

Security Domain Configuration

- [Cams Variables](#)
- [Cams Logger Types](#)
- [Configuring the Trace Logger](#)
- [Configuring the Authentication Service](#)
 - [Login Config Factory](#)
 - [Authentication Pipeline](#)
 - [Authentication Valve](#)
 - [Authentication Log](#)
- [Configuring the Access Control Service](#)
 - [Access Control Policy Factory](#)
 - [Access Request Pipeline](#)
 - [Access Control Valve](#)
 - [Access Control Log](#)
- [Configuring the Session Manager Service](#)
 - [Session Event Handler](#)
 - [Session Manager Log](#)
- [Configuring the Session Access Service](#)
 - [Session Access Pipeline](#)
 - [Session Access Valve](#)
 - [Session Access Log](#)
- [Configuring the Session Control Service](#)
 - [Session Control Pipeline](#)
 - [Session Control Valve](#)
 - [Session Control Log](#)
- [Configuring Service Manager Services](#)
 - [Services](#)
 - [LDAP Connection Pool Service](#)
 - [Cams XML User Repository Service](#)
- [Security Domain Tag Reference](#)

Configuring Cams Server

- [Cams Server Name](#)
- [Connections](#)
- [Email Notification](#)
- [Secret Key Encryption](#)
- [Logger](#)
- [Debug](#)
- [Security Domain Registry](#)
- [Resource Types](#)
- [Server Sockets](#)

Hardening Cams Security

- [Securing Cams Network Connections](#)
 - [Firewall Configuration](#)
 - [Securing Communications](#)
- [Securing Cams Files and Directories](#)
 - [Securing Cams Files and Directories under Unix/Linux](#)
 - [Securing Cams Files and Directories under Windows NT/2000](#)
- [Securing Cams Services and Agents](#)
 - [Cams Server](#)
 - [Security Domains](#)
 - [Agents](#)

XML Tag Library

- [Access Control Policy Tag Reference](#)
- [Login Configuration Tag Reference](#)
- [Security Domain Tag Reference](#)
- [Security Domain Registry Tag Reference](#)

Troubleshooting

Introduction

The Cafésoft Access Management System, Cams™, is an easy-to-use, reliable, and cost-effective security platform that centrally controls, administers, and monitors access to secure network applications and data. Administrators decide who gets access to system resources, and Cams enforces the policy and logs the results. Resources protected by Cams can reside on the corporate Internet, an extranet, or the Internet. Cams makes sites more secure and manageable by centralizing application security administration and access, rather than building custom access control systems into each application. This centralized approach reduces development and administration complexity and costs, while improving time to market, application security, and user satisfaction.

As a Cams administrator, you'll need some basic knowledge on how Cams is organized and how it works before you can configure access control, authentication, and auditing. This chapter provides an overview of Cams and refers you to other chapters that contain details on how to administer Cams servers, security services, and agents.

Centralized Access Management using Cams

Cams is a centralized access management solution, in which distributed agents delegate security decisions to a central authority – the Cams server. Cams is designed to protect virtually any kind of resource, including: web pages, files, datasources, Enterprise Java Beans, and more. When an agent sends an access control request, an access control service hosted within the Cams server decides whether to grant or deny access to the requested resource, and the requesting agent enforces the decision.

Figure 1 shows a possible Cams deployment in which User 1 attempts to access web resources available on two different web servers. Each web server contains a Cams web agent, which delegates access control and authentication requests to a centralized Cams server. User 2 tries to access an application server hosting servlets, Enterprise Java Beans, and datasources via a custom Java application or applet. The application server also contains an embedded Cams application server agent, which delegates access control and authentication decisions to the Cams server.

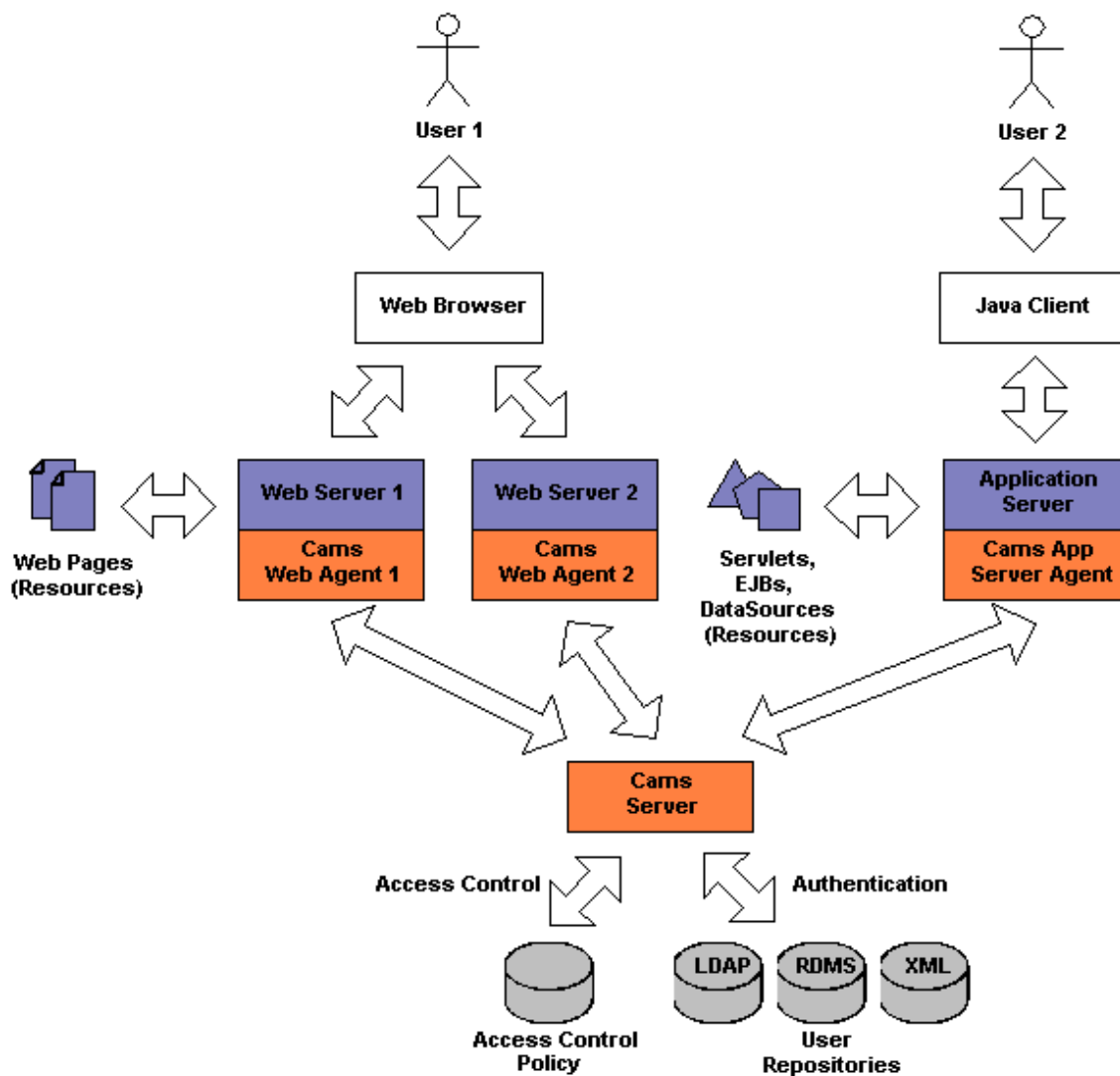


Figure 1 – Using Cams for centralized access management

As you can see, Cams enables access control policy and login configuration to be centralized at the Cams server. The Cams server also centralizes user activity logging and security administration. Also, though Figure 1 shows people making resource requests via graphical user interfaces, keep in mind that a user could also be computer. For example, a user could be a business partner order entry system placing an automated, secure order at your site.

The Cams server

At the core of Cams is the Cams server, which hosts a set of cooperative, high-level security services including:

1. Access Control service – grants or denies access to resources
2. Authentication service – verifies the user identity and establishes a session that exists until the user logs out or the session times out due to inactivity
3. Session Access service – provides information about authenticated users to agents
4. Session Control service – enables modification and explicit closure (logout) of user sessions

The Cams server is both a policy server and an authentication server handling both access control and authentication requests. Services provided by the Cams server are organized into three logical layers as shown in Figure 2.

Cams Server

Network Adapter Layer	Service Connectors Service Adapters
Service Provider Layer	High-level Security Services
Engine Layer	Low-level Security Services

Figure 2 – Logical Cams service layers

The network adapter layer enables a Cams server to offer services on different TCP/IP ports and to support network clients that speak different protocols. The service provider layer provides security services like authentication and access control. This abstraction provides the flexibility to use Cams to be extended protect almost any network resources, while reusing the same Service Provider Layer services.

Cams agents generally access security services on one TCP/IP port, while administrative services that enable Cams server shutdown and service control are available on another port. This enables you to use a firewall to limit access to certain Cams Services. Of course, you'll also use Cams itself to implement access control and authentication for Cams agents.

The Cams engine layer hosts security domain services, which provide a way to partition resources and users according to administrative needs. So what's a security domain? Read on to find out.

Security Domains

In the Cams security model, resources and user accounts are owned by a specific security domain, which enables access management to be partitioned according to organizational or physical boundaries. Different security domains may be securely configured and managed by different individuals, departments, and companies.

Each security domain contains its own set of services, which include:

1. Access control service – grants or denies access to protected resources based on an access control policy
2. Authentication service – verifies the user's identity and establishes a session that exists until the user logs out or the session times out due to inactivity
3. Session access service – provides information about authenticated users to agents
4. Session control service – enables modification and explicit closure (logout) of sessions
5. Session manager service – manages an authenticated user's session and expires it if inactive for a configurable period
6. Service manager service – enables custom security domain-specific services to be used/reused via programmer's APIs

The first four security domain services mimic those available at the service provider–level. The last two services: session manager and service manager, are unique to security domains. As an administrator, the ones that will interest you most are the access control service and the authentication service, because they are the most configurable. The following sections provide an overview of how each security domain's access control and authentication services are organized so you'll understand how to configure them.

The Access Control Service

The access control service hosted within each security domain has its own processing pipeline, called an access control pipeline. This pipeline contains a pluggable sequence of access control valves. By default, the first access control valve configured in Cams logs the request and the last valve (also referred to as the basic valve) uses the security domain–specific access control policy to grant or deny access. The overall structure of components within a security domain's access control service is shown in Figure 3.

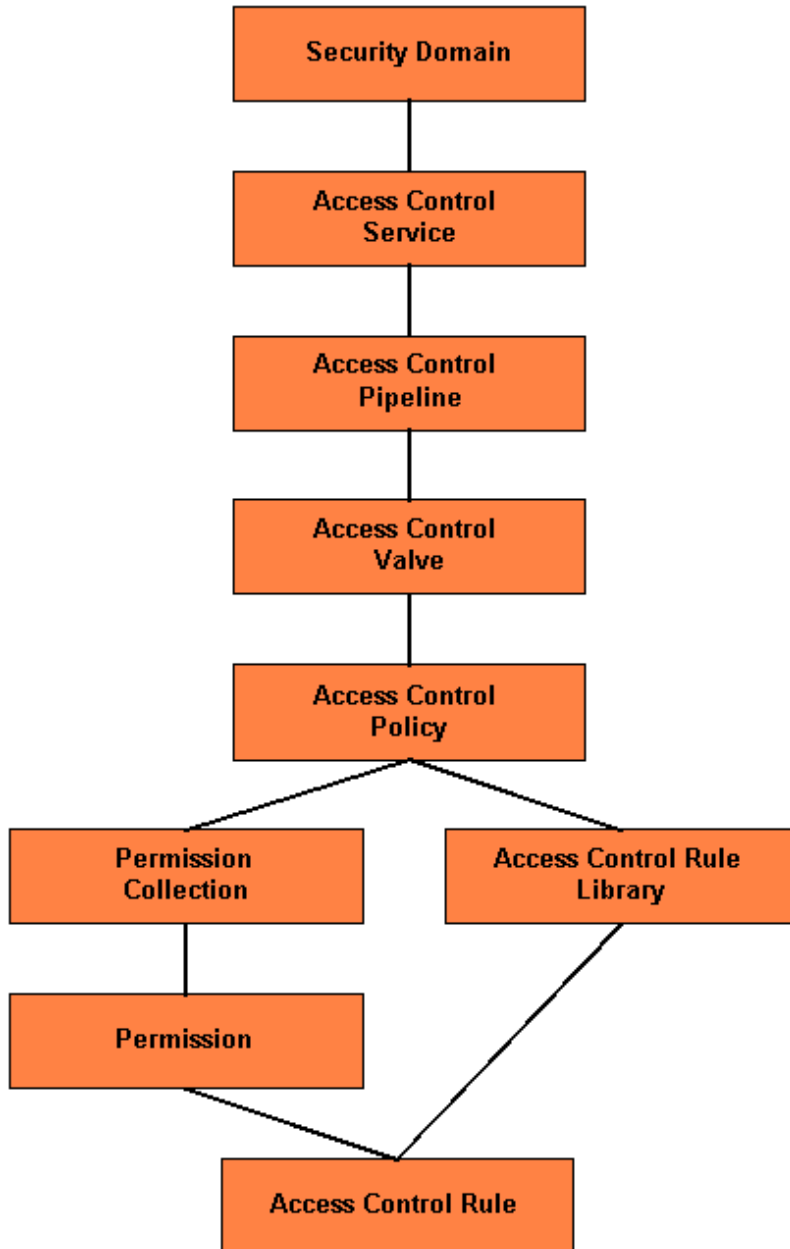


Figure 3 – The structure of a security domain's access control service

Access Control Policies

Each security domain protects the resources that it owns using an access control policy, which contains a set of permissions and a library of access control rules.

Permissions

A permission associates a set of resources (defined using a resource pattern) with one of two possible actions: an access control rule that will be evaluated to grant or deny access to the resource, or a security domain to which access control will be delegated. A special system security domain receives all access control requests and delegates most of them to one or more other security domains that you've configured. The system security domain also authenticates Cams system administrators, agents, and protects system-level resources.

Access Control Rules

Cams includes access control rules that enable you to control access to resources by:

- authenticated user role
- remote host/IP address
- confidential connectivity (e.g., SSL/TLS)
- day/time
- authentication method

These rules can be customized for your needs and combined into reusable, hierarchical expressions using logical "AND", "OR", and "NOT" operators. In addition, new access control rule types can be created using a programmer's API and plugged into any Cams access control policy. For example, you might want to create a custom rule to grant access to a resource based on a database values such as the amount of money in an account.

Access Control Requests/Responses

A Cams agent sends an access control request to a Cams server to check a user's access to a resource. The access control request contains information about the requested resource including a resource type, a unique resource identifier, and one or more requested actions. For example, the following resource request parameters correspond to a user accessing <http://www.cafesoft.com/index.html> via a web browser:

1. Resource type: http
2. Resource id: <http://www.cafesoft.com/index.html>
3. Action: GET

The access control request also contains other parameters that may be important for deciding whether to grant or deny access like:

- The IP address/host name of the client attempting to access the resource
- A Cams session identifier (if the user has already authenticated)
- A flag indicating whether or not a confidential network connection is being used by the client
- The date/time at the resource provider

Ultimately an access control service within a Cams security domain will handle the request and populate an access control response, which will be returned to the agent. You can learn about how Cams access control services work by reading [Access Control Services](#).

The Authentication Service

The Authentication service hosted within each security domain also has its own processing pipeline, called an authentication pipeline. This pipeline contains a pluggable sequence of authentication valves. By default, the first authentication valve logs the authentication request and the last valve (also referred to as the basic valve) attempts authentication based on information contained in an authentication request. The overall structure of components within a security domain's authentication service is shown in Figure 4.

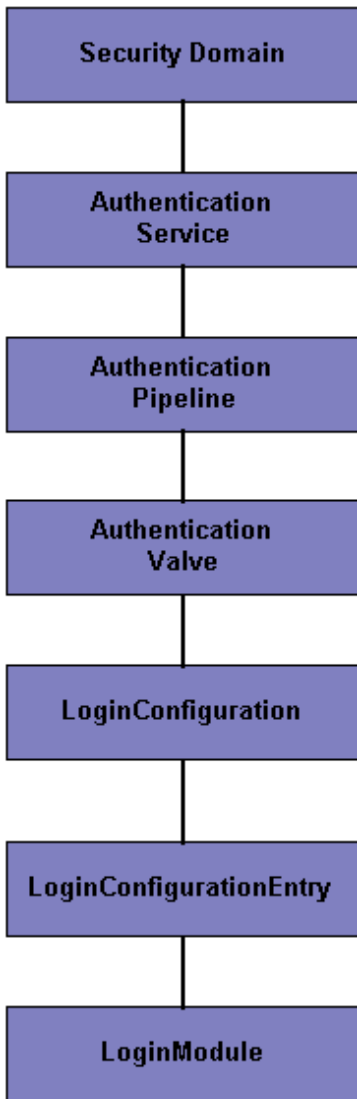


Figure 4 – The structure of a security domain's authentication service

The Login Configuration

Each security domain has its own login configuration, which configures how clients authenticate. The login configuration contains login configuration entries, one for each type of application that may attempt authentication.

Login Modules

Each Login Configuration Entry references one or more login modules, which enable you to configure authentication against a user repository like an LDAP server, a relational database management system (SQL), and an XML file containing users, passwords, groups, and roles. Cams provides a number of configurable login modules, which are compatible with the Java Authentication and Authorization Service (JAAS) API and are ready to support LDAP, RDBMS (SQL), and XML user repositories. You can also create custom login modules to implement authentication for systems not currently supported by Cams.

You can learn about how the Cams authentication services by reading [Login Configuration](#).

Cams Agents

Cams agents are software components that delegate security requests to a Cams server. The purpose of an agent may be to delegate and enforce access control requests to a Cams server or to delegate and enforce authentication requests. The way a Cams agent is installed and configured depends very much on the agent and the environment in which it's installed.

Cams Agent Types

Cams currently supports Web Agents, which are installed within a web servers (like Apache and Tomcat) to enforce access control and authentication on web pages, cgi-bin files, servlets, Java Server Pages, etc. Web agents prompt users requiring authentication using a site specific login page, then delegate the authentication requests to a Cams server with the user's credentials. Web agents support site personalization by making user-specific values available to applications via special HTTP headers.

You can learn how to install and configure the available Cams agents by reading the [Apache Web Agent](#), [IIS Web Agent](#), and [Tomcat Web Agent](#) documents.

Web Single Sign-On (SSO)

Web single sign-on enables a user to authenticate on one web server and access resources hosted on other web servers (or other virtual hosts within the same web server) without having to re-authenticate. Cams currently supports web single sign-on functionality within a single DNS domain plus subdomains. Figure 5 shows a user's experience at a site where each web server implements it's own authentication and access control and SSO is not implemented. After the user authenticates with server `www.travel.com`, then accesses resources on `www2.travel.com` and `payment.travel.com`, he must re-authenticate with each server.

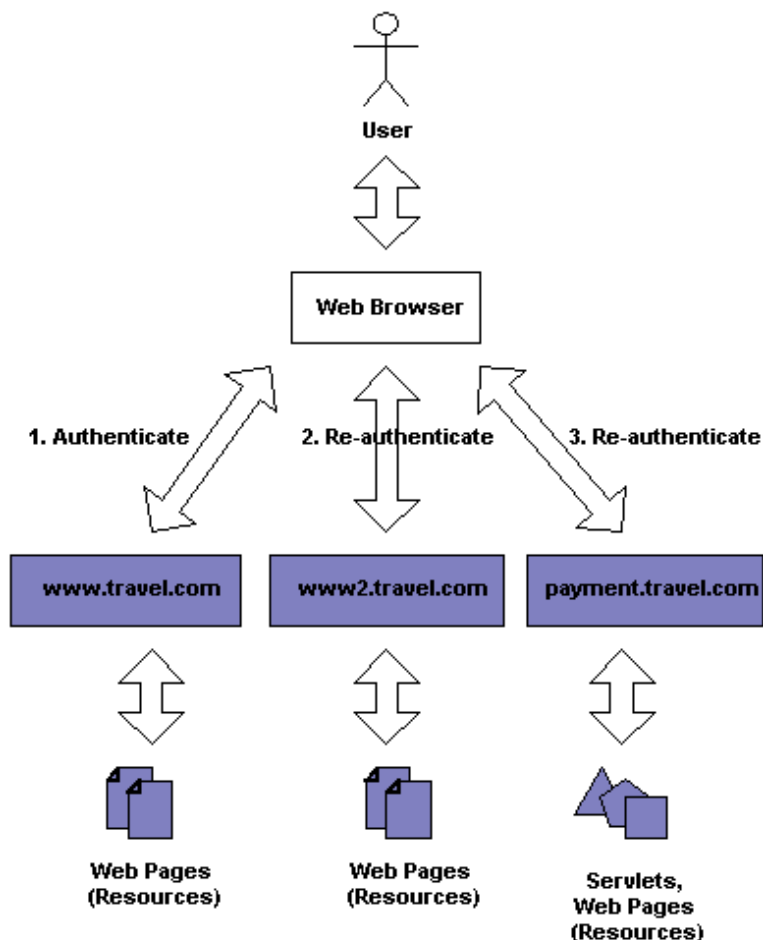


Figure 5 – Accessing multiple web servers in a DNS domain without SSO support

On a site that supports Web SSO, once a user has authenticated he can use any other server within the same DNS domain (or subdomains) without re-authenticating. Figure 6 shows this scenario.

Installation

The following instructions will help you install the Cams server on Windows NT, Windows 2000, Linux, and UNIX systems. In summary, the steps you will take include:

1. [Obtaining the Cams distribution](#)
2. [Unpacking the distribution files](#)
3. [Installing the license key](#)
4. [Installing Java \(if necessary\)](#)
5. [Starting the Cams Server](#)
6. [Testing Cams](#)

Let's get started!

Step 1 – Obtaining the Cams distribution

Cams is available for download from the Caféssoft web site at <http://www.cafesoft.com> in both zip and tar/gzip formats. We recommend you download the zip file for Windows and the gzip (.gz) file for Linux/UNIX systems.

Step 2 – Unpacking the distribution files

The distribution files will unpack into a directory named `cams-1.0`. Change to a directory where you'll install Cams.

Linux/UNIX:

```
cd /var
```

Windows

```
cd c:\
```

Unpack the distribution file (we'll assume it's in a directory named tmp).

Linux/UNIX:

```
gunzip /tmp/cams-1_0.tar.gz  
tar xvf /tmp/cams-1_0.tar
```

Windows

```
copy c:\tmp\cams-1_0.zip .  
pkunzip cams-1_0.zip
```

From here on, we'll use the symbol `CAMS_HOME` to represent the full path to the Cams server root directory and `CAMS_HTTPSERVER_HOME` to represent the full path to the HTTP server. For example, if you unpacked the distribution to "C:\", `CAMS_HOME` would be `C:\Cams1.0\camsServer` and `CAMS_HTTPSERVER_HOME` would be `C:\Cams1.0\httpServer`.

NOTE: The Windows instructions above assume you have the [pkunzip](#) command line utility on your system. You may use any zip program including graphical tools like [Winzip](#).

Step 3 – Installing the license-keys

Cams requires a license-keys file. To obtain evaluation or production license-keys, contact Caféssoft. You need to save the license-keys in a file named `cams-license-keys.xml` in the Cams server's conf directory. These instructions assume you've already obtained the license keys and saved them in the tmp directory to a file named `cams-license-keys.xml`.

Linux/UNIX

```
cp /tmp/cams-license-keys.xml $CAMS_HOME/conf
```

Windows

```
copy c:\tmp\cams-license-keys.xml %CAMS_HOME%\conf
```

WARNING: The license-keys include values that may restrict the use of Cams by version, date, host IP address, and number of concurrent connections. Do not attempt to change these values as it will invalidate the file. If the cams-license-keys.xml file is not valid or not found, you will see an error message when attempting to start Cams. If the number of concurrent sessions is exceeded at any time, a WARNING is logged and a notification is sent no more than once every two hours.

Step 4 – Installing Java (if necessary)

Cams requires that the Java 2 Platform Standard Edition release 1.4 be installed on your system. If J2SDK 1.4 is not already installed, please do so by referring to the Java download and installation instructions at:

<http://java.sun.com/j2se/1.4/download.html>

The Java installation process may set the JAVA_HOME environment variable on your system. For correct operation of Cams, JAVA_HOME must point to the J2SDK 1.4 installation. You may verify the JAVA_HOME setting by typing:

Linux/UNIX:

```
env | grep JAVA_HOME
```

Windows:

```
set j
```

If JAVA_HOME does not point to the correct directory, you may temporarily change it in the command shell you are using. Assuming the J2SDK is installed in the default location, you would enter:

Linux/UNIX (borne or bash shell):

```
JAVA_HOME=/usr/local/j2sdk1.4.0  
export JAVA_HOME
```

Windows:

```
set JAVA_HOME=c:\j2sdk1.4.0
```

That's it, you should now be ready to start the Cams Server!

Step 5 – Starting the Cams Server

The Cams server handles authentication access control and decisions made by Cams agents. You'll usually run the Cams server standalone in it's own Java Virtual Machine (JVM).

If you start the Cams server from the CAMS_HOME or CAMS_HOME/bin directory, you will not need to set the CAMS_HOME environment variable. However, if you start from any other directory, you'll need to set CAMS_HOME. Also, if you want to run the Cams server using a Java Virtual Machine security manager, a "-security" flag can be used for the Linux/UNIX and Windows run scripts. The default Cams server security policy is installed at: CAMS_HOME/conf/cams.policy and grants all permissions for classes installed in directories: CAMS_HOME/lib and CAMS_HOME/classes.

Linux/UNIX

```
CAMS_HOME=/var/cams-1.0  
export CAMS_HOME
```

Windows

```
set CAMS_HOME=C:\cams-1.0
```

To start the Cams server:

Quick Start

This document walks you through a simple example configuration of Cams. The document assumes that you've successfully installed the [Cams server and deployed at least one Cams web agent](#). Other documents in this Cams Administrator's Guide serve as a reference when you require more information.

Cams configuration requires editing of the security domain registry file and a set of XML files for each security domain you deploy. In summary, the steps you'll take are:

1. [Configure the Security Domain Registry](#)
2. [Configure the system Security Domain](#)
3. [Configure the mydomain Security Domain](#)
4. [Configure the Cams Web Agent](#)
5. [Test](#)

The instructions use the symbol CAMS_HOME to represent the directory in which you've installed Cams.

Step 1 – Configure the Security Domain Registry

The file security-domain-registry.xml defines the security domains available within a Cams server. By default, the security domain registry is located in CAMS_HOME/conf/domains. You can change the location by editing the CAMS_HOME/conf/cams.conf value:

```
security.domain.registry.factory.params=${cams.home}/conf/domains/security-domain-registry.xml
```

These instructions assume that you leave the Cams security domains in their default location. Open the file CAMS_HOME/conf/domains/security-domain-registry.xml in a text editor. The default Cams server includes two configured security domains:

- system – always required
- mydomain – a user security domain

In security-domain-registry.xml, you'll see that system and mydomain are enabled:

```
<!-- Register the system security domain -->
<security-domain enabled="true">
  <name>system</name>
  <home>${cams.home}/conf/domains/system</home>
</security-domain>

<!-- Register the mydomain security domain -->
<security-domain enabled="true">
  <name>mydomain</name>
  <home>${cams.home}/conf/domains/mydomain</home>
</security-domain>
```

To disable a security domain, you set the enabled flag to false. To completely remove a security domain, delete the references in security-domain-registry.xml and remove the corresponding directory and files. The system security domain should never be deleted. You should create at least one user security domain to protect your web resources.

Before exiting security-domain-registry.xml, find the following parameters at the top of the file:

```
<var name="http.resource.base.id" value="http://localhost:80"/>
<var name="https.resource.base.id" value="https://localhost:443"/>
<var name="cams.logs.base.dir" value="${cams.home}/logs"/>
```

These parameters are substitution values that make it easier for you to work with permissions and other values in the security domain configuration files. Use of a substitution value such as http.resource.base.id for permissions in your access control policy file (more on this later) enables you to centrally change the root URI value from http://localhost:80 to another DNS name. This greatly facilitates moving a security domain configuration from a development, to a test, to a production system. For example, Change:

```
<var name="http.resource.base.id" value="http://localhost:80"/>
<var name="https.resource.base.id" value="https://localhost:443"/>
```

to:

Quick Start

```
<var name="http.resource.base.id" value="http://myhost:80"/>
<var name="https.resource.base.id" value="https://myhost:443"/>
```

where myhost is the fully qualified hostname (e.g., host.company.com or www.company.com) of the deployment system.

You can define as many substitution values as you require with names that work best for you. Substitution values are hierarchically inherited from the security domain registry by the security domain configuration files. Additionally, you can use the following intrinsic substitution values provided by the Cams server in any security domain configuration file:

- cams.home – The fully qualified path to the Cams server installation directory
- cams.security-domain.home – The fully qualified path to the directory for a given security domain
- cams.security-domain.name – The current security domain name
- cams.server.name – The name of the Cams server
- cams.server.port – The connection port of the Cams server

Step 2 – Configure the system Security Domain

Each security domain configuration requires three files:

- access-control-policy.xml
- login-config.xml
- security-domain.xml

NOTE: You'll also see an optional file, cams-users.xml, in each security domain. This file contains the Cams user directory, which is recommend for use with the system security domain and to facilitate testing and development in user security domains. Most test or production user security domains will have an LDAP or SQL relational database configured as the user directory.

You will not need to change to the system security domain for this example. However, a review of the system security domain's default access control policy configuration file provides an excellent introduction.

Access Control Policy

The access-control-policy.xml file configures a Cams access control policy, which consists of a default bias and rules and permissions. Rules are define how an access control policy will protect resources. Most of the rules you'll use are supplied with Cams (they're called intrinsic rules). You can also create custom rules and/or combine existing rules to create new ones. A permission is the binding of a rule to a specific resource pattern (a URL in web security). The default bias specifies if access to resources without permissions will be granted or denied.

Change directories to the system security domain and open the access-control-policy.xml file in a text editor.

Linux/UNIX:

```
cd $CAMS_HOME/conf/domains/system
vi access-control-policy.xml
```

Windows

```
CD %CAMS_HOME%\conf\domains\system
notepad access-control-policy.xml
```

At the top of the file you see the declaration of an access control policy and the specification of a default bias:

```
<access-control-policy defaultBias="denied">
```

It is recommended that you set the default bias to denied and grant access as required with permissions. This helps prevent situations where access accidentally granted due to a misconfiguration or omission in the access control policy.

Figure 1 shows the system security domain's default http permission collection. The first permission grants access to the cams directory unconditionally. This ensures that Cams login, denied, and error pages can always be accessed. The next two permissions delegate all HTTP and HTTPS resource requests to the mydomain security domain. A permission always has either an <acr-ref> or <owner> tag. The <acr-ref> references an access control rule. The <owner> tag delegates administration of resource permissions to a user security domain. A fourth permission used for testing Cam web agents is commented out (see the [Web Agent Guide](#) for more information). You uncomment it (make it active) by moving the trailing "-->" up four lines.

Quick Start

```
<!-- HTTP and HTTPS resource permissions -->
<permission-collection type="http" desc="HTTP/HTTPS permissions">

<!-- Grant access to all to cams login, denied, and error pages -->
<permission desc="Cams default pages" actions="GET,POST">
  <resource-pattern id="{http.resource.base.id}/cams/*"/>
  <acr-ref id="granted"/>
</permission>

<!-- Delegate HTTP permissions -->
<permission desc="Delegate all HTTP resource permissions" actions="">
  <resource-pattern id="{http.resource.base.id}/*"/>
  <owner id="mydomain"/>
</permission>

<!-- Delegate HTTPS permissions -->
<permission desc="Delegate all HTTPS resource permissions" actions="">
  <resource-pattern id="{https.resource.base.id}/*"/>
  <owner id="mydomain"/>
</permission>

<!-- Only administrators can access the web agent test page
<permission desc="allow administrators to access web agent test page" actions="">
  <resource-pattern id="{http.resource.base.id}/cams/cams-webagent-test*"/>
  <acr-ref id="cams administrator rule"/>
</permission>
-->

</permission-collection>
```

Figure 1 – Permissions from the default system security domain

Before exiting this file, look at the `<permission-collection type="cams">`. This permission collection defines the permissions for Cams web agent connections. These permissions should always be handled by the system security domain. You'll see the access control rule reference:

```
<acr-ref id="cams agent rule"/>
```

Figure 2 shows the access control rules section of the file, marked by the `<acr-lib>` tags. You see that the cams agent rule requires that authenticated Cams web agents have the `cams-agent` role. Now, open the `cams-users.xml` user directory in a text editor to see that the `cams-tomcat-agent` has the `cams-agent` role. Without it, the Cams Tomcat web agent could not authenticate correctly and resource check requests would be denied. Cams web agents typically use an account in the system security domain's `cams-users.xml` file. Optionally, you can configure other user directories for Cams web agents such as LDAP or SQL databases.

The `cams administrator` rule restricts access to a specified resource to users with the `administrator` role. By default, it is only used by the Cams web agent test page, if uncommented.

```
<!-- Library of access control rules -->
<acr-lib>

<!-- Cams administrators must have the "administrator" role -->
<auth-acr id="cams administrator rule">
  <role-constraint>
    <role-name>administrator</role-name>
    <role-class>com.cafesoft.cams.auth.CSRolePrincipal</role-class>
  </role-constraint>
</auth-acr>

<!-- Cams agents must have the "cams-agent" role -->
<auth-acr id="cams agent rule">
  <role-constraint>
    <role-name>cams-agent</role-name>
    <role-class>com.cafesoft.cams.auth.CSRolePrincipal</role-class>
  </role-constraint>
</auth-acr>

</acr-lib>
```

Figure 2 – Rules from the default system security domain

Login Configuration

These instructions assume that you are using the Cams user directory with the system security domain, which is already configured by default. Hence, you will not need to edit the `security-domain.xml` or `login-config.xml` files for the system security domain at this time.

Step 3 – Configure the mydomain Security Domain

Open `CAMS_HOME/conf/domains/mydomain/access-control-policy.xml` in a text editor. The default bias for the mydomain security domain denies access to all resources but the default HTTP and HTTPS permission grant access. This is equivalent to specifying a default bias of granted with no permissions.

Figure 3 shows the default http permission collection for mydomain. The two default permissions are similar to those in the system security domain, except a Cams intrinsic rule (in this case, granted) and HTTP actions are specified. Specifying HTTP actions enables access control to be specific to a resource and the action. So, you could have a different permission on HTTP GETs than POSTs, or allow POST access but not GET. Mydomain has no specified rules.

```
<!-- Grant GET and POST permission to all HTTP resources -->
<permission desc="HTTP Permissions" actions="GET,POST">
  <resource-pattern id="{http.resource.base.id}/*"/>
  <acr-ref id="granted"/>
</permission>

<!-- Grant GET and POST permission to all HTTPS resources -->
<permission desc="HTTPS Permissions" actions="GET,POST">
  <resource-pattern id="{https.resource.base.id}/*"/>
  <acr-ref id="granted"/>
</permission>
```

Figure 3 – The default mydomain security domain permissions

The access control rule reference to the granted rule id, `<acr-ref id="granted"/>`, is an intrinsic Cams rule that grants access to everyone for the specified resource. In this case, it is applied to the root directory for all HTTP and HTTPS resource requests. To tighten security for mydomain HTTP resource requests, first, define an access control rule that requires a user to have the myrole role. Between the `<acr-lib>` and `</acr-lib>` tags, insert:

```
<!-- Requires authenticated users to have the "myrole" role -->
<auth-acr id="require myrole">
  <role-constraint>
    <role-name>myrole</role-name>
  </role-constraint>
</auth-acr>
```

Now, edit the HTTP resource permission by changing the tag `<acr-ref id="granted"/>` to:

```
<acr-ref id="require myrole"/>
```

The permission should now look like:

```
<!-- Grant GET and POST permission to all HTTP resources -->
<permission desc="HTTP Permissions" actions="GET,POST">
  <resource-pattern id="{http.resource.base.id}/*"/>
  <acr-ref id="require myrole"/>
</permission>
```

Alternatively, you might leave the default permissions in tact and add a new permission to protect only a specific directory. For example, to secure a directory on your web server named `mysecuredir`, add the following permission:

```
<!-- Grant GET and POST permission to all HTTP resources -->
<permission desc="HTTP Permissions" actions="GET,POST">
  <resource-pattern id="{http.resource.base.id}/mysecuredir/*"/>
  <acr-ref id="require myrole"/>
</permission>
```

Save and exit the file. Next, you need to add a user with the myrole role to the `cams-user.xml` file in the mydomain security domain. Using a text editor, add the following line to `CAMS_HOME/conf/domains/mydomain/cams-user.xml`:

```
<user name="myuser" password="myuser" roles="myrole"/>
```


Security Domain Registry Configuration

Cams uses security domains to logically partition the management of access control, authentication, and logging. Each Cams server will usually have multiple security domains, each of which must be declared in the security domain registry. Each Cams security domain includes at least the following configuration files:

- access-control-policy.xml
- login-config.xml
- security-domain.xml

The Cams security domain registry maintains basic information about each security domain known to Cams, including the name and location of configuration metadata. The security domain registry also enables you to define global variables that are passed to all security domains and can be referenced within their configuration files.

This document describes the values you will use to edit the Cams security domain registry. See also the [Security Domain Registry Tag Reference](#) for a complete listing of and reference information on all the XML tags used in the security-domain-registry.xml file.

Configuring the Security Domain Registry Factory

A Cams security domain registry factory is a Java class that loads information about the security domains known to Cams from the security-domain-registry.xml file. This factory is configured in the Cams CAMS_HOME/conf/cams.conf file with the values shown in Example 1.

```
#
#--- The factory class used to create the SecurityDomainRegistry
#--- and it's configuration parameter(s)
#
security.domain.registry.factory=\
    com.cafesoft.security.engine.domain.XmlSecurityDomainRegistryFactory
security.domain.registry.factory.params=\
    ${cams.home}/conf/domains/security-domain-registry.xml
```

Example 1 – The default Cams server security domain registry factory configuration

The default Cams security domain registry factory (XmlSecurityDomainRegistryFactory), loads an XML registry file located in the directory specified by the parameter. The parameter's \${cams.home} substitution value is specified by the CAMS_HOME environment variable to the Cams Server via the startup script.

