same name, the last attr element overrides the other elements.
- Other than the defined CA EEM object attributes, an CA EEM object can also have extended attributes. For extended attributes, a SPML attr element should have the name of the attribute and value element should have the value of the attribute. An example of such a case is user attributes.
- The values of attributes can be mentioned as valid XPATH expressions. These expressions are evaluated against an XML element under SPML <attributes> element.
- CA EEM object attributes PasswordDigest and OldPasswordDigest should be simple base64 encoded password and oldpassword respectively.
- CA EEM object attributes of type time or date-time must be valid.
- CA EEM object attributes of type boolean must be valid.
- CA EEM object attributes of type integer must be valid.

Note: For more information on the format for type time or date-time, type integer, or type boolean, see <http://www.w3.org/2001/XMLSchema#dateTime>.

Global User Groups and Application User Groups in SPML Profile

Add, Modify, and Delete operations are not supported when connected to an external directory as the user source.

The following are the mappings from CA EEM object attributes for global and application users to SPML profile attributes:

- All CA EEM Safe::GlobalUserGroup and Safe::UserGroup object attributes mentioned in the CA EEM SDK are supported.
- CA EEM global user groups have the SPML objectclass attribute as SafeGlobalUserGroup.
- CA EEM application user groups have the SPML objectclass attribute as SafeUserGroup.
- All single valued CA EEM object attributes have only a single SPML value element for a SPML attr element.
- All multi valued CA EEM object attributes can have multiple SPML value elements for a single SPML attr element.
- cn is a synonym for CA EEM object attribute *name*.
- An SPML request can have multiple attr elements with the same name. If there are multiple attr elements with the same name, the last attr element overrides the other elements.
- A CA EEM object can also have extended attributes other than the defined CA EEM object attributes. For extended attributes, a SPML attr element should have the name of the attribute and value element should have the value of the attribute. An example of such a case is user attributes.
- The values of attributes can be mentioned as valid XPATH expressions. . These expressions are evaluated against an XML element under SPML <attributes> element.
- CA EEM object attributes of type time or date-time must be valid.
- CA EEM object attributes of type boolean must be valid.
- CA EEM object attributes of type integer must be valid.

Note: For more information on the format for type time or date-time, type integer, or type boolean, see <http://www.w3.org/2001/XMLSchema#dateTime>.

Policies in SPML Profile

The following are the mappings from CA EEM object attributes for policies to SPML profile attributes:

- All CA EEM Safe::Policy object attributes mentioned in the CA EEM SDK are supported.
- CA EEM policies have the SPML objectclass attribute as SafePolicy.
- All single valued CA EEM object attributes have only a single SPML value element for a SPML attr element.
- All multi valued CA EEM object attributes can have multiple SPML value elements for a single SPML attr element.
- cn is a synonym for CA EEM object attribute *name*.
- An SPML request can have multiple attr elements with the same name. If there are multiple attr elements with the same name, the last attr element overrides the other elements.
- The values of attributes can be mentioned as valid XPATH expressions. These expressions are evaluated against an XML element under SPML <attributes> element.
- A CA EEM object can also have extended attributes other than the defined CA EEM object attributes. For extended attributes, a SPML attr element should have the name of the attribute and value element should have the value of the attribute. An example of such a case is user attributes.
- The attr element <value> of the filter attribute of CA EEM Safe::Policy object must start with filter. The properties of a filter can be expressed using multiple <value> elements. The syntax of the value must be filter property/property value. Table 3 describes various filter properties.
- The attr element <value> of the obligation filter attribute of CA EEM Safe::Policy object should start with obligation. The properties of an obligation can be expressed using multiple <value> elements. The syntax of the value should be obligation property/property value.
- CA EEM object attributes of type time or date-time must be valid.
- CA EEM object attributes of type boolean must be valid.
- CA EEM object attributes of type integer must be valid.

Note: For more information on the format for type time or date-time, type integer, or type boolean, see <http://www.w3.org/2001/XMLSchema#dateTime>.

The following are the Filter properties:

Filter Property Name	Filter Property Value
logic	And, OR
lparens	Number of left parenthesis
col	Column Name
optype	Valid CA EEM operation type
oper	Valid CA EEM operation type
val	Column value
tag	Name of the Filter
rparens	Number of right parenthesis
order	Order of evaluation of filter

The following are the obligation properties:

Obligation Property Name	Obligation Property Value
name	Name of the obligation
attribute	Comment about obligation
comment	User defined attribute; must be of the form attribute name/attribute value

Calendars in SPML Profile

The following are the mappings from CA EEM object attributes for policies to SPML profile attributes:

- All CA EEM Safe::Calendar object attributes mentioned in the CA EEM SDK are supported.
- CA EEM calendars have the SPML objectclass attribute as SafeCalendar.
- All single valued CA EEM object attributes have only a single SPML value element for a SPML attr element.
- All multi valued CA EEM object attributes can have multiple SPML value elements for a single SPML attr element.
- cn is a synonym for CA EEM object attribute *name*.
- An SPML request can have multiple attr elements with the same name. If there are multiple attr elements with the same name, the last attr element overrides the other elements.
- The values of attributes can be mentioned as valid XPATH expressions. These expressions are evaluated against an XML element under SPML <attributes> element.
- A CA EEM object can also have extended attributes other than the defined CA EEM object attributes. For extended attributes, a SPML attr element should have the name of the attribute and value element should have the value of the attribute. An example of such a case is user attributes.
- The attr element <value> of the includetimeblock attribute and excludetimeblock attribute of CA EEM Safe::Calendar object should start with includetimeblock and excludetimeblock respectively. The properties of includetimeblock or excludetimeblock can be expressed using multiple <value> elements. The syntax of the value must be includetimeblock or excludetimeblock property/property value.
- CA EEM object attributes of type time or date-time must be valid.
- CA EEM object attributes of type boolean must be valid.
- CA EEM object attributes of type integer must be valid.

Note: For more information on the format for type time or date-time, type integer, or type boolean, see <http://www.w3.org/2001/XMLSchema>

The following table describes the includetimeblock and excludetimeblock properties to be used with web services:

Timeblock Property Name	Timeblock Property Value
name	Name of includetimeblock or excludetimeblock
duration	Duration of the time block

Timeblock Property Name	Timeblock Property Value
monthdaymask	Comma separated list of month days; Valid month days are between 1 and 31
monthmask	Comma separated list of month names; Valid month names are from January through December
weekdaymask	Comma separated list of week days; Valid week days are from Sunday though Saturday
recurringtimeinterval	Recurring time interval of the time block
starttime	Start time of the time block

Application Instances in SPML Profile

All CA EEM `Safe::ApplicationInstance` object attributes mentioned in the CA EEM SDK are supported. During add operation of `Safe::ApplicationInstance` object attributes, `certfile` and `certpwd` are also required.

- CA EEM application instances have the SPML objectclass attribute as `SafeApplicationInstance`.
- During the add operation, a certificate is generated that is placed under `certs` directory under `WEB-INF`. Attribute `certfile` is the name of the certificate file. If not mentioned it defaults to the `label` attribute of `Safe::ApplicationInstance`. Attribute `certpwd` is the password of the generated certificate file and must be base64 encoded.
- All single valued CA EEM object attributes have only a single SPML value element for a SPML `attr` element.
- All multi valued CA EEM object attributes can have multiple SPML value elements for a single SPML `attr` element.
- Application name and `cn` are synonyms for CA EEM object attribute *name*.
- An SPML request can have multiple `attr` elements with the same name. If there are multiple `attr` elements with the same name, the last `attr` element overrides the other elements.
- The values of attributes can be mentioned as valid XPATH expressions. These expressions are evaluated against an XML element under SPML `<attributes>` element.
- A CA EEM object can also have extended attributes other than the defined CA EEM object attributes. For extended attributes, a SPML `attr` element should have the name of the attribute and value element should have the value of the attribute. An example of such a case is user attributes.
- The `attr` element `<value>` of the `resourceclass` attribute of CA EEM `Safe::ApplicationInstance` object should start with `resourceclass`. The properties of a `resourceclass` can be expressed using multiple `<value>` elements. The syntax of the value must be `resourceclass property/property value`.
- CA EEM object attributes of type `time` or `date-time` must be valid.
- CA EEM object attributes of type `boolean` must be valid.
- CA EEM object attributes of type `integer` must be valid.

Note: For more information on the format for type `time` or `date-time`, type `integer`, or type `boolean`, see <http://www.w3.org/2001/XMLSchema>.

The following are the resourceclass properties to be used with web services:

Resourceclass Property Name	Resourceclass Property Value
action	Comma separated list of actions
namedattribute	Comma separated list of named attributes
name	Name of the resource class
bestmatchevaluation	True or false

Example SPML Requests and Responses

The examples in this section describe some sample SPML requests and responses that are sent to a web service. The examples in this section use the RBC_Hospital application that is installed with CA EEM. For these examples it is assumed that the web services are running on localhost. Hence, the request URL for the RBC_Hospital application is:
`http://localhost:8080/oasis/RBC_Hospital.jws.`

Add Global Users

The following request adds a global user.

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

<addRequest>

  <attributes>
    <attr name="objectclass">
      <value>SafeGlobalUser</value>
    </attr>
    <attr name="username">
      <value>JohnDoe</value>
    </attr>
    <attr name="firstname">
      <value>John</value>
    </attr>
    <attr name="workphone">
      <value>123456</value>
    </attr>
    <attr name="description">
      <value>a sample user</value>
    </attr>
    <attr name="address">
      <value>1500 Dexter Ave</value>
      <value>Seattle</value>
      <value>USA</value>
    </attr>
    <attr name="GroupMembership">
      <value>GRP1</value>
    </attr>
  </attributes>
</addRequest>
```

The following is the response to the preceding request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<addResponse xmlns="urn:oasis:names:tc:SPML:1:0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" result="urn:oasis:names:tc:SPML:1:0#success"
xsi:type="addResponse">
<identifier type="urn:oasis:names:tc:SPML:1:0#UserIDAndOrDomainName">
  <id xsi:type="java:java.lang.String">usr1</id>
</identifier>
</addResponse>
```

Delete Global Users

The following example deletes a global user:

```
<?xml version="1.0" encoding="UTF-8" ?>
<deleteRequest>
<identifier type="um:oasis:names:tc:SPML:1:0#GenericString">
<id>/info/data/groupname</id>
<identifierAttributes>
  <attr name="objectclass">
    <value>/info/data/clsName</value>
  </attr>
</identifierAttributes>
</identifier>
  <info>
    <data>
      <groupname>abc</groupname>
      <clsName>SafeGlobalUserGroup</clsName>
    </data>
  </info>
</deleteRequest>
```

The following is the response for the preceding delete global user request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<deleteResponse xmlns="um:oasis:names:tc:SPML:1:0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" result="um:oasis:names:tc:SPML:1:0#success" xsi:type="deleteResponse" />
```

Modify Global User

The following request modifies a global user:

```
<?xml version="1.0" encoding="UTF-8" ?>
<modifyRequest>
<identifier type="um:oasis:names:tc:SPML:1:0#GenericString">
  <id>/info/data/username</id>
  <identifierAttributes>
    <attr name="objectclass">
      <value>/info/data/clsName</value>
    </attr>
  </identifierAttributes>
</identifier>
<modifications>
  <modification name="office" operation="replace">
    <value>ER1</value>
  </modification>
</modifications>
<info>
  <data>
    <username>usr1</username>
    <clsName>SafeGlobalUser</clsName>
  </data>
</info>
</modifyRequest>
```

The following is the response for the preceding modify global user request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<modifyResponse xmlns="um:oasis:names:tc:SPML:1:0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" result="um:oasis:names:tc:SPML:1:0#success" xsi:type="modifyResponse" />
```

Search for Global User

The following request searches for a global user.

```
<?xml version="1.0" encoding="UTF-8" ?>
<searchRequest>
  <searchBase type="um:oasis:names:tc:SPML:1:0#GenericString">
    <id>SafeGlobalUserGroup</id>
  </searchBase>
  <filter>
    <approxMatch name="cn">
      <value>*</value>
    </approxMatch>
  </filter>
  <attributes>
    <attribute name="name" />
    <attribute name="path" />
    <attribute name="description" />
  </attributes>
</searchRequest>
```

The following is the response for the preceding request to search for a global user:

```
<?xml version="1.0" encoding="UTF-8" ?>
<searchResponse xmlns="um:oasis:names:tc:SPML:1:0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" result="um:oasis:names:tc:SPML:1:0#success" xsi:type="searchResponse">
  <searchResultEntry>
    <attributes>
      <attr name="name">
        <ns1:value>
xmlns:ns1="um:oasis:names:tc:DSML:2:0:core">Chiefs</ns1:value>
        </attr>
      <attr name="path">
        <ns2:value>
xmlns:ns2="um:oasis:names:tc:DSML:2:0:core">/Chiefs</ns2:value>
        </attr>
      <attr name="description">
        <ns3:value>
xmlns:ns3="um:oasis:names:tc:DSML:2:0:core">Chiefs</ns3:value>
        </attr>
    </attributes>
  </searchResultEntry>
</searchResponse>
```

