CA Embedded Entitlements Manager

Programming Guide

Release 12.51

CA Technologies Product References

This document references the following CA Technologies products:

- CA Embedded Entitlements Manager (CA EEM)
- CA Directory
- CA SiteMinder®
- CA Integrated Threat Management

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- Using Custom Key Length Certificates for SSL Communication in CA EEM SDK (see page 163)—Added to describe how to use custom key length certificates for the SSL communication.

Contact CA Technologies

Contact CA Support

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

- 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

Providing Feedback About Product Documentation

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

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

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

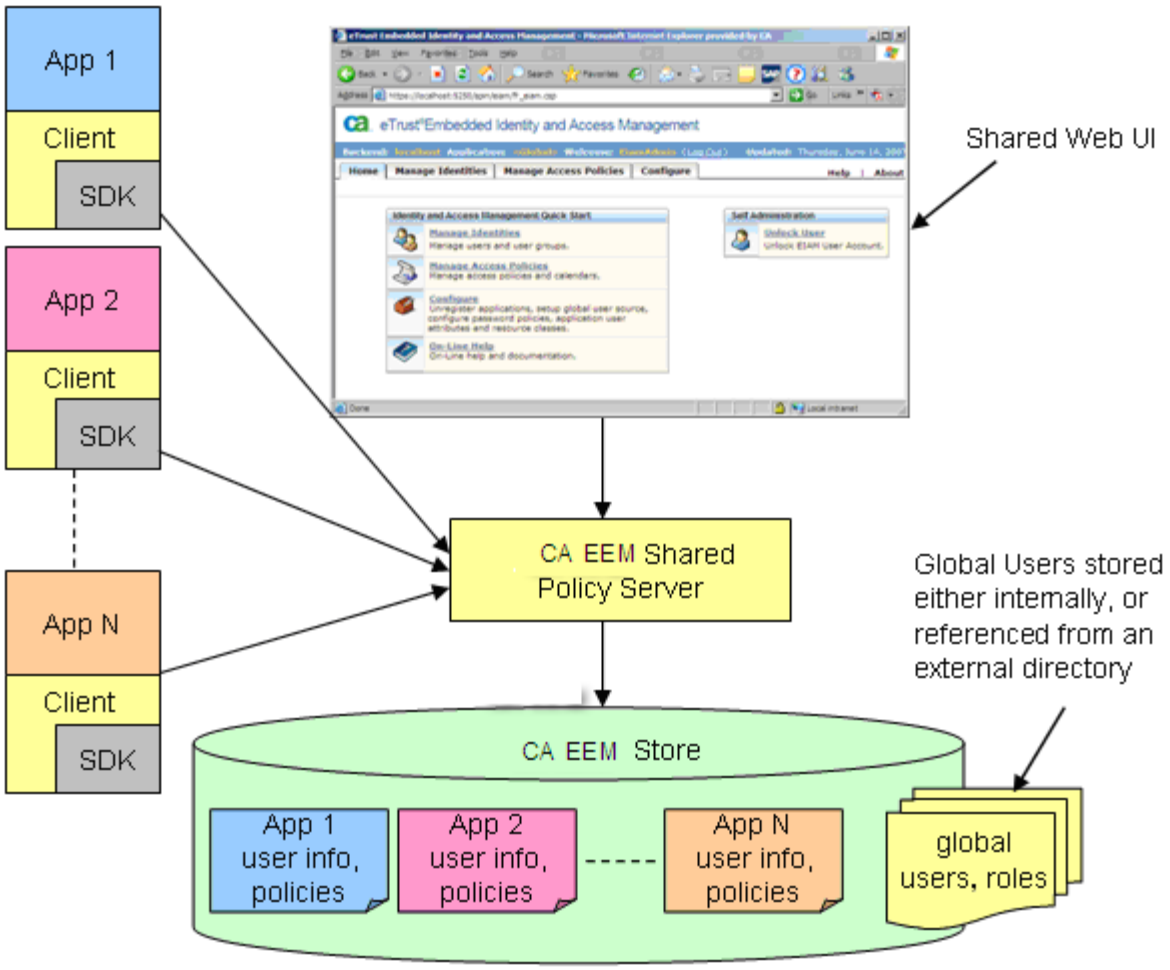
This guide is intended for system programmers, administrators, and integrators who are responsible for defining, monitoring, and managing CA EEM users and tasks. It describes how to integrate CA EEM with your application.

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++
- Java

Architecture

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



More Information:

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 let 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 Entitlements Manager SDK\elsewhere\safex
Safetool source ■ C# ■ C++ ■ Java	C:\Program Files\CA\Embedded Entitlements Manager SDK\safetool

Documentation

The SDK documentation is installed with the CA EEM SDK. The CA EEM SDK documentation is at: <*install_path*>\CA\Embedded Entitlements Manager SDK\Doc.

Default: C:\Program Files\CA\Embedded Entitlements Manager SDK\Doc

It provides the following:

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

Note: For information about installing CA EEM SDK, see the *Implementation Guide*.

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 106)

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: Installing the CA EEM SDK

This section contains the following topics:

Operating System Requirements

To learn about the operating system requirements for the CA EEM SDK, see the CA Embedded Entitlements Manager Compatibility Matrix on the Support site: http://ca.com/support.

System Requirements

The following section describes the CA EEM SDK system requirements.

Windows

The recommended system requirements are as follows:

- An Intel Pentium processor
- 4GB RAM
- 10 GB of hard disk free space
- At least 300 MB disk space is required under the temporary directory %temp% (C:\Documents and Settings\Administrator\Local Settings\Temp\) where the CA EEM installation files are extracted during the installation
- Windows Installer v3 or later
- Winsock-compatible TCP/IP installed and configured

  Windows administrator access to the system
- Internet Browser to run the Web components (Microsoft Internet Explorer 7.0 or higher, or Mozilla Firefox 3.0 or higher).

UNIX and Linux

The recommended system requirements are as follows:

- 4GB RAM
- 10 GB of hard disk free space
- 500 MB disk space is required under the temporary directory (/tmp) where the CA EEM installation files are extracted during the installation.
- A Web browser to run the web components (Firefox 3.0 or higher).

Install on Windows

Install the CA EEM SDK by using one of the following methods:

Interactive installation

Lets you install with user interaction.

Silent installation

Lets you install without any user interaction.

Before installing, gather the required information for the preferred installation parameters.

Install SDK

The CA EEM SDK installation wizard guides you through the installation process.

To install the CA EEM SDK

1. Log in to the computer as a user with administrator privileges.
2. Open the Windows Explorer and double-click the install package EEMSDK_<version number>_win32.exe on the target computer.

   The installation wizard opens.
3. Follow the instructions on the installation wizard to complete the installation.

   CA EEM SDK is installed.

   Note: An environmental variable %EIAM_SDK% is created during the installation to point to the installation path. Use this variable in the explorer path to open the installation folder.

Start SDK

To start CA EEM SDK click Start, All Programs, CA, Embedded Entitlements Manager SDK.

The CA EEM SDK documentation window appears.

Remove SDK

You can uninstall the CA EEM SDK using the Add or Remove Programs of the Control Panel.

Install on UNIX

You can install CA EEM server by performing the following procedure. Before installing, gather the required information for the preferred installation parameters.

Install SDK

CA EEM SDK for Linux and UNIX uses a self-extracting shell script that guides you through the installation process. During the installation process, the script displays the license information and prompts for the installation parameters. After the installation parameters are entered, the installation begins.

To install CA EEM SDK for Linux and UNIX

1. Run the installation script EEMSDK_<version number>_<operating system>.sh on the target computer.

   Example:

   ```
   sh EEMSDK_12.0.0.5_sunos.sh
   ```

   The file is decompressed and installation begins.

2. Enter **Y** to accept the Terms and Conditions of the license agreement (or **N** to decline and abort the installation).

3. Enter the installation path for the CA EEM SDK (or accept the default).

4. Select install product.

   CA EEM SDK is installed on your computer.

   Note: An environmental variable $EIAM_SDK is created during the installation to point to the installation path. Use this variable in the explorer path to open the installation folder.

Remove SDK

You can remove CA EEM SDK from Linux and UNIX operating systems.

To remove CA EEM SDK

1. Log on to the console and navigate to the EiamSDK\Uninstall location.
2. Run eiamuninstall.
3. Follow the instructions on the wizard.

   The CA EEM SDK is uninstalled.

Chapter 3: Configuring CA EEM SDK

This section contains the following topics:

CA EEM SDK Initialization

The following topics explain how to initialize CA EEM SDK. The eiam.config file controls the CA EEM SDK configuration.

About the eiam.config File

Use the eiam.config file to control CA EEM SDK configuration data such as:

- Cyclic buffer
- Logger configuration file
- SAF folder for storing audit files
- FIPS compatible mode
- SafeContext-related information

The eiam.config file consists of the following configurable parameters:

CyclicBuffer size

Specifies the number of log messages contained in a cyclic buffer. The cyclic buffer stores the specified number of latest log messages in the memory. As the buffer reaches the specified size, a new log message replaces the oldest log message in the buffer. If the application crashes, you can recover the latest log messages from the core.

Default: 500

Minimum: 0

Maximum: 1000

Note: This parameter is valid only for the CA EEM C++ SDK.

enable

Specifies if the cyclic buffer is enabled. If enabled is set to false, the cyclic buffer is disabled. So, you need not specify values of the parameters CyclicBuffer size, dump, and file.