Normally, you will only need to change this configuration if you move the security-domain-registry.xml file to another directory. The use of a factory provides the flexibility to store the Cams security domain registry in other document formats or databases should it be required.

Configuring the Cams XML Security Domain Registry

Example 2 shows a Cams XML security domain registry file. This file defines two security domains, system and mydomain, which you'll see at the bottom of the file. To add a new security domain, copy and paste an existing security domain and change the values to reflect the name and location of the new security domain configuration files.

NOTE: Only the system security domain is required, but most installations should delegate resource permissions from the system security domain to one or more custom security domains for administrative purposes.

```
<?xml version="1.0"?>
<!DOCTYPE security-domain-registry SYSTEM
    "http://cafesoft.com/security-domain-registry_1_0.dtd">

<security-domain-registry>

    <!-- Global variables (you can define your own as required)

        http.resource.base.id - the fully qualified root domain (including port)
                               for the connection to resources protected by a
                               Cams http agent
        https.resource.base.id - the fully qualified root domain (including port)
                               for the SSL/TLS connection to resources protected
                               by a Cams http agent
```


Access Control Services

This chapter provides detailed information on how Cams access control services work and how to configure them. Instructions are provided for configuring a security domain's access control policy. See also the [Access Control Policy Tag Reference](#) for a complete listing of and reference information on all XML tags used in the access-control-policy.xml file.

Access Control Conceptual Model

The Cams access control model defines two major responsibilities:

- Policy Enforcement Point – A location in the system where a security policy is enforced. For Cams, this is usually a Cams web agent installed in a resource provider like Apache or Tomcat, but may also be a custom agent embedded in an application, etc.
- Policy Decision Point – A location in the system where the access control decisions are made. For Cams, this is always the Cams server.

To implement access control on a client's request, the policy enforcement point asks the policy decision point whether to grant or deny access to a resource. If the policy decision point grants access, then the policy enforcement point allows access to the resource. If the policy decision point denies access, then the policy enforcement point will not allow access to the resource. Figure 1 shows these abstract access control roles.

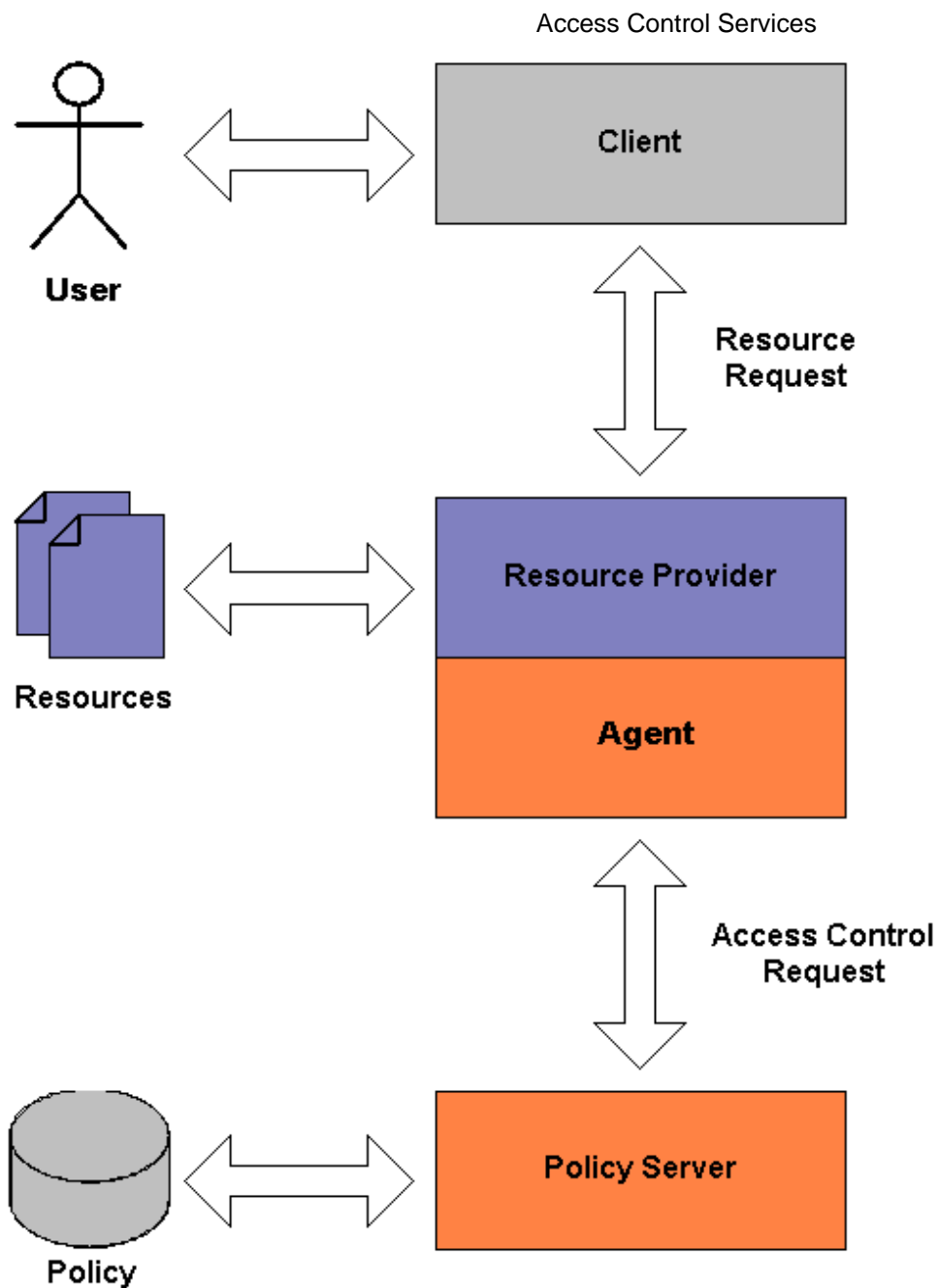


Figure 1 – The Cams centralized access control model

Figure 2 shows a deployment where the resource provider is a web server, the resources are web pages, CGI scripts, and other URL resources accessible via the server. The policy enforcement point is a Cams web agent and the policy decision point is the Cams server. The web browser requests access to an HTTP URL from a web server, where a web server agent intercepts the request and asks the Cams server if access is granted or denied. The Cams server *decides* whether or not access is granted and the agent *enforces* the decision.

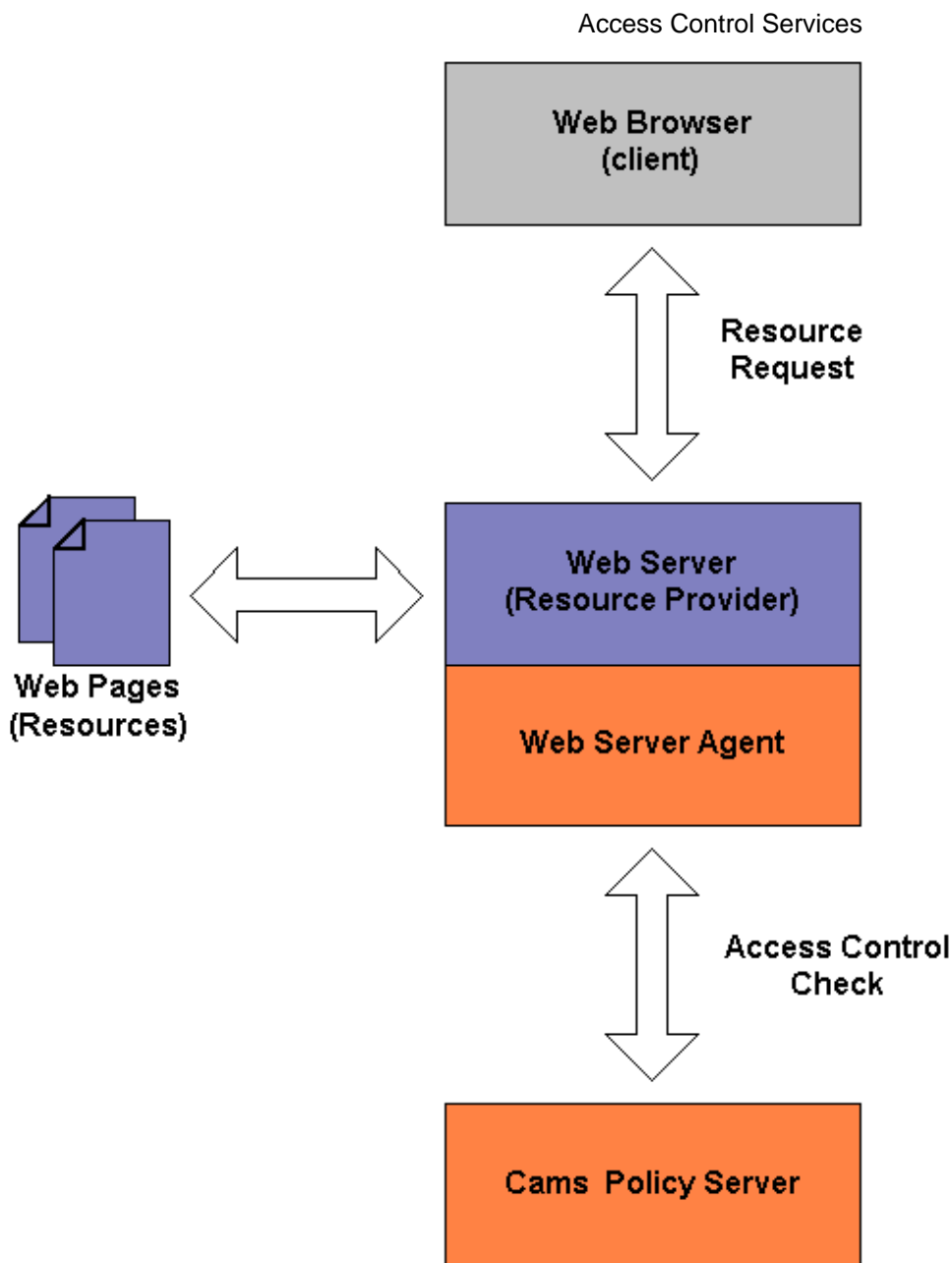


Figure 2 – Example: Web resources access control model

Cams Security Domains

Cams partitions access control, authentication, and session management responsibilities into security domains. Security domains enable a separation of administrative responsibilities based on organizational boundaries. Each security domain has an access control service that uses an access control policy that contains permissions for accessing protected resources. Each permission:

- is associated with a type of resource that it protects (e.g., "http", "cams-service", "wireless")
- contains a pattern for matching resource identifier(s) (e.g., "http://www.cafesoft.com/secure*")
- may specify one or more actions (e.g., "GET", "POST")
- references an access control rule that protects access to matching resources or a security domain to which access control will be delegated (e.g., "acr-ref id=granted")

Permissions are organized into separate permission collections (based on type) and access control rules are stored in an access control rule library, which enables them to be re-used, managed, and referenced from permissions and other access control rules. Figure 3 shows the relationships between these major access control components.

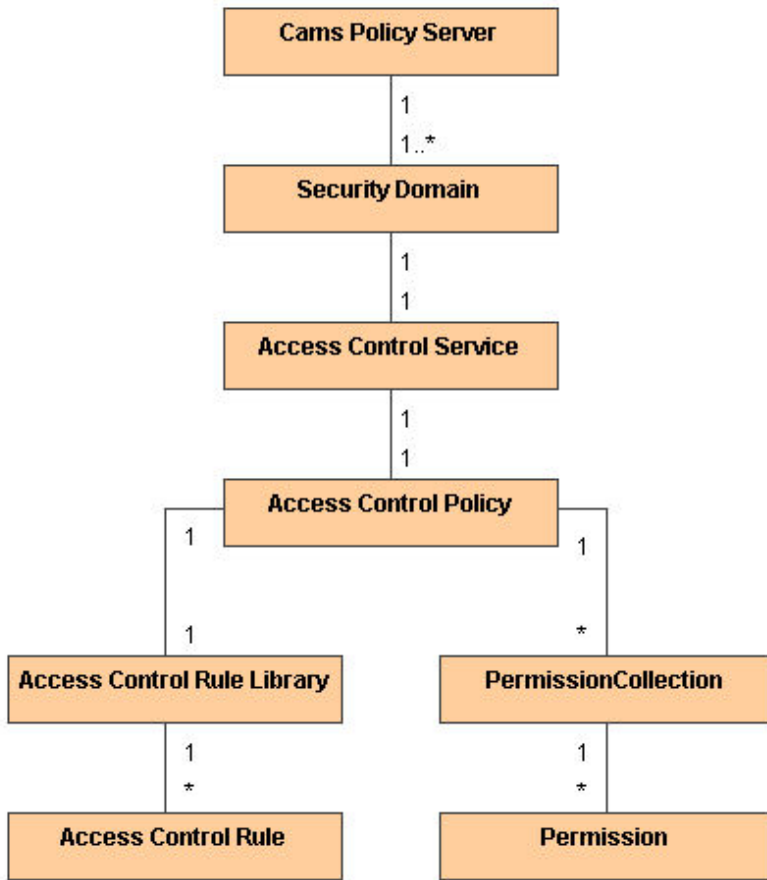


Figure 3 – Relationships between major Cams access control components

How Cams Access Control Works

When a Cams agent needs to check access to a resource, it submits a resource request to a Cams Server. The Cams Server gives the request to the access control service within a special system security domain, which either handles it or delegates the request to another security domain as configured in its access control policy. So, in order for a resource to be protected by Cams, a permission that matches its identifier and action must be configured in the Cams system security domain. In many cases, the system security domain will simply delegate the access control check to another security domain, but it is also responsible for protecting system-level Cams services.

Access control decisions are based on a resource request, which is sent from a Cams agent as part of an access control request. Each resource request includes a resource type, a unique resource identifier, and one or more requested actions. For example, an attempt to access a web page via a web browser might correspond to the following resource request parameters:

1. Resource type: http
2. Resource id: http://www.cafesoft.com:80/index.html
3. Action: GET

The enclosing access control request also contains other parameters that may be important for deciding whether to grant or deny access including:

- The IP address/host name of the client attempting to access the resource
- A Cams session identifier (if a previously authenticated user)
- A flag indicating whether or not a confidential network connection is being used by the client
- The date/time at the Resource provider

When the Cams high-level access control Service receives this request, the following actions are taken:

1. A Cams access control agent, usually installed at a resource provider, will connect with the Cams server and send an access control request (hereafter referred to as the request).
2. The request and its enclosed resource request are given to the Cams access control service.

Access Control Services

3. The access control service gives the request to the system security domain's access control service
4. The system security domain's access control service passes the request to its access control pipeline, which contains a series of access control valves
5. Each access control valve can: ignore the request, partially handle the request, or completely handle the request.
6. By default, an access control valve will log the request
7. By default, a Cams basic access control valve will invoke the security domain's access control policy
8. The access control policy will find the permission collection that matches the resource type
9. The permission collection will find the permission that best matches the resource request. The rules for best match can depend on the resource type, but for HTTP resources, the best match is the one that specifies a matching action and has the longest resource id pattern.
10. If the permission references an access control rule, that rule will be invoked on the request. If the permission references another security domain, the request will be forwarded to that security domain.
11. The invoked access control rule will evaluate the request, possibly invoking other nested access control rules. An access control Response will be populated and returned to the access control agent.

This entire processing pipeline is completely pre-loaded and memory-resident, so it's very fast. The design enables pluggable support for new resource types, new access control rule types, and pluggable access control valves. The Cams engine is thread-safe and can even be used stand-alone or embedded within applications.

Whenever a Cams security domain is asked to check access on a resource, it finds the most specific permission matching the request and either invokes the access control rule associated with the permission or delegates the request to the security domain referenced by the permission. If the access controller does not find a matching permission, then access will be either granted or denied based on default settings for the security domain. In order for a permission to match a specific resource request, the following criteria must be met:

- the id of the requested resource must match the permission's resource pattern
- the action on the requested resource must match one or more of the permission's actions

To understand permission matching rules, consider the example access control policy represented within Table 1.

Permission #	Resource Type	Resource Pattern	Action(s)
1	http	http://www.foo.com/secure/*	GET, POST
2	http	http://www.foo.com/secure/employee/*	GET, POST
3	http	http://www.foo.com/secure/employee/*	PUT, DELETE
4	http	http://www.foo.com/secure/employee/*	POST

Table 1 – Example permissions within an access control policy

Each row represents a permission to access some HTTP (web) resources.

- for request: "GET http://www.foo.com/secure/index.html", only permission #1 matches.
- for request: "POST http://www.foo.com/secure/employee/suggestion" permissions #1 and #2 match the request, but #2 is more specific so it takes precedence. In this case "more specific" means that permission #1's resource pattern is longer than permission #2's resource pattern. (The request does not match permission #3 because the requested "POST" action is not associated with permission #3).
- for request: "DELETE http://www.foo.com/secure/employee/picnic_2002.html" only permission #3 matches because it's the only one that matches the resource pattern and the "DELETE" action.

Permission #4 is incompatible with permission #2 because both have identical resource patterns and a matching action. Cams requires unambiguous rules for assigning access control responsibility within a security domain, so it will not allow insertion of a permission that overlaps another permission. Permissions are considered overlapping if they have the same type, have identical resource patterns, and have at least one action in common. Without this precaution, it would not be able to decide which of the two permissions has responsibility for protecting a matching resource request. If an access control policy contains an overlapping permission during initial access control policy loading, the access control policy will fail to load and the enclosing security domain will not be started.

Once the best matching permission is determined, Cams will take one of two actions:

1. if the permission references another security domain, then access control is delegated to that security domain
2. if the permission references and access control rule, then the rule is invoked on the request to grant or deny access

Cams provides standard access control rules that can be used to protect resources based on:

- the identity of a user
- the remote host on which the user is operating
- the confidentiality of the network connection by which a resource might be accessed
- the type(s) of authentication successfully passed by the user

Furthermore, you can combine existing access control rules using and, or, and not operators. If the standard rules are not sufficient for your site, programmers can implement custom business rules that can be plugged into Cams.

Configuring Cams Access Control Policy

Cams stores each security domain's access control policy in an XML file named *access-control-policy.xml*. This section shows an example of that file in Figure 4 to describe how to configure permissions and access control rules within this file. The example shows how an access control policy might be configured to protect some HTTP (web) resources available from a fictitious application server at: <http://www.acme.com> that hosts:

- web pages to be made available only to authenticated employees
- servlets and/or web applications available only to authenticated managers
- web pages available to everybody

Each of these access control requirements can be expressed as a distinct permission within the access control policy's `<permission collection>` start and end tags. Custom access control rules are defined in the access control rules library within the `<acr-lib>` start and end tags. Please refer to the comments in Example 1 to understand more about each permission and its associated rule reference.

```
<!-- Declare an access control policy -->
<access-control-policy>

  <!-- HTTP (Web) resource permissions -->
  <permission-collection type="http" desc="HTTP (Web) Resources">

    <!-- Declare web pages available to all ACME employees -->
    <permission desc="ACME Employee Web Resources">
      <resource-pattern id="http://www.acme.com/employee*" />
      <acr-ref id="ACME Employee Rule" />
    </permission>

    <!-- Declare web pages available only to ACME manager users -->
    <permission desc="ACME Manager Web Resources">
      <resource-pattern id="http://www.acme.com/manager*" />
      <acr-ref id="ACME Manager LAN-Auth-Conf Rule" />
    </permission>

    <!-- Declare web pages available to the public -->
    <permission desc="ACME Public Web Resources">
      <resource-pattern id="http://www.acme.com/*" actions="GET" />
      <acr-ref id="granted" />
    </permission>

  </permission-collection>

  <!-- Access Control Rule Library -->
  <acr-lib>

    <!-- Require an authenticated user to have the "employee" role. -->
    <auth-acr id="ACME Employee Rule">
      <role-constraint>
        <role-name>employee</role-name>
      </role-constraint>
    </auth-acr>

    <!-- Require hosts to be on the ACME Local Area Network. -->
    <host-acr id="ACME LAN Rule">
      <accept-address-constraint>
        <address>192.168.0.*</address>
      </accept-address-constraint>
    </host-acr>

    <!-- Require an authenticated user to have the "manager" role
         that is implemented by a particular role class. -->
    <auth-acr id="ACME Manager Auth Rule">
      <role-constraint>
        <role-name>manager</role-name>
        <role-class>com.cafesoft.cams.auth.CSRolePrincipal</role-class>
      </role-constraint>
    </auth-acr>
  </acr-lib>
</access-control-policy>
```

Access Control Services

```
</role-constraint>
</auth-acr>

<!-- Require:
    1. the host to be on the ACME Local Area Network
    2. the user to be authenticated with the "manager" role
    3. a confidential (SSL/TLS) connection for privacy -->
<acr id="ACME Manager LAN-Auth-Conf Rule">
  <acr-ref id="ACME LAN Rule">
  <and/>
  <acr-ref id="ACME Manager Auth Rule">
  <and/>
  <confidential/>
</acr>

</acr-lib>

</access-control-policy>
```

Example 1 – An example access control policy file

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Login Configuration

Cams uses the Java Authentication and Authorization Service (JAAS) to provide user authentication services. JAAS was introduced as an optional package to the Java 2 SDK, Standard Edition (J2SDK) 1.3. This was also known as JAAS 1.0. JAAS was officially integrated into the J2SDK with the 1.4. Cams does not use the authorization features of JAAS.

The JAAS authentication API defines a Java version of the Pluggable Authentication Module (PAM) framework, which permits applications to remain independent from underlying authentication technologies. The PAM framework allows the use of new or updated authentication technologies without requiring modification to applications.

Login modules are plugged into the Cams server to configure authentication for different kinds of clients and agents. Currently, the standard login modules included with Cams are:

- JDBC Login Module – for use with user repositories stored in SQL databases
- LDAP Login Module – for use with most LDAPv3 compliant directories
- XML Login Module – for use with the Cams native XML user repository (primarily used to ease evaluation and development)

This chapter describes how to configure these existing login modules for use with Cams. See also the [Login Configuration Tag Reference](#) for a complete listing of and reference information on all the XML tags used in the login-config.xml file.

NOTE: If the standard Cams login modules meet your requirements, read on. If not, then you may want to consult the [Programming Cams JAAS Login Modules](#) chapter of the Cams Programmer's Guide to understand how to create a new Cams login module.

Login Configuration File

Each Cams security domain must have a login configuration defined by a login-config.xml file, which is loaded by Cams during login module initialization. The login-config.xml file contains both required and optional data, specific to each login module. One or more login module configurations can be specified by nesting values within one or multiple <login-config-entry name="name"> and </login-config-entry> tags, as you can see in the [sample login-config.xml](#) file. You invoke a configuration by starting the Cams service and referencing a specific <login-config-entry> name.

Within the <login-config-entry> open and close tags, you can specify how one or more login modules will be used for a configuration. The required parameters for each login module entry are className and flag located in the <login-module-entry> tag shown in Example 1. The class name specifies the fully qualified location of the Java class for the given login module. The login modules currently shipped with Cams include:

- com.cafesoft.cams.auth.login.module.JdbcLoginModule
- com.cafesoft.cams.auth.login.module.LdapLoginModule
- com.cafesoft.cams.auth.login.module.XmlLoginModule

```
<login-config-entry name="http">
  <login-module-entry
    className="com.cafesoft.cams.auth.login.module.XmlLoginModule"
    flag="REQUIRED">
    <options>
      <option name="serviceId" value="cams-user-repository"/>
      <option name="debug" value="true"/>
    </options>
  </login-module-entry>
</login-config-entry>
```

Example 1 – Cams login-config.xml example

The flag parameter is a control value that determines the behavior of multiple modules by specifying whether any particular module is required, requisite, sufficient, or optional.

- REQUIRED – Normally set when authentication by this module is a must. With this flag, any module failure results in the PAM framework returning the error to the caller after executing all other modules on the stack. For the function to be able to return success to the application, all required modules have to report success.
- REQUISITE – The module must be successfully checked in order for the authentication to be successful. If a requisite module check fails, the user is notified immediately with a message reflecting the first failed required or requisite module.
- SUFFICIENT – Used if the module succeeds the PAM framework, then returns success to the application immediately without trying any other modules. For failure cases, the sufficient modules are treated as optional.
- OPTIONAL – Used when the user is allowed access, even if that particular module has failed. With this flag, the PAM framework ignores the module failure and continues processing the next module in sequence.

Login Configuration

Specifying more than one <login-module-entry> for a single <login-config-entry> configuration is also known as stacking. The interaction between flags when stacking is complicated. It is recommended that you keep these configurations as simple as possible. In most cases, you'll specify a single login module with flag value of REQUIRED.

The optional values are specific to each login module and will be explained below in the sections for the JDBC, LDAP, and XML login modules. All options tags require name and value parameters. For example, all Cams login modules include a debug option to toggle on/off debug. The tag is <option name="debug" value="true"/> with the value either true or false.

JDBC Login Module

Example 2 shows the sample entry for the JDBC login module. You use this login module to access user repository information that is stored in SQL databases such as Oracle, DB2, Microsoft SQL, MySQL, Sybase, and virtually any other database with a JDBC driver. The sample configuration uses the JDBC to ODBC bridge that ships with the JDK to access a Windows datasources (DSN) named SampleJdbcLogin. You can configure the [sample Microsoft Access database](#) with the SampleJdbcModule DSN to test the JDBC login module.

```
<login-module-entry
  className="com.cafesoft.cams.auth.login.module.JdbcLoginModule"
  flag="REQUIRED">
  <options>
    <option name="debug" value="false"/>
    <option name="jdbcDriver" value="sun.jdbc.odbc.JdbcOdbcDriver"/>
    <option name="jdbcUrl" value="jdbc:odbc:SampleJdbcLogin"/>
    <option name="jdbcUsername" value=""/>
    <option name="jdbcPassword" value=""/>
    <option name="userPreparedStatement"
  value="SELECT PASSWORD FROM USERS WHERE USERNAME = ? "/>
    <option name="passwordColumn" value="PASSWORD"/>
    <option name="rolePreparedStatement"
  value="SELECT ROLE FROM USER_ROLES WHERE USERNAME = ? "/>
    <option name="roleColumn" value="ROLE"/>
  </options>
</login-module-entry>
<callback-handler
  classname="com.cafesoft.cams.auth.callback.NamePasswordCallbackHandler"/>
```

Example 2 – Cams JdbcLoginModule sample

The required JDBC login module options values are:

- jdbcDriver – The fully qualified name of the JDBC driver for your database.
- jdbcUrl – The URL required to access the database (see the JDBC driver documentation).
- jdbcUsername – The username required to access the database (can be null).
- jdbcPassword – The password required to access the database (can be null).
- userPreparedStatement – The SQL prepared statement to select the user's password from the database. The username entered by the user at login will be substituted for the "?".
- passwordColumn – The table column where the user's password is stored.
- rolePreparedStatement – The SQL prepared statement to select the user's groups/roles from the database. The username entered by the user at login will be substituted for the "?".
- roleColumn – The table column where the roles are stored.

Passwords can be encrypted using Unix Crypt, MD5, and SHA, SMD5, and SSHA where the algorithm is specified within curly brackets in a preceding label (e.g., {CRYPT}, {MD5}, {SHA}, {SMD5}, {SSHA}).

NOTE: If you need to use a driver other than the JDBC to ODBC bridge that ships with the J2SDK, you must supply and install the driver. To use the driver with Cams, copy the driver jar file to the ext directory of your J2SDK installation. For example, if you want to use MySQL with the standard J2SDK installation, you would copy mm.mysql-2.0.14-bin.jar to JAVA_HOME/jre/lib/ext.

Using JDBC Connection Pooling

The overhead of opening and closing database Connections can be quite high. In production or high-volume environments, you can dramatically improve performance by using JDBC Connection Pooling. When a Connection is needed, it is borrowed from the pool. When done with the Connection, it is returned to the pool for reuse later.

Cams provides a JDBC Driver that implements Connection pooling. After configuring a Connection pool using a service provided by Cams, LoginModules (and other components that you can plug into Cams) can then obtain pooled Connections from the Cams JDBC Driver. When the Connection is closed, it is actually just returned to the pool of available Connections. The pooling JDBC Driver does the

Login Configuration

work of expanding and shrinking the pool of Connections within the parameters you configure. It also verifies a Connection is still open before returning it to your JDBC client. In addition, it also automatically destroys broken Connections.

To use JDBC Connection Pooling from JdbcLoginModule, you must:

1. Setup a JDBC Connection Pool using a service provided by Cams
2. Configure the JdbcLoginModule to use the Cams–provided pooling JDBC Driver

Step 1 – Configuring a JDBC Connection Pool (in security–domain.xml)

```
<service-manager className="com.cafesoft.core.service.StandardServiceManager">
  ...
  <!-- Register a Jdbc ConnectionPool with the server -->
  <service id="myusersdb-connection-pool" enabled="true">
    <service-type>com.cafesoft.cams.service.JdbcConnectionPoolService</service-type>
    <service-class>com.cafesoft.security.engine.service.JdbcConnectionPoolServiceImpl</service-class>
    <param-list>
      <param name="poolName" value="myusersdb_pool"/>
      <param name="driver" value="org.hsqldb.jdbcDriver"/>
      <param name="url" value="jdbc:hsqldb:${cams.security-domain.home}/database/myusersdb"/>
      <param name="user" value="sa"/>
      <param name="password" value=""/>
      <param name="initialCapacity" value="1"/>
      <param name="maximumCapacity" value="5"/>
      <param name="minimumCapacity" value="1"/>
      <param name="poolIncrement" value="1"/>
      <param name="allowShrink" value="false"/>
      <param name="shrinkPeriod" value="10"/>
      <param name="refreshPeriod" value="15"/>
      <param name="poolUser" value="my_pool_user"/>
      <param name="poolPassword" value="my_pool_password"/>
      <param name="debug" value="false"/>
    </param-list>
  </service>
  ...
</service-manager>
```

Example 3 – Configuring a JDBC Connection Pool using a Cams–provided Service

1. The service must be configured within the service–manager of the security domain from which you want connection pooling.
2. The service id can be any unique value within the security domain and the service must be enabled
3. The service–type and service–class must be exactly as shown
4. The poolName parameter designates the name of the connection pool. The JDBC URL configured with the JdbcLoginModule will contain this value (shown in Step 2)
5. The driver parameter must be JDBC Driver class used to connect with your SQL database. The one shown here is for Hypersonic SQL. If you're using MySQL, Oracle, MSSQL or some other database, substitute this parameter with the appropriate JDBC Driver class.
6. The url is the JDBC URL appropriate the driver configured in step 5.
7. The user and password are values appropriate for creating connections to your SQL database
8. initialCapacity is the number of Connections initially created for the pool when this service is started
9. maximumCapacity is the maximum number of simultaneous connections that can be managed by the pool
10. minimumCapacity is the minimum number of connections that should be managed by the pool
11. poolIncrement is the number of Connections the pool will be expanded (up to maxCapacity) if all Connections are in use
12. allowShrink is a boolean value that enables the pool to shrink back to the number of in–use Connections or minimumCapacity, which ever is greatest
13. shrinkPeriod is the number of minutes between pool shrink attempts
14. refreshPeriod is the number of minutes between attempts to verify that Connections are still open
15. poolUser and poolPassword are the credentials that must be provided by a JDBC client to the Cams pooling JDBC Driver to get a Connection.

Step 2 – Configure the JdbcLoginModule to use the Cams–provided pooling JDBC Driver (in login–config.xml)

```
<login-module-entry
  className="com.cafesoft.cams.auth.login.module.JdbcLoginModule"
  flag="REQUIRED">
  <options>
    <option name="debug" value="false"/>
    <option name="jdbcDriver" value="com.cafesoft.core.jdbc.PoolingConnectionDriver"/>
    <option name="jdbcUrl" value="jdbc:cafesoft:pool:myusersdb_pool"/>
  </options>
</login-module-entry>
```

Login Configuration

```
<option name="jdbcUsername" value="my_pool_user" />
<option name="jdbcPassword" value="my_pool_password" />
<option name="userPreparedStatement"
value="SELECT PASSWORD FROM USERS WHERE USERNAME = ? " />
<option name="passwordColumn" value="PASSWORD" />
<option name="rolePreparedStatement"
value="SELECT ROLE FROM USER_ROLES WHERE USERNAME = ? " />
<option name="roleColumn" value="ROLE" />
</options>
</login-module-entry>
<callback-handler
classname="com.cafesoft.cams.auth.callback.NamePasswordCallbackHandler" />
```

Example 4 – Using pooled JDBC Connections from JdbcLoginModule

1. The JdbcLoginModule's jdbcDriver parameter must be set to use the Cams PoolingConnectionDriver as shown
2. The jdbcUrl must start with "jdbc:cafesoft:pool:" and must end with the poolName configured in Step 1
3. The jdbcUsername and jdbcPassword must use the poolUser and poolPassword configured in Step 1

LDAP Login Module

Example 5 shows the sample entry for the LDAP login module. You use this login module to access user repository information that is stored in a LDAP directory such as iPlanet Directory Server, Novell eDirectory, OpenLDAP, and virtually any other LDAPv3 compliant directory. The sample entry specifies values that have been tested with iPlanet and OpenLDAP.

This login module requires a Cams service from which connections to an LDAP server can be obtained. By default a Cams service named "ldap-user-repository" is available for this purpose. This service improves authentication efficiency and performance by pooling and reusing LDAP connections. The actual LDAP server connection parameters are configured in security-domain.xml, where the "ldap-user-repository" service is registered. The LDAP login module configuration options include parameters used to flexibly query user and role information from most LDAP repository schemas.

```
<login-module-entry
classname="com.cafesoft.cams.auth.login.module.LdapLoginModule"
flag="REQUIRED">
<options>
<option name="debug" value="false" />
<option name="roleBase" value="ou=groups,dc=sample,dc=com" />
<option name="roleName" value="cn" />
<option name="roleSearch"
value="(uniqueMember=uid={username},ou=people,dc=sample,dc=com)" />
<option name="roleSubtree" value="false" />
<option name="userPattern" value="uid={username},ou=people,dc=sample,dc=com" />
<option name="serviceId" value="ldap-user-repository" />
</options>
</login-module-entry>
```

Example 5 – Cams LDAP login module login-config.xml sample

The required LDAP login module options values are:

- roleBase – The base distinguished name for your LDAP repository.
- roleName – The attribute name where the roles are stored.
- roleSearch – The pattern to match when searching for roles. The username entered by the the user at login will be substituted for the value "{username}".
- roleSubtree – Specifies if the role search will be done only at the level of the roleBase distinguished name (false), or if subtrees of of the roleBase distinguished name will also be searched (true).
- userPattern – The pattern to match when searching for users. The username entered by the the user at login will be substituted for the value "{username}".
- serviceId – The unique name of the Ldap Connection Pool Service that you are assigning to this login module.

Passwords can be encrypted using Unix Crypt, MD5, and SHA, SMD5, and SSHA where the algorithm is specified within curly brackets in a preceding label (e.g., {CRYPT}, {MD5}, {SHA}, {SMD5}, {SSHA}).

See [Security Domain Configuration – LDAP Connection Pool Service](#) section for information on how to configure services.

XML Login Module

Example 6 shows the sample entry for the XML login module. You use this login module to access user repository information that is stored in a [Cams users repository](#). This repository is supplied with Cams to ease deployment and configuration for evaluation and development purposes. In some case where there are a limited number of users and roles, it can be used in production environment. The XML login module does not process the contents of the XML–formatted Cams user repository directly. Instead, it makes use of a Cams service, which loads the XML file and reloads it if modifications are made. This Cams service (defined in security–domain.xml) caches user, password, and role information in memory, greatly improving the efficiency of authentication. The path to Cams user repository and the XML handler that knows how to parse the file are specified in the service manager configuration.

```
<login-module-entry
  className="com.cafesoft.cams.auth.login.module.XmlLoginModule"
  flag="REQUIRED">
  <options>
    <option name="debug" value="false"/>
    <option name="serviceId" value="cams-user-repository"/>
  </options>
</login-module-entry>
```

Example 6 – Cams XML login module login–config.xml sample

The required XML login module options values are:

- `serviceId` – The unique identifier of the Cams XML user repository service (configured in security–domain.xml) that you will use with the XML login module.

Passwords can be encrypted using Unix Crypt, MD5, and SHA, SMD5, and SSHA where the algorithm is specified within curly brackets in a preceding label (e.g., {CRYPT}, {MD5}, {SHA}, {SMD5}, {SSHA}).

See [Security Domain Configuration – Cams XML User Repository Service](#) for information on how to configure services.

Callback Handlers

Cams uses callback handlers to pass login credentials from the calling application to the login module. In the web or HTTP space, most webapps use form authentication, typically requesting a username and password for authentication. In this case you can use the Cams NamePasswordCallbackHandler, which takes a username and password as input and provides them to the login module. It is possible to create login modules that require many types of credentials. In some cases you may need to write and register your own callback handler.

To specify a callback handler for use with a Cams login module, each login–config–entry includes the following required tag:

```
<callback-handler
  classname="com.cafesoft.cams.auth.callback.NamePasswordCallbackHandler" />
```

The Cams JDBC, LDAP, and XML login modules all require the callback handler to supply a username and password. These login modules work well with the NamePasswordCallbackHandler. However, Cams has a pluggable callback framework to associate custom callback handlers with login modules. If the NamePasswordCallbackHandler does not meet your needs, see the [Cams Callback Handlers](#) chapter of the Cams Programmer's Guide to understand how to create a new one. For example, you may want to specify a third authentication credential in addition to username and password, such as the last four digits of a user's social security number or a pin.

Login Parameters

Login parameters are optional tags that further define the login configuration. As shown in Example 7, the login parameter name `camsLoginUrl` with its associated value informs the Cams server of the location of an http resource type login page.

```
<login-parameters>
  <login-parameter name="camsLoginUrl"
    value="{http.resource.base.id}/cams/login.jsp" />
</login-parameters>
```

Example 7 – Cams login paramaters

When a secure resource is requested by a non–authenticated user, Cams uses this value to redirect the user's browser to the appropriate login page.

[Back](#) | [Next](#) | [Contents](#)

Security Domain Configuration

The power of the Cams server is unleashed through high-level services that handle user and Cams agent requests. These services are flexibly configured in each security domain's XML-formatted security-domain.xml file. Table 1 summarizes each security domain service.