Schema for a Global User

The following is a request for the schema of a global user.

```
<?xml version="1.0" encoding="UTF-8" ?>
<schemaRequest execution="um:oasis:names:tc:SPML:1:0#synchronous">
  <providerIdentifier providerIDType="um:oasis:names:tc:SPML:1:0#URN">
    <providerID>eiam</providerID>
  </providerIdentifier>
  <schemalIdentifier schemalIDType="um:oasis:names:tc:SPML:1:0#GenericString">
    <schemalID>SafeGlobalUser</schemalID>
  </schemalIdentifier>
</schemaRequest>
```

The following is the response for the preceding request for the schema of a global user:

```
<?xml version="1.0" encoding="UTF-8" ?>
<schemaResponse xmlns="um:oasis:names:tc:SPML:1:0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" result="um:oasis:names:tc:SPML:1:0#success" xsi:type="schemaResponse">
<schema>
  <providerIdentifier providerIDType="um:oasis:names:tc:SPML:1:0#URN">
    <providerID xsi:type="java:java.lang.String">CA EEM</providerID>
  </providerIdentifier>
  <schemalIdentifier schemalIDType="um:oasis:names:tc:SPML:1:0#GenericString">
    <schemalID xsi:type="java:java.lang.String">SafeGlobalUser</schemalID>
  </schemalIdentifier>
  <attributeDefinition name="UserName" multivalued="false" type="string" />
  <attributeDefinition name="JobTitle" multivalued="false" type="string" />
  <attributeDefinition name="MiddleName" multivalued="false" type="string" />
  <attributeDefinition name="ChangePasswordNextLogin" multivalued="false" type="boolean" />
  <attributeDefinition name="FirstName" multivalued="false" type="string" />
  <attributeDefinition name="PasswordTimeToWarn" multivalued="false" type="boolean" />
  <attributeDefinition name="SuspendedDate" multivalued="false" type="date-time" />
  <attributeDefinition name="OldPasswordDigest" multivalued="true" type="string" />
  <attributeDefinition name="MobilePhoneNumber" multivalued="false" type="string" />
  <attributeDefinition name="MailStop" multivalued="false" type="string" />
  <attributeDefinition name="Name" multivalued="false" type="string" />
  <attributeDefinition name="Department" multivalued="false" type="string" />
  <attributeDefinition name="DisableDate" multivalued="false" type="date-time" />
  <attributeDefinition name="Company" multivalued="false" type="string" />
  <attributeDefinition name="Office" multivalued="false" type="string" />
  <attributeDefinition name="City" multivalued="false" type="string" />
  <attributeDefinition name="Description" multivalued="false" type="string" />
  <attributeDefinition name="OverridePasswordPolicy" multivalued="false" type="boolean" />
  <attributeDefinition name="PasswordExpireTime" multivalued="false" type="date-time" />
  <attributeDefinition name="IncorrectLoginCount" multivalued="false" type="integer" />
  <attributeDefinition name="PasswordChangeDate" multivalued="false" type="date-time" />
  <attributeDefinition name="Alias" multivalued="false" type="string" />
  <attributeDefinition name="Suspended" multivalued="false" type="boolean" />
```

```
<attributeDefinition name="Comments" multivalued="true" type="string" />
<attributeDefinition name="DisplayName" multivalued="false" type="string" />
<attributeDefinition name="PostalCode" multivalued="false" type="string" />
<attributeDefinition name="Path" multivalued="false" type="string" />
<attributeDefinition name="Country" multivalued="false" type="string" />
<attributeDefinition name="State" multivalued="false" type="string" />
<attributeDefinition name="Address" multivalued="true" type="string" />
<attributeDefinition name="GroupMembership" multivalued="true" type="string" />
<attributeDefinition name="FaxPhoneNumber" multivalued="false" type="string" />
<attributeDefinition name="EmailAddress" multivalued="false" type="string" />
<attributeDefinition name="EnableDate" multivalued="false" type="date-time" />
<attributeDefinition name="PasswordDigest" multivalued="false" type="string" />
<attributeDefinition name="WorkPhoneNumber" multivalued="false" type="string" />
<attributeDefinition name="LastName" multivalued="false" type="string" />
<attributeDefinition name="HomePhoneNumber" multivalued="false" type="string" />
</schema>
</schemaResponse>
```


Chapter 12: Event Management

This section contains the following topics:

[Event Policies](#) (see page 153)

[Event Data Model](#) (see page 156)

[Reliable Event Delivery](#) (see page 161)

[Route Events](#) (see page 163)

Event Policies

CA EEM generates events based on the event policies defined in the application. Event policies are used to determine which events are delivered, and which ones are combined (coalesced) into summaries. By using event policies, you can configure the events that must be reported in detail.

How Event Policies are Evaluated

CA EEM generates combined events for both administrative and runtime events. The events generated by CA EEM are recognized by CA Audit and can be delivered to a configured CA Audit system.

The event policies are evaluated as follows:

- All events are coalesced and sent out based on the time set in `Safe::Context::setEventCoalesceTime`.
- By default, events are delivered to the backend server. This can be overridden by setting a new event host in the `Safe::Context::setEventHost`.
- When an event is received, an access check is performed to determine if the event must be sent in detail based on the following:
 - `ResourceClassName` set to `SafeEvent`
 - Action set to `submit`
 - `ResourceName` set to `{action}` from the event
 - Named attribute queue of the event details (Taxonomy, Identity, ResourceClass (for admin) Resource, Error (for runtime))

If the access check is passed, a detailed event is sent.

Controlling Event Delivery

CA EEM coalesces events by unique instances of the host, application instance, and taxonomy (action + success/fail/deny) and forwards the events to the backend server. You can configure the time of delivery for these events. The default value to generate coalesces event is 300 seconds.

Note: Even if the event policies are set to 'not submit' a specified event, Coalesced Events are still generated on a regular interval (defaults to 300 seconds).

The following parameters can be used to modify the default values of events:

Safe::Context::setEventCoalesceTime

Specifies the time span for delivering combined events.

Safe::Context::setEventDeliveryHost

Specifies the Audit host to send the events.

Safe::Context::setEventDrainTime

Specifies time to wait before thrashing the un-sent events.

More Information

[Event Data Model](#) (see page 156)

Default Event Policy

When registering an Application Instance, a default event policy (named DefaultEventPolicy) is created. The default policy sends detailed information on events for the following actions:

- unregisterApplicationInstance
- issueCertificate
- authenticateWithPassword
- authenticateWithCertificate
- authenticateWithArtifact
- authenticateWithDigest
- authenticateWithNative
- authenticateWithCredentials
- fastAuthenticateWithPassword
- fastAuthenticateWithCertificate
- fastAuthenticateWithArtifact
- fastAuthenticateWithDigest
- fastAuthenticateWithNative
- refreshSession
- changePassword
- unlockUser
- pozConfigure
- soInsert
- soRemove
- soModify

Event Data Model

You can view the following events that are cached by CA EEM for the attached application instance:

- Administrative Events
- Runtime Events
- Coalesced Events

Note: The events are displayed based on your rights view.

The following are the standard fields in Audit:

Taxonomy

IAM.*eventname*.{action}.[S|F].I {action}

Where eventname is based on the event model.

Src

Specifies the applicationinstance label from Safe::ApplicationInstance::getLabel method.

Log

Specifies the log file name.

Example: EiamSdk

TimeZone

Specifies the local time offset from GMT.

Location

Specifies the fully qualified hostname (\domain for windows, host.domain.com for *nix).

RecorderHost

Specifies the fully qualified hostname (\domain for windows, host.domain.com for *nix).

Recorder

Specifies the "application name" from Safe::ApplicationInstance::getApplicationName method.

Version

Specifies the version.

Example: 1.0

Administrative Events

Administrative events occur when any SafeStoredObject or folder is inserted, removed, or modified in the policy server store. Admin Events are generated through the following administrative actions:

- pozConfigure
- soInsert (includes addFolder and addGlobalFolder)
- soRemove (includes removeFolder and removeGlobalFolder)
- soModify (includes emptyFolder and emptyGlobalFolder)

Each administrative action generates events along with a 'Tag' field. Each event represents a collection of attributes that are updated, inserted, or removed.

Note: Administrative events use the standard audit fields with Taxonomy set to IAM.Admin.{action}.S.I.

The following are the CA EEM specific fields for administrative events:

Identity

Specifies the identity of the user attached to the context.

Tag

Specifies a unique ID that anchors the group of events together.

Method

Specifies the action.

ResourceClass:

Specifies the resource class being acted upon. Such as, Folder, GlobalFolder, User, GlobalUser, UserGroup, GlobalUserGroup, ApplicationInstance, Calendar, and Policy.

Resource

Specifies the fully qualified name of the resource.

Attribute:

Specifies the name of the field being modified.

OldVal

Specifies the previous value of the attribute.

NewVal

Specifies the new value of the attribute.

Severity

Specifies severity information.

Status

Specifies the status.

Runtime Events

Runtime events are generated when CA EEM methods are invoked, such as authentication and authorization calls. Runtime events are generated through the following runtime actions:

- registerApplicationInstance
- unregisterApplicationInstance
- issueCertificate
- attach
- detach
- authenticateWith*
- fastAuthenticateWith*
- authorizeWithSession
- refreshSession
- removeSession
- changePassword
- changePasswordForIdentity
- unlockUser

Note: Runtime events use the standard audit fields with Taxonomy set to IAM.Runtime.{action}.[S|F].I {action}.

The following are the CA EEM specific fields for runtime events:

Identity

Specifies the identity of the user attached to the context.

Note: If you use authorizeWithSession, then it specifies the identity being checked.

Action

Specifies the action.

Resource

Specifies the resource being acted upon (identity, application instance, or resource).

For authorizeWithSession the resource is the action + '/' + ResourceClassName + '/' + Resource

Example: read/file//tmp/myfile.doc

Error

Specifies the error in numerics.

ErrorCode

Specifies the error code (nmemonic).

Severity

Specifies severity information.

Status

Specifies the status.

Coalesced Events

Coalesced Events are runtime and administrative events, coalesced into 'counts'. Each unique instance of action and success/failure generates a bucket, into which the events are coalesced. Coalesced Events are delivered on a configurable basis to the CA EEM policy server.

Note: Coalesced events use the standard audit fields with Taxonomy set to IAM.Coalesced.{action}.[S|F].I {action}.

The following are the CA EEM specific fields for coalesced events:

Method

Specifies the action.

StartTime

Specifies the time to start coalescing.

StopTime

Specifies the time to stop coalescing.

Count

Specifies the number of these events received/coalesced.

Severity

Specifies severity information.

Status

Specifies the status.

Reliable Event Delivery

CA EEM generates events based on the event policies defined in the application. To ensure all the events generated from the clients reach the CA EEM Server, you must enable Reliable Event Delivery.

Note: All admin events are captured and delivered by default. The admin events cannot be controlled using event policies.

Enable Reliable Event Delivery

You can enable Reliable Event Delivery by using one of the following methods:

Note: Reliable Event Delivery is not enabled by default.

- [Calling setSafLocation Method](#) (see page 162)
- [Using Web Interface](#) (see page 162)

Using SDK Method

You can enable Reliable Event Delivery only once for each safecontext session.

To enable Reliable Event Delivery, you must specify the path to store events by calling the setSafLocation method.

Example: Enable Reliable Event Delivery

The following example enables reliable event delivery in CA EEM

```
//Instantiate a SafeContext Class.  
SafeContext safecontext = new SafeContext();  
//Call the setSafLocation method to enable reliable event delivery  
safecontext.setSafLocation ("C:\\<Folder Name>")  
//Set the backend to the host where Server is running.  
safecontext.setBackend("hostname");
```

Using Web Interface

You can enable Reliable Event Delivery by providing the SAF location using the web interface.

To set the SAF location

1. Log on to the CA EEM Server as EiamAdmin user.
The CA EEM home page appears. For information on how to log on see, *Getting Started guide*.
2. Click Configure and click Embedded IAM Server.
The Embedded IAM Server pane appears.
3. Click Configure SAF Location.
The SAF Configuration page appears.
4. Specify the SAF File Location and Save.

Example: C:\\<Folder Name>

The location is set to enable Reliable Event Delivery.

Route Events

Events that are generated in CA EEM can be routed to another server through the iGateway iControl configuration settings. You must perform the following steps in the server from where you want the events to be routed.

Note: After you enable event routing, events will still be available on the CA EEM Server.

To route events from CA EEM

1. Stop the iGateway service:

Windows

```
net stop igateway
```

Linux and UNIX

```
$(GW_LOC)/S99gateway stop
```

2. Go to the iTechnology installation folder.

Windows (Default)

```
\CA\SharedComponents\iTechnology
```

Linux and UNIX

```
/opt/CA/SharedComponents/iTechnology
```

3. Edit the iControl.conf file and modify the RouteEvent and RouteEventHost tags.

Example:

```
<RouteEvent>>false</RouteEvent>
```

Change to:

```
<RouteEvent>>true</RouteEvent>
```

Example:

```
<RouteEventHost>localhost</RouteEventHost>
```

Change to:

```
<RouteEventHost><hostname></RouteEventHost>
```

4. Start the iGateway service:

Windows

```
net stop igateway
```

Linux and UNIX

```
$IGW_LOC/S99gateway stop
```

All the events that are generated after configuring will be routed to the specified host.