Value: [*true*|*false*]

Default: true

Important! Cyclic buffer is enabled by default. If you enable the cyclic buffer, performance of CA EEM is affected.

dump

Specifies if the contents of cyclic buffer are written to a file if the eiam.config file is modified or updated.

Value: [*true*|*false*]

Default: false

file

Specifies filename of the dump file. If dump is set to false, the log messages are not written to a dump file. The file extension of file is .log.

LoggerConfiguration file

Specifies absolute path of the logger configuration files for CA EEM Java, C# SDK, and C++ SDKs. The CA EEM logging information is stored in the logger configuration files. eiam.log4cxx.config, eiam.log4net.config, and eiam.log4j.config are the logger configuration files for CA EEM C++ SDK, CA EEM C# SDK, and CA EEM Java SDK.

audit

SAF folder where audit files are stored for processing.

Note: For more information about SAF Directory, see the Reliable Event Delivery section in the *Programming Guide*.

Network sockettimeout

Specifies the socket timeout in milliseconds.

Default: 120000 (120 seconds)

Note: This parameter is valid only for the CA EEM C++ SDK and CA EEM Java SDK.

<SDK type ="C#">

Specifies the FIPS mode settings for C# SDK.

FIPSMode

Specifies the FIPS mode for CA EEM SDK. For FIPS-only mode, set the value to On.

Value: [Off|On]

Default: Off

digestAlgorithm

Specifies the cryptographic algorithm used to sign server requests. In FIPS mode, CA EEM C# SDK uses SHA1 as the digest algorithm by default. If FIPS mode is disabled, CA EEM C# SDK uses MD5 as the digest algorithm. MD5 is not supported in FIPS-only mode.

Value: [MD5|SHA1]

Default: MD5 for non-FIPS mode.

Note: In FIPS-only mode, CA EEM C# SDK supports only SHA1 as the digest algorithm.

<SDK type ="Java">

Specifies the FIPS mode settings for Java SDK.

FIPSMode

Specifies the FIPS mode for CA EEM SDK. For FIPS-only mode, set the value to On.

Value: [Off|On]

Default: Off

JCEProvider

Specify the Java Cryptography Extension (JCE) provider to use in the FIPS-only mode.

digestAlgorithm

Specifies the cryptographic algorithm used to sign server requests. For CA EEM SDK enabled in FIPS-only mode, use SHA1 as the digestAlgorithm. FIPS does not support MD5. If FIPS-only mode is disabled, the server requests are signed using MD5.

Value: MD5/SHA1/SHA256/SHA384/SHA512

Default: SHA1 for FIPS-only mode and MD5 for non-FIPs mode.

logLevel

Specifies the log level.

Value: [Error|Warning|Trace|Nolevel]

logToFile

Specifies if the log messages must be stored in a file.

Value: [True|False]

Default: False

logFile

Specifies the absolute path to the log file. This parameter is valid only if logToFile is set to True.

maxLogSize

Specifies the maximum size of the log file in MB.

<SDK type ="C++">

Specifies the FIPS mode settings for C++ SDK.

FIPSMode

Specifies the FIPS mode for CA EEM SDK. For FIPS-only mode, set the value to On.

Value: [Off|On]

Default: Off

etpkiCryptoLib

Specifies the installation path for the etpki libraries.

secureProtocol

Specifies the protocol that the CA EEM SDK uses to communicate with CA EEM Server.

Default: SSLV23

Values: SSLV23 / SSLV3 / TLSV1

Note: FIPS-only communication mode supports only TLSV1. Communication fails if you use SSLV2 or SSLV3 when FIPS mode is set to True.

digestAlgorithm

Specifies the cryptographic algorithm used to sign server requests. For CA EEM SDK enabled in FIPS mode, use SHA1 as the digestAlgorithm. FIPS does not support MD5. If FIPS mode is disabled, the server requests are signed using MD5.

Value: MD5/SHA1/SHA256/SHA384/SHA512

Default: MD5 for non-FIPS and SHA1 for FIPS-only mode. If the tag is empty, CA EEM uses the default values.

logLevel

Specifies the log level.

Value: [Error|Warning|Trace|Nolevel]

logToFile

Specifies if the log messages must stored in a file.

Value: [True|False]

Default: False

logFile

Specifies the absolute path to the log file. This parameter is valid only if logToFile is set to True.

maxLogSize

Specifies the maximum size of the log file in MB.

<SafeContext>

Specifies the information required to generate a SafeContext using the SafeContextFactory method.

Note: You can include more than one SafeContext tag in the eiam.config file. However, the refid must be unique for each SafeContext tag.

refid

Specifies the reference ID for a SafeContext tag. This ID must be unique. The SafeContextFactory uses the reference ID to pick the information required to generate a SafeContext.

Backend

Specifies the hostname of the CA EEM Server.

Application

Specifies the name of the application instance for which the SafeContext is generated. If the application name is not specified, SafeContextFactory attaches to the global application.

Locale

Specifies the locale.

Authentication Type

Specifies the authentication that the SafeContextFactory uses to attach to an application. The following are the supported authentication types:

- password-based authentication; use the UserAuth tags for this authentication
- PEM certificates; use the <Certificate type="pem"> for this authentication
- P12 certificates; use the <Certificate type="p12"> for this authentication
- PKCS#11 certificates; use the <Certificate type="p11"> for this authentication

Note: Use only one authentication type with a SafeContext tag.

UserAuth

Specify password-based authentication method.

Username

Specifies the username of the administrator needed to attach to an application instance or global instance.

Password

Specifies the munged password needed to authenticate the administrator.

PEM certificates

Specifies the details required for a PEM certificate-based authentication.

CertURI

Specifies the path including the certificate filename.

KeyURI

Specifies the path including the key file.

KeyPW

Specifies the munged password required to read the certificate file. This tag is valid only for the CA EEM C++ SDK. In FIPS-only mode, this tag must be blank.

PKCS#11 certificates

Specifies the details required for a P11 certificate-based authentication.

Provider

Specifies the path to the encryption libraries.

Userpin

Specifies the userpin to use with the PKCS#11 device.

ID

Specifies the ID of the PKCS#11 certificate.

P12 certificates

Specifies the details required for a P12 certificate-based authentication.

CertURI

Specifies the path including the P12 certificate filename.

KeyPW

Specifies the munged password to read or write to the certificate file.

Example of a eiam.config File for Java Application

The following is an example of the eiam.config file:

```
<EiamConfiguration>
  <!-- Absolute file path for logger configuration, For Java use:-
file="eiam.log4j.config" -->
  <LoggerConfiguration file="eiam.log4j.config"/>
  <!-- Absolute folder path for SAF folder where audit files will be stored for
processing-->
  <Saf directory="audit"/>
   <!-- Socket timeout in milli seconds. Default value is 2 mins -->
  <Network sockettimeout="120000"/>
<SDK type="Java">
        <iTechSDK>
                <FIPSMode>true</FIPSMode>
                <JCEProvider>JsafeJCE</JCEProvider>
                <Security>
                        <digestAlgorithm>SHA1</digestAlgorithm>
                </Security>
                <Debug>
                        <logLevel>trace</logLevel>
                </Debug>
        </iTechSDK>
  </SDK>
<!-- configuration to create SafeContext instance from SafeContextFactory -->
<SafeContext refid="RBC_Hospital" version="1.0">
        <!-- EEM server hostname -->
        <Backend>eiamServer</Backend>

        <!-- application instance to attach to (leave empty for global) -->
        <Application>RBC_Hospital</Application>

        <!-- locale -->
        <!-- java:format language-country-variant -->
        <Locale>en-us<\Locale>

        <!-- possible values for type are certificate/password/native -->
        <Authentication type="">
        <!-- input for certificate based authentication (PEM) keyPW is valid only for
C++
                <Certificate type="pem">
                        <CertURI>appcert.cer</CertURI>
                        <KeyURI>appcert.key</KeyURI>
                        <KeyPW></KeyPW/>
                </Certificate>
        </Authentication>
</SafeContext>
</EiamConfiguration>
```

Enable iTechnology SDK Logging

You can enable iTechnology SDK logging only for the CA EEM C++ SDK and the CA EEM Java SDK. For CA EEM C# SDK, use the logger configuration file.

To enable iTechnology SDK logging, open the eiam.config file and edit the following tags:

- logLevel
- logToFile
- logFile
- maxLogSize

For CA EEM Java SDK, edit the previously mentioned tags in the <SDK type ="Java"> section. For CA EEM C++ SDK, edit the tags mentioned in the <SDK type ="C++"> section.

More information:

About the eiam.config File (see page 19)

Before You Configure CA EEM Java SDK in FIPS-only Mode

To configure CA EEM Java SDK in FIPS-only mode, do the following tasks:

1. Configure JRE to use third-party Java Cryptography Extension (JCE) libraries.
2. Add the Crypto-J libraries as a JCE provider in the Java.security file.

   Note: For more information about how to configure JRE with JCE, see the respective JCE documentation.
3. Enable FIPS-only mode in the eiam.config file.

Configure CA EEM Java SDK in FIPS-only Mode

When you configure CA EEM SDK in FIPS-only mode, CA EEM uses FIPS 140-2 compliant cryptographic libraries to encrypt and decrypt sensitive data.

To configure CA EEM Java SDK in FIPS-only mode

1. Open eiam.config file and edit the following tags <SDK type="Java"> section:
   - FIPSMode
   - JCEProvider
   - digestAlgorithm
2. Save and close the eiam.config file.
3. Restart your application.

   CA EEM Java SDK is configured in FIPS-only mode.