Service	Description
Authentication Service	Authenticates users according to the login configuration of the enclosing security domain. Authorized Cams agents submit authentication requests and receive authentication responses.
Access Control Service	Hosts the security domain-specific access control policy that controls access to the resources protected by a security domain. Authorized Cams agents submit access control requests and receive access control responses.
Session Manager Service	Manages sessions corresponding to authenticated users. Sessions keep track of who, when, where, and how users authenticated. Sessions may also contain application-specific data that can be accessed via the session access service.
Session Access Service	Provides the information available within an authenticated user's session to an authorized Cams agent. The session information can be used to customize application behaviour and/or user presentation. It can also be used to by access control rules within the Access Control Service to grant or deny access based on business rules.
Session Control Service	Enables authorized Cams agents to update, close, and expire sessions managed within a Cams server. A common use for this service is to logout a user by closing the associated session.
Service Manager Service	Manages a set of services conforming to the Cams service API, which are available to components within a security domain. Cams ships with some standard services used by login modules to share pooled LDAP connections and access users, passwords, and roles stored within an XML file. Using the Cams service API, you can create reusable services that maintains business state or assist other components hosted within a security domain.

Table 1 – Cams services

With a few exceptions, most of the default values supplied with Cams in security-domain.xml will work for you. This document will help you understand how to configure security-domain.xml. The most common configuration activities that will interest you within security-domain.xml include:

- Configuring security domain-wide variables
- Configuring a security domain's trace logger
- Configuring transaction loggers for each security domain service
- Configuring custom services managed by a security domain's service manager

Cams Variables

The following global variables are provided by the Cams server and are always available in every security domain:

- `cams.home` – The fully qualified path to the Cams server installation directory (set based on an environment variable defined within "runcams.bat" or "runcams.sh"). This value is inherited from the Cams server environment and cannot be modified within a security domain.
- `cams.server.name` – The name assigned to the Cams server installation within configuration file: `${cams.home}/conf/cams.conf`. This value is used primarily within the system security domain, which control access to services used by Cams agents and cannot be modified.
- `cams.server.port` – The TCP/IP port number on which the Cams server listens for connection by agents as configured in `${cams.home}/conf/cams.conf`. This value is used primarily within the system security domain, which control access to services used by Cams agents and cannot be modified.
- `cams.security-domain.name` – The security domain-specific name (set based on the `<name>` element of a `<security-domain>` registration), which is inherited from the security domain registry and cannot be modified.
- `cams.security-domain.home` – The security domain-specific home directory (set based on the `<home>` element of a `<security-domain>` registration), which is inherited from the security domain registry and cannot be modified.

Security domains can also define their own local variables, which are inherited by all XML configuration files within the security domain. One such variable is defined within the security-domain.xml files shipped with Cams as shown in Example 1.

```
<security-domain>
  ...
  <var-list>
    <var name="logPrefix"
        value="{cams.logs.base.dir}/{cams.security-domain.name}"/>
  </var-list>
  ...
</security-domain>
```

Example 1– The Cams *logPrefix* variable defined within each Cams *security-domain.xml* file

The *logPrefix* variable is used to centralize the location of Cams log files while giving them security domain–specific names. This variable is substituted throughout the remainder of *security-domain.xml*, where various log files are configured.

Cams Logger Types

Cams has two basic types of loggers: trace loggers and transaction loggers.

A trace logger writes information about the runtime state of Cams services and includes five different levels of messages: DEBUG, INFO, WARNING, ERROR, and FATAL which are summarized in Table 2. It enables you to follow (trace) requests made to a Cams server through the services that handle it. In the case of serious errors, it will log a stack trace, which shows the software context of the error. This information can be important to debugging problems, especially for Cafésoft support representatives. All software components hosted within a security domain log to the same trace logger. The Cams server also has its own trace logger, which contains information from services that are higher–level than Cams security domains. See [Configuring Cams Server](#) for more details on the server–level transaction logger.

Transaction loggers are used by services hosted within a security domain to record transaction–specific requests and responses. For example, a security domain's authentication service records authentication requests and responses, which is quite different from the information logged by an access control service. The format of each transaction log type is detailed within its corresponding service section below.

Message Level	Description
DEBUG	DEBUG–level are intended primarily for administrators and developers that need to see low–level details while Cams services execute and components are invoked. During normal Cams Server production deployments, DEBUG–level messages are disabled.
INFO	INFO–level messages are used for higher–level messages that help an administrator confirm that Cams services are running as expected. When Cams services start, they log a few INFO–level messages to show the startup status.
WARNING	WARNING–level messages are used to indicate a condition that should not have occurred and may need attention, but are not crucial to the continued, normal operation of Cams.
ERROR	ERROR–level messages indicate a condition that was not expected and threatens the continued operation of Cams. The corresponding condition should be investigated and fixed.
FATAL	FATAL–level messages indicate the most serious of error conditions in which Cams cannot start or cannot continue running. FATAL conditions must be fixed as Cams will either not startup or will shutdown as gracefully as possible.

Table 2 – A summary of Cams message levels

Configuring the Trace Logger

Each Cams security domain configures its own centralized trace logger, which logs information about the startup, shutdown, warnings, and errors of its services. If a Cams security domain is misconfigured, fails to load, detects a runtime error, or experiences any other anomaly, a message will be written by this logger. Example 2 shows the XML tag that configures a security domain's trace logger.

```
<security-domain>
  ...
  <logger
    className="com.cafesoft.cams.log.CamsTraceLogger"
    filePath="{logPrefix}-trace.log"/>
  ...
</security-domain>
```

Example 2 – The Cams security domain trace–logger configuration

Table 3 shows properties that can be used to customize a trace logger's behaviour.

Property	Req/Opt	Description
className	Req	The fully qualified name of the Logger class that will be instantiated.
filePath	Req	The fully qualified log file path.
append	Opt	If true new log messages will be appended to the current log file. If false the current log file will be deleted and a new log file will be created. The default value is true.
bufferedIO	Opt	If true the logger will buffer log messages before writing them to the log file. This can significantly improve logging performance. The default value is true.
bufferSize	Opt	Indicates the size of the buffer to fill before writing to the log file. The default value is "4096".
maxSize	Opt	The maximum size the logfile is allowed to grow before creating a new logfile. Suffixes KB, MB, and GB are recognized. When log files are rolled over, the file name is appended with an numeric digit: 1, 2, 3, etc. The default maxSize value is "4MB".
enableConsole	Opt	If true all log statements that are sent to the log file are also sent to the console. The default value is false.
enableDebugFilter	Opt	If true all log statements that have the level "DEBUG" will not be logged. The default value is false.
verbose	Opt	<p>If true all DEBUG, INFO, WARN, ERROR, and FATAL messages logged will contain the following format:</p> <pre>[INFO] Sample log message Class Name: com.cafesoft.cams.log.CamsTraceLogger Method Name: info() Line Number: 121 Timestamp: 25 Jul 2002 11:02:36,339</pre> <p>If false ONLY WARNING, ERROR, and FATAL message level will use the verbose format, while DEBUG and INFO level messages will use the following format:</p> <pre>[INFO] Sample Log Message</pre> <p>The default value is false.</p>
debug	Opt	If true the logger will output diagnostic debug statements to the System.err stream. The default value is false.

Table 3 – Properties for configuring a Cams trace logger

Configuring the Authentication Service

The Authentication Service is responsible for validating the identity of a user who accesses protected resources within the security domain.

Login Config Factory

Loads and initializes the security domain's login configuration.

Authentication Pipeline

The authentication pipeline processes authentication requests issued locally or remotely by a Cams agent. This pipeline implements a chain of responsibility pattern that provides strong, configureable control over who can issue authentication requests, and how the response to authentication requests is created.

Authentication Valve

The authentication valve represents a single node within an authentication pipeline. The valve receives an authentication request and can handle the authentication completely, modify or add to the authentication request, or pass the authentication request to the next authentication valve in the chain.

Authentication Log

The Cams authentication log file contains information about successful and unsuccessful authentication requests within a security domain. Example 3 shows how an authentication valve is configured for logging authentication transactions.

```
<auth-valve
  className="com.cafesoft.security.engine.auth.valves.LogAuthRequestValve">
  <param-list>
    <param name="logPath" value="{logPrefix}-authentication.log"/>
    <param name="maxSize" value="10MB"/>
    <param name="logOnAuthFailure" value="http=username"/>
  </param-list>
  ...
</auth-valve>
```

Example 3 – The Cams security domain authentication transaction logger

Table 4 shows the configuration parameters that can be used to customize this logger.

Field	Req/Opt	Description
logPath	Req	The path to the authentication transaction log file for the enclosing security domain.
append	Opt	If true, then append new records to an existing log file. If false, then overwrite an existing transaction log when the security domain is started. The default value is "true".
maxSize	Opt	The maximum size the logfile is allowed to grow before creating a new logfile. Suffixes KB, MB, and GB are recognized. When log files are rolled over, the log file name is appended with a numeric digit. The default value is "4MB".
logOnAuthFailure	Opt	A mapping of login-config-entry names to callback value names. On failed authentication, the callback value corresponding to the login-config-entry and callback name is logged. This feature can be used to log the "username" (if supplied) associated with the failed login. The actual callback value name can depend on the login-config-entry and login module(s) configured for that entry. This value is optional, but if not supplied a WARNING is logged to the security domain trace log. The following example shows how security domains containing multiple login-config-entry specifications can be configured: <param name="logOnAuthFailure" value="http=username,ws=x509"/>

Table 4 – Properties for configuring a Cams authentication transaction logger

The authentication transaction log is a comma-delimited text file. Each line corresponds to an authentication request/response pair. Table 5 shows the fields (left to right) defined within this file. If an optional field does not apply, its value is written as "-" (a dash character).

Field	Req/Opt	Description
remote-address	Req	The IP address of the client.
remote-hostname	Req	The value of this element may be the hostname of the client if (its address has been resolved by DNS) or the remote IP address of the client (if not resolved by DNS).
date-time	Req	The local date/time of the authentication attempt at the Cams server. Example format: [10/Oct/2000:13:55:36 -0700]. The last value "-0700" represents the offset from universal time (GMT), which enables time comparisons based on absolute time.
app-name	Opt	The name of the application attempting authentication. (This must match a login configuration entry within the security domain's login-config.xml). If this value is not specified, then the default application name "other" is used.
status	Req	The status code of the authentication response: 1 = success, 2 = failed
reason	Opt	If status=2 (authentication failed), a code indicating the reason authentication failed: 1 = general server error 2 = invalid remote IP address 3 = invalid remote host name 4 = unauthorized agent 5 = unknown security domain 6 = unknown login configuration

Security Domain Configuration

		7 = incomplete or invalid authentication credentials (normal login failure) 8 = unrecoverable error during authentication (runtime exception) 12 = unrecoverable error within a callback handler (runtime exception)
session-id	Opt	If status = 1 (authentication succeeded), the ID of the authenticated subject's session.
subject	Opt	If status = 1 (authentication succeeded), the name of the authenticated Subject.

Table 5 – The Cams authentication transaction log fields

Configuring the Access Control Service

The access control service defines the Java classes that control access to the resources protected by a security domain.

Access Control Policy Factory

The access control policy factory specifies the Java class that creates the access control policy that declares the resources protected within a security domain along with the rules for accessing them. Loads and initializes the access control policy. The factory will usually be specific to the persistence format for the configured access control policy. For example, the access control policy might be stored in an XML file, a relational database, an LDAP server, or some other data storage facility.

Access Request Pipeline

The access control pipeline specifies the Java class that processes access requests issued locally or remotely by a Cams agent. This pipeline is composed of a sequence of access control valves, which handle the request using the chain of responsibility design pattern. This enables each access control valve to handle the request altogether or modulate the request for processing by a subsequent valve.

Access Control Value

The access control valve represents a single node within an access control pipeline for handling access requests. The valve receives an access request and can handle the request completely, modify or add to it, or pass the request to the next valve in the chain.

Access Control Log

The Cams access control log file contains information about access control requests made by remote agents on behalf a client wishing to access resources protected by a security domain. Example 4 shows how an access control valve is configured for logging access control transactions.

```
<access-control-valve
  className="com.cafesoft.security.engine.access.valves.LogAccessControlRequestValve">
  <param-list>
    <param name="logPath" value="{logPrefix}-access-control.log" />
  </param-list>
  ...
</access-control-valve>
```

Example 4 – The Cams security domain access control transaction logger

Table 6 shows the configuration parameters that can be used to customize this logger. Additional name=value pairs can be appended to the value of the params attribute (like this: params="logPath={cams.home}/logs/system-authentication.log,append=true").

Field	Req/Opt	Description
logPath	Req	The path to the access control transaction log file for the enclosing security domain.
append	Opt	If true, then append new records to an existing log file. If false, then overwrite an existing transaction log when the security domain is started. The default value is "true".
maxSize	Opt	The maximum size the logfile is allowed to grow before creating a new logfile. Suffixes KB, MB, and GB are recognized. When log files are rolled over, the log file name is appended with a numeric digit. The default value is "4MB".

Table 6 – Properties for configuring the Cams access control transaction logger

The access control transaction log is a comma-delimited text file. Each line corresponds to an access control request/response pair. Table 7 shows the fields (left to right) defined within this file. If an optional field does not apply, its value is written as "-" (a dash character).

Security Domain Configuration

Field	Req/Opt	Description
remote-address	Req	The IP address of the client requesting access to the resource.
remote-hostname	Req	The value of this element may be the name of the host requesting access to the resource (if its address has been resolved by DNS) or the remote IP address of the host (if not resolved by DNS).
date-time	Req	The local date/time of the session access event at the Cams server. Example format: [10/Oct/2000:13:55:36 -0700] . The last value "-0700" represents the offset from universal time (GMT), which enables time comparisons based on absolute time.
session-id	Opt	If an authenticated user session is established, the ID of the session being asserted by the remote agent.
subject	Opt	If a session is being asserted, the name of the Subject being asserted by the remote agent.
resource-type	Req	The type of the resource the client is attempting to access.
resource-id	Req	The identifier of the resource.
resource-actions	Req	The action(s) being requested on the resource. If multiple actions, they are separated by the "and" character (&).
status	Req	The status code of the access response: 1 = success, 2 = failed
reason	Opt	<p>A code indicating the reason access failed/succeeded:</p> <ul style="list-style-type: none"> • 0 = not applicable • 1 = general server error • 2 = invalid remote IP address • 3 = invalid remote host name • 4 = unauthorized agent • 5 = unknown security domain • 6 = unknown resource type • 7 = invalid resource identifier • 8 = unknown resource action • 9 = access denied (unconditionally) • 10 = access denied (authentication required) • 11 = access denied (session expired) • 12 = access denied (access control rule evaluation error) • 13 = access denied (confidentiality is required) • 14 = access denied (session id invalid) • 15 = default bias applied • 16 = access denied (unknown login config entry) • 17 = access denied (general transport error) • 18 = access granted (conditionally) • 19 = access granted (unconditionally) • 20 = access denied (conditionally)

Table 7 – The Cams access control transaction log fields

Configuring the Session Manager Service

The session manager service manages authenticated user sessions. A session is created on successful user authentication, may be closed explicitly by authorized Cams agents, and expires if inactive for a configurable period. The session manager transaction log records session creation, update, and closure events. The session manager also supports registration of application-specific session event handlers, that can be use to customize the contents of sessions or trigger business processes based on session events.

Example 5 shows the default configuration of the the Cams session manager service.

```
<session-manager-service
  className="com.cafesoft.security.engine.session.StandardSessionManager">
  <param-list>
    <param name="maxActiveSessions" value="-1"/>
    <param name="inactiveSessionTimeout" value="30"/>
    <param name="sessionCleanupInterval" value="1"/>
    <param name="sessionIdIPAddrValidationMask" value="255.255.255.255"/>
    <param name="sessionIdAlgorithm" value="SHA"/>
    <param name="sessionIdKey" value="secret-key"/>
  </param-list>
</session-manager-service>
```

Security Domain Configuration

```

</param-list>
...
</session-manager-service>

```

Example 5 – The Cams session manager transaction logger

Table 8 shows the configuration parameters that can be used to configure the session manager service.

Field	Req/Opt	Description
maxActiveSessions	Opt	The maximum number of concurrent active sessions allowed for this SecurityDomain. To specify no maximum use the value -1. The default value is -1.
inactiveSessionTimeout	Opt	The maximum interval (in minutes) an active session is allowed to be inactive before it is expired and removed from the session manager. Value must be greater than 0. The default value is 30 minutes.
sessionCleanupInterval	Opt	The interval (in minutes) the session manager checks and removes expired sessions. Value must be greater than 0. The default value is 1 minute.
sessionIdAlgorithm	Opt	The message digest algorithm to be used for encrypting the session identifier. Valid values include: SHA or MD5. If not specified, SHA is used.
sessionIdIPAddrValidationMask	Opt	<p>A bit mask used to detect possible hijacked Cams session identifiers by validating the associated IP address.</p> <p>When the session is created (at authentication time) the IP address of the remote client is associated with the session. For every subsequent access control, session access, and session control request, the IP address associated with the request is validated against the original authentication IP address using the mask to indicate which bits to compare. For example, a mask value of: 255.255.255.255 indicates that the entire IP address must match. A value of 255.255.255.0 validates the first three triplets of the IP address.</p> <p>When supporting web clients accessing resources via the general internet, it may be necessary to loosen IP address validation to support proxy servers or gateway routers that cause subsequent HTTP requests to arrive via different client IP addresses. For example, some commercial ISPs will route HTTP traffic via one network and HTTPS traffic via another causing subsequent requests arriving at a Cams web agent from the same web browser have different remote client IP addresses.</p> <p>The default value is: 255.255.255.255, the most restrictive IP address validation.</p>
sessionIdKey	Req	The key used to encrypt session id's. The default value is "secret-key". It is highly recommended this value be changed to unique value.

Table 8 – Properties for configuring the Cams access control transaction logger

Session Event Handler

Registers a session event handler with the session manager to handle session events.

Session Manager Log

The Cams session manager log file contains information about the sessions created, closed, and expired. (A session is associated with an authenticated subject within a Cams security domain). Example 6 shows the standard Cams session event handler registered to log session events.

```

<session-event-handler
  className="com.cafesoft.security.engine.session.SessionManagerEventLogger">
  <param-list>
    <param name="logPath" value="{logPrefix}-session-manager.log" />
  </param-list>
  ...
</session-event-handler>

```

Security Domain Configuration

Example 6– The Cams session manager transaction logger

Table 9 shows the configuration parameters that can be used to customize this logger.

Field	Req/Opt	Description
logPath	Req	The path to the session manager transaction log file for the enclosing security domain.
append	Opt	If true, then append new records to an existing log file. If false, then overwrite an existing transaction log when the security domain is started. The default value is "true".
maxSize	Opt	The maximum size the logfile is allowed to grow before creating a new logfile. Suffixes KB, MB, and GB are recognized. When log files are rolled over, the log file name is appended with a numeric digit. The default value is "4MB".

Table 9 – Properties for defining a Cams session manager transaction logger

Table 10 shows the fields defined within this file.

Field	Req/Opt	Description
session-id	Req	The ID of the authenticated subject's session.
subject	Req	The name of the Subject associated with the session.
remote-address	Req	The IP address of the client associated with the session.
remote-hostname	Req	The value of this element may be the hostname of the client if (it's address has been resolved by DNS) or the remote IP address of the client (if not resolved by DNS).
date-time	Req	The local date/time of the session management event at the Cams server. Example format: [10/Oct/2000:13:55:36 -0700] . The last value "-0700" represents the offset from universal time (GMT), which enables time comparisons based on absolute time.
authentication-time	Req	The local date/time at which authentication occurred (which may be slightly different than the time at which the session management event is logged). Example format: [10/Oct/2000:13:55:36 -0700] . The last value "-0700" represents the offset from universal time (GMT), which enables time comparisons based on absolute time.
event-type	Req	The type of event associated with session. Values may include: <ul style="list-style-type: none"> • CREATED – a new session was created due to successful authentication • CLOSED – an existing session was closed due to explicit logout • EXPIRED – an existing session expired for lack of activity
auth-methods	Opt	If the event type is "CREATED", then a list of authentication methods separated by the "and" character (&). Values may include: <ul style="list-style-type: none"> • urn:oasis:names:tc:SAML:1.0:am:password – (password) • urn:ietf:rfc:1510 – (kerberos) • urn:ietf:rfc:2945 – (secure remote password) • urn:oasis:names:tc:SAML:1.0:am:HardwareToken – (hardware token) • urn:ietf:rfc:2246 – (SSL/TLS Certificate-based client authentication) • urn:oasis:names:tc:SAML:1.0:am:X509-PKI – (X.509 Public Key) • urn:oasis:names:tc:SAML:1.0:am:PGP – (PGP Public Key) • urn:oasis:names:tc:SAML:1.0:am:SPKI – (SPKI Public Key) • urn:oasis:names:tc:SAML:1.0:am:XKMS – (XKMS Public Key) • urn:ietf:rfc:3075 – (XML Digital Signature)

Table 10 – The Cams session manager log fields

Configuring the Session Access Service

The Cams session access service enables session information to be queried by qualified Cams agents.

Session Access Pipeline

The session access pipeline processes session access requests by a Cams agent. This pipeline is composed of a sequence of session access valves, which handle the request using the chain of responsibility design pattern. This enables each session access valve to handle the

request altogether or modulate the request for processing by a subsequent valve.

Session Access Valve

The session access valve represents a single node within a session access pipeline for handling access requests. The valve receives a session access request and can handle the request completely, modify or add to it, or pass the request to the next valve in the chain.

Session Access Log

The Cams session access log file contains information about requests for remote access to an authenticated users session. Example 7 shows how a session-access-valve is configured for logging session access transactions.

```
<session-access-valve
  className="com.cafesoft.security.engine.session.access.valves.LogSessionAccessRequestValve">
  <param-list>
    <param name="logPath" value="\${logPrefix}-session-access.log" />
  </param-list>
  ...
</session-access-valve>
```

Example 7 – The Cams security domain session access transaction logger

Table 11 shows the configuration parameters that can be used to customize this logger.

Field	Req/Opt	Description
logPath	Req	The path to the session access transaction log file for the enclosing security domain.
append	Opt	If true, then append new records to an existing log file. If false, then overwrite an existing transaction log when the security domain is started.
maxSize	Opt	The maximum size the logfile is allowed to grow before creating a new logfile. Suffixes KB, MB, and GB are recognized. When log files are rolled over, the log file name is appended with an numeric digit. The default value is "4MB".

Table 11 – Properties for configuring the session access transaction logger

Table 12 shows the fields defined within this file.

Field	Req/Opt	Description
session-id	Req	The ID of the session being asserted by the remote host.
subject	Req	The name of the subject being asserted by the remote host.
remote-address	Req	The IP address of the host requesting access to the session. NOTE: This is NOT the IP address of the client associated with the session, rather, the host on which the browser is running.
remote-hostname	Req	The value of this element may be the name of the host requesting access to the session (if it's address has been resolved by DNS) or the remote IP address of the host (if not resolved by DNS).
date-time	Req	The local date/time of the session access event at the Cams server. Example format: [10/Oct/2000:13:55:36 -0700] . The last value "-0700" represents the offset from universal time (GMT), which enables time comparisons based on absolute time.
status	Req	The status code of the session access response: 1 = success, 2 = failed
reason	Opt	If status = 2 (session access failed), a code indicating the reason session access failed: <ul style="list-style-type: none"> • 0 = not applicable • 1 = general server error • 2 = invalid remote IP address • 3 = invalid remote host name • 4 = unauthorized agent • 5 = unknown security domain • 6 = unknown session • 7 = unknown subject

Table 12 – The Cams session access log fields

Configuring the Session Control Service

The Cams session control service enables sessions to be updated, closed, expired, etc. by qualified Cams agents.

Session Control Pipeline

The session control pipeline processes session control requests by a Cams agent. Existing sessions can be updated, closed, or expired. This pipeline is composed of a sequence of session control valves, which handle requests using the chain of responsibility design pattern. This enables each session control valve to handle the request altogether, modify the request for processing by a subsequent valve, or ignore it completely.

Session Control Valve

Represents a single node within a session control pipeline for handling session control requests. The valve receives a session control request and can handle the request completely, modify or add to it, or pass the request to the next valve in the chain.

Session Control Log

The Cams session control log file contains information about requests to update, close or expire sessions managed by the security domain's session manager service. A common use for these transactions is to logout an authenticated user by closing the associated session. Example 8 shows how a session-control-valve is configured for logging session-control transactions.

```
<session-control-valve
  className="com.cafesoft.security.engine.session.access.valves.LogSessionControlRequestValve">
  <param-list>
    <param> name="logPath" value="\${logPrefix}-session-control.log"/>
  </param-list>
  ...
</session-control-valve>
```

Example 8 – The Cams security domain session control transaction logger

Table 13 shows the configuration parameters that can be used to customize this logger.

Field	Req/Opt	Description
logPath	Req	The path to the session control transaction log file for the enclosing security domain.
append	Opt	If true, then append new records to an existing log file. If false, then overwrite an existing transaction log when the security domain is started. The default value is "true".
maxSize	Opt	The maximum size the logfile is allowed to grow before creating a new logfile. Suffixes KB, MB, and GB are recognized. When log files are rolled over, the log file name is appended with an numeric digit. The default value is "4MB".

Table 13 – Properties for configuring the Cams session control transaction logger

Table 14 shows the fields defined within this file.

Field	Req/Opt	Description
date-time	Req	The local date/time of the session control request at the Cams server. Example format: [10/Oct/2000:13:55:36 -0700] . The last value "-0700" represents the offset from universal time (GMT), which enables time comparisons based on absolute time.
security-domain-name	Req	The name of the security domain name asserted by the remote host.
subject	Req	The name of the Subject being asserted by the remote host.
action	Req	The requested action, which may be: "CLOSE" or "EXPIRE".
session-id	Req	The ID of the session being asserted by the remote host.
status	Req	The status code of the session control response: 1 = success, 2 = failed
reason	Opt	If status = 2 (session access failed), a code indicating the reason session control failed:

Security Domain Configuration

		<ul style="list-style-type: none">• 0 = not applicable• 1 = general server error• 2 = unknown security domain• 3 = invalid session id• 4 = invalid subject• 5 = poisoned session• 6 = session does not exist• 7 = null session id
--	--	--

Table 14 – The Cams session control log fields

Configuring Service Manager Services

Cams includes predefined services that you configure for use with a security domain. Programmers can also use the Cams Services API to create new services for use by other custom components, like session event handlers, access control rules, etc. This section provides information on how to configure the standard services that ship with Cams and those that might be created by your development staff. See the [Cams Programmer's Guide](#) for more details on creating custom Cams services.

By default, the following services are available to each Cams security domain:

- LDAP Connection Pool Service – Provides connection pooling for the LDAP login module (required to use the LDAP login module)
- Cams XML User Repository Service – Loads and manages the Cams XML user repository file (required to use the XML login module)

You can have any number of services, each nested within the security-domain.xml <service-manager> tag, which has three attributes:

```
className="com.cafesoft.core.service.StandardServiceManager"  
params=""  
debug="false"
```

You should not need to change the Java class value. However, if you are having difficulty configuring the service, you might set the debug value to true to provide for more verbose log output. There are no parameters.

Services

The <service> tag starts the registration of a Cams service within the scope of the enclosing security domain. It has the following attributes:

- id – The unique textual identifier or name by which this service will be referenced.
- enabled – A flag to specify if a service is enabled
- debug – If set to true, turns debug on for this service

In addition, a service has three required elements:

- service-type – Declares the type of service being registered. This value is the fully qualified name of the Java interface class implemented by the service. (See examples 7 and 8 to see how this attribute is used)
- service-class – Declares the Java class that implements the Cams service. This value is the fully qualified Java class name which implements the service type)
- param-list – A container for a list of service-specific configuration parameters

For example, suppose a developer at your site has created a custom Cams service called EmailNotifier that can send a short textual messages to an e-mail address. Perhaps the service will be used by a session event handler to notify administrators whenever somebody with the admin role successfully authenticates. The same service might be used to notify a customer at logout time of their account status.

Suppose the programmer created the Cams service by writing two classes:

1. com.myco.message.TextNotifier (a Java interface that extends com.cafesoft.core.service.Service)
2. com.myco.message.EmailTextNotifier (a Java class that implements interface com.myco.message.TextNotifier and uses SMTP to send a brief text message to an e-mail address)

Example 9 shows how you might register and configure this service for use by other custom components (or other services) within a security domain. This service can be looked up by it's identifier: "email-text-notifier-service" or by it's type: com.myco.message.TextNotifier.

Security Domain Configuration

```
<service id="email-text-notifier-service" enabled="true">
  <service-type>com.myco.message.TextNotifier</service-type>
  <service-class>com.myco.message.EmailTextNotifier</service-class>
  <param-list>
    <param name="smtp.host" value="mail.myco.com">
    <param name="smtp.port" value="25">
    <param name="smtp.from" value="cams@myco.com">
  </param-list>
</service>
```

Example 9 – An example Cams service manager service registration

If need arises for message to be sent to a pager, Java class: `com.myco.message.PagerTextNotifier` could be written and the Cams service registration shown in Example 10 could be added to make both e-mail and pager text notification available to components.

```
<service id="pager-text-notifier-service" enabled="true">
  <service-type>com.myco.message.TextNotifier</service-type>
  <service-class>com.myco.message.PagerTextNotifier</service-class>
  <param-list>
    <param name="phone.from" value="800-555-9191">
  </param-list>
</service>
```

Example 10 – Another example Cams service manager service registration

The next sections describes the required configuration values for each standard service supplied with Cams.

LDAP Connection Pool Service

The LDAP connection pool service is intended for use with the Cams [LDAP Login Module](#). The pool optimizes performance by sharing and reusing connections to an LDAP repository. Example 11 shows the configuration values. You'll need to change most of the parameters in the `<param-list>` to values specific for your site.

```
<service id="ldap-user-repository" enabled="true">
  <service-type>com.cafesoft.cams.service.LdapConnectionPoolService</service-type>
  <service-class>com.cafesoft.security.engine.service.LdapConnectionPoolServiceImpl</service-class>
  <param-list>
    <param name="maxConnections" value="10">
    <param name="minConnections" value="2">
    <param name="maxInactiveTime" value="0">
    <param name="hostname" value="host.sample.com">
    <param name="port" value="389">
    <param name="version" value="3">
    <param name="username" value="cn=username,dc=sample,dc=com">
    <param name="password" value="password">
  </param-list>
</service>
```

Example 11 – Cams LDAP connection pool configuration

The `<service>` tag parameter values are:

- `id` – The identity of this service for this security domain. WARNING: The id must match the corresponding `serviceId` value in the LDAP login module configuration file.
- `enable` – Set to true if this service is enabled, false if it is not.

The values `<service-type>` and `<service-class>` should not change unless you have created a custom LDAP service.

The `<param-list>` parameters are:

- `maxConnections` – The maximum number of LDAP repository connections to allow
- `minConnections` – The minimum number of LDAP repository connections to maintain active
- `maxInactiveTime` – The maximum number of seconds that inactive connections live before being closed
- `hostname` – The LDAP server hostname (IP address or fully qualified DNS name)
- `port` – The LDAP repository port
- `username` – The username for LDAP repository (this must be a user account with LDAP repository permissions to lookup users, compare password attributes, fetch group memberships, etc.)
- `password` – The password for the username

- version – The version of LDAP on the server (usually 3)

When you start Cams, this service will attempt to make a connection to the LDAP repository.

Cams XML User Repository Service

The Cams XML user repository service is intended for use with the [XML login module](#). This service loads a [Cams user repository](#) file into memory, and provides methods to lookup user passwords and roles. The service also detects if the Cams user repository file is changed, and reloads if necessary. Example 12 shows the configuration values. Except for the repositoryFilePath, the default values should not change for your site.

```
<Service id="cams-user-repository" enabled="true">
  <service-type>com.cafesoft.cams.service.UserRepositoryService</service-type>
  <service-class>com.cafesoft.security.engine.service.UserRepositoryServiceImpl</service-class>
  <param-list>
    <param name="repositoryFilePath" value="{cams.home}/conf/domains/system/cams-users.xml"/>
    <param name="repositoryFactoryClass"
      value="com.cafesoft.security.engine.auth.login.userrepository.XmlUserRepositoryFactory"/>
    <param name="handlerClass"
      value="com.cafesoft.security.engine.auth.login.userrepository.CamsXmlUserRepositoryHandler"/>
  </param-list>
</service>
```

Example 12 – Cams XML user repository configuration

The <service> tag parameter values are:

- id – The identity of this service for this security domain. **WARNING:** The id must match the corresponding serviceId value in the XmlLoginModule configuration file.
- enable – Set to true if this service is enabled, false if it is not.

The values <service-type> and <service-class> should not change unless you create a custom XML service.

The <param-list> parameters are:

- repositoryFilePath – The fully qualified file path to the user repository file
- handlerClass – The fully qualified class name of the Java class that parses the XML user repository file
- repositoryFactoryClass – The fully qualified Java class name of the user repository factory

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Configuring Cams Server

Cams is highly configurable by editing the `CAMS_HOME/conf/cams.conf` file. Many configuration options are provided, and, you are also empowered to extend some behaviors by implementing custom code. This chapter provides instructions on how to configure existing options and extend others.

Cams Server Name

The Cams server name is defined by `cams.conf` property:

- `cams.server.name=MyCamsServer`

This value is used for logging and to distinguish Cams server installations and their corresponding user sessions from each other. It does not need to correspond to the host's DNS name. You should change its value to something unique for your company or department, etc. For example: `myhost_domain`. Alphanumeric and underscore ("_") characters are valid.

Connections

The Cams server listens for agent connections on a specified TCP/IP port specified by property:

- `cams.server.port=9191`

The default server port is 9191, but you may also configure Cams to use any other port. Cams also accepts SSL connections when used in conjunction with some agents. The default port for SSL connections to the Cams server is 9393. See [Securing Cams Network Connections with SSL](#) for more details.

NOTE: If a Cams agent connects with the Cams server through your firewall, you must ensure that the port you use for the Cams connection is allowed to pass through your firewall.

To limit the number of agents that can connect simultaneously, use property:

- `cams.server.maxConnections=200`

The default maximum is 200. Remember that each agent may have many connections. For example, the Cams Apache web agent uses one connection for each Apache process.

Cams also provides a shutdown service on a different port. This is for security purposes to allow you to further refine access to only allow connections to the Cams shutdown service from administrative systems. Furthermore, a password you specify must be supplied by the client to activate shutdown. The properties for the Cams shutdown service are:

- `cams.server.shutdown.port=9292`
- `cams.server.shutdown.password=theEndIsNear`

The default shutdown port is 9292. The `shutdown.bat` (Windows) and `shutdown.sh` (Unix) scripts execute a client program that is setup to use these properties from `${cams.home}/cams.conf`. So, if these scripts are executed from the system that started the Cams server, the shutdown program will automatically connect to the correct Cams shutdown port and provide the right password.

NOTE: You [should change this password](#) for your site.

Email Notification

Cams server may occasionally use SMTP-based email messages to notify administrators of various configuration or administrative issues. The properties used to configure this service are:

- `cams.server.smtp.host=mail.mycompany.com`
- `cams.server.smtp.from=cams-server@mycompany.com`
- `cams.server.smtp.to=admin@mycompany.com`

These values must be updated for compatibility with your installation environment. The value of `cams.server.smtp.host` must be the hostname or IP address of the mail exchange server registered for your Internet domain. The value of `cams.server.smtp.from` must be an email identity permitted to send messages to the configured SMTP server. In general, an email alias or fictitious name associated with the Cams server should be used. The value of `cams.server.smtp.to` is the address of the administrator to which messages are sent. If this value is omitted, the value registered in your Cams product license contact attribute is used.

NOTE: Don't hand–edit your Cams product license to change the administrators email address as this will corrupt the license–keys file. Instead, set `cams.server.smtp.to` to the desired email address.

Secret Key Encryption

Default Cams server connections use selective encryption of data transported between agents and the Cams server. This is facilitated by use of public encryption algorithms like Blowfish, DES, and DESede (triple DES) along with secret key parameters configured by the Cams server and participating Cams agents. You use the following properties to set the options:

- `cams.skey.algorithm=Blowfish`
- `cams.skey.key=ed28f2c7b60e978277d125d774bd25c1cad3c5c1a7f02757`
- `cams.skey.iv=1a5dce1235fd429e`

Detailed information on configuring the values for these options is provided in [Securing Cams Communications using Secret Keys](#).

Logger

Cams configures it's own centralized trace logger, which logs information about the startup, shutdown, warnings, and errors of it's services. If Cams is not configured correctly, fails to load, detects a runtime error, or experiences any other anomaly, a message will be written by this logger.

Table 1 shows properties that can be used to customize a trace logger's behavior.

Property	Req/Opt	Description
<code>logger.class</code>	Req	The fully qualified name of the Logger class that will be instantiated.
<code>logger.file.path</code>	Req	The fully qualified log file path.
<code>logger.file.append</code>	Opt	If true new log messages will be appended to the current log file. If false the current log file will be deleted and a new log file will be created. The default value is true.
<code>logger.file.bufferedIO</code>	Opt	If true the logger will buffer log messages before writing them to the log file. This can significantly improve logging performance. The default value is true.
<code>logger.file.bufferSize</code>	Opt	Indicates the size of the buffer to fill before writing to the log file. The default value is "4096".
<code>logger.file.maxSize</code>	Opt	The maximum size the logfile is allowed to grow before creating a new logfile. Suffixes KB, MB, and GB are recognized. When log files are rolled over, the file name is appended with an numeric digit: 1, 2, 3, etc. The default maxSize value is "4MB".
<code>logger.file.maxBackupIndex</code>	Opt	The maximum rollover file index. When log files are rolled over, a numeric index is appended to the name, starting with 1 and proceeding to this value. The default maxBackupIndex is 100.
<code>logger.enableConsole</code>	Opt	If true all log statements that are sent to the log file are also sent to the console. The default value is false.
<code>logger.enableDebugFilter</code>	Opt	If true all log statements that have the level "DEBUG" will not be logged. The default value is false.
<code>logger.verbose</code>	Opt	<p>If true all DEBUG, INFO, WARN, ERROR, and FATAL messages logged will contain the following format:</p> <pre>[INFO] Sample log message Class Name: com.cafesoft.cams.log.CamsTraceLogger Method Name: info() Line Number: 121 Timestamp: 25 Jul 2002 11:02:36,339</pre> <p>If false ONLY WARNING, ERROR, and FATAL message level will use the verbose format, while DEBUG and INFO level messages will use the following format:</p> <pre>[INFO] Sample Log Message</pre> <p>The default value is false.</p>

logger.debug	Opt	If true the logger will output diagnostic debug statements to the System.err stream. The default value is false.
--------------	-----	--

Table 1 – Properties for configuring a Cams trace logger

Debug

Cams debug can be turned on or offer at various levels. The cams.conf file enables you to toggle global or service-level debug on/off using the following properties:

- cams.debug=false
- serviceadapter.access-control.cams.debug=false
- serviceadapter.authentication.cams.debug=false
- serviceadapter.session-access.cams.debug=false
- serviceadapter.session-control.cams.debug=false
- cams.debug.message=false
- cams.debug.message.connection.input=false
- cams.debug.message.connection.output=false
- cams.debug.message.packet=false

The default value is false. Generally, you should only need to enable these values if instructed to do so by Cafésoft support.

