



SPRING-SECURITY-CAS

spring-security-cas - Reference Documentation

Authors: Burt Beckwith

Version: 1.0.1

Table of Contents

1. Introduction	3
1.1 History	3
2. Usage	4
3. Configuration	5

1. Introduction

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 History

- Version 1.0.1
 - released 09/01/2010
- Version 1.0
 - released 07/27/2010
- Version 0.1
 - released 06/18/2010

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.

There isn't much that you need to do in your application to be a CAS client. Just install this plugin, and configure any required parameters and whatever optional parameters you want in Config.groovy. These are described in detail in [Chapter 3](#) but typically you only need to set these properties

```
grails.plugins.springsecurity.cas.loginUri = '/login'
grails.plugins.springsecurity.cas.serviceUrl = 'http://localhost:8080/your-app-name/j_spring
grails.plugins.springsecurity.cas.serverUrlPrefix = 'https://your-cas-server/cas'
grails.plugins.springsecurity.cas.proxyCallbackUrl = 'http://localhost:8080/your-app-name/se
grails.plugins.springsecurity.cas.proxyReceptorUrl = '/secure/receptor'
```

where "your-app-name" is the Grails application context (will be blank if deployed as the default context) and "your-cas-server" is the name of your CAS server.

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.plugins.springsecurity.logout.afterLogoutUrl =
    'https://your-cas-server/cas/logout?url=http://localhost:8080/your-app-name/'
```

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 to 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.plugins.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/Config.groovy` using the `grails.plugins.springsecurity` suffix, for example

```
grails.plugins.springsecurity.cas.serverUrlPrefix =  
    'https://cas-server/cas'
```

Name	Default	Meaning
cas.serverUrlPrefix	null, must be set	the 'root' of all CAS server URLs, e.g. <code>https://cas-server</code>
cas.loginUri	null, must be set	the login URI, relative to <code>cas.serverUrlPrefix</code> , e.g. <code>/login</code>
cas.sendRenew	false	if true, ticket validation will only succeed if it was issued from a single sign-on session. Analogous to <code>IS_AUTHENTICATED_FULLY</code> in Spring Security
cas.serviceUrl	null, must be set	the local application login URL, e.g. <code>http://localhost:8080/myapp/j_spring_cas_security_check</code>
cas.key	'grails-spring-security-cas', should be changed	used by <code>CasAuthenticationProvider</code> to identify token authenticated
cas.artifactParameter	'ticket'	the ticket login url parameter
cas.serviceParameter	'service'	the service login url parameter
cas.filterProcessesUrl	'/j_spring_cas_security_check'	the URL that the filter intercepts for login
cas.proxyCallbackUrl	null, should be set	proxy callback url, e.g. <code>'http://localhost:8080/myapp/secure/receptor'</code>
cas.proxyReceptorUrl	null, should be set	proxy receptor url, e.g. <code>'/secure/receptor'</code>
cas.useSingleSignout	true	if true a <code>org.jasig.cas.client.session.SingleSignOut</code> registered in <code>web.xml</code>