More information:

About the eiam.config File (see page 19)

Configure CA EEM C++ SDK in FIPS-only Mode

When you configure CA EEM SDK in FIPS-only mode, CA EEM uses FIPS 140-2 compliant cryptographic libraries to encrypt and decrypt sensitive data.

To configure CA EEM C++ SDK in FIPS-only mode

1. Open eiam.config file and edit the following tags in the <SDK type="C++"> section.
   - FIPSMode
   - etpkiCryptoLib
   - secureProtocol
   - digestAlgorithm
2. Save and close the eiam.config file.
3. Restart your application.

   CA EEM C++ SDK is configured in FIPS-only mode.

More information:

About the eiam.config File (see page 19)

Configure CA EEM C# SDK in FIPS-only Mode

When you configure the CA EEM C# SDK in FIPS-only mode, CA EEM uses only FIPS 140-2 compliant cryptographic libraries to encrypt and decrypt sensitive data.

Note: CA EEM C# SDK does not support P11 certificates.

To configure CA EEM C# SDK in FIPS-only mode

1. Open eiam.config file and edit the following tags in the <SDK type="C#"> section:
   - FIPSMode
   - digestAlgorithm
2. Save and close the eiam.config file.
3. Restart your application.

   CA EEM C# SDK is configured in FIPS-only mode.

More information:

About the eiam.config File (see page 19)

Set SafeContext Information

The <SafeContext> tag in the eiam.config file contains information required to generate a SafeContext using the SafeContextFactory class. Each SafeContext tag in the eiam.config file is identified using a unique refID tag. To generate a SafeContext, you must pass this refID to the SafeContextFactory. Following are the benefits of specifying the SafeContext-related information in the eiam.config file:

To set SafeContext-related information:

1. Open eiam.config file and edit the <SafeContext> section to set the following tags:
   - refID
   - Backend
   - Application
   - Locale
   - Authentication Type
2. Save and close the eiam.config file.

Initialize CA EEM Java SDK

Configure CA EEM SDK using the SafeConfigurator class.

Note: You must configure eiam.config before you configure CA EEM SDK. If you do not configure the eiam.config file, the CA EEM SDK is initialized with the following default configuration:

- non-FIPS mode
- Logging is set to error and only console logging is enabled
- SAF location is disabled

To initialize CA EEM SDK, perform the following process:

1. Include the following API in your code to initialize CA EEM SDK during application startup:

   ```
   SafeConfigurator.getInstance().init(filename);
   ```

   Where

   filename

   Specifies the absolute path of the eiam.config file that you have defined for your application.

   Note: All the CA EEM SDK operations after this line are logged based on tracing levels of logging in the logger configuration.

2. Include the following API in your code during shutdown of your application:

   ```
   m_config.term();
   ```

   Note: For more information about the SafeConfigurator class, see the *Programming Guide*.

Initialize CA EEM C++ SDK

Configure CA EEM SDK using the Safe::Configurator class.

Note: Configure the eiam.config before you configure CA EEM SDK. If you do not configure the eiam.config file, the CA EEM SDK is initialized with the following default configuration:

- non-FIPS mode
- Logging is set to error and only console logging is enabled
- SAF location is disabled
- CyclicBuffer is set to True

To initialize CA EEM SDK, perform the following process:

1. Include the following API in your code to initialize CA EEM SDK during application startup:

   ```
   Safe::Configurator::getInstance()->init(filename);
   ```

   Where

   filename

   Specifies the absolute path of the eiam.config file that you have defined for your application.

   Note: If you do not mention the filename, CA EEM SDK is initialized with the default values.

2. Include the following API in your code during shutdown of your application:

   ```
   Safe::Configurator::getInstance()->term();
   ```

   Note: For more information about the SafeConfigurator class, see the *Programming Guide*.

Important! If there are any invalid attributes in the eiam.config file, the CA EEM SDK aborts.

Initialize CA EEM C# SDK

Configure CA EEM SDK using the SafeConfigurator class. To configure CA EEM SDK, perform the following process:

Note: Configure the eiam.config file before you configure CA EEM SDK. If you do not configure the eiam.config file, the CA EEM SDK is initialized with the following default configuration:

- non-FIPS mode
- Logging is set to error and only console logging is enabled
- SAF location is disabled

To initialize CA EEM SDK, perform the following process:

1. Include the following API in your code to initialize CA EEM SDK during application startup:

   ```
   SafeConfigurator.getInstance().Init(filename);
   ```

   Where

   filename

   Specifies the absolute path of the eiam.config file that you have defined for your application.

   Note: If you do not mention the filename, CA EEM SDK is initialized with the default values.

2. Include the following API in your code during shutdown of your application:

   ```
   SafeConfigurator.getInstance().term();
   ```

   Note: For more information about the SafeConfigurator class, see the *Programming Guide*.

More information:

About the eiam.config File (see page 19)
About the Logger Configuration Files (see page 149)

Package CA EEM Java SDK with Your Applications

To package CA EEM Java SDK, do the following:

1. Update the ClassPath with references to the required jar files from the following location:

   Windows: %EIAM_SDK%\jars

   UNIX and Linux: $EIAM_SDK/jars

2. Update your installer to package and deploy the required jar files and configuration files.

   Note: For more information about the required jars and configuration files, see the Required Files topic.

Required Files

Package the following files in your application:

- safe.jar
- xml-apis.jar
- commons-codec-1.3.jar
- commons-logging-1.1.1.jar
- commons-logging-api-1.1.1.jar
- httpclient-4.0.jar
- httpcore-4.0.1.jar
- not-yet-commons-ssl-0.3.10.jar
- xercesImpl-2.9.1.jar
- xml-apis-2.9.1.jar

Package the following configuration files:

- eiam.config
- eiam.log4j.config

Package CA EEM C++ SDK with Your Applications

To package CA EEM C++ SDK, do the following:

1. Update your build script to search a directory for include files and lib path from the makeflags.mk file in the CA EEM SDK folder.
2. Update your installer to package and deploy the required dlls, include files, and configuration files.

   Note: For more information about the required dlls and configuration files, see the Required Files topic.
3. Include the CAPKI installer set up with your installer. Install the CAPKI installer with your product on the target system. The CAPKI installer is in following location:

   - **Windows**:%EIAM_SDK%\capki
   - **UNIX and Linux:** $EIAM_SDK/CAPKI

   Note: If you run Safex on a computer that is different from the computer where the CA EEM Server is installed, install CAPKI with Safex.

Required Files

Package the following dlls in your application:

Windows

Visual Studio 2010

- 32-bit—$EIAM_SDK\sdk\cpp\lib\

UNIX platforms

- $EIAM_SDK/sdk/cpp/lib/

Package the following configuration files:

- eiam.config
- eiam.log4cxx.config

Include the following header files in your code when compiling your application:

Windows

%EIAM_SDK%\sdk\cpp\include

UNIX platforms

$EIAM_SDK/sdk/cpp/include

Package CA EEM C# SDK with Your Applications

To package CA EEM C# SDK, do the following:

1. Add the dlls from the following folder as referenced assemblies when building your application:

   ```
   %EIAM_SDK%\sdk\cshar\lib\
   ```

2. Update your installer to package and deploy the referenced dlls, the eiam.config file, and the eiam.log4net.config file.

   Note: For more information about the reference dlls and configuration files, see the Required Files topic.

Required Files

Package and deploy the following referenced dlls in your application:

- log4net.dll
- SafeCS.dll

Package and deploy the following configuration files:

- eiam.config
- eiam.log4net.config

Chapter 4: Sample WorkFlow

This section contains the following topics:

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 38)
Designing Safe Objects to Implement (see page 39)
Designing the User Interface (see page 53)
Migrating (see page 53)

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	■ read/write by patient's doctor ■ read by any doctor/nurse assigned to the patient's ward ■ read 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
	■ 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	■ 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	■ read/write by Office users ■ read by Chiefs	ResourceClassName: billingdata ResourceName: patientid Actions: read, write User Groups: Chiefs, Office

More Information

Defining Calendars (see page 46)
Defining Policies (see page 50)
Defining the Application Instance (see page 41)

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.soInsert()
}
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 mornings = new SafeTimeBlock();
mornings.setName("morning");
mornings.setStartTime(10*60);
mornings.setDuration(2*60);
mornings.setMask(SafeEnum.WeekDay.WD_ALL, SafeEnum.MonthDay.MD_ALL,
SafeEnum.Month.M_ALL);
cal.addIncludeTimeBlock(mornings);
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.soInsert()
}
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 *mornings = new Safe::TimeBlock;
mornings->setName("morning");
mornings->setStartTime(10*60);
mornings->setDuration(2*60);
mornings->setMask(Safe::WD_ALL, Safe::MD_ALL, Safe::M_ALL);
cal.addIncludeTimeBlock(mornings);
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.soInsert())
{
    // 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.soInsert()
}
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.soInsert()
}
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.soInsert())
{
    // 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 85)

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.setContext	SafeStoredObject.Context	
Set Name	Safe::StoredObject::setPath("/folder/name")	SafeStoredObject.setPath("/folder/name")	SafeStoredObject.Path = "/folder/name"	<Object name="name" folder="/folder">

Action	C++	Java	C#	Safex
Check Access	Safe::StoredObject::canIdentityRead	SafeStoredObject.canIdentityRead	SafeStoredObject.canIdentityRead	
	Safe::StoredObject::canIdentityWrite	SafeStoredObject.canIdentityWrite	SafeStoredObject.canIdentityWrite	
	Safe::StoredObject::canContextRead	SafeStoredObject.canContextRead	SafeStoredObject.canContextRead	
	Safe::StoredObject::canContextWrite	SafeStoredObject.canContextWrite	SafeStoredObject.canContextWrite	
Retrieve	Safe::StoredObject::soRetrieve	SafeStoredObject.soRetrieve	SafeStoredObject.soRetrieve	
	Safe::StoredObject::soRetrieveByName	SafeStoredObject.soRetrieveByName	SafeStoredObject.soRetrieveByName	
	Safe::StoredObject::soRetrieveByUserName	SafeStoredObject.soRetrieveByUserName	SafeStoredObject.soRetrieveByUserName	
Insert	Safe::StoredObject::soInsert	SafeStoredObject.soInsert	SafeStoredObject.soInsert	<Add>
Modify	Safe::StoredObject::soModify	SafeStoredObject.soModify	SafeStoredObject.soModify	<Modify>
Delete	Safe::StoredObject::soRemove	SafeStoredObject.soRemove	SafeStoredObject.soRemove	<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)

Chapter 5: Application Instances

This section contains the following topics:

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. Create a SafeContext using one of the following methods:
   a. Use SafeContextFactory to create a SafeContext.