Security Domain Registry

A Cams security domain registry factory is a Java class that loads information about the security domains known to Cams from the security-domain-registry.xml file. This factory is configured with the values shown in Example 1.

```
#
#--- The factory class used to create the SecurityDomainRegistry
#--- and it's configuration parameter(s)
#
security.domain.registry.factory=\
    com.cafesoft.security.engine.domain.XmlSecurityDomainRegistryFactory
security.domain.registry.factory.params=\
    ${cams.home}/conf/domains/security-domain-registry.xml
```

Example 1 – The default Cams server security domain registry factory configuration

The default Cams security domain registry factory (XmlSecurityDomainRegistryFactory), loads an XML registry file located in the directory specified by the parameter. The parameter's \${cams.home} substitution value is specified by the CAMS_HOME environment variable to the Cams server via the startup script.

Normally, you will only need to change this configuration if you move the security-domain-registry.xml file to another directory. The use of a factory provides the flexibility to store the Cams security domain registry in other document formats or databases should it be required.

Resource Types

Cams resource types provide an extensible mechanism to configure agent connections. Configurations are supplied with Cams to enable http resource requests and the native cams resource type to provide connection management messages.

Example 2 shows the http resource type configuration.

```
#
#--- Configure the "http" Resource type
#
resource.type.permissionFactoryClass.http=\
    com.cafesoft.security.engine.access.http.HttpPermissionFactory
resource.type.resourceRequestFactoryClass.http=\
    com.cafesoft.cams.access.http.HttpResourceRequestFactory
```

Example 2 – The default Cams server "http" resource type

Configuring Cams Server

Example 3 shows the cams resource type configuration.

```
#
#--- Configure the "cams" (service) Resource type
#
resource.type.permissionFactoryClass.cams=\
com.cafesoft.security.engine.access.cams.CamsServicePermissionFactory
resource.type.resourceRequestFactoryClass.cams=\
com.cafesoft.security.common.access.cams.CamsServiceResourceRequestFactory
```

Example 3 – The default Cams server "cams" resource type

In most case, you will not need to change these values. However, should you need to create custom resource types to support messaging protocols additional to those supplied, these configuration values provide the flexibility to do so.

Server Sockets

Cams server sockets provide an extensible mechanism to configure socket connections. The default configuration supports non-SSL sockets. Example 4 shows the non-SSL server socket configuration.

```
#
#--- Non-SSL ServerSocketFactory and configuration parameters
#
cams.server.socket.factory.class=\
    com.cafesoft.core.message.DefaultServerSocketFactory
cams.server.socket.factory.params=
```

Example 4 – The default Cams non-SSL server socket configuration

Additional values to support SSL server sockets are supplied but commented out. See [Securing Cams Network Connections with SSL](#) for more details.

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Hardening Cams Security

Cams ships with default configurations to make initial setup and integration as easy as possible. However, there's always a tradeoff between convenience and security risk, which means that you should take measures to ensure that your Cams security configuration is lock-tight. This document provides an overview of issues you should consider and specific values that should be changed to harden the security of your Cams installation.

Cams system security issues fall into the following three categories:

1. [Securing Cams Network Connections](#), including: firewall configuration, securing communications
2. [Securing Cams Files and Directories](#), including: program, configuration, and log files
3. [Securing Cams Services and Agents](#), including: the Cams server, security domains, graceful shutdown, agent authentication, and agent access control

Securing Cams Network Connections

Cams works in conjunction with network hardware and other security software to create a cohesive security blanket for your resources. You can use Cams with many different network topologies. Perhaps the most important consideration is ensuring that Cams and your firewall(s) are working together.

Firewall Configuration

Figure 1 shows a topology with internal and external users accessing web servers with embedded Cams web agents. Web agents communicate with a Cams server across a firewall. In this case, there is a firewall between the web tier servers and the external users but not the internal users. You could just as well require a configuration with a firewall between the internal users and the web servers, which may have different restrictions on each user community. From the perspective of your Cams configuration security only, the most important consideration is the firewall configuration between the Cams web agents and the Cams server.

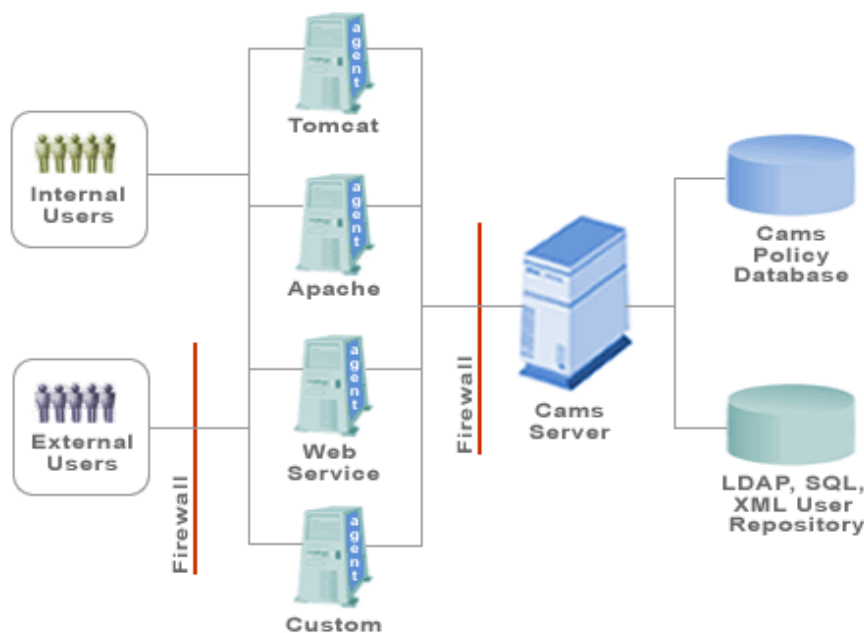


Figure 1 – A possible network topology using Cams

The firewall between Cams web agents and the Cams server should be configured to allow agents to initiate connections from any client-side port to the specific Cams server port listening for agent connections (9191 by default). Though there is nothing inherently insecure about leaving this port set to 9191, you may want to consider changing it to obscure it. You should also configure your firewall to only allow agent connections from known hosts. This is not foolproof as host names and IP addresses can be spoofed, but it does provide a measure of security.

In general, Cams web agents in this topology will be running on hosts within one TCP/IP subnet and the Cams server will be running on an internal subnet. The firewall will use static Network Address Translation (static NAT) to map an IP address from the web agent's subnet to the internal IP address of the host running the Cams server.

Hardening Cams Security

The Cams server also listens for shutdown requests on a different TCP/IP port (9292 by default). You should not open a hole in your external firewall for the shutdown port, as it may allow someone to shutdown your Cams server remotely.

The Cams server communicates on the back end with servers that include user repositories. Typically, these repositories will reside on the same local area network as the Cams server, but this may not always be the case. Hence, you must examine the various issues for communicating with these resources. In fact, it may be desirable to have a firewall between Cams and these resources!

Securing Communications

By default, Cams selectively encrypts sensitive network communications using a configurable PKI algorithm and secret key. Some agents also support SSL communications with the Cams server. Tradeoffs between these confidentiality mechanisms include: performance, security, and ease of administration.

Cams selective data encryption provides adequate confidentiality for most sites with good performance and easy administration. Sensitive information, like user authentication credentials, agent passwords, and shutdown passwords are encrypted, while other less-sensitive data like resource identifiers, network addresses, etc. are sent in clear text.

SSL provides superior security at the cost of lower performance and more difficult administration. When using SSL, all data transferred between the Cams agent and server are encrypted. In addition, the SSL architecture ensures the integrity of data. If the receiving server can decrypt the data, then it has not been altered in transit. A final benefit is that SSL enables Cams agents to verify the identity of a Cams server.

NOTE: Cams is not currently packaged with SSL support of the agent connections to the Cams server. However, it is possible to configure Cams with SSL support. Packaged support will be provided in a future release. If you have need of this level of confidentiality now, please contact [Cafésoft support](#) for further instruction.

The following document provides more information on the configuration of specific confidentiality measures:

- [Securing Cams Communications using Secret Keys](#)

Similarly, communications between the Cams server and user repositories may need encryption. For example, if a login module communicates with an LDAP user repository, you may want to use secure LDAP for the connection. This need is somewhat reduced if passwords are stored as hashed values in the repository, and if the local area network is not accessible by internal users.

Securing Cams Files and Directories

In a typical production environment, the Cams server is started and executed by the administrative "root" user (Unix) or "Administrator" user (Windows NT/2000). The general strategy for securing Cams files and directories is to:

1. Ensure that Cams server or agent are executed as the appropriate administrative user (root or Administrator), hereafter referred to as the owner. This enables Cams and Cams utilities to run within the secured file and directory environment described below.
2. Enable owner read/write/execute permissions on all Cams directories, but no permissions for all other users and groups. This enables owner processes to scan and modify the contents of directories, while prohibiting all other users and groups from seeing or modifying the contents of directories.
3. Enable owner read/write/execute permissions on the Cams programs and scripts in: CAMS_HOME/bin, but no permissions for all others users and groups. This ensures that arbitrary users cannot execute administrative commands available within the Cams environment.
4. Enable owner read/write permissions on configuration files and log files, but no permissions for all other users and groups. This ensures that an arbitrary user cannot replace, overwrite, or redirect log files to obscure security violations or obtain sensitive information via trace logs.

The following sections provide operating system-specific instructions for securing Cams files and directories.

Securing Cams Files and Directories under Linux/UNIX

In the instructions that follow, it is assumed that the Cams server is executed by root on your Linux/UNIX system. For the sake of this example, assume that:

1. Cams is installed at /usr/local/camsServer
2. the configuration directory is at /usr/local/camsServer/conf
3. that you are logged in as root.

As is the case with any command that root executes, you must take care that it is protected from modification by non-root users. Not only must the files themselves be writable only by root, but so must the directories, and parents of all directories. It is assumed that /, /usr, and

/usr/local are only modifiable by root. When you install the Cams server, you should ensure that it is similarly protected.

A shell script for Unix/Linux is provided at CAMS_HOME/bin/setFilePerm.sh. This script hardens the security of files and directories under the default Cams server installation tree. The script executes the following steps:

Step 1 – Change directories to the Cams server installation directory

```
cd /usr/local/camsServer
```

Step 2 – Set user and group ownership of all files and directories

```
chown -R root.root .
```

This command recursively finds all file and directories and sets user and group ownership to root.

Step 3 – Set directory permissions

```
find . -type d -exec chmod 700 {} \;
```

This command recursively finds all directories and gives the read/write/execute permissions only for the owner. Group and world are given no permissions.

Step 4 – Set executable permissions

```
chmod 700 bin/*
```

This command enables read/write/execute permissions for the owner of all files in the bin directory. Group and world are given no permissions.

Step 5 – Set file permissions

```
find conf -type f -exec chmod 600 {} \;  
find logs -type f -exec chmod 600 {} \;  
find lib -type f -exec chmod 600 {} \;  
find classes -type f -exec chmod 600 {} \;  
find templates -type f -exec chmod 600 {} \;  
find src -type f -exec chmod 600 {} \;
```

This series of commands recurses through the conf, logs, lib, classes, templates, and src directory trees looking for files (not directories) and gives them read/write permissions for the owner. Group and world are given no permissions.

Cams agent configuration files and directories should use similar means to secure their contents, although instructions for doing so are necessarily specific to the software application hosting the Cams agent. See the appropriate agent configuration guide for more details.

To test these permissions:

1. If the Unix system hosting Cams supports normal user accounts, try logging in as a normal user.
2. Try to list the contents of: /usr/local/camsServer (permission should be denied)
3. Try changing directories to: /usr/local/camsServer (permission should be denied)
4. Try creating a test file in: /usr/local/cams/test.txt using a text editor like "vi" or "emacs" (permission should be denied)

Securing Cams Files and Directories under Windows NT/2000

In the instructions that follow, it is assumed that the Cams server is executed by the Admin user on your Windows NT/2000 server. For the sake of this example, assume that:

1. The Cams Server is currently stopped
2. Cams is installed at C:\camsServer
3. The configuration directory is at: C:\camsServer\conf
4. You are logged in as Admin to your Windows NT/2000 Server or Domain Controller

As is the case with any command that Admin executes, you must take care that it is protected from modification by non-Admin users. The file permissions graphical user interface varies between Windows NT and Windows 2000, so use the appropriate instructions below for your system.

Securing Windows NT Files and Directories

The following instructions apply to Windows NT 4.0.

Step 1 – Set user and group ownership of all files and directories

This is done using the Windows NT graphical user interface.

1. Using the Windows Explorer file browser, select the top-level Cams installation directory: C:\camsServer
2. Right click on the folder and select "Properties" from the pop-up menu
3. In the dialog box that appears, select the: "Security" tab
4. Click on the "Ownership" button
5. In the dialog box that appear, confirm that: "Admin\Administrators" is the intended owner, then click: "Take Ownership"

Step 2 – Set all directory and file permissions

From the same "Security" tab in the context-sensitive dialog box for directory C:\camsServer used in Step 1:

1. Click on the "Permissions" button
2. In the "Directory Permissions" dialog box that appears, confirm that the directory Owner is now: Admin\Administrators
3. Select check box: "Replace Permissions on Subdirectories" (i.e. make sure it is checked)
4. Select check box: "Replace Permissions on Existing Files" (i.e. make sure it is checked)
5. In the list of all User\Group items listed, "Remove" all items except: "Admin\Administrator"
6. Select the list item: "Admin\Administrator", then select "Type of Access: " = Full Control

Securing Windows 2000 Files and Directories

The following instructions apply to Windows 2000.

Step 1 – Set user and group ownership of all files and directories

This is done using the Windows 2000 graphical user interface.

1. Using the Windows Explorer file browser, right click on the top-level Cams installation directory: C:\camsServer and select the "Properties" menu item. A dialog box entitled: "camsServer Properties" will appear.
2. Select the "Security" tab
3. Select the "Advanced ..." button. A dialog box entitled: "Access Control Settings for camsServer" will appear.
4. Select the "Owner" tab.
5. Select the "Admin" user from the list at the top.
6. Select the "Replace owner on subcontainers and objects" check box
7. Select the "OK" button to apply ownership changes

Step 2 – Set all directory and file permissions

1. In the "camsServer Properties" dialog box, deselect the "Allow inheritable permissions from parent to propagate to this object" check box. A dialog box entitled: "Security" will appear.
2. Select the "Remove" button to remove inherited permissions explicitly specified on the camsServer directory.
3. Select the "Add ..." button. A dialog box entitled: "Select Users or Groups" will appear.
4. Select the "admin" user from list at the top.
5. Select the "Add" button. The Admin user should now be added to the list at the bottom of the dialog box.
6. Select the "OK" button to apply the selected "Admin" user.
7. In the "Permissions" list, select check box: "Allow/Full Control". This will cause all "Allow" column check boxes to become selected.
8. Select the "OK" button to apply the camsServer Properties changes.

To test these directory and file permissions under Windows NT/2000:

1. Log out from the Windows NT/2000 server Admin account and login using a normal user account (not Admin)
2. Using a file browser or command prompt, attempt to view the contents of directory: C:\camsServer. (You should not be able to view any files in this directory).
3. If you will be running the Cams Server as a Windows NT/2000 "Service", make sure the service is run under the Admin user identity. Try starting and stopping the Cams Server Service.

NOTE: Unless necessary, it is also recommended that you avoid "sharing" the Cams Server installation directory and its contents between hosts.

Securing Cams Services and Agents

The Cams server hosts a set of services that should be secured with site-specific configuration values. Cams agents should be configured with compatible values and described in the following sections.

Cams Server

This section explains the Cams server values that you should change to harden your site security. Start by opening the `CAMS_HOME/conf/cams.conf` file in a text editor.

You should change the name of your Cams server from `MyCamsServer` to something unique for your site. This value is used for logging and to distinguish Cams server installations and their corresponding user sessions from each other. It does not need to correspond to the host's DNS name. You should change its value to something unique for your company or department, etc. For example: `myhost_domain`. Alphanumeric and underscore ("`_`") characters are valid. You'll also need to change the corresponding name in the configuration file for each Cams agent.

```
#--- The name of the Cams server. This value and cams.server.url
#--- are used to uniquely identify a Cams server instance and its
#--- available services.
#
cams.server.name=MyCamsServer
```

You can change the Cams server connection port. Though the default port 9191 is not insecure, you can gain a measure of security by obscuring the port you are using from the standard Cams default value. Again, you'll also need to make corresponding changes in the configuration file for each Cams agent.

```
#
#--- The TCP/IP port on which Cams listens for general security
#--- service connections.
#
cams.server.port=9191
```

You may want to change the Cams server shutdown port with the next value. Unlike the connection port, the Cams agents do not require the same value.

```
#
#--- The TCP/IP port on which Cams listens for shutdown
#--- service connections.
#
cams.server.shutdown.port=9292
```

You should definitely change the shutdown password from the default. The shutdown password is required to send a valid shutdown command to the Cams server. If a secret key is installed, the shutdown password is encrypted before being sent from the shutdown client to the Cams server.

```
#
#--- The password that must be supplied by shutdown clients
#--- to authenticate requests.
#
cams.server.shutdown.password=theEndIsNear
```

Security Domains

You must consider values in each security domain that should be hardened. Use a text editor to open the `security-domain.xml` file for each security domain in your configuration. The Cams session manager service uses a key to encrypt the session id for each security domain. Each security domain should have a unique key, which should be changed from the default value of `secret-key`.

```
<session-manager-service ...>
  <param-list>
    ...
    <param name="sessionIdIPAddrValidationMask" value="255.255.255.255"/>
    <param name="sessionIdKey" value="secret-key"/>
  </param-list>
  ...
</session-manager-service>
```

Hardening Cams Security

The session-manager-service's sessionIdIPAddrValidationMask should be made as restrictive as practical. This value is used to validate the IP address of a remote, authenticated client (like a web browser) against the IP address when the client authenticated and its session was created. The mask value: 255.255.255.255 causes the entire IP address associated with client requests to be validated. This may be too restrictive in cases where a web browser client may access protected resources via different network routes. For example, many major Internet Service Providers, will transmit HTTP and HTTPS network traffic via different routes by use of proxy servers. In addition, some routers are setup to used "Dynamic Network Address Translation". Both conditions lead to HTTP/HTTPS requests for the same client that appear to come from different IP addresses. In such cases, it may be necessary to make IP address validation less restrictive by using a validation mask like: 255.255.255.0, which would validate only the first 3 triplets in a class C address.

Other usernames and passwords specified in session event handlers, service manager services, etc. are for connections to user repositories. Any defaults that are provided must obviously be changed to work with your site's data repositories. Data repository connections in login-config.xml also share the same issue.

The access-control.xml file does not require hardening, just configuration to your site specific needs. However, you may desire to make changes to strengthen, for example, the rules associated with agent authentication. In this respect, you could add a list of valid hostnames or IP addresses to the agent authentication rule.

You will need to change cams-users.xml in the system security domain. This is where the default Cams agent profiles are stored. You should minimally change the passwords and possibly the username and role for agents. If you change the name of the role, you must change the corresponding role name for the agent rule in the system security domain's access-control.xml file.

```
<cams-users>
  <!-- Cams http agent connection authentication accounts -->
  <user name="cams-tomcat-agent" password="password" roles="cams-agent" />
  <user name="cams-apache-agent" password="password" roles="cams-agent" />
</cams-users>
```

Agents

Connection of Cams agents to a Cams server should also be carefully secured in a production environment. The system security domain is responsible for authenticating and checking access control rules for all Cams agents.

Securing Agent Authentication

Each Cams agent must authenticate with the Cams server by supplying a username and password. The username and password must match a profile in the corresponding security domain's user repository. By default, this is the cams-users.xml file in the system security domain, but can easily be modified to use an LDAP server or SQL server.

The Cams Tomcat web agent is pre-configured with the following username/password parameter defaults:

```
cams.client.authentication.principal=cams-tomcat-agent
cams.client.authentication.credential=password
```

The Cams Apache web agent is pre-configured with the following username/password parameter defaults:

```
connection.authentication.principal=cams-apache-agent
connection.authentication.credential=password
```

You should change each of these values according to your username and password policies. In addition, you should update any other configuration values for which you may have changed a corresponding value on the Cams server. For example, if you change the Cams server name or connection port, you must change the corresponding value in the agent configuration file.

In addition, you should use Cams secret keys (described in Securing Cams Communications using Secret Keys) to encrypt username/password values sent from agents to a Cams server.

Securing Agent Access Control

The Cams system security domain's access control policy contains default rules for controlling access by Cams agents. These rules are shown in Example 1 and control access by requiring the authenticated agent to belong to the cams-agent role. If an agent has the cams-agent role, then access is granted, else access is denied.

```
<access-control-policy>
...
  <!-- Authenticated agents must have the "cams-agent" role -->
  <auth-acr id="cams agent rule">
```

Hardening Cams Security

```
<role-constraint>
  <role-name>cams-agent</role-name>
</role-constraint>
</auth-acr>
...
</access-control-policy>
```

Example 1 – Default system security domain access control rules for connecting Cams agents

You can harden access control for Cams agents by incorporating IP address constraints into the system security domain's access control policy as shown in Example 2.

```
<access-control-policy>
...
<!-- Authenticated agents must have the "cams-agent" role
and be connecting from an approved remote address -->
<acr id="cams agent rule">
  <auth-acr>
    <role-constraint>
      <role-name>cams-agent</role-name>
    </role-constraint>
  </auth-acr>
  </and>
  <host-acr>
    <allow-address>
      <address>127.0.0.1</address>
      <address>192.168.1.112</address>
      <address>192.168.1.113</address>
    </allow-address>
  </host-acr>
</acr>
...
</access-control-policy>
```

Example 2 – Hardened system access control rules for connecting Cams agents

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Access Control Policy Tag Reference

A security domain's access control policy declares the resources protected within a security domain and the access control rules that protect them. Specifically, each access control policy contains permission collections and an access control rule library, which are containers for permissions and access control rules. This document contains reference information for each of the tags that can be used within a Cams `access-control-policy.xml` file. The following table shows the file structure with links to each of the possible elements.

Tag Name	Instances	Description
access-control-policy	1	declares an access control policy
[permission-collection]	0 ... N	a collection of permissions
[permission]	0 ... N	a specific permission that associates a set of resources and access control rule or owner
resource-pattern	1	a pattern for matching resources
acr-ref owner	1	a reference to an access control rule or a security domain to which access control is delegated
acr-lib	1	a library of access control rules
[acr-type]	0 ... N	a custom type of access control rule registered with the access control rule library
acr-persistence-manager	1	defines the class responsible for creating, loading, storing, and removing access control rule instances for a given type
param-list	0 ... 1	a list of initialization/configuration parameters
param	0 ... N	a generic name/value pair
[acr]	0 ... N	an access control rule expression
[granted]	0 ... N	a Cams intrinsic access control rule that always grants access
[denied]	0 ... N	a Cams intrinsic access control rule that always denies access
[confidential]	0 ... N	a Cams intrinsic access control rule that requires that a resource be accessed via an encrypted connection (usually means SSL/TLS)
[and or not]	0 ... N	boolean operators for use between access control rule invocations
[acr-ref]	0 ... N	a reference to another access control rule to be invoked
[auth-acr]	0 ... N	an authentication access control rule embedded in this rule
[host-acr]	0 ... N	a remote host access control rule embedded in this rule
[custom-acr]	0 ... N	a custom access control rule of a type declared by acr-type and embedded in this rule
[auth-acr]	0 ... N	an authentication access control rule that implements role-based access control
[role-constraint]	0 ... N	grants or denies access based on a role name and optionally a class
role-name	1	a role name
role-class	0 ... 1	a role class
[host-acr]	0 ... N	a remote host access control rule
allow-host	0 ... 1	a list of remote hosts to allow access

Access Control Policy Tag Reference

[host]	0 ... N	a remote host name pattern
deny-host	0 ... 1	a list of remote hosts to deny access
[host]	0 ... N	a remote host name pattern
allow-address	0 ... 1	a list of remote addresses to allow access
[address]	0 ... N	a remote host IP address pattern
deny-address	0 ... 1	a list of remote addresses to deny access
[address]	0 ... N	a remote host IP address pattern
[custom-acr]	0 ... N	a custom access control rule of a type declared by acr-type (see also acr-persistence-manager).

<access-control-policy>

The <access-control-policy> element is the top-level element for a security domain's access control policy.

Item	Description						
Syntax	<pre><access-control-policy defaultBias="granted denied" debug="true false"> <permission-collection> ... </permission-collection> ... <acr-lib> ... </acr-lib> </access-control-policy></pre>						
Attributes	<table border="0"> <tr> <td style="padding-right: 20px;">defaultBias</td> <td style="padding-right: 20px;">Opt</td> <td>Specify the security domain default bias to either grant or deny access to resources (granted or denied). If not specified, the default bias is denied.</td> </tr> <tr> <td>debug</td> <td>Opt</td> <td>Activate debug for the access control policy (true or false).</td> </tr> </table>	defaultBias	Opt	Specify the security domain default bias to either grant or deny access to resources (granted or denied). If not specified, the default bias is denied.	debug	Opt	Activate debug for the access control policy (true or false).
defaultBias	Opt	Specify the security domain default bias to either grant or deny access to resources (granted or denied). If not specified, the default bias is denied.					
debug	Opt	Activate debug for the access control policy (true or false).					
Data	None						
Parent Elements	None						
Child Elements	<table border="0"> <tr> <td style="padding-right: 20px;"><permission-collection></td> <td style="padding-right: 20px;">Opt</td> <td>A type-specific collection of permissions. Any number of permission-collections may exist within an access-control-policy, but each must have a unique type (currently http or cams). The permission-collection/permission types are dictated by the unique types of agents enforcing the access control policy in your Cams deployment.</td> </tr> <tr> <td><acr-lib></td> <td>Req</td> <td>A container for all security domain-specific access control rule types and references. Only one library is permitted.</td> </tr> </table>	<permission-collection>	Opt	A type-specific collection of permissions. Any number of permission-collections may exist within an access-control-policy, but each must have a unique type (currently http or cams). The permission-collection/permission types are dictated by the unique types of agents enforcing the access control policy in your Cams deployment.	<acr-lib>	Req	A container for all security domain-specific access control rule types and references. Only one library is permitted.
<permission-collection>	Opt	A type-specific collection of permissions. Any number of permission-collections may exist within an access-control-policy, but each must have a unique type (currently http or cams). The permission-collection/permission types are dictated by the unique types of agents enforcing the access control policy in your Cams deployment.					
<acr-lib>	Req	A container for all security domain-specific access control rule types and references. Only one library is permitted.					
Example	<pre><access-control-policy> <!-- HTTP (Web) Resource Permissions --> <permission-collection type="http" desc="HTTP (Web) Resources"> <permission desc="ACME Local Area Network Resources"> <resource-pattern id="http://www.acme.com/employee*" /> <acr-ref id="ACME LAN Rule" /> </permission> </permission-collection> <!-- Allow http agents to connect --></pre>						

Access Control Policy Tag Reference

	<pre> <permission-collection type="cams" desc="Agent Permissions"> <permission desc="Cams Agent Connections" actions="ACCESS"> <resource-pattern id="{cams.resource.base.id}"/> <acr-ref id="cams agent rule"/> </permission> </permission-collection> <!-- Access Control Rule Library --> <acr-lib> <host-acr id="ACME LAN Rule"> <accept-address-constraint> <address>192.168.0.*</address> </accept-address-constraint> </host-acr> </acr-lib> </access-control-policy> </pre>
--	---

<permission-collection>

A security domain's access control policy contains one permission collection for each unique type of permission. For example, all http permissions are stored in a single http-specific permission collection. Permission collections organize Cams access control policies and optimize access control performance.

There can be any number of permission-collections within an access-control-policy, once for each unique type of permission.

Item	Description									
Syntax	<pre> <permission-collection type="type code" desc="text description" debug="true false"> <permission ... > ... </permission> ... </permission-collection> </pre>									
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;">type</td> <td style="width: 10%; text-align: center;">Req</td> <td>A code indicating the type of permission contained within the permission collection. Currently http or cams.</td> </tr> <tr> <td>desc</td> <td style="text-align: center;">Opt</td> <td>A description used for readability and for context-sensitive messages. Use unique, descriptive names to facilitate message comprehension and debugging.</td> </tr> <tr> <td>debug</td> <td style="text-align: center;">Opt</td> <td>Activate debug for the permission collection (true or false).</td> </tr> </table>	type	Req	A code indicating the type of permission contained within the permission collection. Currently http or cams.	desc	Opt	A description used for readability and for context-sensitive messages. Use unique, descriptive names to facilitate message comprehension and debugging.	debug	Opt	Activate debug for the permission collection (true or false).
type	Req	A code indicating the type of permission contained within the permission collection. Currently http or cams.								
desc	Opt	A description used for readability and for context-sensitive messages. Use unique, descriptive names to facilitate message comprehension and debugging.								
debug	Opt	Activate debug for the permission collection (true or false).								
Data	None									
Parent Elements	1. <access-control-policy>									
Child Elements	<permission> Opt A type-specific resource and rules for controlling access.									
Example	<pre> <permission-collection type="http" desc="HTTP (Web) Resources"> <!-- Declare web pages available to all ACME employees --> <permission desc="ACME Employee Web Resources"> <resource-pattern id="http://www.acme.com/employee*" /> <acr-ref id="ACME Employee Rule" /> </permission> <!-- Declare web pages available only to ACME management --> <permission desc="ACME Management Web Resources"> <resource-pattern id="http://www.acme.com/management*" /> <acr-ref id="ACME Management Resources Rule" /> </permission> </permission-collection> </pre>									

<permission>

A permission identifies a set of resources via a resource pattern and either an access control rule to protect those resources or an owner security domain to which access control is to be delegated. The rules for defining resource patterns vary with the type of permission.

Any number of permissions may be added to a permission-collection provided the resource pattern and action(s) are valid for the enclosing permission collection and the permissions don't overlap. Two permissions overlap if their fully-qualified resource patterns are identical and they have at least one action in common. Cams will not allow addition of an overlapping permission and an access control policy referencing an overlapping permission will fail to load.

Item	Description		
Syntax	<pre><permission desc="text description" actions="GET,POST,PUT,DELETE,HEAD,OPTIONS,TRACE" debug="true false"> <resource-pattern ... /> <acr-ref ... /> <owner ... /> ... </permission></pre>		
Attributes	<p>desc</p> <p>actions</p> <p>debug</p>	<p>Opt</p> <p>Opt</p> <p>Opt</p>	<p>A description used for readability and for context-sensitive messages about the permission.</p> <p>A pattern for matching actions on incoming access control requests. The list of possible actions is dependent on the type of resource being protected. Here are some example action patterns for different types of permissions:</p> <ul style="list-style-type: none"> • type="http" -> GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE • type="cams" -> ACCESS, START, STOP <p>An access control request will match the actions part of a resource pattern if all of the actions it requests are present in the resource pattern's actions list. If a resource-pattern does not specify a list of actions, the convention is to match any/all actions.</p> <p>Activate debug for the permission (true or false).</p>
Data	None		
Parent Elements	1. <permission-collection>		
Child Elements	<p><resource-pattern></p> <p><acr-ref></p> <p><owner></p>	<p>Req</p> <p>Opt</p> <p>Opt</p>	<p>A type-specific pattern for resource identifiers that match this permission.</p> <p>A reference to an access control rule that protects the resources. Although optional, either <acr-ref> or <owner> must be referenced within a permission.</p> <p>The name of a security domain to which access control is to be delegated. Although optional, either <acr-ref> or <owner> must be referenced within a permission.</p>
Example	<pre><!-- Declare web pages available to the public --> <permission desc="ACME Public Web Resources" actions="GET"> <resource-pattern id="http://www.acme.com/*" /> <acr-ref id="granted"/> </permission> <!-- Delegate human resources content to the "Human Resources" security domain --> <permission desc="ACME Public Web Resources"> <resource-pattern id="http://www.acme.com/human-resources*" /> <owner id="Human Resources" /> </permission></pre>		

<resource-pattern>

A resource-pattern identifies the resources associated with a permission. When a permission receives an access control request, it will compare the requested resource's identifier and action against its identifier pattern and actions pattern. If the pattern matches and it is the most specific pattern for the request, then the permission matches and the access control policy will use it to enforce access control.

Item	Description
Syntax	<code><resource-pattern id="type-specific resource id pattern"/></code>
Attributes	<p>id Req A pattern for matching resources identifiers. The actual syntax of this pattern depends on the type of resource being protected. Here are some example resource id patterns for different types of permissions:</p> <ul style="list-style-type: none"> • type="http" -> http://www.acme.com/* • type="cams" -> cams://www.acme.com:9191 <p>Support for pattern wild card matching and range matching can vary by permission type.</p>
Data	None
Parent Elements	1. <permission>
Child Elements	None
Example	<pre><!-- Declare the public web pages --> <permission desc="ACME Public Web Resources" actions="GET,POST"> <resource-pattern id="http://www.acme.com/*"/> <acr-ref id="granted"/> </permission></pre>

<acr-ref>

A reference to an existing access control rule within the enclosing security domain's access control rule library. Access control rules can be referenced from:

1. permissions to assign the access control rule that protects matching resources
2. other access control rule expressions to be create (combined) custom access control rules

Item	Description
Syntax	<code><acr-ref id="identifier of existing access control rule"/></code>
Attributes	<p>id Req The identifier of an access control rule within the access control rule library. Cams intrinsic access control rules, which are always available for use from permissions and other rules, are:</p> <ul style="list-style-type: none"> • granted • denied • confidential <p>Cams also ships with standard access control rule types, such as auth-acr and host-acr, that can be referenced within the access control rule library, then by this attribute.</p>
Data	None
Parent Elements	1. <permission> 2. <acr>
Child Elements	None
Example	<pre><!-- Declare web pages available to the public --> <permission desc="ACME Public Web Resources" actions="GET"> <resource-pattern id="http://www.acme.com/*"/> <acr-ref id="granted"/> </permission></pre>

Access Control Policy Tag Reference

	<pre> ... <!-- Rule for accessing LAN requires either the user authenticate or the workstation be on the LAN --> <acr id="Local Channel Access Rule"> <acr-ref id="LAN User Auth Rule"/> <or/> <acr-ref id="LAN User Address Rule"/> </acr> </pre>
--	---

<owner>

A reference to an existing security domain to which access control for a permission is delegated.

Item	Description			
Syntax	<code><owner id="security domain identifier"/></code>			
Attributes	<table border="0"> <tr> <td style="padding-right: 20px;">id</td> <td style="padding-right: 20px;">Req</td> <td>the name of an existing security domain.</td> </tr> </table>	id	Req	the name of an existing security domain.
id	Req	the name of an existing security domain.		
Data	None			
Parent Elements	1. <permission>			
Child Elements	None			
Example	<pre> <!-- Delegate ACME web server access control to the ACME security domain --> <permission desc="ACME Web Server"> <resource-pattern id="http://www.acme.com/*" /> <owner id="ACME"/> </permission> </pre>			

<acr-lib>

An access control rule library, which can only be referenced once within an access control policy. It serves as a container for all access control rule references and types available within a security domain.

Item	Description															
Syntax	<pre> <acr-lib debug="true false"> ... </acr-lib> </pre>															
Attributes	<table border="0"> <tr> <td style="padding-right: 20px;">debug</td> <td style="padding-right: 20px;">Opt</td> <td>Activate debug for the library. The default is false.</td> </tr> </table>	debug	Opt	Activate debug for the library. The default is false.												
debug	Opt	Activate debug for the library. The default is false.														
Data	None															
Parent Elements	1. <access-control-policy>															
Child Elements	<table border="0"> <tr> <td style="padding-right: 20px;"><acr-type></td> <td style="padding-right: 20px;">Opt</td> <td>A declaration for a custom access control rule type. This tag registers new access control rule types created using the Cams Access Control Rule API. See the Cams Programmer's Guide for more details.</td> </tr> <tr> <td style="padding-right: 20px;"><acr></td> <td style="padding-right: 20px;">Opt</td> <td>An access control rule reference.</td> </tr> <tr> <td style="padding-right: 20px;"><auth-acr></td> <td style="padding-right: 20px;">Opt</td> <td>An authentication rule.</td> </tr> <tr> <td style="padding-right: 20px;"><host-acr></td> <td style="padding-right: 20px;">Opt</td> <td>A remote host access control rule.</td> </tr> <tr> <td style="padding-right: 20px;">[custom-acr]</td> <td style="padding-right: 20px;">Opt</td> <td>If one or more custom access control rule types are declared, create and use a custom acr tag within the access control rule library. (The actual tag name for custom access control rules is specified as part of the acr-type).</td> </tr> </table>	<acr-type>	Opt	A declaration for a custom access control rule type. This tag registers new access control rule types created using the Cams Access Control Rule API. See the Cams Programmer's Guide for more details.	<acr>	Opt	An access control rule reference.	<auth-acr>	Opt	An authentication rule.	<host-acr>	Opt	A remote host access control rule.	[custom-acr]	Opt	If one or more custom access control rule types are declared, create and use a custom acr tag within the access control rule library. (The actual tag name for custom access control rules is specified as part of the acr-type).
<acr-type>	Opt	A declaration for a custom access control rule type. This tag registers new access control rule types created using the Cams Access Control Rule API. See the Cams Programmer's Guide for more details.														
<acr>	Opt	An access control rule reference.														
<auth-acr>	Opt	An authentication rule.														
<host-acr>	Opt	A remote host access control rule.														
[custom-acr]	Opt	If one or more custom access control rule types are declared, create and use a custom acr tag within the access control rule library. (The actual tag name for custom access control rules is specified as part of the acr-type).														
Example	<pre> <acr-lib> <!-- Require an authenticated user to have "employee" role --> <auth-acr id="ACME Employee Rule"> </pre>															

Access Control Policy Tag Reference

```

    <role-constraint>
      <role-name>employee</role-name>
    </role-constraint>
  </auth-acr>

<!-- Require hosts to be on the ACME Local Area Network -->
<host-acr id="ACME LAN Rule">
  <accept-address-constraint>
    <address>192.168.0.*</address>
  </accept-address-constraint>
</host-acr>

<!-- Require an authenticated user to have the "manager" role
that is implemented by a particular role class -->
<auth-acr id="ACME Manager Rule">
  <role-constraint>
    <role-name>manager</role-name>
    <role-class>com.cafesoft.cams.auth.CSRolePrincipal</role-class>
  </role-constraint>
</auth-acr>

<!-- Require:
1. the host to be on the ACME Local Area Network
2. the user to be authenticated with "manager" role
3. a confidential (SSL) connection for privacy -->
<acr id="ACME Manager Resources Rule">
  <acr-ref id="ACME LAN Rule">
  </acr-ref>
  <acr-ref id="ACME Management Rule">
  </acr-ref>
  <confidential/>
</acr>

<!-- Declare a custom Access Control Rule type for limiting
access by company business hours -->
<acr-type
  name="example:business-hours-acr"
  className="examples.acrs.BusinessHoursAcr"
  desc="Control access by normal business hours"
  >
  <acr-persistence-manager className="examples.acrs.XmlBusinessHoursAcrPm">
    <param-list>
      <param name="clockHours" value="0-23"/>
      <param name="useLocalTimeZone" value="true"/>
    </param-list>
  </acr-persistence-manager>
</acr-type>

<!-- Create an Access Control Rule for limiting access
by company business hours -->
<example:business-hours-acr
  xmlns:example=\
    "http://cafesoft.com/example-business-hours-acr_1_0.dtd"
  id="business hours rule"
  desc="Limit access to M-F business hours">

  <example:business-hours start-hour="8" end-hour="17"/>
</example:business-hours-acr>

</acr-lib>

```

<acr-type>

An access control rule type declares a custom type of access control rule within an access control rule library. Once defined, access control rules can be used within the access control rule library directly or nested within an access control rule expression.