Chapter 13: Server Configuration

This section contains the following topics:

[Server Configuration](#) (see page 165)

[Using Java Authentication and Authorization Service](#) (see page 166)

Server Configuration

You can retrieve and modify the CA EEM Server configuration. To retrieve information on CA EEM Server configuration you must use the `SafeContext.pozInfo` method.

Note: To use these methods, you must have administrative scoping privileges to read (`pozInfo`) and write (`pozConfigure`) the "iPoz" object.

To configure CA EEM Server configuration you must use the following methods based on the environment:

C++

`Safe::Context::pozConfigure`

Java

`SafeContext.pozConfigure`

C#

`SafeContext.pozConfigure`

Safex

`<GlobalSettings>`

Using Java Authentication and Authorization Service

You can use the Java Authentication and Authorization Service (JAAS) with CA EEM. CA EEM has a JAAS LoginModule (com.ca.eiam.jaas.EiamLoginModule) that authenticates against CA EEM. The EiamLoginModule implements the following LoginModule interface methods:

Initialize

The initialize method initializes the EiamLoginModule with relevant CA EEM authentication information.

Login

The login method authenticates the credentials using CA EEM. It will get the credentials and attempt to use one of the AuthenticateWith method to authenticate the user. Upon a successful authentication, the users group is loaded (global and non-global).

Logout

The logout method logs out a subject.

Commit

The commit method commits the authentication process.

Abort

The Abort method stops the authentication process.

Example: Using LoginModule for a Non-Web Application

The following example displays how to use the LoginModule for a non-web application:

```
LoginContext lc = null;
try
{
    //the EiamLoginModule must be configured as ca_portal_security
    lc = new LoginContext("ca_portal_security", new TextCallbackHandler());
}
catch (LoginException le)
{
    logger.error("Cannot create LoginContext. ", le);
    System.exit(-1);
}
catch (SecurityException se)
{
    logger.error("Cannot create LoginContext. ", se);
    System.exit(-1);
}
try
{
```

```
lc.login();
}
catch (LoginException e)
{
    logger.error("Login failed.");
    System.exit(-1);
}
logger.info("Login Successful: "+lc.getSubject());
```

In the case of the sample above, the configuration file would contain:

```
ca_portal_security {
    com.ca.eiam.jaas.EiamLoginModule required;
};
```


Chapter 14: Managing with CA Products

This section contains the following topics:

[Provisioning through CA Admin](#) (see page 169)

[Security Management with CA SCC](#) (see page 170)

Provisioning through CA Admin

CA Admin is a user-provisioning product that is used to manage, manipulate, and provision to multiple namespaces.

CA EEM is a CA Admin-managed namespace, as a result, any application that integrates with CA EEM can have its users and user groups provisioned by CA Admin.

CA Admin offers enterprise administrators the following identity and access management features:

- Set up a single provisioning policy for multiple CA EEM application instances
- Synchronize CA EEM global user information with other namespaces (when global users are stored in the CA-MDB)
- Manage the external directory that CA EEM references (when referencing global users in an external directory)

For more information on configuring CA Admin to provision to CA EEM applications, see CA Admin's documentation.

Security Management with CA SCC

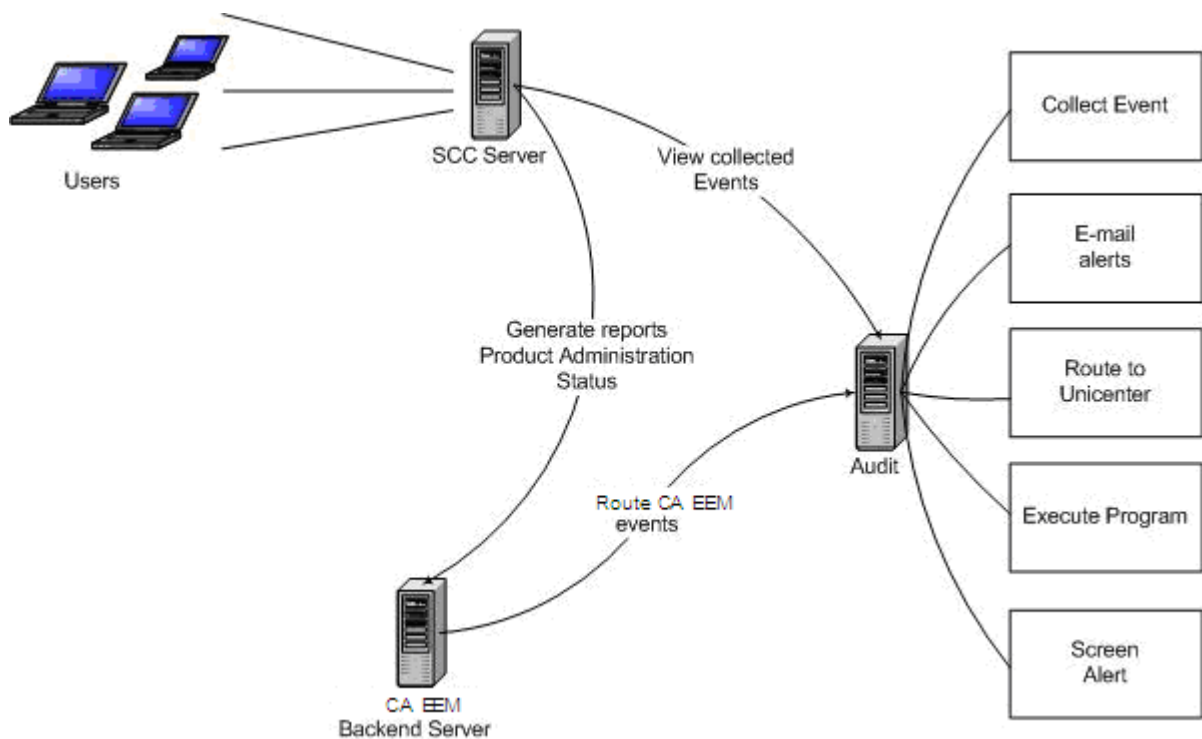
The CA Security Command Center (CA SCC) provides enterprises with a centralized security management console from which you can monitor and manage multiple security applications.

CA EEM is a CA SCC-managed security application, as a result, any application that integrates with CA EEM can be managed by CA SCC.

CA SCC offers the enterprise administrator the following security management features:

- Reporting and Analysis of the backend server
- Event viewing
- Backend server status
- Access to product administration tasks

The following illustration displays the basic architecture:



Reporting and Analysis

The reporting and analysis component contains general backend configuration information, repository information, and application data. The following are the details:

General Configuration

Provides details about the backend configuration settings and event settings for the given backend server.

Repository Information

Provides backend statistics, directory information, password policy information, session data, and application statistics.

Application Details

Application details include the name, label, brand, version, and install information (host name, address, identity, and date).

Work with Audit Events

You can perform the following tasks using the audit events by configuring and customizing CA EEM events:

- Take direct action on an event. This includes sending an e-mail alert, executing a program, and routing events
- Save the event in the collector database
- Send the event to the security monitor

Note: The status monitor overview displays the status of all the backend servers administered by SCC. You can monitor the status to the component level of CA EEM.

Chapter 15: Sample WorkFlow

This section contains the following topics:

[Overview](#) (see page 173)

[Defining Identity and Access Requirements](#) (see page 174)

[Designing Safe Objects to Implement](#) (see page 175)

[Designing the User Interface](#) (see page 188)

[Migrating](#) (see page 188)

[Modifying StoredObjects](#) (see page 190)

Overview

To explain the workflow, a sample hospital management software package is created.

To create the software package, enable, and embed the application in CA EEM, you must define the following objects:

1. Identity and access requirements
2. Safe objects to implement
3. User interfaces
4. Migrate existing application data

More Information

[Defining Identity and Access Requirements](#) (see page 174)

[Designing Safe Objects to Implement](#) (see page 175)

[Designing the User Interface](#) (see page 188)

[Migrating](#) (see page 188)

Defining Identity and Access Requirements

To specify the application's identity and access requirements you must define the business policies for your application.

The following are the sample business policies for the hospital management software:

- Business policies for medical records
 - Medical records can be read/written by a patient's doctor
 - Any doctor or nurse in the patient's ward can read a patient's medical record
 - The chief doctor and nurse can read any patient's medical records
- Business policies for patients
 - Patients can only be admitted to the ER, and only by ER staff or the patient's doctor
 - A patient's doctor can discharge the patient, and prescribe them medicine
 - The Chief doctor and nurse can transfer patients between wards
 - Any doctor in patient's ward can discharge and prescribe medication to a patient
 - Any doctor or nurse can locate any patient
 - Receptionists can locate patients during visiting hours
- Business policies for wards
 - Maintenance and security can enter any ward
 - Any employee can enter their assigned ward
 - The chief doctor and nurse can enter all wards except the office
- Business policies for billing records
 - Office employees can read/write billing data
 - The chief doctor and nurse can read billing data

Designing Safe Objects to Implement

After the business policies are defined, you must identify and define the various Safe Objects. The objects to define include:

ResourceClasses

The resource names to use, actions and 'named attributes' that are used in policy evaluation.

User Attributes

The application-specific user attributes that is used in policy evaluation.

Calendars

To limit when policies are effective.

Policies

Rules attached to the users that define their access.

The following table provides the resource classes names, actions/attributes identified based on the business policies:

Resource Class Name	Business Policies	Safe Information
medicalrecords	<ul style="list-style-type: none">read/write by patient's doctorread by any doctor/nurse assigned to the patient's wardread by Chiefs	ResourceClassName: medicalrecord ResourceName: patientid Actions: read, write Named Attributes: patient's ward, patient's doctor User Attributes: staff's ward User Groups: Chiefs, Doctors, Nurses

Resource Class Name	Business Policies	Safe Information
	<ul style="list-style-type: none"> ■ admit to ER by any staff assigned to ER, or the patient's doctor ■ discharge/prescribe by patient's doctor, or any doctor assigned to patient's ward ■ discharge/transfer by global usergroup Chiefs ■ locate by Doctors and Nurses ■ locate by JobTitle Receptionist, during visiting hours 	ResourceClassName: patient ResourceName: patientid Actions: admit, discharge, prescribe, transfer, locate Named Attributes: patient's ward, patient's doctor User Attributes: staff's ward, staff JobTitle User Groups: Chiefs, Doctors, Nurses Calendar: visitinghours
wards	<ul style="list-style-type: none"> ■ entry by Maintenance and Security ■ entry by anybody to their assigned wards ■ entry by Chiefs to everywhere except the office 	ResourceClassName: ward ResourceName: wardname Actions: enter User Attributes: staff's ward User Groups: Chiefs, Maintenance, Security
billingdata	<ul style="list-style-type: none"> ■ read/write by Office users ■ read by Chiefs 	ResourceClassName: billingdata ResourceName: patientid Actions: read, write User Groups: Chiefs, Office

More Information

[Defining the Application Instance](#) (see page 177)

[Defining Calendars](#) (see page 182)

[Defining Policies](#) (see page 185)

Defining the Application Instance

After defining the SafeObjects, you can build your ApplicationInstance object with the resource classes and user attributes.

Example code to register an application using Safex script

```

<Safex>
  <Attach/>
  <Register>
    <ApplicationInstance name="HospitalMgmt" label="elsewhere">
      <Brand>ABC</Brand>
      <MajorVersion>1</MajorVersion>
      <MinorVersion>0</MinorVersion>
      <Description>Demo App</Description>
      <UserAttribute>text:ward</UserAttribute>
      <ResourceClass>
        <Name>medicalrecord</Name>
        <Action>read</Action>
        <Action>write</Action>
        <NamedAttr>doctor</NamedAttr>
        <NamedAttr>ward</NamedAttr>
      </ResourceClass>
      <ResourceClass>
        <Name>patient</Name>
        <Action>admit</Action>
        <Action>discharge</Action>
        <Action>prescribe</Action>
        <Action>transfer</Action>
        <Action>locate</Action>
        <NamedAttr>ward</NamedAttr>
        <NamedAttr>doctor</NamedAttr>
      </ResourceClass>
      <ResourceClass>
        <Name>ward</Name>
        <Action>enter</Action>
      </ResourceClass>
      <ResourceClass>
        <Name>billingdata</Name>
        <Action>read</Action>
        <Action>write</Action>
      </ResourceClass>
    </ApplicationInstance>
  </Register>
</Safex>

```

Example code to register an application using C#

```
// new application instance
SafeApplicationInstance ai = null;
ai = new SafeApplicationInstance();
ai.Context = sc;
ai.Label = "elsewhere";
ai.ApplicationName = "HospitalMgmt";
ai.MajorVersion = "1";
ai.MinorVersion = "0";
ai.Brand = "ABC";
ai.Description = "Demo App";

// user attribute "ward"
ai.addUserAttribute("text:ward");

// resourceclass medicalrecord
SafeResourceClass rc_medicalrecord = new SafeResourceClass();
rc_medicalrecord.Name = "medicalrecord";
rc_medicalrecord.addAction("read");
rc_medicalrecord.addAction("write");
rc_medicalrecord.addNamedAttr("doctor");
rc_medicalrecord.addNamedAttr("ward");
ai.addResourceClass(rc_medicalrecord);

// resourceclass patient
SafeResourceClass rc_patient = new SafeResourceClass();
rc_patient.Name = "patient";
rc_patient.addAction("admit");
rc_patient.addAction("discharge");
rc_patient.addAction("prescribe");
rc_patient.addAction("transfer");
rc_patient.addAction("locate");
rc_patient.addNamedAttr("doctor");
rc_patient.addNamedAttr("ward");
ai.addResourceClass(rc_patient);

// resourceclass ward
SafeResourceClass rc_ward = new SafeResourceClass();
rc_ward.Name = "ward";
rc_ward.addAction("enter");
ai.addResourceClass(rc_ward);

// resourceclass billingdata
SafeResourceClass rc_billingdata = new SafeResourceClass();
rc_billingdata.Name = "billingdata";
rc_billingdata.addAction("read");
rc_billingdata.addAction("write");
ai.addResourceClass(rc_billingdata);
```

```
// register product instance
try
{
    sc.registerApplicationInstance(ai, "mycert.p12", "certpass")
}
catch(SafeException e)
{
    // handle error
}
```