      Note: For more information about using a SafeContextFactory, see the Create a SafeContext using SafeContextFactory topic.

   or

   b. Instantiate a SafeContext class.

   c. Set the backend server to the host where CA EEM Policy Server is running.

   d. Call the authenticateWithpassword method to verify the authenticity of the user.

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

2. 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 a backend server using SafeContextFactory

The following example creates a SafeContext by using the details specified in the Example: SafeContext Tag section:

```
SafeConfigurator.getInstance().init(eiam.config);

//Passing the reference ID for the SafeContext tag specified in the eiam.config file.
The SafeContext tag with the specified reference ID contains information about the
backend server, application instance, locale, and user credentail details.
SafeContext safecontext = SafeContextFactory.getSafeContext("RBC_Hospital");
```

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");
```

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("mvtext:memberID");
```

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);
```

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 and a SafeCertificate (CA EEM C++ SDK) or a SafeCertificateData (CA EEM C# and CA EEM Java SDKs) is issued for the application instance.

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
SafeCertificateData certdata = safecontext.registerApplicationInstance(sai);
```

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();
```

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");
```

Create a SafeContext using SafeContextFactory

Use the SafeContextFactory to create a SafeContext. The information required to generate a SafeContext is set in the CA EEM SDK configuration file. Each SafeContext instance in the eiam.config file is identified using a unique refID tag. To create a SafeContext, you must pass this refID to the SafeContextFactory.

To create a SafeContext using SafeContextFactory

1. Set the SafeContext tag in the CA EEM SDK configuration file.

   Note: For information about how to set the SafeContext tag and the eiam.config file, see the *Implementation Guide*.

2. Pass the refID from the SafeContext tag to the SafeContextFactory.

Example: SafeContext tag in the CA EEM configuration file.

```
<SafeContext refid="RBC_Hospital" version="1.0">
        <!-- EEM server hostname -->
        <Backend>eiamServer</Backend>

        <!-- application instance to attach to (leave empty for global) -->
        <Application>RBC_Hospital</Application>

        <!-- locale -->
        <!-- java:format language-country-variant -->
        <Locale>en-us<\Locale>

        <!-- possible values for type are certificate/password/native -->
        <Authentication type="">
        <!-- input for certificate based authentication (PEM) keyPW is valid only for
C++
                <Certificate type="pem">
                        <CertURI>appcert.cer</CertURI>
                        <KeyURI>appcert.key</KeyURI>
                        <KeyPW></KeyPW/>
                </Certificate>
        </Authentication>
</SafeContext>
```

Example: Create a SafeContext using SafeContextFactory by passing the SafeContext reference ID from the configuration file.

The following example creates a SafeContext by using the details specified in the Example: SafeContext Tag section:

```
SafeConfigurator.getInstance().init(eiam.config);

//Passing the reference ID for the SafeContext tag specified in the eiam.config file.
The SafeContext tag with the specified reference ID contains information about the
backend server, application instance, locale, and user credentail details.
SafeContext safecontext = SafeContextFactory.getSafeContext("RBC_Hospital");
```

Chapter 6: Users

This section contains the following topics:

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 SafeGloablUser 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-specifc 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

Follow these steps:

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 soretrive 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.
    SafeUser u = new SafeUser();
    u.setContext(sc);
    //Path should be same as globaluser:UserName
    u.setPath("erdoctor");
    //Associate with application-specific group name.
    u.addGroup("Staff");
    u.soInsert();
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 soretrive 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 99).

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()){
        SafeGloabalUser gu = (SafeGloablUser) 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 75)
Retrieve an Application-Specific User (see page 76)

Chapter 7: Groups

This section contains the following topics:

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 73)

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 99).

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 83)
Retrieve a User Group (see page 83)

Chapter 8: Access Management

This section contains the following topics:

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 polices are divided into six parts:

Identities

Specifies the identities to which policy 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

How Policies Are Evaluated (see page 122)
Filters (see page 93)
Policy Evaluation (see page 121)

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

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 89)
- Anybody can admit John or John* (see page 90)
- Staff can admit anyone except Sam (see page 91)
- Nobody can admit Sam (see page 92)

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.soInsert();
```

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

More Information:

How You Create Anybody Can Admit John or John* Policy (see page 90)
Types of Authorization Checks (see page 88)

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'.

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 93).

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.addIndentity(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.soInsert();
```

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

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 121).

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.soInsert();
```

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

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

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	■ 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	■ Date EffectiveStart ■ Date EffectiveStop ■ portableobject IncludeTimeBlock ■ portableobject ExcludeTimeBlock ■ string Description
Global User Groups	searchGlobalUserGroups	■ string GroupMembership ■ string Description

Stored Object	Method to Invoke	MappedAttributes
Global Users	searchGlobalUsers	■ 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	■ 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	■ 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	■ string GroupMembership ■ string Description
Users	searchUsers	■ 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}	■ string Name ■ string Parent ■ string Path ■ string[] GroupMembership ■ string Description
UserGroup	ug:	gu:{name} evaluates to the value(s) of the SafeGlobalUser object attributes matching {name}	■ 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}	■ 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

Column Field Type	Value to Prefix	Filter Evaluation	Attributes
Environment	env:	env:{name} evaluates to the value(s) of the environment attributeq (envattrq) matching {name}	String value from envattrq
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}	■ 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}	■ 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.
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 103)
- How to Search on First Name and Designation (see page 104)
- How to Search on First Name, Designation, and Department (see page 105)

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);
```

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 103)
- 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.

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

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

- Build a filter to search on first name field, see Search on First Name (see page 103).
- Build a filter to search on designation field, see Search on First Name and Designation (see page 104).
- 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.

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 9: Authentication

This section contains the following topics:

NTLM Authentication

CA EEM lets you authenticate users with their browser credentials. The NTLM authentication implementation is provided for an HTTP filter without JAAS.

Prerequisites for Configuring NTLM Authentication

Do the following before you configure NTLM authentication:

- Verify that the CA EEM Server is installed on a Windows Server and is connected to an Active Directory.
- Verify that the users launch the application from a Windows computer.
- Verify that the CA EEM Server and the computer where the users are launching the application are part of the same network domain. If the computers are part of nested domains, ensure that the CA EEM Server and the computer where the application is launched belong to domains that have a trust relation established.
- Verify that the domain users are added to the User Groups on the computer where the application is being launched.

HTTP Filter Without JAAS

Do the following procedure on the Web Server where you have deployed your application.

To configure NTLM authentication for CA EEM-enabled application

1. Stop the Tomcat services.
2. Open the web.xml file from the folder where you have deployed your application.
3. Add an NTLM authentication filter as the first filter in the filters section.

```
<filter>
  <filter-name>NtlmAuthFilter</filter-name>
  <filter-class>com.ca.eiam.httpfilter.AuthenticationFilter</filter-class>
  <init-param>
    <param-name>eiamBackendHost</param-name>
    <param-value></param-value>
  </init-param>
  <init-param>
    <param-name>eiamApplication</param-name>
    <param-value>applicationname</param-value>
  </init-param>
  <init-param>
    <param-name>eiamCertFile</param-name>
     <param-value>abc.p12</param-value>
  </init-param>
  <init-param>
    <param-name>eiamMaskedPassword</param-name>
    <param-value>FR0KBAJEXl8=</param-value>
<init-param>
    <param-name>eiamForceNTLM</param-name>
    <param-value>true</param-value>
  </init-param>
  </init-param>
</filter>
```

4. Add the following lines to the filter mapping section of the web.xml file.

```
<filter-mapping>
<filter-name>NtlmAuthFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

   Note: /* indicates that all the URLs are protected using the NTLM authentication method.

5. Save and close the file.
6. Start the Tomcat services.

Filter Description

The NTLM authentication filter has the following parameters:

Note: Each parameter is identified using the <init-param> tag. The parameter name and value are identified by the <param-name> tag and the <param-value> tag respectively.

The following are the parameters names. Enter the values for each of these parameters in the respective <param-value> tag.

eiamBackendHost

Specifies the host name of the CA EEM Server.

eiamApplication

Specifies the application instance name registered with the CA EEM Server.

eiamCertFile

Specifies the P12 certificate file needed to attach to the application instance or global instance to generate a Safecontext object.

eiamUsername

Specifies the username of the administrator needed to attach to an application instance or global instance.

Note: Based on your authentication method use either eiamCertFile or eiamUsername in the filter.

eiamPassword

Specifies the clear text password needed to authenticate the administrator.

eiamMaskedPassword

Specifies the munged password to be used with the certificate file or the username.

eiamForceNTLM

Specifies that the NTLM authentication is forced even if the browser is configured to use Kerberos authentication.

Value: [True|False]

Default: False

Example--NTLM Filter

The following is an example of an NTLM filter:

```
<filter>
    <filter-name>NtlmAuthFilter</filter-name>
        <filter-class>com.ca.eiam.httpfilter.AuthenticationFilter</filter-class>
    <init-param>
        <param-name>eiamBackendHost</param-name>
        <param-value>Server1</param-value>
    </init-param>
    <init-param>
        <param-name>eiamApplication</param-name>
        <param-value>RBC_Hospital</param-value>
    </init-param>
    <init-param>
        <param-name>eiamCertFile</param-name>
        <param-value>C:\Program Files\sample.p12</param-value>
    </init-param>
    <init-param>
        <param-name>eiamMaskedPassword</param-name>
        <param-value>FR0KBAJEXl8=</param-value>
    </init-param>
</filter>
```

Certificate Authentication

CA EEM SDK supports the following authentication certificate types:

- P12--Supported only in non-FIPS mode.
- PEM--Supported by the CA EEM C++, CA EEM C#, and CA EEM Java SDKs.
- PKCS#11--Supported only by the CA EEM C++ and CA EEM Java SDKs.

Use the following methods for certificate authentication:

- authenticateWithCertificate
- fastauthenticateWithCertificate

To use authenticateWithCertificate

1. Instantiate a SafeContext Class.
2. Set the backend server to the host where CA EEM Server is running.
3. Read the certificate using the SafeCertificateReader Class and return a SafeCertificate.
4. Use the SafeCertificate with authenticateWithCertificate method to verify the authenticity of the user and generate a SafeSession.

Example: Authenticate with P12 certificate

The following example attaches an application to the backend server and authenticates using a certificate:

```
//Instantiate a SafeContext Class.
SafeContext safecontext = new SafeContext();
//Set the backend to the host where Server is running.
safecontext.setBackend("<hostname>");
SafeCertificateData certdata = SafeCertificateReader.readP12(certfile, password);
//authentication call
SafeSession safesession = safecontext.authenticateWithCertificate(certdata);
```

Example: Authenticate with PEM certificate

The following example attaches an application to the backend server and authenticates using a certificate:

```
//Instantiate a SafeContext Class.
SafeContext safecontext = new SafeContext();
//Set the backend to the host where Server is running.
safecontext.setBackend("<hostname>");
SafeCertificateData certdata = SafeCertificateReader.readPEM(certfile,
privatekeyfile);
//authentication call
SafeSession safesession = safecontext.authenticateWithCertificate(certdata);
```

Example: Authenticate with PKCS#11 certificate

The following example attaches an application to the backend server and authenticates using a certificate:

Note: The CA EEM C# SDK does not support PKCS#11 device.

```
//Instantiate a SafeContext Class.
SafeContext safecontext = new SafeContext();
//Set the backend to the host where Server is running.
safecontext.setBackend("<hostname>");
SafeCertificateData certdata = SafeCertificateReader.readP11(provider1, userpin,
id);
//authentication call
SafeSession safesession = safecontext.authenticateWithCertificate(certdata);
```

Issue Certificate

To issue certificates, you must be attached to an application instance.

1. Attach to a backend server.
2. Issue certificate using the issueCertificate method.
3. Write the SafeCertificate to a P12, PEM, or PKCS#11 file using the SafeCertificateWriter method.

Example: Issue a certificate and store the certificate as a P12

The following example issues P12 certificates:

```
//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 the application instance
safecontext.attach(appname, safession)
//Issue certificate
SafeCertificateData certdata = safecontext.issueCertificate();
//Write SafeCertificateData to P12
SafeCertificateWriter.writeToP12(certData, "certfile.p12", Pwd);
//Write SafeCertificateData to PEM
SafeCertificateWriter.writeToPEM(certData, "certfile.cer", "keyfile.key");
//Write SafeCertificateData to PKCS#11
SafeCertificateWriter.writeToP11(certData, provider1, userpin, id);
```

Note: When you are using the writeToPEM method in CA EEM C++ SDK, you must also pass a password as an argument. This password can be empty. In FIPS-only mode, the password must be empty as FIPS does not support password encryption.

Using writeToPEM in CA EEM C++ SDK

```
Safe::CertificateWriter::writeToPEM(certData, "certfile.cer", "keyfile.key",
password);
```

Issue Certificate for a Session

To issue certificates, you must be attached to an application instance.

1. Attach to a backend server.
2. Pass a session to the issueCertificateForSession method.
3. Write the SafeCertificate to a P12, PEM, or PKCS#11 file using the SafeCertificateWriter method.

Example: Issue a certificate for a SafeSession

The following example generates a certificate for a SafeSession:

```
//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");
//Issue Certificate for a session
SafeCertificateData CertData = issueCertificateForSession(safesession);
//Write SafeCertificateData to P12
SafeCertificateWriter.writeToP12(CertData, "certfile.p12", Pwd);
//Write SafeCertificateData to PEM
SafeCertificateWriter.writeToPEM(CertData, "certfile.cer", "keyfile.key");
//Write SafeCertificateData to PKCS#11
SafeCertificateWriter.writeToP11(CertData, provider1, userpin, id);
```

Issue Certificate For Users

You can only issue certificates only for valid application-specific users.

1. Pass a SafeUser to the issueCertificateForUser method.
2. Write the SafeCertificate to a P12, PEM, or PKCS#11 file using the SafeCertificateWriter method.

Example: Issue a certificate for a SafeUser

The following example generates a certificate for a SafeUser:

```
SafeContext safecontext = new SafeContext();
SafeSession session = safecontext.authenticateWithPassword("EiamAdmin", "password")
//Attach to the application using the session.
obj.attach("RBC_Hospital", session);
SafeUser user = new SafeUser();
user.setContext(safecontext);
User.soRetriveByUserName("erdoctor");
SafeCertificateData CertData = issueCertificateForUser(user);
//Write SafeCertificateData to P12
SafeCertificateWriter.writeToP12(certData, "certfile.p12", Pwd);
//Write SafeCertificateData to PEM
SafeCertificateWriter.writeToPEM(certData, "certfile.cer", "keyfile.key");
//Write SafeCertificateData to PKCS#11
SafeCertificateWriter.writeToP11(certData, provider1, userpin, id);
```

Chapter 10: Certificate Validation

After you verify the public key and private key of a certificate in a SSL handshake, you can use CA EEM to validate the revocation status of a certificate. CA EEM uses the SafeContext.validateUserCertificate or SafeContext.fastValidateUserCertificate APIs to validate a certificate. If the validation of a certificate is successful, CA EEM extracts the username and initiates a safe session.

CA EEM supports the following revocation mechanisms:

- Certificate Revocation List (CRL)
- CRL Distribution Point (CRLDP)
- Online Certificate Status Protocol (OCSP)

This section contains the following topics:

Validate a Certificate (see page 119)

Validate a Certificate

Follow these steps:

1. Configure CA EEM server for certificate validation. For information about configuring certificate validation, see the *Implementation Guide*.
2. Assign permission to validate certificates.
3. Invoke the validateUserCertificate API.

Assign Permission to the CertificateValidation Resource

You must assign a user with the permission to validate certificates. You can update the existing AdministerObjects policy or create a policy.

To assign permission to validate a certificate

1. Go to Scoping Policies, and click the AdministerObjects policy in the right pane.
2. Select CertificateValidation from Add resource list in the Access Policy Configuration section.
3. Click the Add resource icon.

   The CertificateValidation resource appears in the Resources field. The user is assigned the permission to validate a certificate.

Note: If you can want to create a policy and assign the user with permission to validate a certificate, create a policy and perform the steps 2–3.

How to Invoke the API

When you invoke the API, CA EEM performs the following steps:

1. CA EEM SDK calls the CA EEM server to validate user certificate.
2. CA EEM server verifies that the caller of the CA EEM SDK has the permission to validate a certificate.
3. CA EEM server validates the certificate with selected revocation mechanism.
4. CA EEM server extracts user mapped attribute.
5. (Optional) If configured, CA EEM server uses the extracted subject to map the certificate with a user from the configured external LDAP directory.
6. CA EEM returns a SafeSession.

Chapter 11: Policy Evaluation

This section contains the following topics:

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 1and 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 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 12: Exception Handling

This section contains the following topics:

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
int SUCCESS = 0;	EE_SUCCESS Success
int EXCEPTION = 1;	EE_EXCEPTION Exception
int NOCREDS = 2;	EE_NOCREDS No Credentials
int NOBACKEND = 3;	EE_NOEIAMNODES No SafeNodes Defined
int SPONSORERROR = 4;	EE_SPONSORERROR iSponsor Error
int NOTATTACHED = 5;	EE_NOTATTACHED Not Attached to Repository
int NOTFOUND = 6;	EE_NOTFOUND 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
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.eiam.SafeException
        +- com.ca.eiam.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.eiam.SafeException
        +- com.ca.eiam.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 13: Identity Management

This section contains the following topics:

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 136)
- Applying password policies
- Identity self-administration (see page 136)

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.

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.

   `./S99igateway 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.

   `./S99igateway 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.

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

Chapter 14: Event Management

This section contains the following topics:

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 a 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.

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

More Information

Event Data Model (see page 142)

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::ApplicationInstace::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::getApplcationName 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

Notes:

- Runtime events use the standard audit fields with Taxonomy set to IAM.Runtime.{action}.[S|F].I {action}.
- For every authorization call, an event will be generated that will log the named attributes used by the authorization call.

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.

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

   ```
   $IGW_LOC/S99igateway 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/S99igateway stop
   ```

   All the events that are generated after configuring will be routed to the specified host.

Chapter 15: CA EEM SDK Logging

The CA EEM SDK logging lets you do the following:

- Application log levels can be changed at run time.
- Configure logging information such as filename, file size, number of backup log files, and so on.

The following files control the logging in the CA EEM SDK:

- eiam.log4cxx.config and eiam.config files for CA EEM C++ SDK
- eiam.log4net.config for CA EEM C# SDK
- eiam.log4j.config and eiam.config files for CA EEM Java SDK

The samples of these logger configurations are shipped with the CA EEM SDK package and are placed in Bin folder:

UNIX

$EIAM_SDK/bin

Windows

%EIAM_SDK%\bin

This section contains the following topics:

About the Logger Configuration Files (see page 149)

About the Logger Configuration Files

The logger configuration files, eiam.log4cxx.config, eiam.log4net.config, and eiam.log4j.config, are used to configure CA EEM SDK logging. These files contain the following major components:

- Appenders
- Loggers
- Root Logger

These components contain configurable parameters that let you customize the logging process based on your business requirements.

Appender

An appender contains parameters that control the logging of each logger. By default, the logger configuration files contains the following appenders:

SDK

Logs the SDK messages into a log file. Specifies the path including the file name of the log file.

Default: eiam.cppsdk.log for C++ SDK, EIAM.C#SDK.log for C#, and eiam.javasdk.log for Java SDK.

Note: If you are deploying your application under Tomcat server on Windows, verify that you use forward slash '/' in the path instead of the backward slash '\'. If you use backward slash, the log file is not created at the path you have specified; instead, the log file is created in the Apache Tomcat folder.

Network

Logs the network call related messages into a log file.

Default:eiam.network.cpp.log for C++ SDK, EIAM.NETWORK.C#SDK.log for C# SDK, and eiam.javasdk.log for Java SDK.

Performance

Logs the performance call related messages into a log file.

Default: eiam.performance.cpp.log for C++ SDK, EIAM.PERFORMANCE.C#SDK.log for C# SDK, and eiam.performance.java.log

Console

Displays the log messages on the console.

SDK appender is enabled by default. To enable other appenders, remove the comment strings (<!-- and -->) from their respective code.

An appender consists of the following configurable parameters:

file

Specifies the log filename of the appender.

append

Specifies if a set of log messages is appended to the log file. If the value is true, the set of log message is appended to the last log message in the log file.

Note: This parameter is named appendToFile in the eiam.log4net.config file.

BufferedIO

Specifies if the latest log message is buffered. If the value is true, latest few log messages are kept in memory before writing to log file. This option minimizes IO operation and is beneficial if the log level is higher.

Value: [*true*|*false*]

Default*:* false

Note: The default size of BufferedIO is 8 KB.

maxFileSize

Specifies the maximum size of the log file. If a log file exceeds the maximum size, a new log file filename log.1 is created and the contents of log file are transferred to log.1 file. The log file now contains latest log messages. If this file too exceeds the maximum size, a new log file filename log.2 is created, the contents of log.1 are transferred to log.2 file, and the contents of log file are transferred to log.1 file.

Default: 10 MB

Minimum: 10 KB

Maximum: 2 GB

Note: The minimum size of the maxFileSize must be greater than or equal to the size of BufferedIO. This parameter is named maximumFileSize in the eiam.log4net.config file.

maxBackupIndex

Specifies the maximum number of backup log files used for keeping old logs. If the number of log files exceeds the maximum backup index value, the file with the oldest log messages is deleted.

Default: 1

Minimum: 1

Maximum: 12

Note: This parameter is named maxSizeRollBackups in the eiam.log4net.config file.

rollingStyle

Specifies the criteria for creating log files. When this parameter is set to Size, if a log file exceeds the maximumFileSize, a new log file is created and the contents of the current log file are backed up.

Default*: Size*

ConversionPattern

Specifies the formatting of a log message. Configure the format modifiers and conversion characters to define the conversion pattern.

Note: For more information about conversion patterns, refer the topic log4j in www.apache.org.

Example: SDK Appender

```
<appender name="SDK" class="org.apache.log4j.RollingFileAppender">
    <!-- The active sdk log file -->
    <param name="file" value="eiam.cppsdk.log" />
    <param name="append" value="true" />
    <param name="BufferedIO" value="false"/>
    <param name="maxFileSize" value="10000KB" />
    <param name="maxBackupIndex" value="1" />
    <layout class="org.apache.log4j.PatternLayout">
    <!-- The log message pattern -->
    <param name="ConversionPattern" value="%5p %d{ISO8601} [%t] [%c] %m%n"/>
    </layout>
</appender>
```

Appender in eiam.log4net.config

An appender contains parameters that control the logging of each logger. By default, the logger configuration files contains the following appenders:

SDK

Logs the SDK messages into a log file. Specifies the path including the file name of the log file.

Default: EIAM.C#SDK.log

Note: If you are deploying your application under Tomcat server on Windows, ensure that you use forward slash '/' in the path instead of the backward slash '\'. If you use backward slash, the log file is not created at the path you have specified; instead, the log file is created in the Apache Tomcat folder.

Network

Logs the network call related messages into a log file.

Default:EIAM.NETWORK.C#SDK.log

Performance

Logs the performance call related messages into a log file.

Default: EIAM.PERFORMANCE.C#SDK.log

Console

Displays the log messages on the console.

SDK appender is enabled by default. To enable other appenders, remove the comment strings (<!-- and -->) from their respective code.

An appender consists of the following configurable parameters:

file

Specifies the log filename of the appender.

appendToFile

Specifies if a set of log messages is appended to the log file. If the value is true, the set of log message is appended to the last log message in the log file.

maxSizeRollBackups

Specifies the maximum number of backup log files used for keeping old logs. If the number of log files exceeds the maximum backup index value, the file with the oldest log messages is deleted.

Default: 1

Minimum: 1

Maximum: 12

rollingStyle

Specifies the criteria for creating log files. When this parameter is set to Size, if a log file exceeds the maximumFileSize, a new log file is created and the contents of the current log file are backed up.

Default*: Size*

maximumFileSize

Specifies the maximum size of the log file. If a log file exceeds the maximum size, a new log file filename log.1 is created and the contents of log file are transferred to log.1 file. The log file now contains latest log messages. If this file too exceeds the maximum size, a new log file filename log.2 is created, the contents of log.1 are transferred to log.2 file, and the contents of log file are transferred to log.1 file.

Default: 10 MB

Minimum: 10 KB

Maximum: 2 GB

Note: The minimum size of the maxFileSize must be greater than or equal to the size of rollingStyle.

ConversionPattern

Specifies the formatting of a log message. Configure the format modifiers and conversion characters to define the conversion pattern.

Note: For more information about conversion patterns, refer the topic log4net in www.apache.org.

Logger

Loggers let you control the log messages for CA EEM SDK. To enable a logger, remove the comment strings from their respective code.

A logger contains the following parameters:

logger name

Specifies the name of a logger.

additivity

Specifies if the log messages are duplicated in the SDK log file.

Value: [*true*|*false*]

Default: false

level value

Specifies log level of a logger.

Value: [*Trace*|*Debug*|*Info*|*Warn*|*Error*|*Fatal*|*Off*]

The following are the log levels, in the order of their precedence:

Note: Higher the log level, lesser is the performance of CA EEM.

Trace

Indicates low level debugging. It contains control flow and passes arguments.

Debug

Indicates messages used for problem diagnosis. It contains contextual information.

Info

Indicates contextual information that traces execution at a coarse-grained level in a production environment.

Warn

Indicates a potential problem in the system. For example, if the message category corresponds to security, a warning message must display if a dictionary attack is detected.

Error

Indicates a serious problem in the system. The problem is non-recoverable and requires manual intervention.

Fatal

Indicates a very sever error that may lead the application to abort.

Off

Indicates the absence of logging.

Note: The log level of the default SDK appender must be Error.

Example: Performance Logger

```
<logger name="Perform" additivity="false">
    <level value="trace"/>
    <appender-ref ref="Performance" />
</logger>
```

Root Logger

Root Logger controls the log level of all the appenders. However, if the log level of the referenced appender in root logger is different from the log level specified in the parent appender, the higher priority log level overrides the lower priority log level.

For example, if the log level of a the root logger is Error and the log level of the Network appender is Trace, the log level Trace overrides Error and the system considers log messages with log level Trace at the runtime.

Example: Root Logger