Item	Description
Syntax	<pre> <acr-type> ... </acr-type> </pre>

Access Control Policy Tag Reference

Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%; padding: 5px;">name</td> <td style="width: 10%; padding: 5px;">Req</td> <td style="padding: 5px;">The name associated with the access control rule type. In the case of XML, the name corresponds to the element tag name that denotes the beginning of a new reference of the access control rule type.</td> </tr> <tr> <td style="padding: 5px;">className</td> <td style="padding: 5px;">Req</td> <td style="padding: 5px;">The fully-qualified Java class name implementing the access control rule.</td> </tr> <tr> <td style="padding: 5px;">desc</td> <td style="padding: 5px;">Opt</td> <td style="padding: 5px;">A textual description of the access control rule type.</td> </tr> </table>	name	Req	The name associated with the access control rule type. In the case of XML, the name corresponds to the element tag name that denotes the beginning of a new reference of the access control rule type.	className	Req	The fully-qualified Java class name implementing the access control rule.	desc	Opt	A textual description of the access control rule type.
name	Req	The name associated with the access control rule type. In the case of XML, the name corresponds to the element tag name that denotes the beginning of a new reference of the access control rule type.								
className	Req	The fully-qualified Java class name implementing the access control rule.								
desc	Opt	A textual description of the access control rule type.								
Data	None									
Parent Elements	1. <acr-lib>									
Child Elements	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%; padding: 5px;"><acr-persistence-manager></td> <td style="width: 10%; padding: 5px;">Req</td> <td style="padding: 5px;">Configures the Java class that is responsible for storing, loading, and creating new access control rules.</td> </tr> </table>	<acr-persistence-manager>	Req	Configures the Java class that is responsible for storing, loading, and creating new access control rules.						
<acr-persistence-manager>	Req	Configures the Java class that is responsible for storing, loading, and creating new access control rules.								
Example	<pre style="font-family: monospace; font-size: 0.9em;"> <!-- Declare a custom Access Control Rule type for limiting <acr-type name="example:business-hours-acr" className="examples.acrs.BusinessHoursAcr" desc="Control access by normal business hours" > <acr-persistence-manager className="examples.acrs.XmlBusinessHoursAcrPm"> <param-list> <param name="clockHours" value="0-23"/> <param name="useLocalTimeZone" value="true"/> </param-list> </acr-persistence-manager> </acr-type> <!-- Create an Access Control Rule for limiting access by company business hours --> <example:business-hours-acr xmlns:example="http://cafesoft.com/example-business-hours-acr_1_0.dtd" id="business hours rule" desc="Limit access to M-F business hours"> <example:business-hours start-hour="8" end-hour="17"/> </example:business-hours-acr> </pre>									

<acr-persistence-manager>

An access control rule persistence manager is responsible for persisting (storing), loading, and creating new access control rules.

Item	Description			
Syntax	<pre style="font-family: monospace; font-size: 0.9em;"> <acr-persistence-manager debug="true false"> ... </acr-persistence-manager> </pre>			
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%; padding: 5px;">className</td> <td style="width: 10%; padding: 5px;">Req</td> <td style="padding: 5px;">The fully-qualified Java class name implementing the access control rule persistence manager.</td> </tr> </table>	className	Req	The fully-qualified Java class name implementing the access control rule persistence manager.
className	Req	The fully-qualified Java class name implementing the access control rule persistence manager.		
Data	None			
Parent Elements	1. <acr-type>			
Child Elements	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%; padding: 5px;"><param-list></td> <td style="width: 10%; padding: 5px;">Opt</td> <td style="padding: 5px;">An optional list of parameters that can be used to set the default configuration.</td> </tr> </table>	<param-list>	Opt	An optional list of parameters that can be used to set the default configuration.
<param-list>	Opt	An optional list of parameters that can be used to set the default configuration.		
Example	<pre style="font-family: monospace; font-size: 0.9em;"> <!-- Declare a custom Access Control Rule type for limiting access by company business hours --> <acr-type name="example:business-hours-acr" className="examples.acrs.BusinessHoursAcr" desc="Control access by normal business hours" > <acr-persistence-manager className="examples.acrs.XmlBusinessHoursAcrPm"> </pre>			

Access Control Policy Tag Reference

```

<param-list>
  <param name="clockHours" value="0-23"/>
  <param name="useLocalTimeZone" value="true"/>
</param-list>
</acr-persistence-manager>
</acr-type>

<!-- Create an Access Control Rule for limiting access
by company business hours -->
<example:business-hours-acr
xmlns:example="http://cafesoft.com/example-business-hours-acr_1_0.dtd"
id="business hours rule"
desc="Limit access to M-F business hours">

  <example:business-hours start-hour="8" end-hour="17"/>
</example:business-hours-acr>

```

<param-list>

A list of parameters that can be used to set the default configuration.

Item	Description
Syntax	<pre> <param-list> <param ... /> ... </param-list> </pre>
Attributes	None
Data	None
Parent Elements	1. <acr-persistence-manager>
Child Elements	<param> Opt An initialization/configuration parameter as a generic name/value pair.
Example	<pre> <param-list> <param name="clockHours" value="0-23"/> <param name="useLocalTimeZone" value="true"/> </param-list> </pre>

<param>

A parameter used to set the default configuration.

Item	Description						
Syntax	<pre> <param name="textual name" value="value"/> </pre>						
Attributes	<table> <tr> <td>name</td> <td>Req</td> <td>The textual param name.</td> </tr> <tr> <td>value</td> <td>Req</td> <td>The param value.</td> </tr> </table>	name	Req	The textual param name.	value	Req	The param value.
name	Req	The textual param name.					
value	Req	The param value.					
Data	None						
Parent Elements	1. <param-list>						
Child Elements	None						
Example	<pre> <param-list> <param name="clockHours" value="0-23"/> <param name="useLocalTimeZone" value="true"/> </param-list> </pre>						

<auth-acr>

An authentication access control rule (auth-acr) forces user authentication and controls access by the user's roles. It is composed of one or more role constraints, each of which identifies the name of a role that is assigned at authentication time. Each role constraint may also specify a role-class, which is the Java class implementing the role.

The auth-acr grants access if:

Access Control Policy Tag Reference

1. the user is authenticated
2. and at least one role–constraint that grants access matches a role associated with the authenticated user
3. and no role–constraints that deny access match the authenticated user

A single `auth–acr` enables you to grant and deny access to any number of authenticated users by role name and/or role class. When a user is authenticated the Cams authentication configured for the protected resource assigns one or more roles to the user. Let's clarify via the following two scenarios:

Scenario 1: A user authenticates as johndoe

In this scenario, a user logs in with a user name/password. The following identities (roles) are assigned by the Cams authentication system:

1. user: johndoe
2. role: employee
3. role: engineering

Don't let the user designation fool you, johndoe is also a role. He has logged in using his personal identity. Now consider Scenario 2:

Scenario 2: A user authenticates as administrator

In this scenario, a shared user name administrator is used by multiple individuals. After logging in a user has the following identity:

1. user: administrator

The authenticated user has the administrator role, but there's no telling who the real user might be. So, when formulating an `auth–acr`, a role–name corresponds to either a user or a role. In other words, a user name is also a role name, but a role name is not necessarily a user name.

Item	Description
Syntax	<pre><auth–acr id="authentication rule identifier" debug="true false"> <role–constraint> ... </role–constraint> ... </auth–acr></pre>
Attributes	<p><code>id</code> Opt The <code>auth–acr</code> id is used to improve access control policy readability and for context–sensitive error and debugging messages. Use unique, descriptive ids to improve message comprehension and debugging.</p> <p><code>debug</code> Opt Activate debug for the authentication constraint. The default is false.</p>
Data	None
Parent Elements	1. <acr–lib>
Child Elements	<p><role–constraint> Opt A condition placed on a role that the user must have (or not have). Any number of role–constraints may be added to an <code>auth–acr</code>.</p>
Example	<pre><!-- The following auth–acr is true only if: 1) the user is authenticated 2) and the user has the "administrator" role 3) and the user has the "employee" role 4) and the user does not have the "contractor" role, implemented by a specific Java class An auth–acr will evaluate to "true" ONLY if: 1) at least one role–constraint matches accessGranted="true" and no role–constraints deny access 2) the auth–acr is completely empty of role–constraints --> <auth–acr ID="Employee Role Rule"> <role–constraint> <role–name>administrator</role–name> </role–constraint></pre>

Access Control Policy Tag Reference

```

<role-constraint>
  <role-name>employee</role-name>
</role-constraint>

<role-constraint accessGranted="false">
  <role-name>contractor</role-name>
  <role-class>
    com.cafesoft.security.engine.auth.CSRolePrincipal
  </role-class>
</role-constraint>
</auth-acr>

```

<role-constraint>

A role-constraint grants or denies an authenticated user access by virtue of the identities (roles) associated at login time. A role-constraint applies to an authenticated user if:

1. it contains only a <role-name> and the user has that role
2. if contains a <role-name> and a <role-class> and the user has that role implemented by the specified Java class

A role-constraint grants access if:

1. it applies to the authenticated user
2. the role-constraint implicitly or explicitly has attribute: accessGranted="true"

If a role-constraint applies to the user and denies access, then the auth-acr as a whole will immediately deny access. To optimize performance, place any role-constraints that deny access at the beginning of the auth-acr.

Item	Description						
Syntax	<pre> <role-constraint accessGranted="true false"> <role-name> ... </role-name> <role-class> ... </role-class> </role-constraint> </pre>						
Attributes	<table style="width: 100%; border: none;"> <tr> <td style="width: 15%;"><code>accessGranted</code></td> <td style="width: 10%; text-align: center;">Opt</td> <td>Indicates whether a matching role-constraint grants or denies access. If not specified, the role-constraint grants access.</td> </tr> </table>	<code>accessGranted</code>	Opt	Indicates whether a matching role-constraint grants or denies access. If not specified, the role-constraint grants access.			
<code>accessGranted</code>	Opt	Indicates whether a matching role-constraint grants or denies access. If not specified, the role-constraint grants access.					
Data	None						
Parent Elements	1. <auth-acr>						
Child Elements	<table style="width: 100%; border: none;"> <tr> <td style="width: 15%;"><role-name></td> <td style="width: 10%; text-align: center;">Req</td> <td>the name of a role</td> </tr> <tr> <td><role-class></td> <td style="text-align: center;">Opt</td> <td>the fully-qualified name of the Java class implementing the role. (See documentation on the LoginModules used for authentication to see the role classes they use).</td> </tr> </table>	<role-name>	Req	the name of a role	<role-class>	Opt	the fully-qualified name of the Java class implementing the role. (See documentation on the LoginModules used for authentication to see the role classes they use).
<role-name>	Req	the name of a role					
<role-class>	Opt	the fully-qualified name of the Java class implementing the role. (See documentation on the LoginModules used for authentication to see the role classes they use).					
Example	<pre> <!-- Denies access to CSUserPrincipal "richard" --> <role-constraint accessGranted="false"> <role-name>richard</role-name> <role-class>com.cafesoft.cams.auth.CSUserPrincipal</role-class> </role-constraint> </pre>						

<role-name>

An role-name identifies the name of a role possibly assigned to an authenticated user. It is one of two possible criteria (<role-class> is the other) that indicates whether or not a role-constraint applies to the access request.

Item	Description
Syntax	<code><role-name>role name</role-name></code>
Attributes	None
Data	the name of a role or user name assigned to the user when authenticated.
Parent Elements	1. <role-constraint>

Child Elements	None
Example	<pre><!-- Denies access to "richard" --> <role-constraint accessGranted="false"> <role-name>richard</role-name> </role-constraint></pre>

<role-class>

An `role-class` identifies the Java class that implements a role possibly assigned to an authenticated user. It is one of two possible criteria (`role-name` is the other) that indicates whether or not a `role-constraint` applies to the access request. If specified, it must be used in conjunction with `role-name` and the `role-constraint` applies only an authenticated user has the `role-name` that is implemented by the specified `role-class`.

To determine the classes assigned to authenticated users, you'll need to know the Login Modules that were used to authenticate the user and the classes they use to add principals (roles). For more information, see the [Login Configuration](#) document.

Item	Description
Syntax	<code><role-class>fully.qualified.JavaClassName</role-class></code>
Attributes	None
Data	<p>The name of a role class associated with a <code>role-name</code> at login time. For example:</p> <ul style="list-style-type: none"> • <code>com.cafesoft.security.engine.auth.CSUserPrincipal</code> • <code>com.cafesoft.security.engine.auth.CSRolePrincipal</code> • <code>com.cafesoft.security.engine.auth.CSDomainPrincipal</code>
Parent Elements	1. <role-constraint>
Child Elements	None
Example	<pre><!-- Denies access to CSUserPrincipal "richard" --> <role-constraint accessGranted="false"> <role-name>richard</role-name> <role-class>com.cafesoft.cams.auth.CSUserPrincipal</role-class> </role-constraint></pre>

<host-acr>

The `host-acr` grants or denies access to a user based on the host name and/or IP address of the computer from which he's attempting to access a protected resource. This rule does not require that the user be authenticated, but does require that the remote host and/or remote address be known.

WARNING: It is not always possible to resolve a remote host's name using the Internet's Domain Name Service (DNS). Even if a host name can be resolved, performance is sometimes poor, which can adversely effect performance. We recommend that you use host name patterns only in a controlled environments (like an intranet), where you're likely to have more control over DNS configuration and performance.

Item	Description						
Syntax	<pre><host-acr id="access control rule identifier" debug="true false"> ... </host-acr></pre>						
Attributes	<table> <tr> <td><code>id</code></td> <td>Opt</td> <td>Uniquely identifies this rule so it can be referenced from permissions and access control rule expressions. It is required if used directly within an access control rule library, but optional if used within an access control rule expression.</td> </tr> <tr> <td><code>debug</code></td> <td>Opt</td> <td>Activate debug for the host access control rule. The default is false.</td> </tr> </table>	<code>id</code>	Opt	Uniquely identifies this rule so it can be referenced from permissions and access control rule expressions. It is required if used directly within an access control rule library, but optional if used within an access control rule expression.	<code>debug</code>	Opt	Activate debug for the host access control rule. The default is false.
<code>id</code>	Opt	Uniquely identifies this rule so it can be referenced from permissions and access control rule expressions. It is required if used directly within an access control rule library, but optional if used within an access control rule expression.					
<code>debug</code>	Opt	Activate debug for the host access control rule. The default is false.					
Data	None						
Parent Elements	<ol style="list-style-type: none"> 1. <acr-lib> 2. <acr> 						

Access Control Policy Tag Reference

Child Elements	<p><code><allow-host></code> Opt Lists granted remote hosts by name, 0 or more are permitted.</p> <p><code><deny-host></code> Opt Lists denied remote hosts by name, 0 or more are permitted.</p> <p><code><allow-address></code> Opt Lists accepted remote hosts by IP address, 0 or more are permitted.</p> <p><code><deny-address></code> Opt Lists denied remote hosts by IP address, 0 or more are permitted.</p>
Example	<pre> <!-- The following example shows a host-acr which is true only if: 1) the remote host has a name that ends with ".mycompany.com" or ".partner.com" 2) but the remote host does not include: "badcompany.com" 3) or the remote host has IP address: 208.175.100.5 4) or the remote host has an IP address that matches pattern: 192.168.0.* 5) but the remote host does not have IP address: 192.168.0.1 --> <host-acr ID="Extranet Partners Rule"> <allow-host> <host>*.mycompany.com</host> <host>*.partner.com</host> </allow-host> <deny-host> <host>*badcompany.com</host> </deny-host> <allow-address> <address>208.175.100.5</address> <address>192.168.0.*</address> </allow-address> <deny-address> <address>192.168.0.1</address> </deny-address> </host-acr> <!-- A more typical example might grant permission to remote hosts based on IP address --> <host-acr ID="Simpler Extranet Partners Rule"> <allow-address> <address>208.175.100.5</address> <address>192.168.0.*</address> </allow-address> </host-acr> </pre>

A host-acr will evaluate to true ONLY if:

1. the it contains no constraints whatsoever (i.e. it's empty)
2. a remote host matches an accept pattern, but does not match a deny pattern

<allow-host>

Defines one or more remote host name patterns. If a remote host attempts to access a resource protected by the security constraint associated with this rule and it's host name matches a pattern within this constraint, then that remote host is eligible to have access granted. Access will be granted by the enclosing rule if the host does not match a deny pattern. If the remote host name is not available, then it cannot match this pattern so this constraint does not apply.

WARNING: It is not always possible to resolve a remote host's name using the Internet's Domain Name Service (DNS). Even if a host name can be resolved, performance is sometimes poor, which can adversely effect performance. We recommend that you use host name patterns only in a controlled environments (like an intranet), where you're likely to have more control over DNS configuration and performance.

Item	Description
------	-------------

Access Control Policy Tag Reference

Syntax	<pre><allow-host> <host> ... </host> ... </allow-host></pre>
Attributes	None
Data	None
Parent Elements	1. <host-act>
Child Elements	<host> Opt The pattern for a remote host name with support for regular expression matching. Zero or more are required. See the Regular Expressions document for more information on pattern matching.
Example	<pre><!-- Allow connections from know sites --> <allow-host> <host>*.mycompany.com</host> <host>*.partner.com</host> </allow-host></pre>

<deny-host>

Defines one or more remote host name patterns. If a remote host attempts to access a resource protected by the security-constraint associated with this rule and it's host name matches a pattern within this constraint, then that remote host is immediately denied access. If the remote host name is not available, then it cannot match this pattern so this constraint does not apply.

WARNING: It is not always possible to resolve a remote host's name using the Internet's Domain Name Service (DNS). Even if a host name can be resolved, performance is sometimes poor, which can adversely effect performance. We recommend that you use host name patterns only in a controlled environments (like an intranet), where you're likely to have more control over DNS configuration and performance.

Item	Description
Syntax	<pre><deny-host> <host> ... </host> ... </deny-host></pre>
Attributes	None
Data	None
Parent Elements	1. <host-act>
Child Elements	<host> Opt The pattern for a remote host name with support for regular expression matching. Zero or more are required. See the Regular Expressions document for more information on pattern matching.
Example	<pre><!-- Deny all connections from Bad Company --> <deny-host> <host>*badcompany.com</host> </deny-host></pre>

<host>

The pattern for a remote host name with support for regular expression matching. Zero or more are required. See the [Regular Expressions](#) document for more information on pattern matching.

Item	Description
Syntax	<pre><host>host name or host pattern</host></pre>
Attributes	None
Data	host name or host pattern
Parent Elements	1. <allow-host> 2. <deny-host>
Child Elements	None

Access Control Policy Tag Reference

Example	<pre> <!-- Match only a host with DNS name "www1.mycompany.com" --> <host>www1.mycompany.com</host> <!-- Match all hosts with names that end with "mycompany.com" --> <host>*mycompany.com</host> <!-- Match all hosts with names containing "mycompany" --> <host>*mycompany*</host> </pre>
---------	--

<allow-address>

Defines one or more remote host IP address patterns. If a remote host attempts to access a resource protected by the security constraint associated with this rule and its host IP address matches a pattern within this constraint, then that remote host is eligible to have access granted. Access will be granted by the enclosing rule only if the host does not match a deny pattern. If the remote host IP address is not available, then it cannot match this pattern so this constraint does not apply.

Item	Description
Syntax	<pre> <allow-address> <address> ... </address> ... </allow-address> </pre>
Attributes	None
Data	None
Parent Elements	1. <host-acr>
Child Elements	<p><address> Opt The pattern for a remote host IP address with support for regular expression matching. Zero or more are required. See the Regular Expressions document for more information on pattern matching.</p>
Example	<pre> <!-- Grant access from specific IP addresses --> <allow-address> <address>208.175.100.5</address> <address>192.168.0.*</address> </allow-address> </pre>

<deny-address>

Defines one or more remote host IP address patterns. If a remote host attempts to access a resource protected by the security constraint associated with this rule and its host IP address matches a pattern within this constraint, then that remote host is immediately denied access. If the remote host name is not available, then it cannot match this pattern so this constraint does not apply.

Item	Description
Syntax	<pre> <deny-address> <address> ... </address> ... </deny-address> </pre>
Attributes	None
Data	None
Parent Elements	1. <host-acr>
Child Elements	<p><address> Opt The pattern for a remote host IP address with support for regular expression matching. Zero or more are required. See the Regular Expressions document for more information on pattern matching.</p>
Example	<pre> <!-- Deny access to specific IP addresses --> <deny-address> <address>192.168.0.1</address> </deny-address> </pre>

<address>

The pattern for a remote host IP address with support for regular expression matching. Zero or more are required. See the [Regular Expressions](#) document for more information on pattern matching.

Item	Description
Syntax	<address>host IP address or host IP address pattern</address>
Attributes	None
Data	host IP address or host IP address pattern
Parent Elements	1. <allow-address> 2. <deny-address>
Child Elements	None
Example	<pre><!-- Match only a host that has IP address "192.168.0.1" --> <address>192.168.0.1</address> <!-- Match all hosts with IP addresses that start with "192.168.0." --> <address>192.168.0.*</address></pre>

<acr>

Creates a custom access control rule by combining other elements attributes. Once defined, access control rule expressions can be referenced directly from permissions or nested within other access control rule expression.

Item	Description															
Syntax	<pre><acr id="access control rule identifier" debug="true false"> ... </acr></pre>															
Attributes	<table border="0"> <tr> <td>id</td> <td>Req</td> <td>Uniquely identifies this rule so it can be referenced from permissions and other access control rule expressions. It is required if used directly within an access control rule library, but optional if used within another access control rule expression.</td> </tr> <tr> <td>debug</td> <td>Opt</td> <td>Activate debug for the authentication constraint (true or false).</td> </tr> </table>	id	Req	Uniquely identifies this rule so it can be referenced from permissions and other access control rule expressions. It is required if used directly within an access control rule library, but optional if used within another access control rule expression.	debug	Opt	Activate debug for the authentication constraint (true or false).									
id	Req	Uniquely identifies this rule so it can be referenced from permissions and other access control rule expressions. It is required if used directly within an access control rule library, but optional if used within another access control rule expression.														
debug	Opt	Activate debug for the authentication constraint (true or false).														
Data	None															
Parent Elements	1. <acr-lib>															
Child Elements	<table border="0"> <tr> <td><granted></td> <td>Opt</td> <td>Intrinsic Cams operator, unconditionally grant access to a resource.</td> </tr> <tr> <td><denied></td> <td>Opt</td> <td>Intrinsic Cams operator, unconditionally deny access to a resource.</td> </tr> <tr> <td><confidential></td> <td>Opt</td> <td>Intrinsic Cams operator, require a confidential (typically SSL/TLS) connection.</td> </tr> <tr> <td><and> <or> <not></td> <td>Opt</td> <td>Boolean operators for use between access control rules.</td> </tr> <tr> <td><acr-ref></td> <td>Opt</td> <td>A reference to an existing access control rule within the enclosing security domain's access control rule library. The referenced access control rule must already have been defined within the acr-lib or an "Undefined Access Control Rule" will be reported.</td> </tr> </table>	<granted>	Opt	Intrinsic Cams operator, unconditionally grant access to a resource.	<denied>	Opt	Intrinsic Cams operator, unconditionally deny access to a resource.	<confidential>	Opt	Intrinsic Cams operator, require a confidential (typically SSL/TLS) connection.	<and> <or> <not>	Opt	Boolean operators for use between access control rules.	<acr-ref>	Opt	A reference to an existing access control rule within the enclosing security domain's access control rule library. The referenced access control rule must already have been defined within the acr-lib or an "Undefined Access Control Rule" will be reported.
<granted>	Opt	Intrinsic Cams operator, unconditionally grant access to a resource.														
<denied>	Opt	Intrinsic Cams operator, unconditionally deny access to a resource.														
<confidential>	Opt	Intrinsic Cams operator, require a confidential (typically SSL/TLS) connection.														
<and> <or> <not>	Opt	Boolean operators for use between access control rules.														
<acr-ref>	Opt	A reference to an existing access control rule within the enclosing security domain's access control rule library. The referenced access control rule must already have been defined within the acr-lib or an "Undefined Access Control Rule" will be reported.														

Access Control Policy Tag Reference

	<p><code><auth-acr></code> Opt Requires user authentication and controls access based on the user's role(s).</p> <p><code><host-acr></code> Opt Grants or denies access to a user based on the host from which he's attempting to access a protected resource.</p> <p><code>[custom-acr]</code> Opt A custom access control rule of a type declared by <code>acr-type</code> and embedded in this rule. See acr-type and acr-persistence-manager.</p>
Example	<pre> <!-- Require: 1. the host to be on the ACME Local Area Network 2. the user to be authenticated with the "manager" role 3. a confidential (SSL) connection for privacy --> <acr id="ACME Manager Resources Rule"> <acr-ref id="ACME LAN Rule"> </acr-ref> <acr-ref id="ACME Management Rule"> </acr-ref> <confidential/> </acr> </pre>

<granted> <denied> <confidential>

Cams intrinsic access control rules that are available to all security domains. These access control rules can be referenced by their identifiers: granted, denied, and confidential. They can also be embedded as XML elements anywhere within an access control rule expression (`<acr ...>`).

Item	Description
Syntax	<pre> <granted/> <denied/> <confidential/> </pre>
Attributes	None
Data	None
Parent Elements	<ol style="list-style-type: none"> 1. <code><acr></code> 2. <code><permission></code>
Child Elements	None
Example	<pre> <!-- Grant GET access by all to "public" --> <permission desc="Grant access to public" actions="GET"> <resource-pattern id="http://localhost:8080/public/*"/> <acr-ref id="granted"/> </permission> <!-- Deny GET access by all to "private" --> <permission desc="Deny access to private" actions="GET"> <resource-pattern id="http://localhost:8080/private/*"/> <acr-ref id="denied"/> </permission> <!-- Require a non-confidential connection --> <acr id="non-confidential"> <not/> <confidential/> </acr> </pre>

<and> <or> <not>

Boolean operators for use between access control rule invocations and available to all security domains.

Item	Description
------	-------------

Access Control Policy Tag Reference

Syntax	condition <and/> condition condition <or/> condition </not> condition
Attributes	None
Data	None
Parent Elements	1. <acr>
Child Elements	None
Example	<pre> <!-- Match 1. the host to be on the ACME Local Area Network 2. the user to be authenticated with the "manager" role or "executive" role 3. a confidential (SSL) connection for privacy --> <acr id="ACME Manager Resources Rule"> <acr-ref id="ACME LAN Rule"> <and/> <acr-ref id="ACME Management Rule"> <or/> <acr-ref id="ACME Executive Rule"> <and/> <confidential/> </acr> </pre>

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Login Configuration Tag Reference

A security domain's login configuration declares the login modules, callback handlers, and login parameters required for user authentication. This document contains reference information for each of the tags that can be used within a Cams login-config.xml file. The following table shows the file structure and provides links to each of the possible elements.

Tag Name	Instances	Description
login-config	1	declares the login configuration
[login-config-entry]	1 ... N	a collection of login module entrys
[login-module-entry]	1 ... N	register a login module class and provides it's configuration options
options	0 ... 1	encloses a list of configuration options for a login module
[option]	0 ... N	provides an initialization/configuration parameter as a generic name/value pair
callback-handler	1	registers a callback handler class used for authentication to the given login-config-entry
login-parameters	0 ... N	an optional collection of login-parameter elements
[login-parameter]	0 ... N	a name/value pair of contextual login information

<login-config>

The <login-config> element is the top-level element for a security domain's login configuration.

Item	Description
Syntax	<pre><login-config> <login-config-entry ... > ... </login-config-entry> ... </login-config></pre>
Attributes	None
Data	None
Parent Elements	None
Child Elements	<p><login-config-entry> Req Declares a single named collection of login-module-entrys that will be used to authenticate a user. Typically, each login-config-entry is setup to handle a set of client applications, depending on their authentication needs.</p> <p>The JAAS-compatible login modules associated with each login-module-entry will be executed in the order in which the occur within this element.</p>
Example	<pre><login-config> <!-- HTTP login configurations --> <login-config-entry name="http"> ... </login-config-entry> ... </login-config></pre>

<login-config-entry>

Declares a single named collection of login-module-entrys that will be used to authenticate a user. Typically, each login-config-entry is setup to handle a set of client applications, depending on their authentication needs. Multiple login-config-entrys can be declared to

Login Configuration Tag Reference

handle the requirements of different client applications.

The JAAS-compatible LoginModules associated with each login-module-entry will be executed in the order in which they occur within this element.

Item	Description									
Syntax	<pre><login-config-entry name="authentication type"> <login-module-entry ... > ... </login-module-entry> ... <callback-handler ... /> <login-parameters> ... </login-parameters> </login-config-entry></pre>									
Attributes	<table> <tr> <td>name</td> <td>Req</td> <td>An identifier, usually indicating the type of authentication configuration contained within the login config entry. For example, use http for HTTP resources or cams agents.</td> </tr> </table>	name	Req	An identifier, usually indicating the type of authentication configuration contained within the login config entry. For example, use http for HTTP resources or cams agents.						
name	Req	An identifier, usually indicating the type of authentication configuration contained within the login config entry. For example, use http for HTTP resources or cams agents.								
Data	None									
Parent Elements	1. <login-config>									
Child Elements	<table> <tr> <td><login-module-entry></td> <td>Req</td> <td>A type-specific resource and rules for controlling access.</td> </tr> <tr> <td><callback-handler></td> <td>Req</td> <td>Registers a callback handler class for use with authentication.</td> </tr> <tr> <td><login-parameters></td> <td>Opt</td> <td>An optional collection of login parameter elements.</td> </tr> </table>	<login-module-entry>	Req	A type-specific resource and rules for controlling access.	<callback-handler>	Req	Registers a callback handler class for use with authentication.	<login-parameters>	Opt	An optional collection of login parameter elements.
<login-module-entry>	Req	A type-specific resource and rules for controlling access.								
<callback-handler>	Req	Registers a callback handler class for use with authentication.								
<login-parameters>	Opt	An optional collection of login parameter elements.								
Example	<pre><!-- HTTP login configurations --> <login-config-entry name="http"> <!-- Cams native XML user repository --> <login-module-entry className=\ "com.cafesoft.security.engine.auth.login.module.XmlLoginModule" flag="REQUIRED"> <options> <option name="debug" value="false"/> <option name="serviceId" value="cams-user-repository"/> </options> </login-module-entry> <!-- Register the username password callback handler --> <callback-handler className=\ "com.cafesoft.security.engine.auth.callback.NamePasswordCallbackHandler" /> <!-- Specify the default login page --> <login-parameters> <login-parameter name="camsLoginUrl" value="\\${http.resource.base.id}/login/login.jsp"/> </login-parameters> </login-config-entry></pre>									

<login-module-entry>

Registers a login module class and provides its configuration options. One to many login-module-entries can be declared and will be executed in order in accordance with the flag value specified.

Item	Description
Syntax	<pre><login-module-entry className="fully.qualified.java.Classname" flag="REQUIRED SUFFICIENT REQUISITE OPTIONAL"> <options> ... </options></pre>

Login Configuration Tag Reference

	</login-module-entry>		
Attributes	className	Req	The login module's fully qualified Java class name.
	flag	Req	REQUIRED, SUFFICIENT, REQUISITE, OPTIONAL
Data	None		
Parent Elements	1. <login-config-entry>		
Child Elements	<options>	Opt	A list of configuration options for a login module.
Example	<pre><!-- Cams native XML user repository --> <login-module-entry className=\ "com.cafesoft.security.engine.auth.login.module.XmlLoginModule" flag="REQUIRED"> <options> <option name="debug" value="false"/> <option name="serviceId" value="cams-user-repository"/> </options> </login-module-entry></pre>		

<options>

A list of configuration options for a login module.

Item	Description
Syntax	<pre><options> <option ... /> ... </options></pre>
Attributes	None
Data	None
Parent Elements	1. <login-module-entry>
Child Elements	<option> Opt An initialization/configuration parameter as a generic name/value pair.
Example	<pre><options> <option name="debug" value="false"/> <option name="serviceId" value="cams-user-repository"/> </options></pre>

<option>

A list of configuration options for a login module.

Item	Description
Syntax	<option name="service identifier" value="cams-user-repository"/>
Attributes	name Req An initialization/configuration parameter name.
	value Req An initialization/configuration parameter value.
Data	None
Parent Elements	1. <options>
Child Elements	None
Example	<pre><options> <option name="debug" value="false"/> <option name="serviceId" value="cams-user-repository"/> </options></pre>

<callback-handler>

Registers a Java callback handler class for use with authentication. The callback handler is specific to the authentication client. For example, an HTTP Web interface will use a different callback handler from a Java application to garner username and password information.

Item	Description			
Syntax	<code><callback-handler className="fully.qualified.JavaClassName"/></code>			
Attributes	<table> <tr> <td>className</td> <td>Req</td> <td>The fully qualified name of the Java class that implements the callback handler.</td> </tr> </table>	className	Req	The fully qualified name of the Java class that implements the callback handler.
className	Req	The fully qualified name of the Java class that implements the callback handler.		
Data	None			
Parent Elements	1. <login-config-entry>			
Child Elements	None			
Example	<pre> <!-- Register the username password callback handler --> <callback-handler className=\ "com.cafesoft.security.engine.auth.callback.NamePasswordCallbackHandler " /> </pre>			

<login-parameters>

A collection of login parameter elements. The login parameters can have 0 to many login-parameter elements.

Item	Description			
Syntax	<pre> <login-parameters> <login-paramter ... /> ... </login-parameters> </pre>			
Attributes	None			
Data	None			
Parent Elements	1. <login-config-entry>			
Child Elements	<table> <tr> <td><login-parameter></td> <td>Opt</td> <td>A name/value pair of contextual login information.</td> </tr> </table>	<login-parameter>	Opt	A name/value pair of contextual login information.
<login-parameter>	Opt	A name/value pair of contextual login information.		
Example	<pre> <!-- Specify the default login page --> <login-parameters> <login-parameter name="camsLoginUrl" value="\${http.resource.base.id}/login/login.jsp"/> </login-parameters> </pre>			

<login-parameter>

A collection of login-parameter elements. The login-parameters can have 0 to many login-parameter elements.

Item	Description						
Syntax	<code><login-parameter name="textual name" value="value"/></code>						
Attributes	<table> <tr> <td>name</td> <td>Req</td> <td>A login parameter name.</td> </tr> <tr> <td>value</td> <td>Req</td> <td>A login parameter value.</td> </tr> </table>	name	Req	A login parameter name.	value	Req	A login parameter value.
name	Req	A login parameter name.					
value	Req	A login parameter value.					
Data	None						
Parent Elements	1. <login-parameters>						
Child Elements	None						
Example	<pre> <!-- Specify the default login page --> <login-parameters> <login-parameter name="camsLoginUrl" value="\${http.resource.base.id}/login/login.jsp"/> </login-parameters> </pre>						

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Security Domain Tag Reference

The security domain's service configuration is defined by the Cams security-domain.xml file. This document contains reference information for each of the tags that can be used within security-domain.xml. The following table shows the file structure with links to each of the possible elements.