Example code to register an application using Java

```
// new application instance
SafeApplicationInstance ai = null;
ai = new SafeApplicationInstance();
ai.setContext(sc);
ai.setLabel("elsewhere");
ai.setApplicationName("HospitalMgmt");
ai.setMajorVersion("1");
ai.setMinorVersion("0");
ai.setBrand("ABC");
ai.setDescription("Demo App");

// user attribute "ward"
ai.addUserAttribute("text:ward");

// resourceclass medicalrecord
SafeResourceClass rc_medicalrecord = new SafeResourceClass();
rc_medicalrecord.setName("medicalrecord");
rc_medicalrecord.addAction("read");
rc_medicalrecord.addAction("write");
rc_medicalrecord.addNamedAttr("doctor");
rc_medicalrecord.addNamedAttr("ward");
ai.addResourceClass(rc_medicalrecord);

// resourceclass patient
SafeResourceClass rc_patient = new SafeResourceClass();
rc_patient.setName("patient");
rc_patient.addAction("admit");
rc_patient.addAction("discharge");
rc_patient.addAction("prescribe");
rc_patient.addAction("transfer");
rc_patient.addAction("locate");
rc_patient.addNamedAttr("doctor");
rc_patient.addNamedAttr("ward");
ai.addResourceClass(rc_patient);

// resourceclass ward
SafeResourceClass rc_ward = new SafeResourceClass();
rc_ward.setName("ward");
```

```
rc_ward.addAction("enter");
ai.addResourceClass(rc_ward);

// resourceclass billingdata
SafeResourceClass rc_billingdata = new SafeResourceClass();
rc_billingdata.setName("billingdata");
rc_billingdata.addAction("read");
rc_billingdata.addAction("write");
ai.addResourceClass(rc_billingdata);

// register product instance
try
{
    sc.registerApplicationInstance(ai, "mycert.p12", "certpass")
}
catch(SafeException e)
{
    // handle error
}
```

Example code to register an application using C++

```
// new application instance
Safe::ApplicationInstance ai;
ai.setContext(sc);
ai.setLabel("elsewhere");
ai.setApplicationName("HospitalMgmt");
ai.setMajorVersion("1");
ai.setMinorVersion("0");
ai.setBrand("ABC");
ai.setDescription("Demo App");

// user attribute "ward"
ai.addUserAttribute("text:ward");

// resourceclass medicalrecord
Safe::ResourceClass *rc_medicalrecord = new Safe::ResourceClass;
rc_medicalrecord->setName("medicalrecord");
rc_medicalrecord->addAction("read");
rc_medicalrecord->addAction("write");
rc_medicalrecord->addNamedAttr("doctor");
rc_medicalrecord->addNamedAttr("ward");
ai.addResourceClass(rc_medicalrecord);

// resourceclass patient
Safe::ResourceClass *rc_patient = new Safe::ResourceClass;
rc_patient->setName("patient");
rc_patient->addAction("admit");
rc_patient->addAction("discharge");
rc_patient->addAction("prescribe");
```

```
rc_patient->addAction("transfer");
rc_patient->addAction("locate");
rc_patient->addNamedAttr("doctor");
rc_patient->addNamedAttr("ward");
ai.addResourceClass(rc_patient);

// resourceclass ward
Safe::ResourceClass *rc_ward = new Safe::ResourceClass;
rc_ward->setName("ward");
rc_ward->addAction("enter");
ai.addResourceClass(rc_ward);

// resourceclass billingdata
Safe::ResourceClass *rc_billingdata = new Safe::ResourceClass;
rc_billingdata->setName("billingdata");
rc_billingdata->addAction("read");
rc_billingdata->addAction("write");
ai.addResourceClass(rc_billingdata);

// register product instance
if(!sc.registerApplicationInstance(ai, "mycert.p12", "certpass", ee))
{
    // handle error
}
else
{
    // successful registration
}
```

Defining Calendars

You can use calendars to control access to a resource for a selected period. You can create a calendar with label 'visiting hours' and limit the Receptionist's right to locate patients by implementing policies.

Example code to create a calendar using Safex script

```
<Safex>
  <Attach label="elsewhere"/>
  <Add>
    <Calendar folder="/" name="visitinghours">
      <Description>Visiting hours calendar: 10am to noon; 8pm to 9pm</Description>
      <TimeBlock name="morning" type="include" starttime="600" duration="120" recurringtimeinterval="0"
weekdaymask="ALL" monthdaymask="ALL" monthmask="ALL"/>
      <TimeBlock name="evening" type="include" starttime="1200" duration="60" recurringtimeinterval="0"
weekdaymask="ALL" monthdaymask="ALL" monthmask="ALL"/>
    </Calendar>
  </Add>
</Safex>
```

Example code to create a calendar using C#

```
// new calendar
SafeCalendar cal = new SafeCalendar();
cal.Context = sc;
cal.Path = "/visitinghours";
cal.Description = "Visiting hours calendar: 10am to noon; 8pm to 9pm";
// Timeblocks
SafeTimeBlock mornings = new SafeTimeBlock();
mornings.Name = "morning";
mornings.StartTime = 10*60;
mornings.Duration = 2*60;
mornings.setMask(SafeEnum.WeekDay.WD_ALL, SafeEnum.MonthDay.MD_ALL, SafeEnum.Month.M_ALL);
cal.addIncludeTimeBlock(mornings);
SafeTimeBlock evenings = new SafeTimeBlock();
evenings.Name = "evening";
evenings.StartTime = 20*60;
evenings.Duration = 1*60;
evenings.setMask(SafeEnum.WeekDay.WD_ALL, SafeEnum.MonthDay.MD_ALL, SafeEnum.Month.M_ALL);
cal.addIncludeTimeBlock(evenings);
// insert calendar
try
{
  cal.solInsert()
}
Catch (SafeException e)
{
  //handle error
}
```

Example code to create a calendar using Java

```

// new calendar
SafeCalendar cal = new SafeCalendar();
cal.setContext(sc);
cal.setPath("/visitinghours");
cal.setDescription("Visiting hours calendar: 10am to noon; 8pm to 9pm");
// Timeblocks
SafeTimeBlock momings = new SafeTimeBlock();
momings.setName("morning");
momings.setStartTime(10*60);
momings.setDuration(2*60);
momings.setMask(SafeEnum.WeekDay.WD_ALL, SafeEnum.MonthDay.MD_ALL, SafeEnum.Month.M_ALL);
cal.addIncludeTimeBlock(momings);
SafeTimeBlock evenings = new SafeTimeBlock();
evenings.setName("evening");
evenings.setStartTime(20*60);
evenings.setDuration(1*60);
evenings.setMask(SafeEnum.WeekDay.WD_ALL, SafeEnum.MonthDay.MD_ALL, SafeEnum.Month.M_ALL);
cal.addIncludeTimeBlock(evenings);
// insert calendar
try
{
cal.solInsert()
}
Catch (SafeException e)
{
//handle error
}

```

Example code to create a calendar using C++

```

// new calendar
Safe::Calendar cal;
cal.setContext(sc);
cal.setPath("/visitinghours");
cal.setDescription("Visiting hours calendar: 10am to noon; 8pm to 9pm");
// Timeblocks
Safe::TimeBlock *momings = new Safe::TimeBlock;
momings->setName("morning");
momings->setStartTime(10*60);
momings->setDuration(2*60);
momings->setMask(Safe::WD_ALL, Safe::MD_ALL, Safe::M_ALL);
cal.addIncludeTimeBlock(momings);
Safe::TimeBlock *evenings = new Safe::TimeBlock;
evenings->setName("evening");
evenings->setStartTime(20*60);
evenings->setDuration(1*60);
evenings->setMask(Safe::WD_ALL, Safe::MD_ALL, Safe::M_ALL);
cal.addIncludeTimeBlock(evenings);

```

```
// insert calendar
if(!cal.solInsert())
{
    // handle error
}
else
{
    // successful insert
}
```


Defining Policies

The policies implement the business rules. The following are the policies implementing the 'patient' resource class.

Example code to create policy using Safex Script

```

<Safex>
  <Attach label="elsewhere"/>
  <Add>
    <Policy folder="/" name="patient er admission">
      <Description>patient can be admitted to the ER by any staff assigned to ER, or the patient's
doctor</Description>
      <Action>admit</Action>
      <Identity>ug:Doctors</Identity>
      <Identity>ug:Nurses</Identity>
      <ResourceClassName>patient</ResourceClassName>
      <Filter logic="AND" lparens="0" col="name:ward" optype="STRING" oper="EQUAL" val="val:ER"
rparens="0"/>
      <Filter logic="AND" lparens="1" col="name:ward" optype="STRING" oper="EQUAL" val="u:ward"
rparens="0"/>
      <Filter logic="OR" lparens="1" col="name:doctor" optype="STRING" oper="EQUAL" val="gu:UserName"
rparens="1"/>
      <Filter logic="AND" lparens="0" col="ug:Name" optype="STRING" oper="EQUAL" val="val:Doctors"
rparens="2"/>
    </Policy>
    <Policy folder="/" name="patient discharge-prescribe">
      <Description>patient can be discharged/prescribed by patient's doctor, or any doctor assigned to patient's
ward</Description>
      <Action>discharge</Action>
      <Action>prescribe</Action>
      <Identity>ug:Doctors</Identity>
      <ResourceClassName>patient</ResourceClassName>
      <Filter logic="AND" lparens="0" col="name:ward" optype="STRING" oper="EQUAL" val="u:ward"
rparens="0"/>
      <Filter logic="OR" lparens="0" col="name:doctor" optype="STRING" oper="EQUAL" val="gu:UserName"
rparens="0"/>
    </Policy>
    <Policy folder="/" name="patient discharge-transfer">
      <Description>patient can be discharged/transferred by Chiefs</Description>
      <Action>discharge</Action>
      <Action>transfer</Action>
      <Identity>gug:Chiefs</Identity>
      <ResourceClassName>patient</ResourceClassName>
    </Policy>
    <Policy folder="/" name="patient locate doctor-nurse">
      <Description>patient can be located by any doctor or nurse</Description>
      <Action>locate</Action>
      <Identity>ug:Doctors</Identity>
      <Identity>ug:Nurses</Identity>

```

```

    <ResourceClassName>patient</ResourceClassName>
  </Policy>
  <Policy folder="/" name="patient locate receptionist">
    <Description>patient can be located by any Staff receptionist during visiting hours</Description>
    <Action>locate</Action>
    <ResourceClassName>patient</ResourceClassName>
    <Calendar>visitinghours</Calendar>
    <Identity>ug:Staff</Identity>
    <Filter logic="AND" lparens="0" col="gu:JobTitle" optype="STRING" oper="EQUAL" val="val:Receptionist"
rparens="0"/>
  </Policy>
</Add>
</Safex>

```

Example code to create policy using C#

```

// new policy
SafePolicy pol = new SafePolicy();
pol.Context = sc;
pol.Path = "/patient er admission";
pol.Description = "patient can be admitted to the ER by any staff assigned to ER, or the patient's doctor";
pol.ResourceClassName = "patient";
pol.addIdentity("ug:Doctors");
pol.addIdentity("ug:Nurses");
pol.addAction("admit");
pol.addFilter(new SafeFilter(SafeEnum.Logic.AND, 0, "name:ward",
    SafeEnum.OpType.STRING, SafeEnum.Oper.EQUAL, "val:ER", 0);
pol.addFilter(new SafeFilter(SafeEnum.Logic.SAFE_LOGIC_AND, 1, "name:ward",
    SafeEnum.OpType.STRING, SafeEnum.Oper.EQUAL, "u:ward", 0);
pol.addFilter(new SafeFilter(SafeEnum.Logic.OR, 1, "name:doctor",
    SafeEnum.OpType.STRING, SafeEnum.Oper.EQUAL, "gu:UserName", 1);
pol.addFilter(new SafeFilter(SafeEnum.Logic.AND, 0, "ug:Name",
    Safe.Enum.OpType.STRING, Safe.Enum.Oper.EQUAL, "val:Doctors", 2);
// insert policy
try
{
    pol.solInsert()
}
Catch (SafeException e)
{
    //handle error
}