```
<root>
    <priority value="error" />
    <appender-ref ref="SDK" />
</root>
```

Configure the Logger Files

CA EEM lets you configure the log messages related to network, performance, console, and SDK classes.

To configure logger files

1. Open the logger configuration file, eiam.log4cxx.config, eiam.log4net.config, or eiam.log4j.config, in a text editor.
2. Enable loggers and appenders. Remove the comment strings from the logger and appender code to enable loggers and appenders.
3. Update the appender parameters.
4. Save the logger configuration file.

Example of a eiam.log4cxx.config File

The following is an example of the eiam.log4cxx.config file:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<!-- Note that this file is read by the sdk every 60 seconds -->

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

    <appender name="SDK" class="org.apache.log4j.RollingFileAppender">
        <!-- The active sdk log file -->
        <param name="file" value="eiam.cppsdk.log" />
        <param name="append" value="true" />
        <param name="BufferedIO" value="false"/>
        <param name="maxFileSize" value="10000KB" />
        <param name="maxBackupIndex" value="1" />
        <layout class="org.apache.log4j.PatternLayout">
        <!-- The log message pattern -->
        <param name="ConversionPattern" value="%5p %d{ISO8601} [%t] [%c] %m%n"/>
        </layout>
    </appender>

    <appender name="Network" class="org.apache.log4j.RollingFileAppender">
        <!-- The file to log Network calls -->
        <param name="file" value="eiam.network.cpp.log" />
        <param name="append" value="true" />
        <param name="BufferedIO" value="true"/>
        <param name="maxFileSize" value="10000KB" />
        <param name="maxBackupIndex" value="1" />
        <layout class="org.apache.log4j.PatternLayout">
        <!-- The log message pattern -->
        <param name="ConversionPattern" value="%5p %d{ISO8601} [%t] [%c] %m%n"/>
        </layout>
    </appender>

    <appender name="Performance" class="org.apache.log4j.RollingFileAppender">
        <!-- The file to log Performance calls -->
        <param name="file" value="eiam.performance.cpp.log" />
        <param name="append" value="true" />
        <param name="BufferedIO" value="true"/>
        <param name="maxFileSize" value="10000KB" />
        <param name="maxBackupIndex" value="1" />
        <layout class="org.apache.log4j.PatternLayout">
        <!-- The log message pattern -->
        <param name="ConversionPattern" value="%5p %d{ISO8601} [%t] [%c] %m%n"/>
        </layout>
    </appender>

    <appender name="Console" class="org.apache.log4j.ConsoleAppender">
        <!-- Logs to Console -->
        <layout class="org.apache.log4j.PatternLayout">
```

```
            <!-- The log message pattern -->
            <param name="ConversionPattern" value="%5p %d{ISO8601} [%t] [%c] %m%n"/>
            </layout>
        </appender>

        <!-- Remove comment to enable Performance Logging -->
        <!--
        <logger name="Perform" additivity="false">
            <level value="trace"/>
            <appender-ref ref="Performance" />
        </logger>
        -->

        <!-- Remove comment to enable Network Logging -->
        <!--
        <logger name="Network" additivity="false">
            <level value="trace"/>
            <appender-ref ref="Network" />
        </logger>
        -->

        <root>
            <priority value="error" />
            <appender-ref ref="SDK" />
            <!-- <appender-ref ref="Console" /> -->
        </root>
    </log4j:configuration>
```

Example of a eiam.log4net.config File

The following is an example of a eiam.log4net.config file:

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

<log4net>
        <appender name="SDK" type="log4net.Appender.RollingFileAppender">
                <file value="EIAM.C#SDK.log" />
                <appendToFile value="true" />
                <maxSizeRollBackups value="1" />
                <maximumFileSize value="10000KB" />
                <rollingStyle value="Size" />
                <layout type="log4net.Layout.PatternLayout">
                        <conversionPattern value="%date [%thread] %-5level %logger
- %message%newline" />
                </layout>
        </appender>

        <appender name="Network" type="log4net.Appender.RollingFileAppender">
                <file value="EIAM.NETWORK.C#SDK.log" />
                <appendToFile value="true" />
                <maxSizeRollBackups value="1" />
                <maximumFileSize value="10000KB" />
                <rollingStyle value="Size" />
                <layout type="log4net.Layout.PatternLayout">
                        <conversionPattern value="%date [%thread] %-5level %logger
- %message%newline" />
                </layout>
        </appender>

        <appender name="Performance" type="log4net.Appender.RollingFileAppender">
                <file value="EIAM.PERFORMANCE.C#SDK.log" />
                <appendToFile value="true" />
                <maxSizeRollBackups value="1" />
                <maximumFileSize value="10000KB" />
                <rollingStyle value="Size" />
                <layout type="log4net.Layout.PatternLayout">
                        <conversionPattern value="%date [%thread] %-5level %logger
- %message%newline" />
                </layout>
        </appender>

        <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
                <layout type="log4net.Layout.PatternLayout">
                        <conversionPattern value="%date [%thread] %-5level %logger
- %message%newline" />
                </layout>
        </appender>
```

```
        <!-- Uncomment to enable Performance Logging -->
        <!--
        <logger name="Perform" additivity="false">
                <level value="ERROR"/>
                <appender-ref ref="Performance" />
        </logger>-->

        <!-- Uncomment to enable Network Logging -->
        <!--<logger name="Network" additivity="false">
                <level value="ERROR"/>
                <appender-ref ref="Network" />
        </logger>-->

        <root>
                <level value="ERROR" />
                <appender-ref ref="SDK" />
                <!--    <appender-ref ref="ConsoleAppender" />   -->
        </root>
</log4net>
```

Example of a eiam.lo4j.config File

The following is an example of the eiam.log4cxx.config file:

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

<!-- Note that this file is read by the sdk every 60 seconds -->

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

        <appender name="SDK" class="com.ca.eiam.log4j.RollingFileAppender">
                <!-- The active sdk log file -->
                <param name="file" value="eiam.javasdk.log" />
                <param name="append" value="true" />
                <param name="BufferedIO" value="false"/>
                <param name="maxFileSize" value="10000KB" />
                <param name="maxBackupIndex" value="1" />
                <layout class="com.ca.eiam.log4j.PatternLayout">
                        <!-- The log message pattern -->
                        <param name="ConversionPattern" value="%5p %d{ISO8601}
[%t] [%c] %m%n"/>
                </layout>
        </appender>

        <appender name="Network" class="com.ca.eiam.log4j.RollingFileAppender">
                <!-- The file to log Network calls -->
                <param name="file" value="eiam.network.java.log" />
                <param name="append" value="true" />
                <param name="BufferedIO" value="false"/>
                <param name="maxFileSize" value="10000KB" />
                <param name="maxBackupIndex" value="1" />
                <layout class="com.ca.eiam.log4j.PatternLayout">
                        <!-- The log message pattern -->
                        <param name="ConversionPattern" value="%5p %d{ISO8601}
[%t] [%c] %m%n"/>
                </layout>
        </appender>

        <appender name="Performance"
class="com.ca.eiam.log4j.RollingFileAppender">
                <!-- The file to log Performance calls -->
                <param name="file" value="eiam.performance.java.log" />
                <param name="append" value="true" />
                <param name="BufferedIO" value="false"/>
                <param name="maxFileSize" value="10000KB" />
                <param name="maxBackupIndex" value="1" />
                <layout class="com.ca.eiam.log4j.PatternLayout">
                        <!-- The log message pattern -->
```

```
                        <param name="ConversionPattern" value="%5p %d{ISO8601}
[%t] [%c] %m%n"/>
                </layout>
        </appender>

        <appender name="Console" class="com.ca.eiam.log4j.ConsoleAppender">
                <!-- Logs to Console -->
                <layout class="com.ca.eiam.log4j.PatternLayout">
                        <!-- The log message pattern -->
                        <param name="ConversionPattern" value="%5p %d{ISO8601}
[%t] [%c] %m%n"/>
                </layout>
        </appender>

        <!-- Uncomment to enable Performance Logging -->
        <!--
        <logger name="Perform" additivity="false">
                <level value="trace"/>
                <appender-ref ref="Performance" />
        </logger>
        -->

        <!-- Uncomment to enable Network Logging -->
        <!--
        <logger name="Network" additivity="false">
                <level value="trace"/>
                <appender-ref ref="Network" />
        </logger>
        -->

        <root>
                <priority value="error" />
                <appender-ref ref="SDK" />
                <!-- <appender-ref ref="Console" /> -->
        </root>

</log4j:configuration>
```

Chapter 16: Using Custom Key Length Certificates for SSL Communication in CA EEM SDK

You can use the custom key length certificates for the following two purposes:

- Communicate over SSL between CA EEM SDK and CA EEM Server.
- Authenticate the application against the CA EEM Server.

This section contains the following topics:

Communicate Over SSL between CA EEM SDK and CA EEM Server

For the SSL communication between CA EEM SDK and CA EEM Server using custom key length certificates, do the following steps:

1. Generate the certificate.
2. Update the eiam.config file.
3. Restart the application.

Generate the Certificate

You can generate a certificate using the OpenSSL utility or can obtain the certificate from a Certificate Authority (CA).

You need the certificate file name and key file name to update the configuration file of CA EEM SDK.

Note: For information about the steps to generate a certificate, see the OpenSSL documentation.

CA EEM supports both PEM and P12 certificates for the SSL communication in CA EEM SDK.

Update the eiam.config File

To use certificates for the SSL communication, update the eiam.config file of CA EEM SDK with the details of the certificate that you want to use.