Tag Name	Instances	Description
security-domain	1	declares the security domain
var-list	0 ... 1	an optional list of Cams variables available within all security-domain specific configuration files
[var]	0 ... N	provides an initialization/configuration parameter as a generic name/value pair
logger	1	logs debug, info, warning, error, and fatal messages to a security domain-specific log
auth-service	1	authenticates users and creates new sessions
param-list	0 ... 1	a list of initialization/configuration parameters
[param]	0 ... N	an initialization/configuration parameter
login-config-factory	1	loads and initializes the login configuration
param-list	0 ... 1	a list of initialization/configuration parameters
[param]	0 ... N	an initialization/configuration parameter
auth-pipeline	1	processes authentication requests issued by a qualified Cams agent
param-list	0 ... 1	a list of initialization/configuration parameters
[param]	0 ... N	an initialization/configuration parameter
[auth-value]	0 ... N	a single request processing node within the authentication pipeline
param-list	0 ... 1	a list of initialization/configuration parameters
[param]	0 ... N	an initialization/configuration parameter
access-control-service	1	controls access to the resources protected by a security domain
param-list	0 ... 1	a list of initialization/configuration parameters
[param]	0 ... N	an initialization/configuration parameter
access-control-policy-factory	1	loads and initializes the access control policy for security domain's resources
param-list	0 ... 1	a list of initialization/configuration parameters
[param]	0 ... N	an initialization/configuration parameter
access-control-pipeline	1	processes access control requests issued by a qualified Cams agent
param-list	0 ... 1	a list of initialization/configuration parameters
[param]	0 ... N	an initialization/configuration parameter

Security Domain Tag Reference

<u>[access-control-value]</u>	0 ... N	a single request processing node within an access control pipeline
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters
<u>[param]</u>	0 ... N	an initialization/configuration parameter
<u>session-manager-service</u>	1	manages the sessions for authenticated users
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters
<u>[param]</u>	0 ... N	an initialization/configuration parameter
<u>[session-event-handler]</u>	0 ... N	registers a session event handler with the session manager
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters
<u>[param]</u>	0 ... N	an initialization/configuration parameter
<u>session-access-service</u>	1	enables session information to be queried by qualified Cams agents
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters
<u>[param]</u>	0 ... N	an initialization/configuration parameter
<u>session-access-pipeline</u>	1	processes session access requests
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters
<u>[param]</u>	0 ... N	an initialization/configuration parameter
<u>[session-access-valve]</u>	0 ... N	a single request processing node within a session access pipeline
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters
<u>[param]</u>	0 ... N	an initialization/configuration parameter
<u>session-control-service</u>	1	enables sessions to be closed, touched, updated, etc. by qualified Cams agents
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters
<u>[param]</u>	0 ... N	an initialization/configuration parameter
<u>session-control-pipeline</u>	1	processes session control requests
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters
<u>[param]</u>	0 ... N	an initialization/configuration parameter
<u>[session-control-valve]</u>	0 ... N	a single request processing node within a session control pipeline
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters
<u>[param]</u>	0 ... N	an initialization/configuration parameter
<u>service-manager</u>	1	provides for management of security domain-wide services
<u>param-list</u>	0 ... 1	a list of initialization/configuration parameters

Security Domain Tag Reference

[param]	0 ... N	an initialization/configuration parameter
[service]	0 ... N	creates and registers a service implementation with the service manager
service-type	1	declares the type of service being registered
service-class	1	declares the implementation of the service
param-list	1	a list of initialization/configuration parameters
[param]	0 ... N	an initialization/configuration parameter

<security-domain>

The top-level element used to define services within a Cams security domain.

Item	Description																								
Syntax	<pre><security-domain debug="true false"> ... </security-domain></pre>																								
Attributes	<table border="0"> <tr> <td style="padding-right: 20px;">debug</td> <td style="padding-right: 20px;">Opt</td> <td>Activate debug for the security domain. The default is false.</td> </tr> </table>	debug	Opt	Activate debug for the security domain. The default is false.																					
debug	Opt	Activate debug for the security domain. The default is false.																							
Data	None																								
Parent Elements	None																								
Child Elements	<table border="0"> <tr> <td style="padding-right: 20px;"><var-list></td> <td style="padding-right: 20px;">Opt</td> <td>List of parameters available within all security domain specific configuration files.</td> </tr> <tr> <td><logger></td> <td>Req</td> <td>Logs debug, info, warning, error, and fatal messages to a security domain-specific log.</td> </tr> <tr> <td><auth-service></td> <td>Req</td> <td>Authenticates users and creates new sessions.</td> </tr> <tr> <td><access-control-service></td> <td>Req</td> <td>Controls access to the resources protected by a security domain.</td> </tr> <tr> <td><session-manager-service></td> <td>Req</td> <td>Manages the sessions for authenticated users.</td> </tr> <tr> <td><session-access-service></td> <td>Req</td> <td>Enables session information to be queried by qualified Cams agents.</td> </tr> <tr> <td><session-control-service></td> <td>Req</td> <td>Enables sessions to be closed, touched, updated, etc. by qualified Cams agents.</td> </tr> <tr> <td><service-manager></td> <td>Opt</td> <td>Provides optional security domain-wide services.</td> </tr> </table>	<var-list>	Opt	List of parameters available within all security domain specific configuration files.	<logger>	Req	Logs debug, info, warning, error, and fatal messages to a security domain-specific log.	<auth-service>	Req	Authenticates users and creates new sessions.	<access-control-service>	Req	Controls access to the resources protected by a security domain.	<session-manager-service>	Req	Manages the sessions for authenticated users.	<session-access-service>	Req	Enables session information to be queried by qualified Cams agents.	<session-control-service>	Req	Enables sessions to be closed, touched, updated, etc. by qualified Cams agents.	<service-manager>	Opt	Provides optional security domain-wide services.
<var-list>	Opt	List of parameters available within all security domain specific configuration files.																							
<logger>	Req	Logs debug, info, warning, error, and fatal messages to a security domain-specific log.																							
<auth-service>	Req	Authenticates users and creates new sessions.																							
<access-control-service>	Req	Controls access to the resources protected by a security domain.																							
<session-manager-service>	Req	Manages the sessions for authenticated users.																							
<session-access-service>	Req	Enables session information to be queried by qualified Cams agents.																							
<session-control-service>	Req	Enables sessions to be closed, touched, updated, etc. by qualified Cams agents.																							
<service-manager>	Opt	Provides optional security domain-wide services.																							
Example	<pre><security-domain debug="false"> <!-- Configure the logger --> <logger className="com.cafesoft.cams.log.CamsTraceLogger" filePath="{cams.home}/logs/system-trace.log" debug="false"/> <!-- Configure the authorization service --> <auth-service className="com.cafesoft.security.engine.auth.StandardAuthService" debug="false"> ... </auth-service></pre>																								

Security Domain Tag Reference

```

<!-- Configure the access control service -->
<access-control-service
  className="com.cafesoft.security.engine.access.StandardAccessControlService"
  debug="false">
  ...
</access-control-service>

<!-- Configure the session manager service -->
<session-manager-service
  debug="false">
  ...
</session-manager-service>

<!-- Configure the session access service -->
<session-access-service
  className="com.cafesoft.security.engine.session.access.StandardSessionAccessService"
  debug="false">
  ...
</session-access-service>

<!-- Configure the session control service -->
<session-control-service
  className="com.cafesoft.security.engine.session.control.StandardSessionControlService"
  debug="false">
  ...
</session-control-service>

<!-- Register services accessible within this security domain -->
<service-manager
  className="com.cafesoft.core.service.StandardServiceManager"
  debug="false">
  ...
</service-manager>

</security-domain>

```

<logger>

Logs DEBUG, INFO, WARNING, ERROR, and FATAL messages to a security domain-specific log.

Item	Description															
Syntax	<pre> <logger className="fully.qualified.JavaClassName" filePath="fully qualified file path" append="true false" bufferedIO="true false" bufferSize="integer value" maxSize="string value" maxBackupIndex="integer value" enableConsoleDebug="true false" enableDebugFilter="true false" verbose="true false" debug="true false"/> </pre>															
Attributes	<table border="0"> <tr> <td style="padding-right: 20px;">className</td> <td style="padding-right: 20px;">Req</td> <td>The fully qualified name of the Java logger class that will be instantiated.</td> </tr> <tr> <td>filePath</td> <td>Req</td> <td>The fully qualified path where the log file is written (the value can use forward or back slashes).</td> </tr> <tr> <td>append</td> <td>Opt</td> <td>If set to true new log messages will be appended to the current log file. If the value is false the current log file will be deleted and a new log file will be created. The default value is true.</td> </tr> <tr> <td>bufferedIO</td> <td>Opt</td> <td>If set to true the logger will buffer log messages before writing them to the log file. The default value is true.</td> </tr> <tr> <td>bufferSize</td> <td>Opt</td> <td></td> </tr> </table>	className	Req	The fully qualified name of the Java logger class that will be instantiated.	filePath	Req	The fully qualified path where the log file is written (the value can use forward or back slashes).	append	Opt	If set to true new log messages will be appended to the current log file. If the value is false the current log file will be deleted and a new log file will be created. The default value is true.	bufferedIO	Opt	If set to true the logger will buffer log messages before writing them to the log file. The default value is true.	bufferSize	Opt	
className	Req	The fully qualified name of the Java logger class that will be instantiated.														
filePath	Req	The fully qualified path where the log file is written (the value can use forward or back slashes).														
append	Opt	If set to true new log messages will be appended to the current log file. If the value is false the current log file will be deleted and a new log file will be created. The default value is true.														
bufferedIO	Opt	If set to true the logger will buffer log messages before writing them to the log file. The default value is true.														
bufferSize	Opt															

Security Domain Tag Reference

	<p>Indicates the size of the buffer to fill before writing to the log file. The default value is "4096".</p>
maxSize	<p>Opt Indicates the maximum size the log file is allowed to grow before creating a new logfile. Suffixes KB, MB, and GB are recognized. The default value is "4MB".</p>
maxBackupIndex	<p>Opt The maximum rollover file index. When log files are # rolled over, a numeric index is appended to the name, starting with 1 # and proceeding to this value. The default value is 100.</p>
enableConsoleDebug	<p>Opt If set to true all log statements that are sent to the log file are also sent to the console. The default value is false.</p>
enableDebugFilter	<p>Opt If set to true all log statements that have the level "DEBUG" will be filtered out (not logged). If set to false, all DEBUG-level log statements will be logged.</p>
verbose	<p>Opt If set to true all DEBUG, INFO, WARN, ERROR, FATAL messages logged will contain the following format:</p> <p>[INFO] Sample log message Class Name: com.cafesoft.cams.log.CamsTraceLogger Method Name: info() Line Number: 121 Timestamp: 25 Jul 2002 11:02:36,339</p> <p>If set to false ONLY messages with the WARNING, ERROR, and FATAL message level will use the verbose format, while DEBUG and INFO level messages will use the following format:</p> <p>[INFO] Sample Log Message</p> <p>The default value is false.</p>
debug	<p>Opt If set to true the logger will output diagnostic debug statements to system.err. The default value is false.</p>
Data	None
Parent Elements	1. <security-domain>
Child Elements	None
Example	<pre><logger className="com.cafesoft.cams.log.CamsTraceLogger" filePath="\${cams.home}/logs/system-trace.log" append="true" bufferIO="true" bufferSize="4096" maxSize="1GB" maxBackupIndex="10" enableConsoleDebug="false" enableDebugFilter="false" verbose="false" debug="false"/></pre>

<auth-service>

Implements a security domain's authentication service, which is responsible for validating the identity of a user who accesses protected resources within the security domain.

Item	Description
Syntax	<pre><auth-service className="fully.qualified.JavaClassName" debug="true false"></pre>

Security Domain Tag Reference

	<pre> <param-list> ... </param-list> <login-config-factory ... /> <auth-pipeline ... > ... </auth-pipeline> </auth-service> </pre>									
Attributes	<table> <tr> <td>className</td> <td>Req</td> <td>The fully qualified name of the Java authentication service class that will be instantiated.</td> </tr> <tr> <td>debug</td> <td>Opt</td> <td>Activate debug for this security domain's authentication service. The default value is false.</td> </tr> </table>	className	Req	The fully qualified name of the Java authentication service class that will be instantiated.	debug	Opt	Activate debug for this security domain's authentication service. The default value is false.			
className	Req	The fully qualified name of the Java authentication service class that will be instantiated.								
debug	Opt	Activate debug for this security domain's authentication service. The default value is false.								
Data	None									
Parent Elements	1. <security-domain>									
Child Elements	<table> <tr> <td><param-list></td> <td>Opt</td> <td>A container for a list of initialization/configuration parameters.</td> </tr> <tr> <td><login-config-factory></td> <td>Req</td> <td>Loads and initializes the login configuration.</td> </tr> <tr> <td><auth-pipeline></td> <td>Req</td> <td>Processes authentication requests issued by a qualified Cams agent.</td> </tr> </table>	<param-list>	Opt	A container for a list of initialization/configuration parameters.	<login-config-factory>	Req	Loads and initializes the login configuration.	<auth-pipeline>	Req	Processes authentication requests issued by a qualified Cams agent.
<param-list>	Opt	A container for a list of initialization/configuration parameters.								
<login-config-factory>	Req	Loads and initializes the login configuration.								
<auth-pipeline>	Req	Processes authentication requests issued by a qualified Cams agent.								
Example	<pre> <!-- Configure the authorization service --> <auth-service className="com.cafesoft.security.engine.auth.StandardAuthService" debug="false"> <login-config-factory className="com.cafesoft.security.engine.auth.login.XmlLoginConfigurationFactory" debug="false"> <param-list> <param name="configPath" value="{cams.security-domain.home}/login-config.xml" /> </param-list> </login-config-factory> <auth-pipeline className="com.cafesoft.security.engine.auth.StandardAuthPipeline" debug="false"> <auth-valve className="com.cafesoft.security.engine.auth.valves.LogAuthRequestValve" debug="false"> <param-list> <param name="logPath" value="{cams.home}/logs/system-authentication.log" /> </param-list> </auth-valve> </auth-pipeline> </auth-service> </pre>									

<login-config-factory>

Loads and initializes the security domain's login configuration.

Item	Description
Syntax	<pre> <login-config-factory className="fully.qualified.JavaClassName" params="configPath=fully qualified file path" debug="true false" /> </pre>

Security Domain Tag Reference

Attributes	<p>className Req The fully qualified name of the Java login configuration factory that will be instantiated.</p> <p>debug Opt Activate debug for this security domain's login-config-factory. The default value is false.</p>
Data	None
Parent Elements	1. <auth-service>
Child Elements	<p><param-list> Opt A container for a list of initialization/configuration parameters.</p>
Example	<pre><login-config-factory className="com.cafesoft.security.engine.auth.login.XmlLoginConfigurationFactory" debug="false"> <param-list> <param name="configPath" value="{cams.security-domain.home}/login-config.xml" /> </param-list> </login-config-factory></pre>

<auth-pipeline>

Processes authentication requests issued locally or remotely by a Cams agent. This pipeline implements a chain of responsibility pattern that provides strong control over who can issue authentication requests, and how the response to authentication requests is created.

Item	Description
Syntax	<pre><auth-pipeline className="com.cafesoft.security.engine.auth.StandardAuthPipeline" debug="true false"> <param-list> <param name="parameter name" value="parameter value" /> </param-list> <auth-valve ... /> ... </auth-pipeline></pre>
Attributes	<p>className Req The fully qualified name of the Java authentication pipeline class that will be instantiated.</p> <p>debug Opt Activate debug for this security domain's authentication pipeline. The default value is false.</p>
Data	None
Parent Elements	1. <auth-service>
Child Elements	<p><param-list> Opt A container for a list of initialization/configuration parameters.</p> <p><auth-valve> Opt Represents a single node within an auth-pipeline.</p>
Example	<pre><auth-pipeline className="com.cafesoft.security.engine.auth.StandardAuthPipeline" debug="false"> <auth-valve className="com.cafesoft.security.engine.auth.valves.LogAuthRequestValve" debug="false" /></pre>

Security Domain Tag Reference

```

<param-list>
  <param name="logPath"
    value="{cams.home}/logs/system-authentication.log"/>
</param-list>
</auth-valve>
</auth-pipeline>

```

<auth-valve>

Represents a single node within an authentication pipeline. The valve receives an authentication request and can handle the authentication completely, modify or add to the authentication request, or pass the authentication request to the next auth-valve in the chain.

Item	Description						
Syntax	<pre> <auth-valve className="fully.qualified.JavaClassName" debug="true false"> <param-list> ... </param-list> </auth-valve> </pre>						
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; padding: 5px;">className</td> <td style="width: 10%; padding: 5px; text-align: center;">Req</td> <td style="padding: 5px;">The fully qualified name of the Java authenticatoin valve class that will be instantiated.</td> </tr> <tr> <td style="padding: 5px;">debug</td> <td style="padding: 5px; text-align: center;">Opt</td> <td style="padding: 5px;">Activate debug for this valve node of this security domain. The default value is false.</td> </tr> </table>	className	Req	The fully qualified name of the Java authenticatoin valve class that will be instantiated.	debug	Opt	Activate debug for this valve node of this security domain. The default value is false.
className	Req	The fully qualified name of the Java authenticatoin valve class that will be instantiated.					
debug	Opt	Activate debug for this valve node of this security domain. The default value is false.					
Data	None						
Parent Elements	1. <auth-pipeline>						
Child Elements	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; padding: 5px;"><param-list></td> <td style="width: 10%; padding: 5px; text-align: center;">Opt</td> <td style="padding: 5px;">A container for a list of initialization/configuration parameters.</td> </tr> </table>	<param-list>	Opt	A container for a list of initialization/configuration parameters.			
<param-list>	Opt	A container for a list of initialization/configuration parameters.					
Example	<pre> <auth-valve className="com.cafesoft.security.engine.auth.valves.LogAuthRequestValve" debug="false"> <param-list> <param name="logPath" value="{cams.home}/logs/system-authentication.log"/> </param-list> </auth-valve> </pre>						

<access-control-service>

Specifies the Java class that controls access to the resources protected by a security domain. This element specifies the Java class that implements the access control service.

Item	Description
Syntax	<pre> <access-control-service className="fully.qualified.JavaClassName" debug="true false"> <access-control-policy-factory> ... </access-control-policy-factory> <access-control-pipeline ... > ... </access-control-pipeline> </access-control-service> </pre>

Security Domain Tag Reference

Parent Elements	1. <access-control-service>
Child Elements	<param-list> Opt A container for a list of initialization/configuration parameters.
Example	<pre><access-control-policy-factory className="com.cafesoft.security.engine.access.XmlAccessControlPolicyFactory" debug="false"/></pre>

<access-control-pipeline>

The access control pipeline specifies the Java class that processes access requests issued locally or remotely by a Cams agent. This pipeline is composed of a sequence of access control valves, which handle the request using the chain of responsibility design pattern. This enables each access control valve to handle the request altogether or modulate the request for processing by a subsequent valve.

Item	Description						
Syntax	<pre><access-control-pipeline className="fully.qualified.JavaClassName" debug="true false"/> <access-control-valve> <param-list> ... </param-list> </access-control-valve> ... </access-control-pipeline></pre>						
Attributes	<table> <tr> <td>className</td> <td>Req</td> <td>The fully qualified Java class name of the access control pipeline that will be instantiated.</td> </tr> <tr> <td>debug</td> <td>Opt</td> <td>Activate debug for this security domain's access control pipeline. The default value is false.</td> </tr> </table>	className	Req	The fully qualified Java class name of the access control pipeline that will be instantiated.	debug	Opt	Activate debug for this security domain's access control pipeline. The default value is false.
className	Req	The fully qualified Java class name of the access control pipeline that will be instantiated.					
debug	Opt	Activate debug for this security domain's access control pipeline. The default value is false.					
Data	None						
Parent Elements	1. <auth-service>						
Child Elements	<table> <tr> <td><param-list></td> <td>Opt</td> <td>A container for a list of initialization/configuration parameters.</td> </tr> <tr> <td><access-control-valve></td> <td>Opt</td> <td>Represents a single node within an access control pipeline.</td> </tr> </table>	<param-list>	Opt	A container for a list of initialization/configuration parameters.	<access-control-valve>	Opt	Represents a single node within an access control pipeline.
<param-list>	Opt	A container for a list of initialization/configuration parameters.					
<access-control-valve>	Opt	Represents a single node within an access control pipeline.					
Example	<pre><access-control-pipeline className="com.cafesoft.security.engine.access.StandardAccessControlPipeline" params=" " debug="false"> <access-control-valve className="com.cafesoft.security.engine.access.valves.LogAccessControlRequestValve" debug="false"> <param-list> <param name="logPath" value="{cams.home}/logs/system-access-control.log"/> </param-list> </access-control-valve> </access-control-pipeline></pre>						

<access-control-valve>

Represents a single node within an access control pipeline for handling access requests. The valve receives an access request and can handle the request completely, modify or add to it, or pass the request to the next valve in the chain.

Security Domain Tag Reference

Item	Description						
Syntax	<pre><access-control-valve className="fully.qualified.JavaClassName" debug="true false"/></pre>						
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"><code>className</code></td> <td style="width: 10%; text-align: center;">Req</td> <td>The fully qualified Java class name of the access control valve that will be instantiated.</td> </tr> <tr> <td><code>debug</code></td> <td style="text-align: center;">Opt</td> <td>Activate debug for this valve node of this security domain. The default value is false.</td> </tr> </table>	<code>className</code>	Req	The fully qualified Java class name of the access control valve that will be instantiated.	<code>debug</code>	Opt	Activate debug for this valve node of this security domain. The default value is false.
<code>className</code>	Req	The fully qualified Java class name of the access control valve that will be instantiated.					
<code>debug</code>	Opt	Activate debug for this valve node of this security domain. The default value is false.					
Data	None						
Parent Elements	1. <auth-pipeline>						
Child Elements	<param-list> Opt A container for a list of initialization/configuration parameters.						
Example	<pre><access-control-valve className="com.cafesoft.security.engine.access.valves.LogAccessControlRequestValve" debug="false"> <param-list> <param name="logPath" value="{cams.home}/logs/system-access-control.log"/> </param-list> </access-control-valve></pre>						

<session-manager-service>

Specifies the Java class that manages authenticated user sessions.

Item	Description						
Syntax	<pre><session-manager-service className="fully.qualified.JavaClassName" debug="true false"> <param-list> <param name="maxActiveSessions" value="-1"/> <param name="inactiveSessionTimeout" value="30"/> <param name="sessionCleanupInterval" value="1"/> <param name="sessionIdKey" value="secret-key"/> </param-list> <session-event-handler ... /> </session-manager-service></pre>						
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"><code>className</code></td> <td style="width: 10%; text-align: center;">Req</td> <td>The fully qualified Java class name of the session manager service that will be instantiated.</td> </tr> <tr> <td><code>debug</code></td> <td style="text-align: center;">Opt</td> <td>Activate debug for this security domain's session manager service. The default value is false.</td> </tr> </table>	<code>className</code>	Req	The fully qualified Java class name of the session manager service that will be instantiated.	<code>debug</code>	Opt	Activate debug for this security domain's session manager service. The default value is false.
<code>className</code>	Req	The fully qualified Java class name of the session manager service that will be instantiated.					
<code>debug</code>	Opt	Activate debug for this security domain's session manager service. The default value is false.					
Data	None						
Parent Elements	1. <security-domain>						
Child Elements	<p><param-list> Opt A list of initialization/configuration parameters.</p> <p>For example:</p> <pre><param-list> <param name="maxActiveSessions" value="-1"/> <param name="inactiveSessionTimeout" value="30"/> <param name="sessionCleanupInterval" value="1"/> <param name="sessionIdAlgorithm" value="SHA"/></pre>						

Security Domain Tag Reference

```
<param name="sessionIdIPAddrValidationMask"
value="255.255.255.255"/>
<param name="sessionIdKey" value="secret-key"/>
</param-list>
```

maxActiveSession is the maximum number of concurrent authenticated user sessions that can be managed by the session manager within the enclosing security domain. The value -1 specifies unlimited sessions.

inactiveSessionTimeout is the number of a minutes a user session can be inactive before it will be expired by the session manager. Whener an access control check is done on a session, its "last touched" time is updated.

sessionCleanupInterval is the frequency (in minutes) that the session manager service will check for and expire inactive sessions.

sessionIdAlgorithm is the message digest algorithm to be used for encrypting the Cams session identifier associated with an authenticated user. Valid values include: SHA or MD5. If not specified, SHA is used.

sessionIdIPAddrValidationMask A bit mask used to detect possible hijacked Cams session identifiers by validating the associated IP address.

When the session is created (at authentication time) the IP address of the remote client is associated with the session. For every subsequent access control, session access, and session control request, the IP address associated with the request is validated against the original authentication IP address using the mask to indicate which bits to compare. For example, a mask value of: 255.255.255.255 indicates that the entire IP address must match. A value of 255.255.255.0 validates the first three triplets of the IP address.

When supporting web clients accessing resources via the general internet, it may be necessary to loosen IP address validation to support proxy servers or gateway routers that cause subsequent HTTP requests to arrive via different client IP addresses. For example, some commercial ISPs will route HTTP traffic via one network and HTTPS traffic via another causing subsequent requests arriving at a Cams web agent from the same web browser have different remote client IP addresses.

The default value is: 255.255.255.255, the most restrictive IP address validation.

sessionIdKey is a password used to encode the session id for the security domain. Using a unique value for this key keeps other security domains and malicious users from guessing the parameters used to construct a session identifier.

<session-event-handler> Opt Registers a session event handler with the session manager.

Example	<pre> <!-- Configure the session manager service --> <session-manager-service className="com.cafesoft.security.engine.session.StandardSessionManager"> <param-list> <param name="maxActiveSessions" value="-1"/> <param name="inactiveSessionTimeout" value="30"/> <param name="sessionCleanupInterval" value="1"/> <param name="sessionIdKey" value="secret-key"/> </param-list> <session-event-handler className="com.cafesoft.security.engine.session.SessionManagerEventLogger"> <param-list> <param name="logPath" value="{cams.home}/logs/system-session-manager.log,append=false"/> </param-list> </session-event-handler> </session-manager-service> </pre>
---------	---

<session-event-handler>

Registers a session event handler with the session manager.

Item	Description						
Syntax	<pre> <session-event-handler className="fully.qualified.JavaClassName" debug="false"/> </pre>						
Attributes	<table border="0"> <tr> <td style="padding-right: 20px;">className</td> <td style="padding-right: 20px;">Req</td> <td>The fully qualified Java class name of the access control policy factory that will be instantiated.</td> </tr> <tr> <td>debug</td> <td>Opt</td> <td>Activate debug for this security domain's session event handler. The default value is false.</td> </tr> </table>	className	Req	The fully qualified Java class name of the access control policy factory that will be instantiated.	debug	Opt	Activate debug for this security domain's session event handler. The default value is false.
className	Req	The fully qualified Java class name of the access control policy factory that will be instantiated.					
debug	Opt	Activate debug for this security domain's session event handler. The default value is false.					
Data	None						
Parent Elements	1. <session-manager-service>						
Child Elements	<param-list> Opt A container for a list of initialization/configuration parameters.						
Example	<pre> <session-event-handler className="com.cafesoft.security.engine.session.SessionManagerEventLogger" debug="false"> <param-list> <param name="logPath" value="{cams.home}/logs/system-session-manager.log,append=false"/> </param-list> </session-event-handler> </pre>						

<session-access-service>

Enables session information to be queried by qualified Cams agents.

Item	Description
Syntax	<pre> <session-access-service className="fully.qualified.JavaClassName" debug="true false"> <session-access-pipeline ... > ... </session-access-pipeline> </session-access-service> </pre>

Security Domain Tag Reference

Example	<pre><session-access-pipeline className="com.cafesoft.security.engine.session.access.StandardSessionAccessPipeline" debug="false"> <session-access-valve className="com.cafesoft.security.engine.session.access.valves.LogSessionAccessRequestValve" debug="false"/> <param-list> <param name="logPath" value="{cams.home}/logs/system-session-access.log"> </param-list> </session-access-valve> </session-access-pipeline></pre>
---------	--

<session-access-valve>

Represents a single node within a session access pipeline for handling access requests. The valve receives a session access request and can handle the request completely, modify or add to it, or pass the request to the next valve in the chain.

Item	Description						
Syntax	<pre><session-access-valve className="fully.qualified.JavaClassName" debug="true false"/></pre>						
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"><code>className</code></td> <td style="width: 10%; text-align: center;">Req</td> <td>The fully qualified Java class name of the session access valve that will be instantiated.</td> </tr> <tr> <td><code>debug</code></td> <td style="text-align: center;">Opt</td> <td>Activate debug for this valve node of this security domain. The default value is false.</td> </tr> </table>	<code>className</code>	Req	The fully qualified Java class name of the session access valve that will be instantiated.	<code>debug</code>	Opt	Activate debug for this valve node of this security domain. The default value is false.
<code>className</code>	Req	The fully qualified Java class name of the session access valve that will be instantiated.					
<code>debug</code>	Opt	Activate debug for this valve node of this security domain. The default value is false.					
Data	None						
Parent Elements	1. <session-access-pipeline>						
Child Elements	<param-list> Opt A container for a list of initialization/configuration parameters.						
Example	<pre><session-access-valve className="com.cafesoft.security.engine.session.access.valves.LogSessionAccessRequestValve" debug="false"> <param-list> <param name="logPath" value="{cams.home}/logs/system-session-access.log"> </param-list> </session-access-valve></pre>						

<session-control-service>

Enables sessions to be closed, touched, updated, etc. by qualified Cams agents.

Item	Description						
Syntax	<pre><session-control-service className="fully.qualified.JavaClassName" debug="true false"> <session-control-pipeline ... > ... </session-control-pipeline> </session-control-service></pre>						
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"><code>className</code></td> <td style="width: 10%; text-align: center;">Req</td> <td>The fully qualified Java class name of the session control service that will be instantiated.</td> </tr> <tr> <td><code>debug</code></td> <td style="text-align: center;">Opt</td> <td>Activate debug for this security domain's session control service. The default value is false.</td> </tr> </table>	<code>className</code>	Req	The fully qualified Java class name of the session control service that will be instantiated.	<code>debug</code>	Opt	Activate debug for this security domain's session control service. The default value is false.
<code>className</code>	Req	The fully qualified Java class name of the session control service that will be instantiated.					
<code>debug</code>	Opt	Activate debug for this security domain's session control service. The default value is false.					
Data	None						
Parent Elements	1. <security-domain>						

Security Domain Tag Reference

Child Elements	<p><code><param-list></code> Opt A container for a list of initialization/configuration parameters.</p> <p><code><session-control-pipeline></code> Req Processes session control requests from qualified Cams agent.</p>
Example	<pre> <!-- Configure the session control service --> <session-control-service className="com.cafesoft.security.engine.session.control.StandardSessionControlService" debug="false"> <session-control-pipeline className="com.cafesoft.security.engine.session.control.StandardSessionControlPipeline" debug="false"> <session-control-valve className="com.cafesoft.security.engine.session.control.valves.LogSessionControlRequestValve" debug="false"> <param-list> <param name="logPath" value="{cams.home}/logs/system-session-control.log"> </param-list> </session-control-valve </session-control-pipeline> </session-control-service> </pre>

<session-control-pipeline>

Processes session control requests by a Cams agent. This pipeline is composed of a sequence of session control valves, which handle requests using the chain of responsibility design pattern. This enables each session control valve to handle the request altogether or modulate the request for processing by a subsequent valve.

Item	Description						
Syntax	<pre> <session-control-pipeline className="fully.qualified.JavaClassName" debug="true false"> <session-control-valve ... /> ... </session-control-pipeline> </pre>						
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;"><code>className</code></td> <td style="width: 10%; text-align: center;">Req</td> <td>The fully qualified Java class name of the session control pipeline that will be instantiated.</td> </tr> <tr> <td><code>debug</code></td> <td style="text-align: center;">Opt</td> <td>Activate debug for this security domain's session control pipeline. The default value is false.</td> </tr> </table>	<code>className</code>	Req	The fully qualified Java class name of the session control pipeline that will be instantiated.	<code>debug</code>	Opt	Activate debug for this security domain's session control pipeline. The default value is false.
<code>className</code>	Req	The fully qualified Java class name of the session control pipeline that will be instantiated.					
<code>debug</code>	Opt	Activate debug for this security domain's session control pipeline. The default value is false.					
Data	None						
Parent Elements	1. <code><session-control-service></code>						
Child Elements	<p><code><param-list></code> Opt A container for a list of initialization/configuration parameters.</p> <p><code><session-control-valve></code> Opt Represents a single node within a session control pipeline.</p>						
Example	<pre> <session-control-pipeline className="com.cafesoft.security.engine.session.control.StandardSessionControlPipeline" debug="false"> <session-control-valve className="com.cafesoft.security.engine.session.control.valves.LogSessionControlRequestValve" debug="false"> <param-list> <param name="logPath" value="{cams.home}/logs/system-session-control.log"> </param-list> </session-control-valve> </session-control-pipeline> </pre>						

<session-control-valve>

Represents a single node within a session control pipeline for handling session control requests. The valve receives a session control request and can handle the request completely, modify or add to it, or pass the request to the next valve in the chain.

Item	Description						
Syntax	<code><session-control-valve className="fully.qualified.JavaClassName" debug="true false"/></code>						
Attributes	<table> <tr> <td>className</td> <td>Req</td> <td>The fully qualified Java class name of the session control valve that will be instantiated.</td> </tr> <tr> <td>debug</td> <td>Opt</td> <td>Activate debug for this valve node of this security domain. The default value is false.</td> </tr> </table>	className	Req	The fully qualified Java class name of the session control valve that will be instantiated.	debug	Opt	Activate debug for this valve node of this security domain. The default value is false.
className	Req	The fully qualified Java class name of the session control valve that will be instantiated.					
debug	Opt	Activate debug for this valve node of this security domain. The default value is false.					
Data	None						
Parent Elements	1. <session-control-pipeline>						
Child Elements	<param-list> Opt A container for a list of initialization/configuration parameters.						
Example	<code><session-control-valve className="com.cafesoft.security.engine.session.control.valves.LogSessionControlRequestValve" debug="false"> <param-list> <param name="logPath" value="{cams.home}/logs/system-session-control.log"> </param-list> </session-control-valve></code>						

<service-manager>

Provides for management of security domain-wide services.

Item	Description						
Syntax	<code><service-manager className="fully.qualified.JavaClassName" debug="true false"> <service ... > ... </service> ... </service-manager></code>						
Attributes	<table> <tr> <td>className</td> <td>Req</td> <td>The fully qualified name of the Java service manager class that will be instantiated.</td> </tr> <tr> <td>debug</td> <td>Opt</td> <td>Activate debug for this security domain's service manager. The default value is false.</td> </tr> </table>	className	Req	The fully qualified name of the Java service manager class that will be instantiated.	debug	Opt	Activate debug for this security domain's service manager. The default value is false.
className	Req	The fully qualified name of the Java service manager class that will be instantiated.					
debug	Opt	Activate debug for this security domain's service manager. The default value is false.					
Data	None						
Parent Elements	1. <security-domain>						
Child Elements	<table> <tr> <td><param-list></td> <td>Opt</td> <td>A container for a list of initialization/configuration parameters.</td> </tr> <tr> <td><service></td> <td>Req</td> <td>Creates and registers a service implementation with the service manager.</td> </tr> </table>	<param-list>	Opt	A container for a list of initialization/configuration parameters.	<service>	Req	Creates and registers a service implementation with the service manager.
<param-list>	Opt	A container for a list of initialization/configuration parameters.					
<service>	Req	Creates and registers a service implementation with the service manager.					
Example	<code><!-- Register services accessible within this security domain --> <service-manager className="com.cafesoft.core.service.StandardServiceManager" debug="false"> <!-- Register a user repository service for cams-users.xml --></code>						

Security Domain Tag Reference

```

<service id="cams-user-repository" enabled="true">
  <service-type>com.cafesoft.security.engine.service.UserRepositoryService</service-type>
  <service-class>com.cafesoft.security.engine.service.UserRepositoryService</service-class>
  <param-list>
    <param name="repositoryFilePath"
      value="{cams.security-domain.home}/cams-users.xml"/>
    <param name="repositoryFactoryClass"
      value="com.cafesoft.security.engine.auth.login.userrepository.XmlUserRepositoryFactory"/>
    <param name="handlerClass"
      value="com.cafesoft.security.engine.auth.login.userrepository.CamsXmlUserRepositoryHandler"/>
    <param name="debug" value="false"/>
  </param-list>
</service>
</service-manager>

```

<service>

Creates and registers a service implementation with the service manager.

Item	Description									
Syntax	<pre> <service id="textual identifier" enabled="true false" debug="true false"> <service-type>fully.qualified.JavaClassName</service-type> <service-class>fully.qualified.JavaClassName</service-class> <param-list> ... </param-list> </service> </pre>									
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; padding: 2px;">id</td> <td style="width: 10%; padding: 2px;">Req</td> <td style="padding: 2px;">The unique textual identifier or name by which this service will be referenced.</td> </tr> <tr> <td style="padding: 2px;">enabled</td> <td style="padding: 2px;">Req</td> <td style="padding: 2px;">If true, initializes and make this service available. The default is true.</td> </tr> <tr> <td style="padding: 2px;">debug</td> <td style="padding: 2px;">Opt</td> <td style="padding: 2px;">Activate debug for this security domain's session control pipeline. The default value is false.</td> </tr> </table>	id	Req	The unique textual identifier or name by which this service will be referenced.	enabled	Req	If true, initializes and make this service available. The default is true.	debug	Opt	Activate debug for this security domain's session control pipeline. The default value is false.
id	Req	The unique textual identifier or name by which this service will be referenced.								
enabled	Req	If true, initializes and make this service available. The default is true.								
debug	Opt	Activate debug for this security domain's session control pipeline. The default value is false.								
Data	None									
Parent Elements	1. <service-manager>									
Child Elements	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%; padding: 2px;"><service-type></td> <td style="width: 10%; padding: 2px;">Req</td> <td style="padding: 2px;">Declares the type of service being registered.</td> </tr> <tr> <td style="padding: 2px;"><service-class></td> <td style="padding: 2px;">Req</td> <td style="padding: 2px;">Declares the implementation of the service.</td> </tr> <tr> <td style="padding: 2px;"><param-list></td> <td style="padding: 2px;">Req</td> <td style="padding: 2px;">A container for a list of initialization/configuration parameters.</td> </tr> </table>	<service-type>	Req	Declares the type of service being registered.	<service-class>	Req	Declares the implementation of the service.	<param-list>	Req	A container for a list of initialization/configuration parameters.
<service-type>	Req	Declares the type of service being registered.								
<service-class>	Req	Declares the implementation of the service.								
<param-list>	Req	A container for a list of initialization/configuration parameters.								
Example	<pre> <!-- Register a user repository service for cams-users.xml --> <service id="cams-user-repository" enabled="true" debug="false"> <service-type>com.cafesoft.security.engine.service.UserRepositoryService</service-type> <service-class>com.cafesoft.security.engine.service.UserRepositoryService</service-class> <param-list> <param name="repositoryFilePath" value="{cams.security-domain.home}/cams-users.xml"/> <param name="repositoryFactoryClass" value="com.cafesoft.security.engine.auth.login.userrepository.XmlUserRepositoryFactory"/> <param name="handlerClass" value="com.cafesoft.security.engine.auth.login.userrepository.CamsXmlUserRepositoryHandler"/> <param name="debug" value="false"/> </param-list> </service> </pre>									

<service-type>

Declares the type of service being registered (a Java interface).

Item	Description
Syntax	<code><service-type>fully.qualified.JavaClassName</service-type></code>
Attributes	None
Data	None
Parent Elements	1. <service>
Child Elements	None
Example	<pre> <!-- Register a user repository service for cams-users.xml --> <service id="cams-user-repository" enabled="true" debug="false"> <service-type>com.cafesoft.security.engine.service.UserRepositoryService</service-type> <service-class>com.cafesoft.security.engine.service.UserRepositoryService</service-class> <param-list> ... </param-list> </service> </pre>

<service-class>

Declares the Java class that implements the service.