```

Example code to create policy using Java

```

// new policy
SafePolicy pol = new SafePolicy();
pol.setContext(sc);
pol.setPath("/patient er admission");
pol.setDescription("patient can be admitted to the ER by any staff assigned to ER, or the patient's doctor");
pol.setResourceClassName("patient");
pol.addIdentity("ug:Doctors");
pol.addIdentity("ug:Nurses");
pol.addAction("admit");
pol.addFilter(new SafeFilter(SafeEnum.Logic.AND, 0, "name:ward",
    SafeEnum.OpType.STRING, SafeEnum.Oper.EQUAL, "val:ER", 0);
pol.addFilter(new SafeFilter(SafeEnum.Logic.SAFE_LOGIC_AND, 1, "name:ward",
    SafeEnum.OpType.STRING, SafeEnum.Oper.EQUAL, "u:ward", 0);
pol.addFilter(new SafeFilter(SafeEnum.Logic.OR, 1, "name:doctor",
    SafeEnum.OpType.STRING, SafeEnum.Oper.EQUAL, "gu:UserName", 1);
pol.addFilter(new SafeFilter(SafeEnum.Logic.AND, 0, "ug:Name",
    Safe.Enum.OpType.STRING, Safe.Enum.Oper.EQUAL, "val:Doctors", 2);
// insert policy
try
{
    pol.solInsert()
}
Catch (SafeException e)
{
    //handle error
}

```

Example code to create policy using C++

```

// new policy
Safe::Policy pol;
pol.setContext(sc);
pol.setPath("/patient er admission");
pol.setDescription("patient can be admitted to the ER by any staff assigned to ER, or the patient's doctor");
pol.setResourceClassName("patient");
pol.addIdentity("ug:Doctors");
pol.addIdentity("ug:Nurses");
pol.addAction("admit");
pol.addFilter(new Safe::Filter(Safe::SAFE_LOGIC_AND, 0, "name:ward",
    Safe::SAFE_OPTYPE_STRING, Safe::SAFE_OPER_EQUAL, "val:ER", 0);
pol.addFilter(new Safe::Filter(Safe::SAFE_LOGIC_AND, 1, "name:ward",
    Safe::SAFE_OPTYPE_STRING, Safe::SAFE_OPER_EQUAL, "u:ward", 0);
pol.addFilter(new Safe::Filter(Safe::SAFE_LOGIC_OR, 1, "name:doctor",
    Safe::SAFE_OPTYPE_STRING, Safe::SAFE_OPER_EQUAL, "gu:UserName", 1);
pol.addFilter(new Safe::Filter(Safe::SAFE_LOGIC_AND, 0, "ug:Name",
    Safe::SAFE_OPTYPE_STRING, Safe::SAFE_OPER_EQUAL, "val:Doctors", 2);
// insert policy
if(!pol.solInsert())

```

```
{
    // handle error
}
else
{
    // successful insert
}
```

Designing the User Interface

After designing and building safe objects, you must integrate the applications user interface with CA EEM web interface. The integration of the user interface avoids applications to create individual screens to manage identities and access policies.

How to Design User Interface

To integrate the user interface, CA EEM supports launch in context from the web application in the following process:

1. Log into the interface using the `Safe::Context::authenticateWithPassword` or `::authenticateWithCertificate`. Application will track the session object associated with the user.
2. Build a `Safe LaunchRequest` object with the requested page, object, action, return page, and other attributes.
3. Application invokes the `Safe::Context::generateURI` method to export the user's session and `LaunchRequest` and return artifacts to the session and request 'encoded' in URI string.
4. Application sends a redirect back to the user's browser, specifying the URI.
5. CA EEM Web user interface receives the request, obtains the session/request artifacts, looks up the objects, and presents the appropriate page.

Upon exiting, CA EEM Web user interface redirects to the specified return page.

Migrating

You can migrate an application from using its product's internal identity and access management to CA EEM. Migration of an application involves understanding and converting the information based on the migrating application's data format and storage methods to create XML files that can then be imported into CA EEM through the Safex utility.

Identity

You can migrate application identities into CA EEM.

To migrate identities

1. Create an XML file that represents the applications global users.
2. Map data into the associated fields for each global user represented in the XML file.

Note: During migration, if the application prompts global users to be referenced from an external directory (Microsoft Active Directory) you can move to step three.

3. Import this data into CA EEM using the XML file created, using the following command:

```
safex -h hostname -u user -p pw -f globaluserimport.xml
```

Note: Do not attach CA EEM with an application instance.

4. Define the following parameters for the application:
 - Prompt user to provide a name of the application instance
Example: If the product is Harvest, the instance might be 'Harvest - Engineering Dept'.
 - For each resource class determine the associated actions, and the attributes.
 - Determine the user attributes for the application. You can use this information to create an XML file for registration.
 - Import information into CA EEM

Note: You must have global users (internally or externally) and your application instance must be defined.
5. Determine the attributes stored in the application for each user (not the global user mode). For each user stored in the application, you must format the XML file by mapping the user data and formatting an AI extended attribute user load file.
6. If the application's internal identity management includes group management, you must add a group membership to each of the users defined and define the AI user group.

More Information

[Access Management](#) (see page 41)

Accessing

If the application includes an internal access control method and you want to implement it using CA EEM, automation can be used to create an CA EEM representation of the application's policies, depending on the complexity and structure of your current ACLs or policies. Alternatively, you can also create the corresponding CA EEM policies through the Web UI.

If the application uses calendars along with access policies, you may can automation for converting them to CA EEM clendars.

Modifying StoredObjects

The StoredObject class provides the interface to the repository on the CA EEM backend server. The following CA EEM objects are derived from the StoredObject class:

- ApplicationInstance
- GlobalUser
- GlobalUserGroup
- User
- UserGroup
- Policy
- Calendar
- AppObject

You can modify StoredObjects as follows, based on the environment:

Action	C++	Java	C#	Safex
Set Context	Safe::StoredObject: :setContext	SafeStoredObject.s etContext	SafeStoredObject.C ontext	
Set Name	Safe::StoredObject: :setPath("/folder/na me")	SafeStoredObject.s etPath("/folder/na me")	SafeStoredObject.P ath = "/folder/name"	<Object name="name" folder="/folder">

Action	C++	Java	C#	Safex
Check Access	Safe::StoredObject: :canIdentityRead	SafeStoredObject.c anIdentityRead	SafeStoredObject.c anIdentityRead	
	Safe::StoredObject: :canIdentityWrite	SafeStoredObject.c anIdentityWrite	SafeStoredObject.c anIdentityWrite	
	Safe::StoredObject: :canContextRead	SafeStoredObject.c anContextRead	SafeStoredObject.c anContextRead	
	Safe::StoredObject: :canContextWrite	SafeStoredObject.c anContextWrite	SafeStoredObject.c anContextWrite	
Retrieve	Safe::StoredObject: :soRetrieve	SafeStoredObject.s oRetrieve	SafeStoredObject.s oRetrieve	
	Safe::StoredObject: :soRetrieveByName	SafeStoredObject.s oRetrieveByName	SafeStoredObject.s oRetrieveByName	
	Safe::StoredObject: :soRetrieveByUserN ame	SafeStoredObject.s oRetrieveByUserNa me	SafeStoredObject.s oRetrieveByUserNa me	
Insert	Safe::StoredObject: :soInsert	SafeStoredObject.s oInsert	SafeStoredObject.s oInsert	<Add>
Modify	Safe::StoredObject: :soModify	SafeStoredObject.s oModify	SafeStoredObject.s oModify	<Modify>
Delete	Safe::StoredObject: :soRemove	SafeStoredObject.s oRemove	SafeStoredObject.s oRemove	<Remove>

Folders and Paths

StoredObjects folder path must be named with a fully qualified path. The path is a concatenation of the folder hierarchy (separated by '/'), and the object name. The paths on objects are useful for sorting and organizing objects, and for setting access rights on folders.

Example: Qualified path

A GlobalUser can have the path of `"/North America/Users"` and a name of `"johndoe"`.

- The fully qualified path (`::getPath`) is `"/North America/Users/johndoe"`, and is passed as `"pozPath"` in access checks.
- The parent (`::getParent`) is `"/North America/Users"`, and is passed as `"pozFolder"` in access checks
- The name (`::getName`) is `"johndoe"`, and is passed as `"cn"` in access checks

You can split a fully qualified path into the parent and name by invoking `Safe::Util::splitPath` method.

Search Size

CA EEM limits the number of objects returned in a search to 2000. You can adjust the search size by invoking the `Safe::Context::setMaxSearchSize` method.

If the number of objects returned exceeds the maximum search size, CA EEM will do the following:

Note: These actions are specific to C++ environment.

- Return results upto the maximum size
- Set the `Safe::Error` object error to `EE_MAXSIZEEXCEEDED` (retrieve with `::getErrorCode`)
- Set the `Safe::Error` object's search size to the actual size returned (retrieve with `::getSearchSize`)

Appendix A: Safex Command Line Reference

Safex is a Command Line Interface (CLI) provided by CA EEM. Safex lets you generate XML files to perform product registration, include objects such as policies, users, and calendars. It can also be used to export the data from CA EEM to an XML file.

The safex syntax has the following format:

```
{path} safex [-h backend] [-I locale] [-u user -p password] [-c cert -p password] [-n]
[-v verbose level (0-4)] [-s simulate] [-pause] [-b bulk insert] [-f xml_file]
```

Example: C:\Program Files\CA\Embedded IAM SDK\bin\>safex -h localhost -u eiamAdmin -p eiam -f abc.xml

Where:

- h backend

Specifies the hostname of the CA EEM backend server to communicate.

Note: You can set localhost as your backend server.

- I locale (Optional)

Specifies the language to be used.

Default: us-en

- u user

Specifies the username to attach.

- c certificate

Specifies the certificate filename.

- p password

Specifies the password of the user specified or password used to encrypt the certificate passed with the -c option.

- n (Optional)

Specifies CA EEM to use existing the native authentication.

Note: This parameter is valid only on Windows.

- v verbose (Optional)

Specifies the level of feedback provided by the utility. Higher levels are useful for support personnel.

Limits: 0 - 4

- s simulate (Optional)

Specifies safex to scan the XML file. You can scan the file to verify the syntax and logic, however issues such as trying to add a duplicate object, will not be flagged.

- b bulk (Optional)

Processes large number of objects.

- pause (Optional)

Specifies safex to pause before processing the data and prior to completion.

- f XML file

Specifies the XML file to process.

- munge (Optional)

Converts plaintext password to encrypted versions and displays.

Example: [-munge password1 (password2 ...)]

- digest (Optional)

Converts plaintext password to MD5 hash equivalent and displays.

Note: CA EEM uses MD5 hash format to store global user passwords, when stored in CA-MDB.

Exit Codes

Exit codes are generated when an application encounters an error. The following are the exit codes that are generated by CA EEM and their descriptions:

Exit Code	Description
0	Indicates CA EEM successfully processed the XML input.
1	Indicates usage error. Example: An invalid or incorrect number of parameters entered.
2	Indicates the XML file is not found or an error occurred reading or writing a file.

Exit Code	Description
3	Indicates an error in authentication. Example: Invalid user, certificate, or password.
4	Indicates XML parsing error. XML file line and column number is listed. Example: Tags not matching.
5	Indicates an internal logic error.
6	Indicates no XML data is available to process, Example: Empty file
7	Indicates memory allocation failed.
8	Indicates backend server hostname is invalid or backend server is inactive

Appendix B: Example Safex XML Scripts

This section contains the following topics:

[Register](#) (see page 198)

[Unregister](#) (see page 198)

[Export](#) (see page 199)

[Export Multiple](#) (see page 199)

[CreatedExportMultiple](#) (see page 200)

[Export Global Settings](#) (see page 201)

[Global Settings](#) (see page 201)

[Translations](#) (see page 201)

[Global User](#) (see page 202)

[User](#) (see page 203)

[UserGroups](#) (see page 203)

[GlobalUserGroup](#) (see page 203)

[Policy](#) (see page 204)

[Calendar](#) (see page 205)

[Extended User Attributes](#) (see page 206)

[Sample Application](#) (see page 207)

Register

```
<Safex>
<Attach/>
<Register certfile="appcertfile.p12" password="123">

<ApplicationInstance name="product name" label="application instance label">
<Brand>brand_name</Brand>
<MajorVersion>12</MajorVersion>
<MinorVersion>0</MinorVersion>
<Translations>product_trans_file</Translations>
<Description>description of the product</Description>

<ResourceClass>
<Name>file</Name>
<Action>read</Action>
<Action>write</Action>
<Action>delete</Action>
<NamedAttr>Size</NamedAttr>
<NamedAttr>Count</NamedAttr>
</ResourceClass>

<ResourceClass>
<Name>menu</Name>
<Action>open</Action>
<Action>update</Action>
<NamedAttr>Name</NamedAttr>
</ResourceClass>

<UserAttribute>text:location</UserAttribute>
<UserAttribute>password:extrapassword</UserAttribute>
<UserAttribute>number:pagesize</UserAttribute>
<UserAttribute>text:menu_preference</UserAttribute>
<UserAttribute>number:writefilecount</UserAttribute>

</ApplicationInstance>
</Register>
</Safex>
```

Unregister

```
<Safex>
<Attach/>
<UnRegister>
<ApplicationInstance name="product name" label="application instance label" />
</UnRegister>
</Safex>
```

Export

```
<Safex>
```

```
<Attach label="application name registered with CA EEM"/>
```

<!-- You can control the data to be exported by specifying the Yes(Y) or No(N). If you store the global users and global groups in CA's Management Database (CA-MDB) all the objects are exported.