Follow these steps:

1. Navigate to the following CA EEM SDK installation folder.

   In Windows, the default installation path is as follows:

   ```
   C:\Program Files\CA\Embedded Entitlements Manager SDK\bin
   ```

   In UNIX, the default installation path is as follows:

   ```
   /opt/CA/SharedComponents/EmbeddedEntitlementManagerSDK/bin
   ```

2. Open the eiam.config file.

3. In the SDK type section, navigate to the corresponding SDK that the application uses and update the Security tag with the following code:

   - For a PEM certificate, update the following code:

     ```
     <SslCertificate>
      <certType>pem</certType>
      <certURI/>
      <keyURI/>
     </SslCertificate>
     ```

     Update certURI tag with the certificate name as follows:

     ```
     <certURI>certificate file name</certURI>
     ```

     Update keyURI tag with the certificate name as follows:

     ```
     <keyURI>key file name</keyURI>
     ```

   - For a P12 certificate, update the following code:

     ```
     <SslCertificate>
      <certType>p12</certType>
      <certURI/>
      <certPW/>
     ```

     ```
     </SslCertificate>
     ```

     Update certURI tag with the certificate name as follows:

     ```
     <certURI>certificate file name</certURI>
     ```

     Update certPW tag with the certificate name as follows:

     ```
     <certPW>munged format of the certificate password</certPW>
     ```

4. Copy the certificates to a location of your choice.

5. Update the certURI and keyURI tags with the absolute location of the certificates including the file name.

Restart the Application

To apply the changes, restart the application.

Authenticate the Application against CA EEM Server

You can use CA EEM generated application certificates to authenticate and attach an application against the CA EEM Server.

Regenerate the Application Certificates

If you modify the CA EEM Server certificates, regenerate the application certificates.

Follow these steps:

1. On the CA EEM Server, navigate to the following location:

   ```
   EIAM_HOME/bin
   ```

2. Create an XML file with the following content:

   ```
   <Safex>
       <Attach label="application for which the certificate has to be issued"/>
       <IssueCertificate certtype="pem" certfile="<cert file name>.cer"
       keyfile="<key file name>.key" />
       <Detach/>
   </Safex>
   ```

   Example:

   To issue an application certificate for the RBC_Hospital application, the XML file is as follows:

   ```
   <Safex>
       <Attach label="RBC_Hospital"/>
       <IssueCertificate certtype="pem" certfile="safex.cer" keyfile="safex.key"
       />
       <Detach/>
   </Safex>
   ```

3. Execute the following command:

   ```
   safex –h <CA EEM Server computer name> -u EiamAdmin –p <EiamAdmin password> -f
   <XML file name>
   ```

   The certificate file and key file are created in the current directory.

4. Copy the certificate file and key file to the location where the original application certificates existed.
5. Restart the application.

   The application uses the regenerated certificates.

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] [-l locale] [-u user -p password] [-c cert -p password] [-n] [-v verbose level (0-4)] [-s simulate] [-pause] [-b bulk insert] [-f xml_file] [-munge password] [-sdkconfig configfile] [-sha1encodeddigest password]
```

Example: C:\Program Files\CA\Embedded Entitlements Manager 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.

- l locale (Optional)

Specifies the locale.

Default: us-en

- u user

Specifies the username to attach.

- c certificate

Specifies the certificate filename.

- p password

Specifies the password of the user 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

- b bulk (Optional)

Processes large number of objects.

- pause (Optional)

Specifies safex to pause before processing the data.

- f XML file

Specifies the XML file to process.

- munge (Optional)

Converts plaintext password to encrypted versions and displays.

Example: [-munge password1 (password2 …)]

- -sha1encodeddigest (Optional)

Converts plaintext password to SHA1 encoded digest. Encode your password using this option, and specify the encoded password in the XML file using the DirectoryPasswordDigest tag. The DirectoryPasswordDigest tag is used when creating global users.

-sdkconfig

Specifies the absolute path to the eiam.config.

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.
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.

Exit Code	Description
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

Register

Example: Register application and store certificates in P12 format

```
<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>
```

Example: Register application and store certificates in PKCS#11 device

```
<Safex>
<Attach/>
<Register certtype="p11" pkcs11lib="pkcs11lib " token="lodocs" userpin="userpin"
id="id" sensitive="false">

<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>
```

Example: Register application and store certificates in PEM format

```
<Safex>
<Attach/>
<Register certtype="pem" certfile="sample.pem" keyfile="sample.key"
password="sample">

<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 specifing 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 specifed 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 "jjd*". All
global objects are processed in the first Export pass (note global Attach). This can
further controled with the ExportMultiple acion 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 begining 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="jjd 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="jjd2 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="jjd3 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="jjd4 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="jjd 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>
<PwAllowId>true</PwAllowId>
<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>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<IncorrectLoginCount>0</IncorrectLoginCount>
<SuspendDate>0</SuspendDate>
<DisableDate>0</DisableDate>
<EnableDate>0</EnableDate>
<Description>Working in Finance</Description>
<Comments>12 month temp</Comments>
<JobTitle>Billing Manaeger</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>Engineering 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>nnnnnnn</EffectiveStart>
<EffectiveStop>nnnnnnn</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>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<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>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<Description>Security Guard</Description>
<JobTitle>Lieutenant</JobTitle>
<FirstName>Security</FirstName>
<LastName>Guard</LastName>
<DisplayName>Security Guard</DisplayName>
</GlobalUser>

<GlobalUser>
<UserName>icudoctor</UserName>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<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>
```

```
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<Description>Head Nurse</Description>
<JobTitle>Nurse</JobTitle>
<FirstName>Head</FirstName>
<LastName>Nurse</LastName>
<DisplayName>Head Nurse</DisplayName>
<GroupMembership>Chiefs</GroupMembership>
</GlobalUser>

<GlobalUser>
<UserName>ernurse</UserName>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<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>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<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>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<Description>maintenance employee</Description>
<JobTitle>Sanitary Engineer</JobTitle>
<FirstName>Jan</FirstName>
<LastName>itor</LastName>
<DisplayName>Jan Itor</DisplayName>
</GlobalUser>
<GlobalUser>
<UserName>receptionist</UserName>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<Description>Receptionist</Description>
<JobTitle>Receptionist</JobTitle>
```

```
<FirstName>Re</FirstName>
<LastName>Ceptionist</LastName>
<DisplayName>Re Ceptionist</DisplayName>
</GlobalUser>

<GlobalUser>
<UserName>officeworker</UserName>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<Description>Office Worker for billing</Description>
<JobTitle>Accountant</JobTitle>
<FirstName>Office</FirstName>
<LastName>Worker</LastName>
<DisplayName>Office Worker</DisplayName>
</GlobalUser>

<GlobalUser>
<UserName>icunurse</UserName>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<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>
<DirectoryPasswordDigest>{SHA}AYqS+j5LAwtQqbaMy8EfnJjXrqk=</DirectoryPasswordDige
st>
<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>ernurse</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>acl</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 delgates 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 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.so Modify	SafeApplicationInstance.so Modify	Safe::ApplicationInstance.so Modify
Define Resource Classes	<Register/>	SafeResourceClass	SafeResourceClass	Safe::ResourceClass
		SafeApplicationInstace.addResourceClass	SafeApplicationInstace.addResourceClass	Safe::ApplicationInstance.addResourceClass
		SafeApplicationInstance.so Modify	SafeApplicationInstance.so Modify	Safe::ApplicationInstance.so Modify

Action	Safex	Java	C#	C++
Change Password		SafeContext.changePassword	SafeContext.changePassword	Safe::Context::changePassword
Unlock User Account		SafeContext.unlockUser	SafeContext.unlockUser	Safe::Context::unlockUser
Insert a Global User	<Add> <GlobalUser/> </Add>	SafeGlobalUser.soInsert	SafeGlobalUser.soInsert	Safe::GlobalUser::soInsert
Search Global User		SafeContext.searchGlobalUsers	SafeContext.searchGlobalUsers	Safe::Context::searchGlobalUsers
Retrieve Global User		SafeGlobalUser.soRetrieveByName SafeGlobalUser.soRetrieveByUserName SafeGlobalUser.soRetrieve SafeContext.searchGlobalUsers	SafeGlobalUser.soRetrieveByName SafeGlobalUser.soRetrieveByUserName SafeGlobalUser.soRetrieve SafeContext.searchGlobalUsers	Safe::GlobalUser::soRetrieveByName Safe::GlobalUser::soRetrieveByUserName Safe::GlobalUser::soRetrieve Safe::Context::searchGlobalUsers
Modify Global User	<Modify> <GlobalUser/> </Modify>	SafeGlobalUser.soModify	SafeGlobalUser.soModify	Safe::GlobalUser::soModify

Action	Safex	Java	C#	C++
Remove Global User	<Remove> <GlobalUser/> </Remove>	SafeGlobalUser.soRemove	SafeGlobalUser.soRemove	Safe::GlobalUser::soRemove
Insert User Details	<Add> <User/> </Add>	SafeUser.soInsert	SafeUser.soInsert	Safe::User::soInsert
Search User Details		SafeContext.searchUsers	SafeContext.searchUsers	Safe::Context::searchUsers
Retrieve User Details		SafeUser.soRetrieveByName SafeUser.soRetrieve SafeContext.searchUsers	SafeUser.soRetrieveByName SafeUser.soRetrieve SafeContext.searchUsers	Safe::User::soRetrieveByName Safe::User::soRetrieve Safe::Context::searchUsers
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

Action	Safex	Java	C#	C++
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
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

Action	Safex	Java	C#	C++
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
Search Access Policies		SafeContext.searchCalendars	SafeContext.searchCalendars	Safe::Context::searchCalendars
Retrieve a Calendar		SafeCalendar.soRetrieveByName SafeCalendar.soRetrieve SafeContext.searchCalendars	SafeCalendar.soRetrieveByName SafeCalendar.soRetrieve SafeContext.searchCalendars	Safe::Calendar::soRetrieveByName Safe::Calendar::soRetrieve 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::exportSesssion

Action	Safex	Java	C#	C++
Import a Session		SafeContext.authenticateWithArtifact	SafeContext.authenticateWithArtifact	Safe::Context::authenticateWithArtifact

Index