Item	Description
Syntax	<code><service-class>fully.qualified.JavaClassName</service-class></code>
Attributes	None
Data	None
Parent Elements	1. <service>
Child Elements	None
Example	<pre> <!-- Register a user repository service for cams-users.xml --> <service id="cams-user-repository" enabled="true" debug="false"> <service-type>com.cafesoft.security.engine.service.UserRepositoryService</service-type> <service-class>com.cafesoft.security.engine.service.UserRepositoryService</service-class> <param-list> ... </param-list> </service> </pre>

<var-list>

An optional list of Cams variables that can be used to set security domain substitution values. These variables are useful in defining values that are frequently used in security domain configuration files.

Item	Description
Syntax	<pre> <var-list> <var ... /> ... </var-list> </pre>
Attributes	None
Data	None
Parent Elements	1. <security-domain>

Security Domain Tag Reference

Child Elements	<code><var></code> Opt An initialization/configuration parameter as a generic name/value pair.
Example	<pre><var-list> <var name="name1" value="value1" /> </var-list></pre>

<var>

A Cams variable is used to set a global substitution value. These values are useful in working with a security domains configuration files, especially where test and production deployments are on distinct hosts.

Item	Description						
Syntax	<code><var name="textual name" value="value" /></code>						
Attributes	<table border="0" style="width: 100%;"> <tr> <td style="width: 15%;">name</td> <td style="width: 15%;">Req</td> <td style="width: 70%;">The textual param name.</td> </tr> <tr> <td>value</td> <td>Req</td> <td>The param value.</td> </tr> </table>	name	Req	The textual param name.	value	Req	The param value.
name	Req	The textual param name.					
value	Req	The param value.					
Data	None						
Parent Elements	1. <code><var-list></code>						
Child Elements	None						
Example	<pre><var-list> <var name="name1" value="value1" /> <var name="name2" value="value2" /> <var name="\${name1}_substituted" value="\${name1} and \${name2}" /> </var-list></pre>						

<param-list>

A list of parameters that can be used to set initialization or configuration values.

Item	Description
Syntax	<pre><param-list> <param ... /> ... </param-list></pre>
Attributes	None
Data	None
Parent Elements	<ol style="list-style-type: none"> 1. <code><auth-service></code> 2. <code><login-config-factory></code> 3. <code><auth-pipeline></code> 4. <code><auth-valve></code> 5. <code><access-control-service></code> 6. <code><access-control-policy-factory></code> 7. <code><access-control-pipeline></code> 8. <code><access-control-valve></code> 9. <code><session-manager-service></code> 10. <code><session-event-handler></code> 11. <code><session-access-service></code> 12. <code><session-access-pipeline></code> 13. <code><session-access-valve></code> 14. <code><session-control-service></code> 15. <code><session-control-pipeline></code> 16. <code><session-control-valve></code> 17. <code><service-manager></code> 18. <code><service></code>
Child Elements	<code><param></code> Opt An initialization/configuration parameter as a generic name/value pair.
Example	<code><param-list></code>

Security Domain Tag Reference

```
<param name="textual name" value="value">
</param-list>
```

<param>

A parameter used to set a single initialization or configuration value.

Item	Description						
Syntax	<code><param name="textual name" value="value"/></code>						
Attributes	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">name</td> <td style="width: 20%; text-align: center;">Req</td> <td style="width: 50%;">The textual param name.</td> </tr> <tr> <td>value</td> <td style="text-align: center;">Req</td> <td>The param value.</td> </tr> </table>	name	Req	The textual param name.	value	Req	The param value.
	name	Req	The textual param name.				
value	Req	The param value.					
Data	None						
Parent Elements	1. <param-list>						
Child Elements	None						
Example	<pre><param-list> <param name="textual name" value="value"> </param-list></pre>						

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Security Domain Registry Tag Reference

The Cams security domain registry configuration is defined by the `security-domain-registry.xml` file. This document contains reference information for each of the tags that can be used within `security-domain-registry.xml`. The following table shows the file structure with links to each of the possible elements.

Tag Name	Instances	Description
security-domain-registry	1	declares a security domain registry
var-list	0 ... 1	a list of Cams variables
[var]	0 ... N	a Cams variable
[security-domain]	1 ... N	declares a security domain
name	1	the unique security domain name
home	1	the location of security domain's configuration files

<security-domain-registry>

The top-level element that declares a security domain registry. A security domain registry manages a collection of security domain objects.

Item	Description						
Syntax	<pre><security-domain-registry> <var-list> ... </var-list> <security-domain> ... </security-domain> ... </security-domain-registry></pre>						
Attributes	None						
Data	None						
Parent Elements	None						
Child Elements	<table border="0"> <tr> <td><var-list></td> <td>Opt</td> <td>An optional list of Cams variables that can be used to set global substitution values. These variables are useful in working with a security domains configuration files, especially where test and production deployments are on distinct hosts.</td> </tr> <tr> <td><security-domain></td> <td>Req</td> <td>Declares a single named security domain within the registry with its associated configuration files. Minimally, the system security domain must always be declared. You can declare as many security domains as you require.</td> </tr> </table>	<var-list>	Opt	An optional list of Cams variables that can be used to set global substitution values. These variables are useful in working with a security domains configuration files, especially where test and production deployments are on distinct hosts.	<security-domain>	Req	Declares a single named security domain within the registry with its associated configuration files. Minimally, the system security domain must always be declared. You can declare as many security domains as you require.
<var-list>	Opt	An optional list of Cams variables that can be used to set global substitution values. These variables are useful in working with a security domains configuration files, especially where test and production deployments are on distinct hosts.					
<security-domain>	Req	Declares a single named security domain within the registry with its associated configuration files. Minimally, the system security domain must always be declared. You can declare as many security domains as you require.					
Example	<pre><security-domain-registry> <!-- Global substitution variables --> <var-list> <var name="http.resource.base.id" value="http://localhost:8080" /> <var name="https.resource.base.id" value="https://localhost:8443" /> <var name="cams.resource.base.id" value="cams://localhost:9191" /> <var name="cams.logs.base.dir" value="{cams.home}/logs" /> </var-list></pre>						

Security Domain Registry Tag Reference

	<pre> <!-- Register the system security domain --> <security-domain enabled="true"> <name>system</name> <home>\${cams.home}/conf/domains/system</home> </security-domain> <!-- Register the mydomain security domain --> <security-domain enabled="true"> <name>default</name> <home>\${cams.home}/conf/domains/mydomain</home> </security-domain> </security-domain-registry> </pre>
--	---

<var-list>

An optional list of Cams variables that can be used to set global substitution values. These variables are useful in defining values that are frequently used in security domain configuration files.

Item	Description			
Syntax	<pre> <var-list> <var ... /> ... </var-list> </pre>			
Attributes	None			
Data	None			
Parent Elements	1. <security-domain-registry>			
Child Elements	<table style="width: 100%; border: none;"> <tr> <td style="width: 15%; border: none;"><var></td> <td style="width: 20%; border: none; text-align: center;">Opt</td> <td style="border: none;">A Cams variable as a generic name/value pair.</td> </tr> </table>	<var>	Opt	A Cams variable as a generic name/value pair.
<var>	Opt	A Cams variable as a generic name/value pair.		
Example	<pre> <!-- Global substitution variables --> <var-list> <var name="http.resource.base.id" value="http://localhost:8080"/> <var name="https.resource.base.id" value="https://localhost:8443"/> <var name="cams.resource.base.id" value="cams://localhost:9191"/> <var name="cams.logs.base.dir" value="\${cams.home}/logs"/> </var-list> </pre>			

<var>

A Cams variable is used to set a global substitution value. These values are useful in working with a security domains configuration files, especially where test and production deployments are on distinct hosts.

Item	Description						
Syntax	<pre> <var name="textual name" value="value"/> </pre>						
Attributes	<table style="width: 100%; border: none;"> <tr> <td style="width: 15%; border: none;">name</td> <td style="width: 20%; border: none; text-align: center;">Req</td> <td style="border: none;">The variable name.</td> </tr> <tr> <td style="border: none;">value</td> <td style="border: none; text-align: center;">Req</td> <td style="border: none;">The variable value.</td> </tr> </table>	name	Req	The variable name.	value	Req	The variable value.
name	Req	The variable name.					
value	Req	The variable value.					
Data	None						
Parent Elements	1. <var-list>						
Child Elements	None						
Example	<pre> <!-- Global substitution variables --> <var-list> <var name="http.resource.base.id" value="http://localhost:8080"/> <var name="https.resource.base.id" value="https://localhost:8443"/> </pre>						

Security Domain Registry Tag Reference

```
<var name="cams.resource.base.id"
      value="cams://localhost:9191"/>
<var name="cams.logs.base.dir"
      value="{cams.home}/logs"/>
</var-list>
```

<security-domain>

Declares a single named security domain within the registry with its associated configuration files. Minimally, the system security domain must always be declared. You can declare as many security domains as you require.

Security domains represent a logic division of resources to facilitate management and functional requirements. For example, you might place resources in different security domains to delegate security management responsibilities across an organization. Or, you might have different authentication requirements depending upon the resources (some content may require a higher level of security).

Item	Description						
Syntax	<pre><security-domain enabled="true false"> <name> ... </name> <home> ... </home> </security-domain></pre>						
Attributes	<table> <tr> <td>enabled</td> <td>Req</td> <td>Enable this security domain (true or false).</td> </tr> </table>	enabled	Req	Enable this security domain (true or false).			
enabled	Req	Enable this security domain (true or false).					
Data	None						
Parent Elements	1. <security-domain-registry>						
Child Elements	<table> <tr> <td><name></td> <td>Req</td> <td>The unique security domain name.</td> </tr> <tr> <td><home></td> <td>Req</td> <td>The location of security domain's configuration files.</td> </tr> </table>	<name>	Req	The unique security domain name.	<home>	Req	The location of security domain's configuration files.
<name>	Req	The unique security domain name.					
<home>	Req	The location of security domain's configuration files.					
Example	<pre><!-- Register the system security domain --> <security-domain enabled="true"> <name>system</name> <home>{cams.home}/conf/domains/system</home> </security-domain></pre>						

<name>

The unique security domain name.

Item	Description
Syntax	<code><name>textual name</name></code>
Attributes	None
Data	None
Parent Elements	1. <security-domain>
Child Elements	None
Example	<pre><!-- Register the system security domain --> <security-domain enabled="true"> <name>system</name> <home>{cams.home}/conf/domains/system</home> </security-domain></pre>

<home>

The location of security domain's configuration files. The path can be specified using forward or back slashes, and can use substitution values.

Item	Description
Syntax	<code><home>path to files</home></code>
Attributes	None

Security Domain Registry Tag Reference

Data	None
Parent Elements	1. <security-domain>
Child Elements	None
Example	<pre><!-- Register the system security domain --> <security-domain enabled="true"> <name>system</name> <home>\${cams.home}/conf/domains/system</home> </security-domain></pre>

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Troubleshooting Cams FAQ

This section contains a list of questions and answers to access control configuration errors, debugging techniques, etc.

1. [How can I find out if a security domain's access control service is loading, initializing, starting, and stopping correctly?](#)
2. [How can I tell if a security domain's access control policy is correctly loading and initializing?](#)
3. [How can I tell if a security domain is handling a specific access control request?](#)
4. [How can I see the contents of an access control request when it is being evaluated by an access control policy?](#)
5. [How can I see which permission and access control rule are protecting a resource?](#)
6. [How can I tell if a security domain is delegating an access control request?](#)
7. [How can I see the contents of an access control response?](#)
8. [How can I see how an access control rule evaluates an access control request?](#)

How can I find out if a security domain's access control service is loading, initializing, starting, and stopping correctly?

As a security domain's access control service is being loaded, initialized, started, and stopped, various DEBUG, INFO, WARNING, ERROR, and/or FATAL messages may be written to the security domain-specific trace log.

If the security domain's access control service is correctly loaded, you will see the following two (not necessarily consecutive) messages in that security domain's trace log file:

```
INFO - Loading AccessControlService
INFO - Loaded AccessControlService
```

If the security domain's access control service is correctly initialized, you will see the following two (not necessarily consecutive) messages in that security domain's trace log file:

```
INFO - Initializing AccessControlService
INFO - Initialized AccessControlService
```

If the security domain's access control service is correctly started, you will see the following two messages that security domain's trace log file:

```
INFO - Starting AccessControlService
INFO - Started AccessControlService
```

If the security domain's access control service is correctly stopped, you will see the following two messages that security domain's trace log file:

```
INFO - Stopping AccessControlService
INFO - Stopped AccessControlService
```

If any of these message pairs has an intervening "ERROR" or "FATAL" message, then the load, initialization, start, or stop process failed and the message should give sufficient context to start debugging the configuration problem.

How can I tell if a Cams security domain's access control policy is correctly initializing and loading?

As a security domain's access control policy is being initialized and loaded, various DEBUG, INFO, WARNING, ERROR, and/or FATAL messages may be written to the security domain-specific trace log.

If the security domain's access control policy is correctly loaded, you will see the following two (not necessarily consecutive) messages in that security domain's trace log file:

```
INFO - Initializing AccessControlService
INFO - Initialized AccessControlService
```

If the security domain's access control service is correctly initialized, you will see the following two (not necessarily consecutive) messages in that security domain's trace log file:

```
INFO - Loading AccessControlService
INFO - Loaded AccessControlService
```

If any of these message pairs has an intervening "ERROR" or "FATAL" message, then the load, initialization, start, or stop process failed and the message should give sufficient context to start debugging the configuration problem.

How can I tell if a security domain is handling a specific access control request?

The easiest way to tell if a security domain is handling an access control request for a particular resource is to look in the access control transaction Log for that security domain. For example, suppose a security domain is supposed to protect an "http" resource corresponding to the following resource request:

```
Actions=GET,
Resource Id=http://localhost:8080/examples/index.jsp
```

If you attempt to access the resource via a web browser and the security domain access control transaction logging is enabled, you should see a new record resembling the following line appear at the bottom of the transaction log file:

```
[10/Jul/2002:09:23:59 -0700] 127.0.0.1 -- http http://localhost:8080/examples/index.jsp "GET" 2 10
```

If the record does not appear, then the security domain is either not receiving the request or an error is occurring during the access control check. Check the security domain's trace log for an ERROR or FATAL message related to the access control request. If not present, then check the general cams-server.log and the system security domain's trace log.

Another possible cause of the problem could be that the system (or some other) security domain is not properly delegating the access control request. Check the system security domain's access control transaction log to make sure it is receiving the initial access control request. If it is, you may need to enable DEBUG-level messages for the system security domain's access control policy. Those DEBUG messages will tell you whether the access control check is being handled by the system security domain or if it is being delegated to another security domain. If delegated, you will need to check each intermediate security domain's access control policy to ensure that delegation rules are setup properly.

How can I see the contents of an access control request when it is being evaluated by an access control policy?

To see the contents of an access control request as received by a security domain's access control policy, you'll need to:

- Enable DEBUG-level messages for the security domain's trace logger
- Enable DEBUG-level messages for the security domain's access control policy component

Example 1 shows what you'll see for each access control request in the trace log once debugging is enabled:

```
DEBUG - ----- Start Access Control Check -----
DEBUG - AccessRequest
DEBUG - {
DEBUG -   Security Domain=examples
DEBUG -   Session Id=6578616d706c6573-31303236353136323034363934-6775657374-\
           4d2f4338a97cdae51f2ccb35e94b2e8dcdce66
DEBUG -   Remote Addr=127.0.0.1
DEBUG -   Remote Host=localhost
DEBUG -   Confidential=false
DEBUG -   App Name=xml
DEBUG -   ResourceRequest
DEBUG -   {
DEBUG -     Id=http://localhost:8080/examples/index.jsp
DEBUG -     Type=http
DEBUG -     Actions=GET
DEBUG -     Actions Mask=1 (dec) , 1 (bin)
DEBUG -   }
DEBUG -   Session
DEBUG -   {
DEBUG -     Id=6578616d706c6573-31303236353136323034363934-6775657374-\
           4d2f4338a97cdae51f2ccb35e94b2e8dcdce66
DEBUG -     Creation Time=Fri Jul 12 16:23:24 PDT 2002
DEBUG -     Last Touched Time=Fri Jul 12 16:23:25 PDT 2002
DEBUG -     Status=ACTIVE
DEBUG -     Subject
DEBUG -     {
DEBUG -       username=guest
DEBUG -       principal: name=everyone, class=com.cafesoft.cams.auth.CSRolePrincipal
DEBUG -       principal: name=guest, class=com.cafesoft.cams.auth.CSUserPrincipal
DEBUG -     }
DEBUG -     Attributes
DEBUG -     {
DEBUG -       default:namespace
DEBUG -       {
```

```

DEBUG -         }
DEBUG -     }
DEBUG - }
DEBUG -
DEBUG - RequestDispatchStack
DEBUG - {
DEBUG -     [1]=examples
DEBUG -     [0]=system
DEBUG - }
DEBUG - }
DEBUG - }
...
DEBUG - ----- End Access Control Check -----

```

Example 1 – Sample access control request DEBUG–level messages

More information is provided on how to interpret these messages in [Troubleshooting Cams Access Control](#).

How can I see which permission and access control rule are protecting a resource?

To see which permission and access control rule are protecting a resource:

- Enable DEBUG–level messages for the security domain's trace logger
- Enable DEBUG–level messages for the security domain's access control policy component

The security domain's trace logger will contain DEBUG message like those in Example 2.

```

DEBUG - ----- Start Access Control Check -----
DEBUG - AccessRequest
DEBUG - {
...
DEBUG - }
DEBUG -
DEBUG - Permission
DEBUG - {
DEBUG -     Description: Examples Content
DEBUG -     ResourcePattern
DEBUG -     {
DEBUG -         Pattern=http://localhost:8080/examples/
DEBUG -         Owner=NONE
DEBUG -     }
DEBUG -     Type=http
DEBUG -     Actions=GET,POST
DEBUG -     Actions Mask=3 (dec), 11 (bin)
DEBUG -     ACR=allow role everyone
DEBUG - }
DEBUG -
DEBUG - ACR evaluated to "true": ACCESS GRANTED
...
DEBUG - AccessResponse
DEBUG - {
...
DEBUG - }
DEBUG - ----- End Access Control Check -----

```

Example 2 – Sample access control policy permission DEBUG–level messages

The permission and access control rule information for the request are shown in red.

How can I tell if a security domain is delegating an access control request?

To see is a security domain is delegating an access control request:

- Enable DEBUG–level messages for the security domain's trace logger
- Enable DEBUG–level messages for the security domain's access control policy component

The security domain's trace logger will contain DEBUG message like those shown in Example 3.

```

DEBUG - ----- Start Access Control Check -----
DEBUG - AccessRequest

```

Troubleshooting Cams FAQ

```
DEBUG - {
...
DEBUG - }
DEBUG -
DEBUG - Permission
DEBUG - {
DEBUG -     Description: Examples Content
DEBUG -     ResourcePattern
DEBUG -     {
DEBUG -         Pattern=http://localhost:8080/examples/
DEBUG -         Owner=examples
DEBUG -     }
DEBUG -     Type=http
DEBUG -     Actions=GET,POST
DEBUG -     Actions Mask=3 (dec), 11 (bin)
DEBUG -     ACR=NONE
DEBUG - }
DEBUG -
DEBUG - ... Forwarding access control check to security domain: 'examples'
DEBUG - AccessResponse
DEBUG - {
...
DEBUG - }
DEBUG - ----- End Access Control Check -----
```

Example 3 – Sample access control policy DEBUG messages showing forwarding of an access control request

In the example, the "Owner" security domain is declared to be "examples" (which means that these DEBUG messages are from another security domain's trace log). The "... Forwarding ..." message will indicate the security domain to which the access control check is being delegated.

How can I see the contents of an access control response?

To see the contents of an access control response as returned by a security domain's access control policy, you'll need to:

- Enable DEBUG–level messages for the security domain's trace logger
- Enable DEBUG–level messages for the security domain's access control policy component

Example 4 shows part of what you'll see for each access control response in the trace log once debugging enabled:

```
DEBUG - ----- Start Access Control Check -----
DEBUG - AccessRequest
DEBUG - {
...
DEBUG - }
DEBUG -
DEBUG - Permission
...
DEBUG - }
DEBUG -
DEBUG - ACR evaluated to "true": ACCESS GRANTED
DEBUG - AccessResponse
DEBUG - {
DEBUG -     status=GRANTED
DEBUG -     reason=0
DEBUG -     message=null
DEBUG -     LoginParameters
DEBUG -     {
DEBUG -     }
DEBUG - }
DEBUG - ----- End Access Control Check -----
```

Example 4 – access control response DEBUG–level messages

A sample access control response is shown in red.

How can I see how an access control rule evaluates an access control request?

To see how the access control rule referenced by a security domain's permission is evaluated by an access control policy:

- Enable DEBUG–level messages for the security domain's trace logger

Troubleshooting Cams FAQ

- Enable DEBUG–level messages for the security domain's access control policy component

Example 5 shows part of what you'll see for each access control response in the trace log once debugging enabled:

```
DEBUG - ----- Start Access Control Check -----
DEBUG - AccessRequest
DEBUG - {
...
DEBUG - }
DEBUG -
DEBUG - Permission
...
DEBUG - }
DEBUG -
DEBUG - ACR evaluated to "true": ACCESS GRANTED
DEBUG - AccessResponse
DEBUG - {
DEBUG -     status=GRANTED
DEBUG -     reason=0
DEBUG -     message=null
DEBUG -     LoginParameters
DEBUG -     {
DEBUG -     }
DEBUG - }
DEBUG - ----- End Access Control Check -----
```

Example 5 – Sample DEBUG–level message showing the result of evaluating an access control rule

The access control rule evaluation result is shown above in red. If the invoked access control rule is a compound rule like access control rule expression (XML "acr" tag), you may enable DEBUG–level messages for each nested access control rule to see more details on how each ACR evaluates the access control request.

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Troubleshooting Cams Access Control

The Cams access control system provides fine-grained information that can help you debug your security domains, access control policies, and access control rules. This document contains:

- A summary of the Cams access control system structure, which will enable you to understand the context of configuration issues
- Cams trace log file locations where you'll find DEBUG, INFO, WARNING, ERROR, and FATAL messages
- How to enable Cams trace logs to include DEBUG-level messages
- How to configure specific Cams access control components to issue DEBUG-level messages

The Cams Access Control System Structure

In order to effectively debug Cams access control issues, it's important to understand how the Cams access control system is organized. You'll be able to isolate the Cams components that you want to provide DEBUG-level messages so you can avoid generating too much data, which can make debugging more difficult.

Figure 1 shows the high-level, hierarchical structure of the Cams access control services.

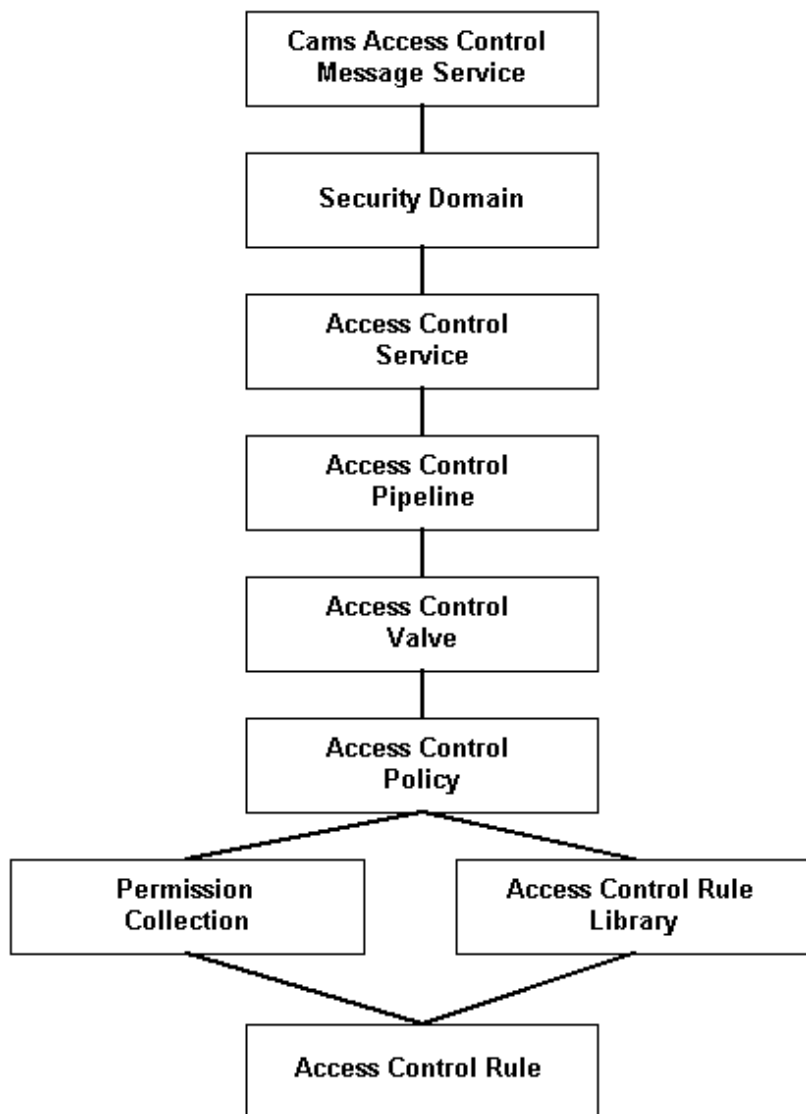


Figure 1 – High-level structure of Cams access control services

Table 1 describes each of these components and their roles within the Cams architecture.

Component	Description
-----------	-------------

Troubleshooting Cams Access Control

Cams Access Control Message Service	<p>Receives incoming access control requests from network agents and hands them to the system security domain, then returns the associated access control response. This service handles access-control requests for all configured Cams security domains and its debug information is written to:</p> <p><code>\${cams.home}/logs/cams-server.log</code></p>
Security Domain	<p>Hosts a set of standard security services that are configured to manage a subset of an enterprise's overall security responsibilities. security domains essentially partition responsibility for managing users, authentication, and access control according to organizational, structural, or political boundaries. Each security domain has its own trace log file, which is generally written to security domain-specific file name. security domains also have various transaction log files for authentication, access control, session management, session access, etc, but debug information is available in the trace log. By default, the "system" security domain's trace log is written to:</p> <p><code>\${cams.home}/logs/system-trace.log</code></p> <p>DEBUG-level messages for all sub-components: access control service, access control pipeline, access control valve, access control policy, and access control rule are written to their security domain-specific trace logs.</p>
Access Control Service	<p>The service with a security domain responsible for implementing access control to resource protected by that security domain.</p>
Access Control Pipeline	<p>A component with a security domain's access control service that hosts a configurable set of "access control valves". This architecture implements a flexible "Chain of Responsibility" design pattern that enables logging, debugging, auditing, etc components to be plugged into a security domain's access control service.</p> <p>When the access control pipeline receives a request, it is passed through its sequence of access control valves. Each valve may handle the request, modify the request, and/or pass the request to the next valve. An internally-defined "Basic" access control valve is always the last one in the sequence and is usually the one that does the work of invoking the associated Cams access control policy.</p>
Access Control Valve	<p>A component within a security domain's access control pipeline that may handle the request, modify the request, and/or pass the request to the next valve.</p>
Access Control Policy	<p>Contains all of the access control permissions for resource protected within a security domain. Also contains all access control rule instances that may be invoked by permissions and access control rule type definitions if custom access control rules have been developed and plugged into Cams.</p>
PermissionCollection	<p>A permission collection is a homogeneous collection of permissions. For example, all http permissions are in the same permission collection and all cams permissions are in another permission collection. Each permission associates a set of resources with an access control action. The action may either be:</p> <ol style="list-style-type: none"> 1. invoke an access control rule that will check access to the requested resource 2. delegate the request to another Cams security domain
Access Control Rule Library	<p>A Container for access control rule instances and access control rule type declarations.</p>
Access Control Rule	<p>In general an access control policy will contain many access control rules, which are used to decide whether or not access will be granted or denied to a protected resource. Cams comes with a number of standard access control rules for controlling access by: authenticated user role, network host, confidential (encrypted) network traffic, etc. Cams also includes a special access control rule that enables you to combine access control rules into logical expressions using "and", "or", and "not" operators.</p>

Table 1 – Summary of Cams access control services/components

Enabling/Disabling Cams DEBUG–level Messages

To generate DEBUG–level messages for a particular Cams access control system component, you'll need to:

1. enable DEBUG–level messages for the appropriate Cams Logger
2. enable DEBUG messages for the desired Cams Service/Component

Configuring Loggers to Enable DEBUG–level Messages

Table 2 summarizes the configuration file locations for Cams trace loggers. In general, each security domain will configure its own trace logger. See configuration file: `${cams.home}/domains/security–domain–registry.xml` for the list of registered security domains and their configuration root directories.

Component	Debug Configuration Properties
Cams Access Control Message Service	Configuration file: <code>\${cams.home}/conf/cams.conf</code> Property: <code>debug=true</code>
Security Domain Access Control Service Access Control Pipeline Access Control Valve Access Control Policy Permission Collection Access Control Rule Library Access Control Rule	Configuration file: <code>security–domain.xml</code> Element: <pre><logger className="com.cafesoft.cams.log.CamsTraceLogger" params="debug=true,logger.file.path=\${cams.home}/logs/system–trace.log,logger.file.append=false"/></pre>

Table 2 – Configuration parameters enabling/disabling Cams Logger DEBUG–level messages

Configuring Components to Enable DEBUG–level Messages

Table 3 summarizes the configuration properties for enabling/disabling component–level DEBUG messages.

Component	Debug Configuration Properties
Cams Access Control Message Service	Configuration file: <code>\${cams.home}/conf/cams.conf</code> Property: <code>cams.service.access–control.debug=true</code>
Security Domain	Configuration file: security–domain.xml Element: <code><security–domain debug="true"></code> if the optional "debug" attributes is not present, the default is "false".
Access Control Service	Configuration file: <code>security–domain.xml</code> Element: <pre><access–control–service className="com.cafesoft.security.engine.access.StandardAccessController" debug="true"></pre> if the optional "debug" attributes is not present, the default is "false".
Access Control Pipeline	Configuration file: <code>security–domain.xml</code> Element: <pre><access–control–pipeline className="com.cafesoft.security.engine.access.StandardAccessPipeline" debug="true"></pre> if the optional "debug" attributes is not present, the default is "false".

Troubleshooting Cams Access Control

	<p>NOTE: enabling debug for the access control pipeline also enables debug for it's internal "basic" access control valve.</p>
Access Control Valve	<p>Configuration file: security-domain.xml</p> <p>Element:</p> <pre><access-control-valve className="com.cafesoft.security.engine.access.valves.LogAccessRequestValve" debug="true"> <param-list> <param name="logPath" value="{cams.home}/logs/system-access-control.log"/> </param-list> </access-control-valve></pre> <p>if the optional "debug" attributes is not present, the default is "false".</p>
Access Control Policy	<p>Configuration file: access-control-policy.xml</p> <p>Element: <access-control-policy defaultBias="denied" debug="true"></p> <p>if the optional "debug" attributes is not present, the default is "false".</p>
Permission Collection	<p>Configuration file: access-control-policy.xml</p> <p>Element:</p> <pre><permission-collection type="http" desc="HTTP Server Permissions" debug="true"></pre> <p>if the optional "debug" attributes is not present, the default is "false".</p>
Access Control Rule Library	<p>Configuration file: access-control-policy.xml</p> <p>Element: <acr-lib debug="true"></p> <p>if the optional "debug" attributes is not present, the default is "false".</p>
Access Control Rule	<p>Configuration file: access-control-policy.xml</p> <p>Various access control rule element types exist. Each of the following elements supports a "debug" attribute:</p> <p>Elements:</p> <ul style="list-style-type: none"> • <auth-acr id="auth rule id" debug="true"> • <host-acr id="host rule id" debug="true"> • <acr id="access control rule expression id" debug="true"> <p>if the optional debug attributes is not present, the default is false. If custom access control rule are developed, we recommend that they support a debug attribute.</p>

Table 3 – Configuration parameters enabling/disabling component DEBUG-level messages

Troubleshooting Cams Access Control

This sections describes how to troubleshoot various Cams access control configuration problems and how to verify your configuration settings.

Troubleshooting a Cams Access Control Policy

Suppose you want to verify that the correct security domain is receiving an access control request for a given resource and that the appropriate permission and access control rule are being invoked. You'll need to:

Troubleshooting Cams Access Control

1. Enable DEBUG–level messages for the security domain protecting the resource
2. Enable DEBUG–level messages for the security domain's access control policy component

(See sections: [Configuring Loggers to enable DEBUG–level Messages](#) and [Configuring Components to enable DEBUG–level Messages](#)). Once you've enabled these and either restarted the Cams Policy Server or reloaded the modified security domain, you should see DEBUG message like those in Example 1 within the security domain–specific trace log. If you don't see messages of this format, then you may not have the right debug flags turned on or some other security domain may be receiving the access control requests. These messages are generated by the standard Cams access control policy.

```
DEBUG - ----- Start Access Control Check -----
DEBUG - AccessRequest
DEBUG - {
DEBUG -     Security Domain=examples
DEBUG -     Session Id=6578616d706c6573-31303236353136323034363934-6775657374-\
        4d2f4338a97cdae51f2ccb35e94b2e8dc66
DEBUG -     Remote Addr=127.0.0.1
DEBUG -     Remote Host=localhost
DEBUG -     Confidential=false
DEBUG -     App Name=xml
DEBUG -     ResourceRequest
DEBUG -     {
DEBUG -         Id=http://localhost:8080/examples/index.jsp
DEBUG -         Type=http
DEBUG -         Actions=GET
DEBUG -         Actions Mask=1 (dec) , 1 (bin)
DEBUG -     }
DEBUG -     Session
DEBUG -     {
DEBUG -         Id=6578616d706c6573-31303236353136323034363934-6775657374-\
        4d2f4338a97cdae51f2ccb35e94b2e8dc66
DEBUG -         Creation Time=Fri Jul 12 16:23:24 PDT 2002
DEBUG -         Last Touched Time=Fri Jul 12 16:23:25 PDT 2002
DEBUG -         Status=ACTIVE
DEBUG -         Subject
DEBUG -         {
DEBUG -             username=guest
DEBUG -             principal: name=everyone, class=com.cafesoft.cams.auth.CSRolePrincipal
DEBUG -             principal: name=guest, class=com.cafesoft.cams.auth.CSUserPrincipal
DEBUG -         }
DEBUG -         Attributes
DEBUG -         {
DEBUG -             default:namespace
DEBUG -             {
DEBUG -             }
DEBUG -         }
DEBUG -     }
DEBUG -     RequestDispatchStack
DEBUG -     {
DEBUG -         [1]=examples
DEBUG -         [0]=system
DEBUG -     }
DEBUG - }
DEBUG - Permission
DEBUG - {
DEBUG -     Description: Examples Content
DEBUG -     ResourcePattern
DEBUG -     {
DEBUG -         Pattern=http://localhost:8080/examples/
DEBUG -         Owner=NONE
DEBUG -     }
DEBUG -     Type=http
DEBUG -     Actions=GET,POST
DEBUG -     Actions Mask=3 (dec), 11 (bin)
DEBUG -     ACR=allow role everyone
DEBUG - }
DEBUG - ACR evaluated to "true": ACCESS GRANTED
DEBUG - AccessResponse
DEBUG - {
DEBUG -     status=GRANTED
DEBUG -     reason=0
```

Troubleshooting Cams Access Control

```

DEBUG -      message=null
DEBUG -      LoginParameters
DEBUG -      {
DEBUG -      }
DEBUG - }
DEBUG - ----- End Access Control Check -----
    
```

Example 1 – Access control request DEBUG-level messages

Table 4 defines the high-level sections within these debug messages and some of the values you might see under different conditions. Curly brackets and indentation are used to show logical collections of parameters and child/parent relationships. A dot notation is used in Table 4 to show parent/child relationships, but for brevity, only enough context is shown for the Debug Tag to uniquely identify it.