You can override the maximum number of items that are returned by the backend server. The default is 2000. To change the maximum number of items to return, include the maxsearchsize="Value" -->

```
<Export file="path of XML file to be exported" globalfolders="y" globalusergroups="y" globalusers="y"
globalsettings="y" folders="y" usergroups="y" users="y" calendars="y" policies="y" appobjects="y"/>
```

```
<Detach/>
```

```
</Safex>
```

Export Multiple

```
<Safex>
```

```
<Attach/>
```

<!-- ExportMultiple creates an Export XML file for all of the application instances which match "labelmask".

The mask can contain a leading and/or trailing asterisk to match a subset of the registered application instances. A mask of "*" or omitting label will create an Export file for all application instances. In addition to file= and label=, all of the "y/n" object attribute switches that can be passed to Export can also be specified on the ExportMultiple line. These switches are then passed on to the created Export lines. These include:

```
globalsettings="y/n"
globalfolders="y/n"
globalusergroups="y/n"
globalusers="y/n"
folders="y/n"
usergroups="y/n"
users="y/n"
calendars="y/n"
policies="y/n" -->
```

```
<ExportMultiple file="filename" label="labelmask*" />
```

```
</Safex>
```

CreatedExportMultiple

```
<Safex>
```

<!-- This file was created using the ExportMultiple action and a label of "jld*". All global objects are processed in the first Export pass (note global Attach). This can further be controlled with the ExportMultiple action by specifying the overriding the associated global switches. By default, all global objects will be exported unless an external directory is being used in which case on the globalsettings will be exported. To support exporting multiple application instance data to a single file, two additional attributes were introduced:

Truncate="y/n" - this sets the file pointer to the beginning of the file but leaves it open for subsequent Export actions. This is used by the initial Export action which handles global objects in order to reset the file while leaving it open.

Append="y/n" - positions the file pointer at the end of the file for the subsequent Export actions. -->

```
<Attach />
```

```
<Export file="_expmulti.xml" truncate="y" globalfolders="n" globalusergroups="n" globalusers="n" globalsettings="y" />
```

```
<Detach />
```

```
<Attach label="jld test app" />
```

```
<Export file="_expmulti.xml" append="y" globalfolders="n" globalusergroups="n" globalusers="n" globalsettings="n" folders="y" usergroups="y" users="y" calendars="y" policies="y" />
```

```
<Detach />
```

```
<Attach label="jld2 test app" />
```

```
<Export file="_expmulti.xml" append="y" globalfolders="n" globalusergroups="n" globalusers="n" globalsettings="n" folders="y" usergroups="y" users="y" calendars="y" policies="y" />
```

```
<Detach />
```

```
<Attach label="jld3 test app" />
```

```
<Export file="_expmulti.xml" append="y" globalfolders="n" globalusergroups="n" globalusers="n" globalsettings="n" folders="y" usergroups="y" users="y" calendars="y" policies="y" />
```

```
<Detach />
```

```
<Attach label="jld4 test app" />
```

```
<Export file="_expmulti.xml" append="y" globalfolders="n" globalusergroups="n" globalusers="n" globalsettings="n" folders="y" usergroups="y" users="y" calendars="y" policies="y" />
```

```
<Detach />
```

```
<Attach label="jld elsewhere" />
```

```
<Export file="_expmulti.xml" append="y" globalfolders="n" globalusergroups="n" globalusers="n" globalsettings="n" folders="y" usergroups="y" users="y" calendars="y" policies="y" />
```

```
<Detach />
```

```
</Safex>
```


Export Global Settings

```
<Safex>
<Attach />
<!-- Export only GlobalSettings to the GlobalSettings.xml file -->
<Export file="GlobalSettings.xml" globalsettings="y" globalfolders="n" globalusergroups="n" globalusers="n"
folders="n" usergroups="n" users="n" calendars="n" policies="n" />
</Safex>
```

Global Settings

```
<Safex>
<Attach />
<Add>
<GlobalSettings>
<UseExternalDirectory>true</UseExternalDirectory>
<ExternalDirType>ADS</ExternalDirType>
<ExternalDirHost>usildc04</ExternalDirHost>
<ExternalDirPort>389</ExternalDirPort>
<ExternalDirExchangeGroups>true</ExternalDirExchangeGroups>
<PwUnlockAllowed>true</PwUnlockAllowed>
<PwMinLength>0</PwMinLength>
<PwMaxLength>0</PwMaxLength>
<PwMinNumeric>0</PwMinNumeric>
<PwAllowld>true</PwAllowld>
<PwMinAge>0</PwMinAge>
<PwMaxAge>0</PwMaxAge>
<PwReuseCount>0</PwReuseCount>
<PwFailureCount>0</PwFailureCount>
<PwWarningAge>0</PwWarningAge>
<PwMaxRepeatChar>0</PwMaxRepeatChar>
</GlobalSettings>
</Add>
</Safex>
```

Translations

```
<Safex>
<Attach label="application specific label" />
<!-- Create a file containing all the target strings for translation for a given application -->
<GenerateTranslations file="translations.xml" />
</Safex>
```

Global User

```
<Safex>
<Attach>
<Add>
<GlobalUser folder="/GlobalUsers" name="doejo33">
<UserName>doejo33</UserName>
<GroupMembership>Administrators</GroupMembership>
<FirstName>john</FirstName>
<MiddleName>dennis</MiddleName>
<LastName>doe</LastName>
<EmailAddress>jdoe@acme.com</EmailAddress>
<Alias>jdoe</Alias>
<Department>accounting</Department>
<DisplayName>John D Doe</DisplayName>
<HomePhoneNumber>718-264-8966</HomePhoneNumber>
<WorkPhoneNumber>508-628-7076</WorkPhoneNumber>
<MobilePhoneNumber>508-593-0963</MobilePhoneNumber>
<FaxPhoneNumber>508-628-2319</FaxPhoneNumber>
<Address>331 Main St</Address>
<Address>Jones Building</Address>
<Address>Suite 3200</Address>
<Address>Acme Corp.</Address>
<City>Smallville</City>
<State>Quebec</State>
<PostalCode>H4M2X4</PostalCode>
<Country>Canada</Country>
<Office>C-42</Office>
<Company>Acme</Company>
<PasswordDigest>xxxxxxxxxxxx</PasswordDigest>
<IncorrectLoginCount>0</IncorrectLoginCount>
<SuspendDate>0</SuspendDate>
<DisableDate>0</DisableDate>
<EnableDate>0</EnableDate>
<Description>Working in Finance</Description>
<Comments>12 month temp</Comments>
<JobTitle>Billing Manager</JobTitle>
<MailStop>C-42-2-12</MailStop>
</GlobalUser>
</Add>
</Safex>
```

User

```

<Safex>
<Add>
<User folder="/Users" name="doejo33">
<!-- UserAttribues follow (see the Register file)-->
<location>Cube 333</location>
<menu_preference>Accounts Receivable</menu_preference>
</User>
</Add>
</Safex>

```

UserGroups

```

<Safex>
<Attach label="application instance label" />
<Add>
<UserGroup folder="/" name="salesman">
<Description>Sales team</Description>
</UserGroup>
<UserGroup folder="/" name="marketing">
<Description>Marketing team</Description>
</UserGroup>
<UserGroup folder="/" name="engineering">
<Description>Marketing team</Description>
</UserGroup>
</Add>
</Safex>

```

GlobalUserGroup

```

<Safex>
<Attach />
<Add>
<GlobalUserGroup folder="/" name="Staff">
<Description>Staff group description</Description>
</GlobalUserGroup>
<GlobalUserGroup folder="/" name="Administrators">
<GroupMembership>Staff</GroupMembership>
<Description>Administrator group description</Description>
</GlobalUserGroup>
</Add>
</Safex>

```

Policy

```
<Safex>
<Attach label="application instance label" />
<Add>
<Policy folder="/Policies" name="policyname">
<Description>policy description</Description>
<Calendar>workday</Calendar>
<Identity>u.name</Identity>
<Identity>ug.ProductAdministrators</Identity>
<Identity>gug.Administrators</Identity>
<Action>read</Action>
<Action>write</Action>
<ResourceClassName>file</ResourceClassName>
<Resource>*.doc</Resource>
<Resource>*.txt</Resource>

<!-- You can set filters for further enhancement of policy -->

<Filter logic="OR" lparens="1" col="size" optype="INT32" oper="GREATER" val="10240" rparens="1" />
<Filter logic="AND" lparens="1" col="count" optype="INT32" oper="EQUAL" val="1" rparens="1" />
</Policy>
</Add>
</Safex>
```

Calendar

```
<Safex>
<Attach label="application specific label" localtimeoffset="0" />
<Add>
<Calendar folder="/Calendars" name="workday">
```

<!-- EffectiveStart and EffectiveStop:

You can specify date and time when the calendar must effective by specifying an integer value.

NOTE: If you want to set as be permanently effective, specify as zero or remove the tags

→

```
<EffectiveStart>nnnnnn</EffectiveStart>
<EffectiveStop>nnnnnn</EffectiveStop>
<Description>workdays except xmas</Description>
```

<!-- Timeblock values:

Name : Provide a descriptive name

Type : Include or Exclude

Starttime : Minutes from midnight

Duration : Specify the duration

Recurringtimeinterval : Repeat interval.

weekdaymask : 0...7|ALL (1=Sunday, 7=Saturday)

monthdaymask : 0...31|LAST|ALL (LAST is the last day of the month)

Example: monthdaymask="15 LAST"

monthmask : 0...12|ALL (1=January, 12=December)

→

```
<TimeBlock type="include" name="weekdays" starttime="480" duration="600" recurringtimeinterval="0"
weekdaymask="2 3 4 5 6" monthdaymask="ALL" monthmask="ALL" />
<TimeBlock type="exclude" name="xmas" starttime="0" duration="1440" recurringtimeinterval="0"
weekdaymask="ALL" monthdaymask="25" monthmask="12" />
</Calendar>
```

</Add>

</Safex>

Extended User Attributes

```
<Safex>
<Attach label="application instance label"/>
<Add>
<User folder="users" name="Doe">
<Attribute name="favorite color">Red</Attribute>
<Attribute name="make of car">PSK</Attribute>
</User>
<User folder="users" name="John">
<Attribute name="favorite color">Blue</Attribute>
<Attribute name="make of car">MWR</Attribute>
</User>
<User folder="users" name="Jane">
<Attribute name="favorite color">Green</Attribute>
<Attribute name="make of car">ABC</Attribute>
</User>
</Add>
</Safex>
```

