Spring Security CAS Plugin - Reference Documentation

Burt Beckwith

Version 3.0.0

Table of Contents

1. Introduction to the Spring Security CAS Plugin

The CAS plugin adds CAS single sign-on support to a Grails application that uses Spring Security. It depends on the Spring Security Core plugin.

Once you have configured a CAS server and have configured your Grails application(s) as clients, you can authenticate to any application that is a client of the CAS server and be automatically authenticated to all other clients.

1.1. Release History

- Version 3.0.0
  - released December 8, 2015
- Version 2.0.0
  - released December 7, 2015
- Version 3.0.0.M1
  - released September 23, 2015
- Version 2.0-RC1
  - released November 11, 2013
- Version 1.0.4
  - released July 11, 2012
- Version 1.0.3
  - released July 4, 2012
- Version 1.0.2
  - released February 12, 2011
- Version 1.0.1
  - released September 1, 2010
- Version 1.0
  - released July 27, 2010
- Version 0.1
  - released June 18, 2010

2. Usage

Configuring your CAS server is beyond the scope of this document. There are many different approaches and this will most likely be done by IT staff. It's assumed here that you already have a running CAS server.

CAS is a popular single sign-on implementation. It's open source and has an Apache-like license, and is easy to get started with but is also highly configurable. In addition it has clients written in Java, .Net, PHP, Perl, and other languages.

2.1. Installation

There isn't much that you need to do in your application to be a CAS client. Add a dependency in `build.gradle` for this plugin:

```
dependencies {
   ...
   compile 'org.grails.plugins:spring-security-cas:3.0.0'
   ...
```

then configure any required parameters and whatever optional parameters you want in application.yml or application.groovy. These are described in detail in the Configuration section but typically you only need to set these properties:

```
grails:
   plugin:
      springsecurity:
         cas:
            loginUri: /login
            serviceUrl: http://localhost:8080/login/cas
            serverUrlPrefix: https://your-cas-server/cas
            proxyCallbackUrl: http://localhost:8080/secure/receptor
            proxyReceptorUrl: /secure/receptor
```

2.2. Single Signout

Single signout is enabled by default and enables signing out for all CAS-managed applications with one logout. This works best in the plugin when combined with the `afterLogoutUrl` parameter, for example:

```
grails:
   plugin:
      springsecurity:
         logout:
            afterLogoutUrl: https://your-cas-server/cas/logout?url=http://localhost:8080/
```

With this configuration, when a user logs out locally by navigating to `/logout/` they'll then be redirected to the CAS server's logout URL. This request includes a local URL to redirect back afterwards. When the whole process is finished they'll be logged out locally and at the CAS server, so subsequent secure URLs at the local server or other CAS-managed servers will require a new login.

If you don't want the single signout filter registered, you can disable the feature:

```
grails:
   plugin:
      springsecurity:
         cas:
            useSingleSignout: false
```

3. Configuration

There are a few configuration options for the CAS plugin.

All of these property overrides must be specified in `grails-app/conf/application.yml` (or `application.groovy`) using the `grails.plugin.springsecurity` suffix, for example

```
grails:
   plugin:
      springsecurity:
         cas:
            serverUrlPrefix: https://cas-server/cas
```

Name	Default	Meaning
cas.active	`true`	whether the plugin is enabled (e.g. to disable per-environment)
cas.serverUrlPrefix	`null`, must be set	the 'root' of all CAS server URLs, e.g. `https://cas-server/cas`
cas.serverUrlEncoding	'UTF-8'	encoding for the server URL

Name	Default	Meaning
cas.loginUri	`null`, must be set	the login URI, relative to `cas.serverUrlPrefix`, e.g. `/login`
cas.sendRenew	`false`	if true, ticket validation will only succeed if it was issued from a login form, but will fail if it was issued from a single sign-on session. Analogous to `IS_AUTHENTICATED_FULLY` in Spring Security
cas.serviceUrl	`null`, must be set	the local application login URL, e.g. `http://localhost:8080/login/cas`
cas.key	'grails-spring-security-cas', should be changed	used by `CasAuthenticationProvider` to identify tokens it previously authenticated
cas.artifactParameter	`'ticket'`	the ticket login url parameter
cas.serviceParameter	`'service'`	the service login url parameter
cas.filterProcessesUrl	'/login/cas'	the URL that the filter intercepts for login
cas.proxyCallbackUrl	`null`, should be set	proxy callback url, e.g. 'http://localhost:8080/secure/receptor'
cas.proxyReceptorUrl	`null`, should be set	proxy receptor url, e.g. '/secure/receptor'
cas.useSingleSignout	`true`	if `true` a `org.jasig.cas.client.session.SingleSignOutFilter` is registered