Debug Tag	Debug Configuration Properties
AccessRequest	The incoming access control request, which encapsulates the information used to evaluate whether access to a resource will be granted or denied. All of the parameters contained within the access request are included between the curly brack that follows this line and the matching curly bracket farther down the listing.
AccessRequest.Security Domain	The name of the security domain receiving the access control request. This will correspond with the security domain-specific log file containing this message.
AccessRequest.Session Id	The identifier of a session asserted for the access control request. Existence of this value asserts that the user has already authenticated. If the session object corresponding to this value exists and is valid, then the session object is also referenced by the access request.
AccessRequest.Remote Addr	The TCP/IP address of the host from which a user is attempting to access a protected resource.
AccessRequest.Remote Host	<p>The DNS name of the host from which a user is attempting to access a protected resource. Cams currently depends on the access control agent (or the application containing the access control agent) to implement reverse DNS name lookups (IP address to name lookups). These name lookups can be very slow and unreliable, so it is not advisable or practical to have the Cams Policy Server resolve them unless you have complete control over the DNS configuration for the clients from which users will access resources.</p> <p>If the reverse DNS is not enabled in the agent or the agent's application, then this value will contain the TCP/IP address (same as Remote Addr).</p>
AccessRequest.Confidential	If "true", then the resource is being accessed via "confidential" means. For example, if the resource is a Web page, then SSL is in use.
AccessRequest.App Name	The name of the application containing the agent from which the access control request was received. This value is used to select a Login Configuration Entry from the security domain's Login Configuration if authentication is required.
ResourceRequest	Encapsulates information about the protected resource that a user wants to access.
ResourceRequest.Id	The identifier of the resource. The actual format for a resource identifier depends on it's type. In this case, an "http" resource is uniquely identified by it's fully-qualifier URL.
ResourceRequest.Type	The type of resource. This value is sent from the access control agent and must match one of the type supported by the Cams Policy Server. See the Cams Policy Server configuration file at: <code>#{cams.home}/conf/cams.conf</code> for the list of all registered resource types.
ResourceRequest.Actions	The action or actions being requested on the resource. For example, the user may be attempting to "GET", "POST", "DELETE", etc a web URL. (If multiple actions are requested on the resource, then all actions must be granted in order for access to the resource to be granted.)
ResourceRequest.Actions Mask	A bit mask used to efficiently compare resource request actions with corresponding permission actions. This value will be of interest to developers that add support for new resource types and permission types.
Session	

Troubleshooting Cams Access Control

	If an authenticated user is associated with the access request, then this tag encapsulates everything about that user's session. If no authenticated user is associated with the access request, then the content of this tag will be "NONE". If the value of AccessRequest.Session Id is populated and the contents of this tag is NONE, then the session has either expired, been closes by an explicit "logout", or the reference Session Id is invalid and possibly counterfeit.
Session.Creation Time	The date/time at which the session was created.
Session.Last Touched Time	The date/time at which access control was last checked for the session. (This value is NOT updated by the "session-access" service or by session event handlers).
Session.Status	The status of the session: "ACTIVE", "EXPIRED", or "CLOSED". In practice, the Session tag will be empty unless the session object is "ACTIVE". "EXPIRED" or "CLOSED" session states are used primarily by the session manager to purge session objects and to report that status of sessions queried via the session-access service.
Session.Subject	Information about the authenticated Subject (user) associated with a Session. A A Subject will alway exist for an "ACTIVE" Session.
Subject.username	The name and class of the user principal within the Subject. A Subject contains a list of Principal instances. The name of the first Principal in the list implemented by class: com.cafesoft.cams.auth.CSUserPrincipal is deemed the "user Principal".
Subject.principal	The name and class of each Principal (including the user Principal) within the authenticated Subject. This tag may repeat any number of times, once for each Principal added by LoginModules at user authentication time.
Session.Attributes	Name/value pairs associated with the authenticated user's session. These values are generally added when a session is first created by registered SessionEventHandlers. These values are segregated by "namespace" to help avoid collisions between applications different applications. (See the Cams Programmer's Guide, Chapter 5 – Programming Cams Sessions for more details).
RequestDispatchStack	The stack of security domains to through which this access control request has traveled. The first name on the stack will be the current security domain, the last name on the stack will be the system security domain.
Permission	The permission within the access control policy that controls access to the requested resource. (This is established by pattern matching on the resource identifier and action(s)). If the indicated permission is not the one you'd like to control access to the resource, then you'll need to refine the resource pattern associated with the desired permission to "best match" the resource.
Permission.Description	The textual description of the permission as specified in access-control-policy.xml.
ResourcePattern	A container for the pattern for resource identifiers that match this permission and possibly the name of a security domain to which access control is to be delegated. Note: even if a resource identifier matches a permission's ResourcePattern, that will not guarantee that the permission will be selected to control access to the resource. In general, the "most specific" matching ResourcePattern will select the permission that applies.
ResourcePattern.Pattern	The actual resource pattern used to match resource identifiers and choose the best matching permission.
ResourcePattern.Owner	If "THIS", then the permission is enforced by the current security domain. Otherwise, the value indicates the next security domain to which the access control request will be delegated.
Permission.Type	The type of permission. This will match the resource request.Type and corresponds to the permission collection type.
Permission.Actions	The action(s) associated with the permission. Actions are specific to the type of resource. For example, "http" resource actions correspond to the different "HTTP" request types: "GET", "POST", "PUT", "DELETE", etc. A "file" resource type might support "CREATE", "READ", "WRITE", "EXECUTE", "DELETE", etc. The order of actions is not important when comparing with the actions within an access request's resource request. (The developer of a permission assigns a unque

Troubleshooting Cams Access Control

	bit within the ActionMask to each unique action).
Permission.Actions Mask	Decimal and Binary versions of an integer mask used for efficient comparison of actions within resource requests and other permissions. This value will be most useful to programmers creating custom permission types.
Permission.ACR	The identifier of the access control rule that is evaluated to grant or deny access to the resource. If this value is "NONE", then the value of resource pattern.Owner must be populated with the security domain to which access control checks will be delegated.
"ACR evaluated to "true"": ACCESS GRANTED	A message that shows the results after evaluating the access control rule associated with the permission.
AccessResponse	A tag that encapsulates information returned to the access control agent for a given access response.
AccessResponse.status	A code that indicates whether access to the protected resource was: GRANTED or DENIED
AccessResponse.reason	If access to the protected resource was denied, then this value indicates why. (See AccessResponse.getReason()).
AccessResponse.message	If access to the protected resource was denied, then this value contains a textual message describing why access was denied.
LoginParameters	If access to the protected resource was denied because authentication is required, then is block indicates the application-specific login parameters returned to the agent. These values may indicate the address of the login server, login page, etc and are agent-specific.

Table 4 – Debug message sections and corresponding values

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2002 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Securing Cams Communications using Secret Keys

Cams may be configured to use a secret key to encrypt and decrypt sensitive values (like authentication credentials) sent between Cams agents and a Cams server. Standard PKI algorithms: Blowfish, DES, and DESede (triple DES) are available using key sizes of 16, 8, and 24 bytes respectively. Cams includes a utility program for generating site-wide secret key parameters. In addition, steps should be taken to secure the file containing the secret key to keep its value out of the hands of would be hackers.

In summary, the following steps are required to secure a Cams environment using a secret key:

1. [Generate a secret key and associated parameters](#)
2. [Configure the Cams Server with the secret key parameters](#)
3. [Configure Cams Agents with the secret key parameters](#)
4. [Set Configuration File Permissions](#)

Generating a Cams Secret Key

The Cams secret key generator can be launched using scripts available in the CAMS_HOME/bin directory (Use: secretKeyGen.bat for Windows and secretKeyGen.sh for Unix).

To generate a Cams secret key, use command:

Linux/UNIX

```
$CAMS_HOME/bin/secretKeyGen.sh [-a algorithm] [-out file] [-debug]
```

Windows

```
%CAMS_HOME%\bin\secretKeyGen.bat [-a algorithm] [-out file] [-debug]
```

All command line arguments are optional and may have the values shown in Table 1.

Option	Description
-a	This option specifies one of the secret key algorithms: Blowfish , DES , DESede (also known as triple DES), or None . If not provided, Blowfish is the default. Blowfish was invented by Bruce Schneier of Applied Cryptography and is widely accepted as a secure and fast encryption/decryption algorithm. It uses a 16 byte key. DES was invented at RSA and uses a 7 byte (56 bit) key. It is not considered secure due to its key length, which can be cracked using easily available modern computing power. DESede is the "triple DES" algorithm invented by RSA, which uses a 24 byte key to encrypt, decrypt, then encrypt again: using different 8 byte keys for each of the three operations. This algorithm is considered secure, but has serious performance issues compared with Blowfish, which is preferred base on its relative performance and security. None indicates that secret key encryption/decryption should be disabled. WARNING – This mode should be avoided in production environments unless used in conjunction with other security measures (like SSL connectivity between agents and the Cams server).
-out	This option enables output to be written to a specified file. If not provided, output is written to stdout (System.out), which can easily be redirected to file if desired.
-debug	This option turns on debugging, which writes [DEBUG] messages to stderr (System.err).

Table 1 – Cams Secret Key Generator command line arguments.

Secret Key Parameters

The Cams secret key generator writes three parameters to stdout or the specified file:

- `cams.skey.algorithm` – the selected encryption/decryption algorithm
- `cams.skey.key` – a 24 byte secret key encoded in hexadecimal. The number of bytes used for encryption/decryption depend on the selected algorithm: Blowfish=16, DES=7, DESede=24

Securing Cams Communications using Secret Keys

- **cams.skey.iv** – an initialization vector, which wards against dictionary attacks by scrambling the first few bytes of encrypted values. Without an initialization vector, hackers can sometimes see similar starting patterns at the beginning of different encrypted values, making them more susceptible to cracking.

Example 1 shows sample output using the Cams secret key generator without any options specified

```
./secretKeyGen.sh
cams.skey.algorithm=Blowfish
cams.skey.key=ed28f2c7b60e978277d125d774bd25c1cad3c5c1a7f02757
cams.skey.iv=1a5dce1235fd429e
```

Example 1 – Sample Cams Secret Key Parameters generated from a Unix shell script

WARNING – Don't copy and paste these values into your configuration files. Generate your own values and keep them secret!

Configuring a Cams Server Secret Key

Cams server secret key parameters are stored in file: CAMS_HOME/conf/cams.conf.

NOTE – If you change your secret key settings while the Cams server is running, it can only be gracefully shutdown using the old settings. If your Cams server is currently running, you should either gracefully shutdown before reconfiguring cams.conf or decide to copy the current secret key parameters so the Cams server can be gracefully shutdown after modifications. See: [Keeping Old Secret Key Parameters for Future Graceful Shutdowns](#) for more details.

To change the secret key configuration settings, simply insert or update the cams.skey.* parameters in cams.conf as shown in Example 2.

```
...

# Encryption/Decryption Cipher properties:
#
# cams.skey.algorithm - the algorithm to be used when encrypting
#   and decrypting selected values sent to/received from the Cams agents.
#   Valid values include: None, Blowfish, DES, and DESede (triple DES).
#   If None, then selective encryption is disabled. Blowfish uses a 16
#   byte encryption key, DES uses an 8 byte key, and DESede
#   uses a 24 byte key.
#
# cams.skey.key - the secret encryption/decryption key in
#   hexadecimal format. The actual number of bytes used depends on the
#   algorithm, although it is legal to supply more key bytes than needed.
#
# cams.skey.iv - the encryption/decryption initialization vector
#   in hexadecimal format. This should be an 8 byte (16 hex digit) value.
#
# NOTE: Use ${cams.home}/bin/camsSecretKeyGen.bat or camsSecretKeyGen.sh
#   to generate these values.
#
cams.skey.algorithm=Blowfish
cams.skey.key=ed28f2c7b60e978277d125d774bd25c1cad3c5c1a7f02757
cams.skey.iv=1a5dce1235fd429e

...
```

Example 2 – Configuring Cams Server Secret Key Parameters

WARNING – Don't copy and paste these values into your configuration files. Generate your own values and keep them secret!

Keeping Old Secret Key Parameters for Future Graceful Shutdowns

Graceful shutdown of the Cams server is generally initiated using one of the scripts CAMS_HOME/bin/shutdown.bat (Windows) or CAMS_HOME/bin/shutdown.sh (Linux). The shutdown client executed by these scripts reads the Cams server configuration file, which contains the shutdown password and secret key values used to encrypt the password before sending it to the Cams server. If the Cams server is running while you change secret key configuration values, then later when you invoke the shutdown client it will encrypt the shutdown password using the new secret key. The Cams server will be expecting the shutdown password to be encrypted with the old secret key.

Securing Cams Communications using Secret Keys

To avoid this situation, you can:

1. Gracefully shutdown the Cams server before modifying secret key values
2. Keep the old secret key parameters in CAMS_HOME/conf/cams.conf by appending ".old" to property names

The third solution is simple. When editing cams.conf, simply rename the secret key properties as follows:

- cams.skey.algorithm to cams.skey.algorithm.old
- cams.skey.key to cams.skey.key.old
- cams.skey.iv to cams.skey.iv.old

If the shutdown client fails when using the primary secret key parameters, it will look for the "old" secret key parameters and try again.

```
...

# Encryption/Decryption Cipher properties:
#
# cams.skey.algorithm - the algorithm to be used when encrypting
#   and decrypting selected values sent to/received from the Cams agents.
#   Valid values include: None, Blowfish, DES, and DESede (triple DES).
#   If None, then selective encryption is disabled. Blowfish uses a 16
#   byte encryption key, DES uses an 8 byte key, and DESede
#   uses a 24 byte key.
#
# cams.skey.key - the secret encryption/decryption key in
#   hexadecimal format. The actual number of bytes used depends on the
#   algorithm, although it is legal to supply more key bytes than needed.
#
# cams.skey.iv - the encryption/decryption initialization vector
#   in hexadecimal format. This should be an 8 byte (16 hex digit) value.
#
# NOTE: Use ${cams.home}/bin/camsSecretKeyGen.bat or camsSecretKeyGen.sh
#   to generate these values.
#
cams.skey.algorithm=DESede
cams.skey.key=d774bd25c1cad3c5c1a7f02757ed28f2c7b60e978277d125
cams.skey.iv=35fd429e1a5dce12

#
# Remove these old values after Cams Server is gracefully shutdown
#

cams.skey.algorithm.old=Blowfish
cams.skey.key.old=ed28f2c7b60e978277d125d774bd25c1cad3c5c1a7f02757
cams.skey.iv.old=1a5dce1235fd429e

...
```

Example 3 – Renaming Cams Server Secret Key Parameters to "old" values

Configuring a Cams Agent Secret Key

Cafésoft-supplied Cams agents use the same configuration file format and secret key parameters. Simply edit the agent's configuration file using the same values configured under the Cams server, then start or restart the agent.

In addition, agents have numerous ways to authenticate with a Cams server. When a secret key is configured, using the EncryptedUsernamePassword authentication type will ensure that sensitive credentials are encrypted before being sent from an agent to the Cams server. Example 4 shows how a typical agent configuration file configures use of the EncryptedUsernamePassword authentication type.

```
...

#
#---   Configure general CamsClient parameters
#
# cams.client.authentication.type
```

Securing Cams Communications using Secret Keys

```
# The type of authentication the web agent will use to authenticate
# connections that it establishes with the Cams server. Options
# include: UsernamePassword and EncryptedUsernamePassword. The
# encrypted version requires use of a secret key (See: cams.skey.*
# configuration properties).
#
# cams.client.authentication.principal
# The principal the web agent will use to authenticate connections
# it establishes with the Cams server.
#
# cams.client.authentication.credential
# The credential the web agent will use to authenticate connections
# it establishes with the Cams server.
#
# cams.client.authentication.timeout
# The maximum time (in seconds) that the web agent will wait for a
# response from the Cams server.
#
cams.client.class=\
    com.cafesoft.security.common.client.StandardCamsClient
cams.client.authentication.type=EncryptedUsernamePassword
cams.client.authentication.principal=my-agent-username
cams.client.authentication.credential=my-agent-password
cams.client.authentication.timeout=10

...
```

Example 4 – Typical configuration of an agent to use encrypted authentication

Setting Configuration File Permissions

When using secret keys, it is important to set permissions on configuration files to keep their contents from would-be hackers. Specific information on setting Cams file permissions is available in [Hardening Cams Security: Securing Cams Files and Directories](#). In summary, all configuration files and the directories containing them should be owned by the operating system user identity that runs the Cams server or agent and should have read/write permission only for that user. All other users and groups should have read, write, and execute permissions.

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Securing Cams Communications using SSL

Cams is not currently packaged with SSL support of the agent connections to the Cams server due to the complexity of the configuration issues. However, it is possible to configure Cams with SSL support.

Make sure that you review [Hardening Cams: Securing Communications](#). If you require SSL at this time, please contact [Cafésoft support](#) for further instructions.

Packaged support of SSL communications between Cams agents and the server will be provided in a future release.

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Access Control Responses

Cams replies to each access request with a status of pending, granted, or denied. If the response is denied, then a reason is also sent. The response and reason values are logged in the security domain's access control log. These values are useful in debugging and analyzing secure resource requests. This document explains the values you'll find in the access control log.

Access Control Log Format

Example 1 shows a typical access control log entry. The access control log format is:

1. date/time
2. requesting host address
3. sessionId of the user (if authenticated)
4. authenticated user name (if authenticated)
5. login config entry from the security domain's login-config.xml file
6. fully-qualified resource identifier of the requested resource
7. the requested action(s) on the resource
8. response code
9. reason code

```
[10/Dec/2002:13:14:59 -0800] 127.0.0.1
MyCamsServer-examples-145e13a8561341691d65c3580d81f3ab37f870ca
guest http http://localhost:8080/examples/styles/cswbapp.css "GET" 1 -
```

Example 1 – A single access control log entry

Response Codes

Response code reflect the definitive Cams server answer to the access control request. Table 1 shows the response codes Cams returns.

Value	Description
0	Access control decision for the resource is in progress
1	Access to the resource is granted
2	Access to the resource is denied

Table 1 – Cams access control response codes

Reason Codes/Quantifiers

A reason code is returned with each access control response to provide additional context. Table 2 shows the possible reason codes.

Value	Description
0	Not applicable
1	General error, probably due to a misconfiguration
2	The remote host IP address is not valid
3	The remote hostname is not valid
4	The agent making the request is not authorized
5	An unknown security domain was referenced
6	An unknown resource type was referenced
7	An invalid resource identifier was specified
8	An unrecognized action was requested on a resource
9	Access was denied unconditionally
10	Authentication is required
11	Authentication is required but the session expired
12	An error occurred while evaluating an access control rule
13	Confidentiality (SSL/TLS connection) is required

Access Control Responses

14	The session id submitted was invalid
15	Authentication is required, but the login configuration for the specified login config entry could not be found
16	Use the default bias (either granted or denied) because no permission was protecting the requested resource
17	A general transport error occurred. Something within the response was corrupted.
18	Access was granted conditionally
19	Access was granted unconditionally
20	Access was denied conditionally

Table 2 – Cams access control reason codes

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Regular Expressions

Cams supports regular expressions for use with pattern matching on various values. For example, you can use regular expressions to define the <host> and <address> child elements of the <host acr> access control rule. Regular expressions can be quite complex, but their usage and scope within Cams should be somewhat simplified. This document introduces newbies to the basics of regular expressions and provides some examples for use with Cams.

Regular Expression Syntax

Regular expressions (regex's) are sets of symbols and syntactic elements used to match patterns of text. The simplest form of a regular expression is a literal string, such as security or programming. Regular expression matching also allows you to test whether a string fits into a specific syntactic form, such as an email address.

Text

Some metacharacters match single characters. Other notations enable you to work with entire text strings.

Pattern	Description
.	Matches any single character
[chars]	Matches any character (chars) between the brackets
[^chars]	Matches any character (chars) except those listed between the brackets
\char	Escape that particular char, for instance, to specify reserved chars such as ".[]()"
text1 text2	Alternative: text1 or text2
(text)	Grouping of text

Quantifiers

The regular expression syntax provides metacharacters which specify the number of times a particular character should match.

Pattern	Description
?	Matches any character zero or one times
*	Matches the preceding element zero or more times
+	Matches the preceding element one or more times
{num}	Matches the preceding element num times
{min, max}	Matches the preceding element at least min times, but not more than max times

Anchors

Often, you need to specify the position at which a particular pattern occurs. This is often referred to as anchoring the pattern.

Pattern	Description
^	Matches at the start of the line
\$	Matches at the end of the line
\<	Matches at the beginning of a word
\>	Matches at the end of a word
\b	Matches at the beginning or the end of a word
\B	Matches any character not at the beginning or end of a word

See the Javadoc for [java.util.regex.Pattern](#) for a more complete listing of regular expression anchors available with Cams.

Cams Examples

Cams uses regular expressions to match DNS hostnames and IP address. For example, suppose you want to match only hosts from the "gov" domain. You could use:

Regular Expressions

```
<allow-host>  
<host>^.*gov</host>  
</allow-host>
```

This expression matches any string starting at the beginning of the line that ends with "gov". Or, you want to deny access to any host not in the 192.168.0 address range. You could use:

```
<deny-address>  
<host>192.168.0.*</host>  
</deny-address>
```

This expression matches any text string that starts with "192.168.0". This example demonstrates that you should be careful with the use of the dot metacharacter (".") with hostnames and IP addresses, but that it usually provides the results you desire. In this case, the regular expression's trailing dot-asterick (".*") matches any characters that follow the string "192.168.0". The fact that the next character of an IP address is a dot is only a coincidence. In fact, as a regular expression, all hostname and IP address dots match any character. If you want to match the dot character instead of using it as a wildcard you must escape it:

```
<deny-address>  
<host>192\.168\.0\.*</host>  
</deny-address>
```

Resources

For a quick tutorial on regular expressions see [Using Regular Expressions](#) by Stephen Ramsay, Assistant Director Electronic Text Center, University of Virginia.

A good Java-centric regular expression article named [Regular Expressions and the Java Programming Language](#) is found at the Java Developer's Connection.

If you are interested in more detailed information about regular expressions and their variants (POSIX regex, Perl regex, etc.) read the following dedicated book to this topic:

[Mastering Regular Expressions](#)
Jeffrey E.F. Friedl
Ntshell Handbook Series
O'Reilly & Associates, Inc. 1997
ISBN 1-56592-257-3

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Running Cams as a Service

Cams is run by default using the command-line scripts `runcams.bat/runcams.sh` to start and `shutdown.bat/shutdown.sh` to stop the server. However, you may want to run Cams as a Microsoft Windows NT/2000/XP service or as a Linux service/daemon. Doing so allows you to configure it to automatically start when the system is booted, as well as set some run-time priorities. This document provides instructions on how to use the [Java Wrapper](#) to do so, which provides some additional benefits including a way to monitor and recover from any virtual machine anomalies.

Installation

[Download the Wrapper](#) and uncompress the archive into the directory of your choice. The installation and configuration documented in this chapter is tested with the Java Wrapper version 2.2.8 under Windows NT 4.0 with Service Pack 6a and Redhat Linux 7.3. You can download a zip or tar file.

Windows

Copy the `Wrapper.exe` file from the `WRAPPER_HOME\bin` directory into the `CAMS_HOME\bin` directory.

```
copy %WRAPPER_HOME%\bin\Wrapper.exe %CAMS_HOME%\bin
```

Template batch files to execute `Wrapper.exe` are available in `WRAPPER_HOME\src\bin`.

- `App.bat.in` – Uses the `"-c"` parameter to run Cams with the Wrapper from the console
- `InstallApp-NT.bat.in` – Uses the `"-i"` parameter to install Cams as a Windows NT/2000 service
- `UninstallApp-NT.bat.in` – Uses the `"-r"` parameter to uninstall Cams as a Windows NT/2000 service

Copy these three files into the `CAMS_HOME\bin` directory and rename to something easier to remember such as:

```
cd %CAMS_HOME%
copy %WRAPPER_HOME%\src\bin\App.bat.in runCamsConsole.bat
copy %WRAPPER_HOME%\src\bin\InstallApp-NT.bat.in installCamsService.bat
copy %WRAPPER_HOME%\src\bin\UninstallApp-NT.bat.in removeCamsService.bat
```

Next, create a new directory in `CAMS_HOME` named `native` and copy `Wrapper.DLL` to it.

```
mkdir %CAMS_HOME%\native
copy %WRAPPER_HOME%\lib\Wrapper.dll %CAMS_HOME%\native
```

Now copy `wrapper.jar` into the `CAMS_HOME/lib` directory.

```
copy %WRAPPER_HOME%\lib\wrapper.jar %CAMS_HOME%\lib
```

Finally, copy the template wrapper configuration file `wrapper.conf.in` from the `WRAPPER_HOME\src\conf` directory into the `CAMS_HOME\conf` directory, renaming it to `wrapper.conf`.

```
copy %WRAPPER_HOME%\src\conf\wrapper.conf.in %CAMS_HOME%\conf\wrapper.conf
```

NOTE: You can save time by using [wrapper.conf](#) which is already configured as outlined in the configuration instructions below.

Linux

Copy the wrapper binary file from the `WRAPPER_HOME\bin` directory into the `CAMS_HOME\bin` directory and make sure that it has execute privileges.

```
cp $WRAPPER_HOME/bin/wrapper $CAMS_HOME/bin
chmod 755 $CAMS_HOME/bin/wrapper
```

Copy the template script file `sh.script.in` from the `WRAPPER_HOME/src/bin/` directory into the `/etc/rc.d/init.d` directory, renaming it to `"cams"`, and making sure that it has execute privileges.

```
cp $WRAPPER_HOME/src/bin/sh.script.in /etc/rc.d/init.d/cams
chmod 755 /etc/rc.d/init.d/cams
```

Running Cams as a Service

NOTE: The Java Wrapper documentation recommends use of the `bash.script.in` template but our tests indicate that the `sh.script.in` file works best when installing a Linux service.

Edit the `cams` script file. At the top of the file, edit the comments with values that will allow the Linux `chkconfig` utility to manage the service.

```
#!/bin/bash# chkconfig: 345 20 80
# description: Provides centralized authentication and access control for \
# secure network resources.
# processname: cams
# pidfile: /var/run/cams.pid
# config: /var/cams/conf/wrapper.conf
```

Below this in the Application section, you will find two tokens which need to be replaced. Replace the token `"@app.name@"` with `"cams"`, and `"@app.long.name@"` with `"Cams Server"`.

```
# Application
APP_NAME="cams"
APP_LONG_NAME="Cams Server"
```

Also, replace the two relative Wrapper environment variables with explicit paths for your installation as shown here (if you do not do this, you need to copy the `realpath` binary file from `WRAPPER_HOME/bin` to `CAMS_HOME/bin`).

```
# Fully qualified Wrapper directories
WRAPPER_DIR="/var/cams"
WRAPPER_CMD="$WRAPPER_DIR/bin/wrapper"
WRAPPER_CONF="$WRAPPER_DIR/conf/wrapper.conf"
cd $WRAPPER_DIR/bin
```

Create a new directory in `CAMS_HOME` named `native` and copy `WRAPPER_HOME/lib/libwrapper.so` to it.

```
mkdir $CAMS_HOME/native
cp $WRAPPER_HOME/lib/libwrapper.so $CAMS_HOME/native
```

Copy the required `wrapper.jar` library from the `WRAPPER_HOME/lib` directory into the `CAMS_HOME/lib` directory.

```
cp $WRAPPER_HOME/lib/wrapper.jar $CAMS_HOME/lib
```

Finally, copy the template wrapper configuration file `wrapper.conf.in` from the `WRAPPER_HOME/src/conf` directory into the `CAMS_HOME/conf` directory, renaming it to `wrapper.conf`.

```
copy $WRAPPER_HOME/src/conf/wrapper.conf.in $CAMS_HOME/conf/wrapper.conf
```

Note: You can save time by using [wrapper.conf](#) which is already configured as outlined in the configuration instructions below.

Configuration

Now that you've installed the wrapper, this section describes how to configure `wrapper.conf` to start and stop Cams. As it turns out, the Cams start and stop scripts under both Windows and Linux work well with the `WrapperStartStopApp` helper class.

Open `CAMS_HOME/conf/wrapper.conf` in your text edit. The first parameter is the command to execute the java virtual machine. It's safest to give the full path, but you can also use a relative path if you have environment variables configured.

```
# Java Application
wrapper.java.command=/j2sdk1.4.0_02/bin/Java
```

Next, you specify that you'll be using the `WrapperStartStopApp` helper class as the main class.

```
# Java Main class
wrapper.java.mainclass=com.silveregg.wrapper.WrapperStartStopApp
```

The next section build the class path. For the default Cams installation, the following is sufficient.

Running Cams as a Service

```
# Java Classpath (include wrapper.jar) Add class path elements as
# needed starting from 1
wrapper.java.classpath.1=.
wrapper.java.classpath.2=./classes
wrapper.java.classpath.3=./lib/*jar
```

Now specify the location of the native library you installed.

```
# Java Library Path (location of Wrapper.DLL or libwrapper.so)
wrapper.java.library.path=./native
```

Add additional JVM parameters required by Cams.

```
# Java Additional Parameters
wrapper.java.additional.1=-server
wrapper.java.additional.2=-Dcams.home=..
wrapper.java.additional.3=-DvalidateXML=true
```

Specify the heap parameters.

```
# Initial Java Heap Size (in MB)
wrapper.java.initmemory=32

# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=64
```

You do not need to change the Wrapper port unless it conflicts with an existing use of the port. The logging parameters can also be maintained in their default state. You should consult the wrapper documentation to learn more about [logging parameters](#).

This completes the general configuration, the next two sections are platform specific.

Windows

Skip the the section of wrapper.conf with the following header:

```
*****
# Wrapper Logging parameters
*****
```

You should not modify any of these parameters when an application using this configuration file has been installed as a service. Uninstall the service before modifying this section. The service can then be reinstalled with the new parameters.

Set the service name, display name, and description (description is only used in Windows 2000/XP, not in NT).

```
# Name of the service
wrapper.ntservice.name=CamsServer

# Display name of the service
wrapper.ntservice.displayname=Cams Server

# Description of the service
wrapper.ntservice.description=Cams Server with Wrapper Service
```

Set the names of any other services which must be running before the Cams service can be started. Stopping any of the listed services will also stop this service. Add dependencies as needed starting from 1. For example, there are no default services required, but you may be using a database service that needs to be started for Cams to work properly.

```
# Service dependencies. Add dependencies as needed starting from 1
#wrapper.ntservice.dependency.1=
```

Specify the mode in which the service is installed. AUTO_START starts the service automatically when the system is rebooted. Or DEMAND_START which requires that the service me started manually.

```
# Mode in which the service is installed. AUTO_START or DEMAND_START
```

Running Cams as a Service

```
wrapper.ntservice.starttype=AUTO_START
```

Specifies the priority at which the Wrapper and its JVM will be run at when run as a service or as a console application. Possible values are NORMAL, LOW, HIGH, and REALTIME. Defaults to NORMAL.

```
# Priority at which the service is run. NORMAL, LOW, HIGH, or REALTIME
wrapper.ntservice.process_priority=NORMAL
```

Linux

Skip the the section of wrapper.conf with the following header:

```
*****
# Wrapper Unix daemon parameters
*****
```

There is only one parameter that you can specify, which is the file to write process ID to.

```
# File to write process ID to
wrapper.pidfile=/var/run/cams.pid
```

Testing

You should now be able to run Cams using the scripts installed above.

On Windows, you can use runCamsConsole.bat to test the configuration from the console before installing it as a service. Once your testing is complete, execute installCamsService.bat. You should see the service appear in the service manager. You can now start and stop the service using the start and stop buttons in the service manager.

On Linux, use the service command to start/stop/restart Cams.

```
service cams start
```

You can also use the "chkconfig --list" command to change startup priorities.

For more information on how to configure and use the Java Wrapper, see the [online documentation](#) (also available in the download).

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Support

Cams product license purchases offer 30 days of free technical product support and software update services. These services include:

- Access to the Cams Knowledgebase
- Guaranteed paid support response times
- Installation and integration help
- Access to product updates
- Online issue tracking

Technical Support Policy

Cafésoft provides support from 8am PST to 5pm PST, Monday through Friday excluding holidays. Support beyond those hours may be provided if staff is working beyond the standard hours.

Guaranteed response time for paid support customers is 24 business hours from the time of receipt. Standard response time for customers without currently valid support contracts is not guaranteed. If you require faster response times, you should purchase a support contract or pay per incident.

NOTE: Response times are for initial responses — not the time it may take to resolve the problem.

Free Support Options

- Knowledgebase
- Documentation
- Pre-sales technical questions

Paid Coverage

- All free support coverage options
- Remote assistance setting up Cams with a particular network topology
- Fixing incorrect settings in a Cams server or Cams agent setup
- Installation support, which includes everything needed to get Cams running on the machine except the operating system installation and configuration

Pricing (in US Dollars)

Support contracts are renewed annually based on the following fee structure:

- Per Incident Support – \$75
 - ◆ 5 Incident Block – \$325
 - ◆ 10 Incident Block – \$600
 - ◆ 20 Incident Block – \$1,000
- 1 Year Unlimited Support – 20% of license cost

Support Process

All Cams support issues are initiated by completing and submitting an issue tracker form found on the [Cams User Portal](#). The customer completes the fields including a severity grade:

- Fatal – The problem prohibits use of the software
- Bug – The problem prohibits the normal software use, but the customer can work around it
- Enhancement – The problem is a request to make the software do something it does not currently do

An incident number is created upon submission and the case remains open until explicitly closed. Support will be provided until it is determined that:

- The problem has been resolved that allows normal use of the product
- The installation environment is not supported
- The software being installed is modified or corrupt
- The issue is a nonfatal bug, the incident then becomes a bug report issue
- The user is attempting to do something that the software was not designed to do, the incident then become an enhancement request

Support

A Cafésoft support representative reviews the problem severity and determines the appropriate course of action for resolution. This may include an update to the software, reinstallation, or referral to a FAQ or documentation. If needed, the CSR will contact the customer by phone or email to gather more information or assist with the resolution.

A support incident first involves determination of the actual problem. The act of determining this is part of the paid support incident and may end in referring the customer to a 3rd party provider. If the problem is with a 3rd party component (database, web server, application server, operating system, customizations, etc.) the support representative will refer the client to the appropriate party for further resolution.

Third-Party Products

Cafésoft provides technical support for its own products, not for 3rd party components or applications that may be required to run with Cams. Cafésoft may optionally provide guidance from experience but support for those products should be obtained from their respective vendors. This includes JDBC drivers, databases, web servers, application servers, operating systems, and hardware.

Product Updates

All registered users are informed of products updates through the Cams news list. Once a product update is made available, all users with current support contracts have immediate access to download the software and send themselves a new license using the [Cams User Portal](#).

Professional Services

Cafésoft professional services are available to help customers with on-site integration and product customizations. Cams is a flexible technology, and no one can help you understand better how to make it work for you than the Cafésoft engineers who designed and wrote it. [Contact Cafésoft](#) to find out more.

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Cafésoft LLC. All rights reserved.

[Back](#) | [Next](#) | [Contents](#)

Cams Administrator's Guide

Glossary

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#)

A

Access Control Pipeline – The Cams server pluggable request processing pipeline with each security domain.

Access Control List – Identifies the users who may access a resource, and the type of access to that resource, that a user is permitted to have. Once a user is authenticated the ACL controls what they are permitted to do.

Access Control Policy – Defines the resources being protected and the rules that control access to them.

Access Control Request – Information about a request by a user to get access to a resource including a type, a resource id, and an action.

Access Control Response – The information return by the Cams server to an agent regard and access control request.

Access Control Service – A Cams server service that grant or denies access to resources based on an access control policy.

Access Control Rule – The implementation or expression of the business logic for controlling access to protected resources.

Access Control Rule Library – Manages access control rule types and instances.

Access Control Value – A Cams server individual processing node within an access control pipeline. By default, the first access control valve configured in Cams logs the request and the last valve (also referred to as the basic valve) uses the security domain–specific access control policy to grant or deny access.

Access Management – The centralized or unified implementation and management of user authentication and entitlement to a site's secure resources.

Agent – Cams software components that delegate security requests to a Cams server. Also known as a plugin. Agents are specific to the application (application agent), web server (web agent), or J2EE server (application server agent) that host them.

Audit – An examination of records and activities to ensure compliance with established security controls, policies, and procedures.

Authentication – Identifies an individual or application through the use of username/password, profiles, digital certificates or other means.

Authentication Pipeline – The Cams server pluggable authentication processing pipeline with each security domain.

Authentication Server – The engine within the Cams server that makes authentication decisions based on a security domain's login configuration.

Authentication Service – A Cams server service that verifies the user identity and establishes a session that exists until the user logs out or the session times out due to inactivity.

Authenticated User – A user that has presented valid and accepted login credentials to a resource controller.

Authentication Valve – A Cams server individual processing node within an authentication pipeline. By default, the first authentication valve logs the authentication request and the last valve (also referred to as the basic valve) attempts authentication based on information contained in an authentication request.

Authorization – Develops rules or policies relating to what information users are allowed to view and manipulate (also know as Access Control).

B

Basic authentication – Internet browser managed base64–encoding the username and password and transmitting the result to the server.

C

Callback – Enables underlying security services to interact with a calling application to retrieve specific authentication data such as usernames and passwords, or to display certain information, such as error and warning messages. See [Callback](#).

Glossary

Callback Handler – Application component that passes authentication credentials from the application to a login module. See [CallbackHandler](#).

Credential – Values (such as a username or password) or tokens (such as a digital certificate) owned by a user and presented to an authentication controller for validation of the user's identity.

E

Engine Layer – Enables a Cams server to host security domain services.

G

Group – A a category of users, classified by common traits to facilitate administration.

L

LDAP – Lightweight Directory Access Protocol. A client–server protocol for accessing a directory service. It runs over TCP and can be used to access a stand–alone LDAP directory service or to access a directory service back–ended by X.509.

Login Configuration – Specifies authentication requirements for a given security domain including login configuration entries, callback handlers, and login parameters.

Login Configuration Entry – Specifies the login modules you will use with Cams. Because LoginModules are pluggable, you can implement them without modification to Cams. Because they are stackable, you can specify how authentication to one or more LoginModules is required to access any resource.

Login Module – The Cams mechanism by which callers prove that they are acting on behalf of specific users or systems. See [LoginModule](#).

Logical Operator – The three Boolean operators AND, OR, and NOT that gather or separate things into neat piles depending on how you use them. Cams uses logical operators between, and in some cases preceding, access control rules.

N

Network Adapter Layer – Enables a Cams server to offer services on different TCP/IP ports and to support network clients that speak different protocols.

P

Permission – Associates a set of resources (defined using a resource pattern) with one of two possible actions: an access control rule that will be evaluated to grant or deny access to the resource, or a security domain to which access control will be delegated.

Policy Server – The components within the Cams server that make access control decisions based on rules and and permissions defined in security domains.

Principal – Any entity such as an individual user, a login id, or groups to which a user belongs.

R

Remote Address – The IP address of the computer on which the browser or client application is running.

Remote Host – The fully qualified DNS hostname of the computer on which the browser or client application is running.

Resource – Content including web pages, files, datasources, Enterprise Java Beans, and more that are network accessible.

Roles – A working description of a user assigned to a user or group at application deployment time. Roles provide users access to application resources or enable programmatic decisions.

S

Security Domain – Enables access management to be partitioned according to organizational or physical boundaries, different security domains may be securely configured and managed by different individuals, departments, and companies.

Glossary

Security Domain Registry – Maintains basic information about each security domain known to Cams, including the name and location of configuration metadata.

Service Manager Service – A Cams server service that enables custom security domain–specific services to be used/reused via programmer's APIs.

Session – The Cams server metadata assigned to a currently authenticated user.

Session Access Service – A Cams server service that provides information about authenticated users to agents.

Session Control Service – A Cams server service that enables modification and explicit closure (logout) of user sessions.

Session Management – The process of capturing and changing metadata about an authenticated user throughout the login.

Session Manager Service – A Cams server service that manages an authenticated user's session and expires it if inactive for a configurable period.

Session Object – The programmatic object where a user's session metadata is persisted.

Service Provider Layer – Enables a Cams server to provide security services like authentication and access control.

Single Sign–On – Enables a user to authenticate on one web server and access resources hosted on other web servers (or other virtual hosts within the same web server) without having to re–authenticate.

Subject – The container that holds authentication information about the user or service being authenticated, including relevant principals and credentials.

T

Trace Logger – A centralized, security domain specific component that logs information about the startup, shutdown, warnings, and errors of it's services.

U

Users – Accounts that usually represents a person (but could be a system).

User Repository – A LDAP server, a database, or file containing users, passwords, groups, and roles.

[Back](#) | [Next](#) | [Contents](#)

© Copyright 1996–2003 Caféssoft LLC. All rights reserved.