Sample Application

```
<Safex>
<Attach />
<Register>

<ApplicationInstance name="HospitalMgmt" label="elsewhere">
<Brand>eTrust</Brand>
<MajorVersion>1</MajorVersion>
<MinorVersion>0</MinorVersion>
<Description>Demo App</Description>
<UserAttribute>text:ward</UserAttribute>

<ResourceClass>
<Name>medicalrecord</Name>
<EventOnAllow>>false</EventOnAllow>
<EventOnDeny>>true</EventOnDeny>
<Action>read</Action>
<Action>write</Action>
<NamedAttr>doctor</NamedAttr>
<NamedAttr>ward</NamedAttr>
</ResourceClass>

<ResourceClass>
<Name>patient</Name>
<EventOnAllow>>false</EventOnAllow>
<EventOnDeny>>true</EventOnDeny>
<Action>admit</Action>
<Action>discharge</Action>
<Action>prescribe</Action>
<Action>transfer</Action>
<Action>locate</Action>
<NamedAttr>ward</NamedAttr>
<NamedAttr>doctor</NamedAttr>
<NamedAttr>severity</NamedAttr>
</ResourceClass>

<ResourceClass>
<Name>ward</Name>
<EventOnAllow>>false</EventOnAllow>
<EventOnDeny>>true</EventOnDeny>
<Action>enter</Action>
</ResourceClass>
<ResourceClass>

<Name>billingdata</Name>
<EventOnAllow>>false</EventOnAllow>
<EventOnDeny>>true</EventOnDeny>
<Action>read</Action>
<Action>write</Action>
```

```
</ResourceClass>
</ApplicationInstance>
</Register>

<Attach label="elsewhere" />
<Add>
<GlobalFolder name="/Medical" />
<GlobalFolder name="/Support" />

<GlobalUserGroup>
<Name>Chiefs</Name>
<Description>Chiefs</Description>
</GlobalUserGroup>
<GlobalUser>
<UserName>itworker</UserName>
<PasswordDigest>1bb60096TKFkhGcPLdGBaStf+ydWPA==</PasswordDigest>
<Description>Application Administrator</Description>
<JobTitle>Programmer</JobTitle>
<FirstName>I</FirstName>
<MiddleName>T</MiddleName>
<LastName>Worker</LastName>
<DisplayName>I T Worder</DisplayName>
</GlobalUser>

<GlobalUser>
<UserName>securityguard</UserName>
<PasswordDigest>a11bda10VyQoaxE3XNXIvi0eTdkdHg==</PasswordDigest>
<Description>Security Guard</Description>
<JobTitle>Lieutenant</JobTitle>
<FirstName>Security</FirstName>
<LastName>Guard</LastName>
<DisplayName>Security Guard</DisplayName>
</GlobalUser>

<GlobalUser>
<UserName>icudoctor</UserName>
<PasswordDigest>72d7414aM0c/8wP8K3kSgYulBom8Rg==</PasswordDigest>
<Description>Doctor in the ICU ward</Description>
<JobTitle>Doctor</JobTitle>
<FirstName>ICU</FirstName>
<LastName>Doctor</LastName>
<DisplayName>ICU Doctor</DisplayName>
</GlobalUser>

<GlobalUser>
<UserName>headnurse</UserName>
<PasswordDigest>2d400f64hG0wGZrUnpj6AAAA2HNIAA==</PasswordDigest>
<Description>Head Nurse</Description>
<JobTitle>Nurse</JobTitle>
```



```
<FirstName>Head</FirstName>
<LastName>Nurse</LastName>
<DisplayName>Head Nurse</DisplayName>
<GroupMembership>Chiefs</GroupMembership>
</GlobalUser>

<GlobalUser>
<UserName>emurse</UserName>
<PasswordDigest>40196fd78Mj9sBQyTNS6Dq//pJS/PQ==</PasswordDigest>
<Description>Nurse in the ER ward</Description>
<JobTitle>Nurse</JobTitle>
<FirstName>E</FirstName>
<MiddleName>R</MiddleName>
<LastName>Nurse</LastName>
<DisplayName>E R Nurse</DisplayName>
</GlobalUser>
<GlobalUser>
<UserName>erdoctor</UserName>
<PasswordDigest>ab64688dtMG3DwPWcK9B0jco76RDSA==</PasswordDigest>
<Description>Doctor in the ER ward</Description>
<JobTitle>Doctor</JobTitle>
<FirstName>E</FirstName>
<MiddleName>R</MiddleName>
<LastName>Doctor</LastName>
<DisplayName>E R Doctor</DisplayName>
</GlobalUser>

<GlobalUser>
<UserName>janitor</UserName>
<PasswordDigest>183fe80cesXJZuUIFND5KJ7EdgaYDw==</PasswordDigest>
<Description>maintenance employee</Description>
<JobTitle>Sanitary Engineer</JobTitle>
<FirstName>Jan</FirstName>
<LastName>itor</LastName>
<DisplayName>Jan Itor</DisplayName>
</GlobalUser>
<GlobalUser>
<UserName>receptionist</UserName>
<PasswordDigest>b42b2b99x32WOS+DI4z2cpQqGPOCyg==</PasswordDigest>
<Description>Receptionist</Description>
<JobTitle>Receptionist</JobTitle>
<FirstName>Re</FirstName>
<LastName>ceptionist</LastName>
<DisplayName>Re Ceptionist</DisplayName>
</GlobalUser>

<GlobalUser>
<UserName>officeworker</UserName>
<PasswordDigest>8a612830IC/0rsm9J/iaYJXGC0TXtA==</PasswordDigest>
```

```
<Description>Office Worker for billing</Description>
<JobTitle>Accountant</JobTitle>
<FirstName>Office</FirstName>
<LastName>Worker</LastName>
<DisplayName>Office Worker</DisplayName>
</GlobalUser>

<GlobalUser>
<UserName>icunurse</UserName>
<PasswordDigest>6e02fe6-ZaV/tKavOPm76Q3RjonlKA==</PasswordDigest>
<Description>Nurse in the ICU ward</Description>
<JobTitle>Nurse</JobTitle>
<FirstName>ICU</FirstName>
<LastName>Nurse</LastName>
<DisplayName>ICU Nurse</DisplayName>
</GlobalUser>
<GlobalUser>
<UserName>headdoctor</UserName>
<PasswordDigest>e0f29323E99Xd4QjECegSKnHDrdWhA==</PasswordDigest>
<Description>Head Doctor</Description>
<JobTitle>Doctor</JobTitle>
<FirstName>Head</FirstName>
<LastName>Doctor</LastName>
<DisplayName>Head Doctor</DisplayName>
<GroupMembership>Chiefs</GroupMembership>
</GlobalUser>

<Folder name="/Support" />
<Folder name="/System" />
<Folder name="/Medical" />

<UserGroup>
<Name>Office</Name>
<Description>Office Workers</Description>
</UserGroup>

<UserGroup>
<Name>Nurses</Name>
<Description>Nurses</Description>
</UserGroup>

<UserGroup>
<Name>Maintenance</Name>
<Description>Maintenance</Description>
</UserGroup>

<UserGroup>
<Name>Security</Name>
<Description>Security Personnel</Description>
```

```
</UserGroup>

<UserGroup>
<Name>Staff</Name>
<Description>All Staff</Description>
</UserGroup>
<UserGroup>
<Name>Doctors</Name>
<Description>Doctors</Description>
</UserGroup>

<User>
<Name>itworker</Name>
<GroupMembership>Staff</GroupMembership>
<ward>Office</ward>
</User>

<User>
<Name>icudoctor</Name>
<GroupMembership>Doctors</GroupMembership>
<ward>ICU</ward>
</User>
<User>
<Name>headdoctor</Name>
<GroupMembership>Doctors</GroupMembership>
</User>
<User>

<Name>officeworker</Name>
<GroupMembership>Office</GroupMembership>
<ward>Office</ward>
</User>

<User>
<Name>janitor</Name>
<GroupMembership>Maintenance</GroupMembership>
</User>

<User>
<Name>erdoctor</Name>
<GroupMembership>Doctors</GroupMembership>
<ward>ER</ward>
</User>

<User>
<Name>icunurse</Name>
<GroupMembership>Nurses</GroupMembership>
<ward>ICU</ward>
</User>
```

```
<User>
<Name>emurse</Name>
<GroupMembership>Nurses</GroupMembership>
<GroupMembership>Staff</GroupMembership>
<ward>ER</ward>
</User>

<User>
<Name>headnurse</Name>
<GroupMembership>Nurses</GroupMembership>
</User>

<User>
<Name>securityguard</Name>
<GroupMembership>Security</GroupMembership>
</User>

<User>
<Name>receptionist</Name>
<GroupMembership>Staff</GroupMembership>
</User>

<Calendar>
<Description>Visiting hours calendar: 10am to noon; 8pm to 9pm</Description>
<EffectiveStart>0</EffectiveStart>
<EffectiveStop>0</EffectiveStop>
<TimeBlock type="include" name="morning" starttime="600" duration="120" recurringtimeinterval="0"
weekdaymask="ALL" monthdaymask="ALL" monthmask="ALL" />
<TimeBlock type="include" name="evening" starttime="1200" duration="60" recurringtimeinterval="0"
weekdaymask="ALL" monthdaymask="ALL" monthmask="ALL" />
</Calendar>

<Policy>
<Description>patient's doctor has read/write access to medical record</Description>
<ResourceClassName>medicalrecord</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>read</Action>
<Action>write</Action>
<Identity>ug:Doctors</Identity>
<Filter logic="AND" lparens="0" col="name:doctor" optype="STRING" oper="EQUAL" val="gu:UserName"
rparens="0" />
</Policy>

<Policy>
<Description>office workers can read/write any safeobject except policies</Description>
<ResourceClassName>SafeObject</ResourceClassName>
<PolicyType>policy</PolicyType>
```

```

<Disabled>False</Disabled>
<Action>read</Action>
<Action>write</Action>
<Identity>ug:Office</Identity>
<Filter logic="AND" lparens="0" col="req:resource" optype="STRING" oper="NEQ" val="val:Policy" rparens="0" />
</Policy>

```

```

<Policy>
<Description>patient can be admitted to the ER by any staff assigned to ER, or the patient's doctor</Description>
<ResourceClassName>patient</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>admit</Action>
<Identity>ug:Doctors</Identity>
<Identity>ug:Nurses</Identity>
<Filter logic="AND" lparens="0" col="name:ward" optype="STRING" oper="EQUAL" val="val:ER" rparens="0" />
<Filter logic="AND" lparens="1" col="name:ward" optype="STRING" oper="EQUAL" val="u:ward" rparens="0" />
<Filter logic="OR" lparens="1" col="name:doctor" optype="STRING" oper="EQUAL" val="gu:UserName"
rparens="1" />
<Filter logic="AND" lparens="0" col="ug:Name" optype="STRING" oper="EQUAL" val="val:Doctors" rparens="2"
/>
</Policy>

```

```

<Policy>
<Description>maintenace and security can enter any ward</Description>
<ResourceClassName>ward</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>enter</Action>
<Identity>ug:Maintenance</Identity>
<Identity>ug:Security</Identity>
</Policy>

```

```

<Policy>
<Description>Chiefs can enter any ward except the office</Description>
<ResourceClassName>ward</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>enter</Action>
<Identity>gug:Chiefs</Identity>
<Filter logic="AND" lparens="0" col="req:resource" optype="STRING" oper="NEQ" val="val:OFFICE" rparens="0"
/>
</Policy>

```

```

<Policy>
<Description>anybody can enter their assigned ward</Description>
<ResourceClassName>ward</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>

```

```
<Action>enter</Action>
<Identity>ug:Staff</Identity>
<Filter logic="AND" lparens="0" col="req:resource" optype="STRING" oper="EQUAL" val="u.ward" rparens="0" />
</Policy>
```

```
<Policy>
<Description>it worker who manages the elsewhere application</Description>
<ResourceClassName>SafeObject</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>read</Action>
<Action>write</Action>
<Identity>itworker</Identity>
</Policy>
```

```
<Policy>
<Description>System Default: everybody gets access to their own delegated policies</Description>
<ResourceClassName>SafeObject</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Resource>*</Resource>
<Resource>Policy</Resource>
<Action>read</Action>
<Action>write</Action>
<Filter logic="AND" lparens="1" col="name:ResourceClassName" optype="STRING" oper="EQUAL"
val="val:SafeDelegation" rparens="0" />
<Filter logic="AND" lparens="0" col="gu:UserName" optype="STRING" oper="EQUAL" val="name:Delegator"
rparens="1" />
</Policy>
```

```
<Policy>
<Description>Chiefs can read any billing data</Description>
<ResourceClassName>billingdata</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>read</Action>
<Identity>gug:Chiefs</Identity>
</Policy>
```

```
<Policy>
<Description>Global usergroup Chiefs have read access to all medical records</Description>
<ResourceClassName>medicalrecord</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>read</Action>
<Identity>gug:Chiefs</Identity>
</Policy>
```

```
<Policy>
```

```
<Description>patient can be discharged/prescribed by patient's doctor, or any doctor assigned to patient's
ward</Description>
<ResourceClassName>patient</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>discharge</Action>
<Action>prescribe</Action>
<Identity>ug:Doctors</Identity>
<Filter logic="AND" lparens="0" col="name:ward" optype="STRING" oper="EQUAL" val="u:ward" rparens="0" />
<Filter logic="OR" lparens="0" col="name:doctor" optype="STRING" oper="EQUAL" val="gu:UserName"
rparens="0" />
</Policy>

<Policy>
<Description>patient's ward staff has read access to medical record</Description>
<ResourceClassName>medicalrecord</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>read</Action>
<Identity>ug:Doctors</Identity>
<Identity>ug:Nurses</Identity>
<Filter logic="AND" lparens="0" col="name:ward" optype="STRING" oper="EQUAL" val="u:ward" rparens="0" />
</Policy>

<Policy>
<Description>System Default: administrative access for the installer and the application instance
certificate</Description>
<ResourceClassName>SafeObject</ResourceClassName>
<PolicyType>ad</PolicyType>
<Disabled>False</Disabled>
<Resource>ApplicationInstance</Resource>
<Resource>Calendar</Resource>
<Resource>Policy</Resource>
<Resource>User</Resource>
<Resource>UserGroup</Resource>
<Resource>GlobalUser</Resource>
<Resource>GlobalUserGroup</Resource>
<Resource>Folder</Resource>
<Resource>GlobalFolder</Resource>
<Resource>iPoz</Resource>
<Action>read</Action>
<Action>write</Action>
<Identity>EiamAdmin</Identity>
<Identity>CERT-elsewhere</Identity>
</Policy>

<Policy>
<Description>office workers can read/write any billing data</Description>
<ResourceClassName>billingdata</ResourceClassName>
```

```
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>read</Action>
<Action>write</Action>
<Identity>ug:Office</Identity>
</Policy>
```

```
<Policy>
<Description>Any staff member can attach to elsewhere</Description>
<ResourceClassName>SafeObject</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Resource>ApplicationInstance</Resource>
<Action>read</Action>
<Identity>ug:Staff</Identity>
</Policy>
```

```
<Policy>
<Description>patient can be located by any doctor or nurse</Description>
<ResourceClassName>patient</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>locate</Action>
<Identity>ug:Doctors</Identity>
<Identity>ug:Nurses</Identity>
</Policy>
```

```
<Policy>
<Description>patient can be discharged/transferred by Chiefs</Description>
<ResourceClassName>patient</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>discharge</Action>
<Action>transfer</Action>
<Identity>gug:Chiefs</Identity>
</Policy>
```

```
<Policy>
<Description>it worker delegates his rights to the head doctor for global users and users in the Medical
subfolder</Description>
<ResourceClassName>SafeDelegation</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Delegator>itworker</Delegator>
<Resource>SafeObject/GlobalUser</Resource>
<Resource>SafeObject/User</Resource>
<Action>inherit</Action>
<Identity>headdoctor</Identity>
```



```
<Filter logic="AND" lparens="1" col="req:resource" optype="STRING" oper="EQUAL"
val="val:SafeObject/GlobalUser" rparens="0" />
<Filter logic="AND" lparens="0" col="name:pozPath" optype="STRING" oper="LIKE" val="val:/Medical/"
rparens="1" />
<Filter logic="OR" lparens="1" col="req:resource" optype="STRING" oper="EQUAL" val="val:SafeObject/User"
rparens="0" />
<Filter logic="AND" lparens="0" col="name:pozPath" optype="STRING" oper="LIKE" val="val:/Medical/"
rparens="1" />
</Policy>

<Policy>
<Description>patient can be located by any Staff receptionist during visiting hours</Description>
<ResourceClassName>patient</ResourceClassName>
<PolicyType>policy</PolicyType>
<Disabled>False</Disabled>
<Action>locate</Action>
<Identity>ug:Staff</Identity>
<Calendar>visitinghours</Calendar>
<Filter logic="AND" lparens="0" col="gu:JobTitle" optype="STRING" oper="EQUAL" val="val:Receptionist"
rparens="0" />
</Policy>
</Add>
</Safex>
```


Appendix C: Reference Matrix

The reference matrix lists the actions and the method that must be used to perform the action.

| Action | Safex | Java | C# | C++ |
|-----------------------------------|-----------------|---|---|--|
| Attach to an Application Instance | <Attach/> | SafeContext.attach | SafeContext.attach | Safe::Context::attach |
| Login | <Authenticate/> | SafeContext.authenticateWithPassword | SafeContext.authenticateWithPassword | Safe::Context::authenticateWithPassword |
| | | SafeContext.authenticateWithCertificate | SafeContext.authenticateWithCertificate | Safe::Context::authenticateWithCertificate |
| | | SafeContext.authenticateWithArtifact | SafeContext.authenticateWithArtifact | Safe::Context::authenticateWithArtifact |
| | | SafeContext.authenticateWithNative | SafeContext.authenticateWithNative | Safe::Context::authenticateWithNative |
| | | SafeContext.authenticateWithDigest | SafeContext.authenticateWithDigest | Safe::Context::authenticateWithDigest |
| | | SafeContext.authenticateWithCredentials | SafeContext.authenticateWithCredentials | Safe::Context::authenticateWithCredentials |
| | | SafeContext.fastAuthenticateWithPassword | SafeContext.fastAuthenticateWithPassword | Safe::Context::fastAuthenticateWithPassword |
| | | SafeContext.fastAuthenticateWithCertificate | SafeContext.fastAuthenticateWithCertificate | Safe::Context::fastAuthenticateWithCertificate |
| | | SafeContext.fastAuthenticateWithArtifact | SafeContext.fastAuthenticateWithArtifact | Safe::Context::fastAuthenticateWithArtifact |
| | | SafeContext.fastAuthenticateWithNative | SafeContext.fastAuthenticateWithNative | Safe::Context::fastAuthenticateWithNative |
| | | SafeContext.fastAuthenticateWithDigest | SafeContext.fastAuthenticateWithDigest | Safe::Context::fastAuthenticateWithDigest |
| Logout | | SafeContext.removeSession | SafeContext.removeSession | Safe::Context::removeSession |

| Action | Safex | Java | C# | C++ |
|---------------------------------|-----------------------------------|---|---|--|
| Permission Check | <Perm/> | SafeContext.authorizeWithIdentity | SafeContext.authorizeWithIdentity | Safe::Context::authorizeWithIdentity |
| | | SafeContext.authorizeWithSession | SafeContext.authorizeWithSession | Safe::Context::authorizeWithSession |
| | | SafeContext.authorizeQWithIdentity | SafeContext.authorizeQWithIdentity | Safe::Context::authorizeQWithIdentity |
| | | SafeContext.authorizeQWithSession | SafeContext.authorizeQWithSession | Safe::Context::authorizeQWithSession |
| | | SafeContext.processAuthorizationQ | SafeContext.processAuthorizationQ | Safe::Context::processAuthorizationQ |
| | | SafeContext.processAuthorizationMatrix | SafeContext.processAuthorizationMatrix | Safe::Context::processAuthorizationMatrix |
| | | SafeContext.authorizeWithSessionDebug | SafeContext.authorizeWithSessionDebug | Safe::Context::authorizeWithSessionDebug |
| | | SafeContext.authorizeWithIdentityDebug | SafeContext.authorizeWithIdentityDebug | Safe::Context::authorizeWithIdentityDebug |
| Register Application Instance | <Register/> | SafeContext.registerApplicationInstance | SafeContext.registerApplicationInstance | Safe::Context::registerApplicationInstance |
| Unregister Application Instance | <UnRegister/> | SafeContext.unregisterApplicationInstance | SafeContext.unregisterApplicationInstance | Safe::Context::unregisterApplicationInstance |
| Define User Attributes | <Register/> | SafeApplicationInstance.soModify | SafeApplicationInstance.soModify | Safe::ApplicationInstance.soModify |
| Define Resource Classes | <Register/> | SafeResourceClass | SafeResourceClass | Safe::ResourceClass |
| | | SafeApplicationInstance.addResourceClass | SafeApplicationInstance.addResourceClass | Safe::ApplicationInstance.addResourceClass |
| | | SafeApplicationInstance.soModify | SafeApplicationInstance.soModify | Safe::ApplicationInstance.soModify |
| Change Password | | SafeContext.changePassword | SafeContext.changePassword | Safe::Context::changePassword |
| Unlock User Account | | SafeContext.unlockUser | SafeContext.unlockUser | Safe::Context::unlockUser |
| Insert a Global User | <Add>
<Global User/>
</Add> | SafeGlobalUser.soInsert | SafeGlobalUser.soInsert | Safe::GlobalUser::soInsert |

| Action | Safex | Java | C# | C++ |
|-----------------------|---|-------------------------------------|-------------------------------------|--|
| Search Global User | | SafeContext.searchGlobalUsers | SafeContext.searchGlobalUsers | Safe::Context::searchGlobalUsers |
| Retrieve Global User | | SafeGlobalUser.soRetrieveByName | SafeGlobalUser.soRetrieveByName | Safe::GlobalUser::soRetrieveByName |
| | | SafeGlobalUser.soRetrieveByUserName | SafeGlobalUser.soRetrieveByUserName | Safe::GlobalUser::soRetrieveByUserName |
| | | SafeGlobalUser.soRetrieve | SafeGlobalUser.soRetrieve | Safe::GlobalUser::soRetrieve |
| | | SafeContext.searchGlobalUsers | SafeContext.searchGlobalUsers | Safe::Context::searchGlobalUsers |
| Modify Global User | <Modify Global User/>
</Modify Global User/> | SafeGlobalUser.soModify | SafeGlobalUser.soModify | Safe::GlobalUser::soModify |
| Remove Global User | <Remove Global User/>
</Remove Global User/> | SafeGlobalUser.soRemove | SafeGlobalUser.soRemove | Safe::GlobalUser::soRemove |
| Insert User Details | <Add User/>
</Add User/> | SafeUser.soInsert | SafeUser.soInsert | Safe::User::soInsert |
| Search User Details | | SafeContext.searchUsers | SafeContext.searchUsers | Safe::Context::searchUsers |
| Retrieve User Details | | SafeUser.soRetrieveByName | SafeUser.soRetrieveByName | Safe::User::soRetrieveByName |
| | | SafeUser.soRetrieve | SafeUser.soRetrieve | Safe::User::soRetrieve |
| | | SafeContext.searchUsers | SafeContext.searchUsers | Safe::Context::searchUsers |

| Action | Safex | Java | C# | C++ |
|------------------------|--|--------------------------------|--------------------------------|-----------------------------------|
| Modify User Details | <Modify>
<User/>
</Modify> | SafeUser.soModify | SafeUser.soModify | Safe::User::soModify |
| Remove User Details | <Remove>
<User/>
</Remove> | SafeUser.soRemove | SafeUser.soRemove | Safe::User::soRemove |
| Add Global User | <Add>
<GlobalFolder/>
</Add> | SafeContext.addGlobalFolder | SafeContext.addGlobalFolder | Safe::Context::addGlobalFolder |
| List Global Folders | | SafeContext.getGlobalFolders | SafeContext.GlobalFolders | Safe::Context::getGlobalFolders |
| Remove a Global Folder | <Remove>
<GlobalFolder/>
</Remove> | SafeContext.removeGlobalFolder | SafeContext.removeGlobalFolder | Safe::Context::removeGlobalFolder |
| Empty a Global Folder | | SafeContext.emptyGlobalFolder | SafeContext.emptyGlobalFolder | Safe::Context::emptyGlobalFolder |
| Add a Folder | <Add>
<Folder/>
</Add> | SafeContext.addFolder | SafeContext.addFolder | Safe::Context::addFolder |
| List Folders | | SafeContext.getFolders | SafeContext.Folders | Safe::Context::getFolders |
| Remove a Folder | <Remove>
<Folder/>
</Remove> | SafeContext.removeFolder | SafeContext.removeFolder | Safe::Context::removeFolder |

| Action | Safex | Java | C# | C++ |
|----------------------------|------------------------------------|---|---|--|
| Empty a Folder | | SafeContext.emptyFolder | SafeContext.emptyFolder | Safe::Context::emptyFolder |
| Insert an Access Policy | <Add>
<Policy/>
</Add> | SafePolicy.soInsert | SafePolicy.soInsert | Safe::Policy::soInsert |
| Search Access Policies | | SafeContext.searchPolicies | SafeContext.searchPolicies | Safe::Context::searchPolicies |
| Retrieve an Access Policy | | SafePolicy.soRetrieveByName
SafePolicy.soRetrieve
SafeContext.searchPolicies | SafePolicy.soRetrieveByName
SafePolicy.soRetrieve
SafeContext.searchPolicies | Safe::Policy::soRetrieveByName
Safe::Policy::soRetrieve
Safe::Context::searchPolicies |
| Modify an Access Policy | <Modify>
<Policy/>
</Modify> | SafePolicy.soModify | SafePolicy.soModify | Safe::Policy::soModify |
| Remove an Access Policy | <Remove>
<Policy/>
</Remove> | SafePolicy.soRemove | SafePolicy.soRemove | Safe::Policy::soRemove |
| Search for Access Policies | | SafeContext.searchMatchingPoliciesBySession
SafeContext.searchMatchingPoliciesByIdentity
SafeContext.searchMatchingPoliciesByResource | SafeContext.searchMatchingPoliciesBySession
SafeContext.searchMatchingPoliciesByIdentity
SafeContext.searchMatchingPoliciesByResource | Safe::Context::searchMatchingPoliciesBySession
Safe::Context::searchMatchingPoliciesByIdentity
Safe::Context::searchMatchingPoliciesByResource |
| Insert a Calendar | <Add>
<Calendar/>
</Add> | SafeCalendar.soInsert | SafeCalendar.soInsert | Safe::Calendar::soInsert |

| Action | Safex | Java | C# | C++ |
|------------------------|--------------------------------------|--|--|---|
| Search Access Policies | | SafeContext.searchCalendars | SafeContext.searchCalendars | Safe::Context::searchCalendars |
| Retrieve a Calendar | | SafeCalendar.soRetrieveByName | SafeCalendar.soRetrieveByName | Safe::Calendar::soRetrieveByName |
| | | SafeCalendar.soRetrieve | SafeCalendar.soRetrieve | Safe::Calendar::soRetrieve |
| | | SafeContext.searchCalendars | SafeContext.searchCalendars | Safe::Context::searchCalendars |
| Modify a Calendar | <Modify>
<Calendar/>
</Modify> | SafeCalendar.soModify | SafeCalendar.soModify | Safe::Calendar::soModify |
| Remove a Calendar | <Remove>
<Calendar/>
</Remove> | SafeCalendar.soRemove | SafeCalendar.soRemove | Safe::Calendar::soRemove |
| Launch In Context | | SafeLaunchRequest
SafeContext.generateURI | SafeLaunchRequest
SafeContext.generateURI | Safe::LaunchRequest
Safe::Context::generateURI |
| Export a Session | | SafeSession.exportSession | SafeSession.exportSession | Safe::Session::exportSession |
| Import a Session | | SafeContext.authenticateWithArtifact | SafeContext.authenticateWithArtifact | Safe::Context::authenticateWithArtifact |

Index

A

- application instance
 - attach back end server • 16
 - create an instance • 17
 - creating • 16
 - definition • 15
 - obligations • 21
 - register application • 22
 - resource class • 20
 - user attributes • 18

C

- cache • 62
- configure external directory
 - configure external directory • 100
 - configure siteminder • 103

E

- exceptions
 - safe authorization exception • 88
 - safe backendserver exception • 88
 - safe exception • 86
 - safe password exception • 89

F

- Filters
 - policies • 54
 - searches • 50
 - structure • 58

G

- groups
 - application specific groups • 37
 - delete • 40
 - global user groups • 36
 - retrieve • 39

H

- how policies are evaluated • 76

I

- integrating with xacml and spml
 - spml • 135

- xacml • 116

P

- Policies
 - types of policies • 42
- policy evaluation
 - best matching algorithm • 79
 - best matching for regex policies • 80
 - calculating obligations • 84
 - delegated authority evaluation • 82
 - matching algorithm evaluation • 78
 - policy filter evaluation • 81

U

- User Principal Name (UPN) • 102
- users
 - application specific • 27
 - delete • 33
 - global users • 26
 - retrieve • 31