EUROPEAN COMMISSION
DIRECTORATE-GENERAL
INFORMATICS
Directorate D - Digital Services
Trans-European Services

European Commission

ECAS Client Installation and Configuration Guide - Advanced For WebLogic Server 10.3 and above

Date:	26/04/2018
Version:	4.26
Authors:	DIGIT ECAS DEVELOPMENT
Revised by:	
Approved by:	
Public:	
Reference Number:	

TABLE OF CONTENTS

1. INTRODUCTION 1

2. THE CONFIGURATION OPTIONS 1

2.1. The EcasIdentityAsserterV2 MBean Configuration 1

2.1.1. excludedContextPaths 3

2.1.2. ConfigFileReloadingCheckIntervalMillis 3

2.2. The configuration of the ECAS Client properties from within the web.xml deployment descriptor (phased out) 4

3. THE CONFIGURATION FILE 5

3.1. Configuration file name and location 5

3.1.1. The static configuration file (deprecated) 5

3.1.1.1. Configuration file inside the archive 5

3.1.1.2. Configuration file outside of the archive 6

3.1.2. Context-specific configuration file (recommended) 6

3.2. Configuration File Templates 7

3.3. System Properties in configuration files 21

3.4. Setting the right configuration property 21

3.5. Configuration properties 24

3.5.1. configFile 24

3.5.2. ~~configurationId~~ (obsolete) 24

3.5.3. serverName 24

3.5.4. serviceUrl (not recommended) 26

3.5.5. ecasServerDirectHostName 26

3.5.6. ecasServerDirectOneWaySslPort 28

3.5.7. ecasServerDirectTwoWaySslPort 28

3.5.8. ecasServerReverseProxyHostName 29

3.5.9. ecasServerReverseProxyPort 29

3.5.10. ecasBaseUrl 29

3.5.11. initLoginUrl 30

3.5.12. loginUrl 30

3.5.13. validateUrl 31

3.5.14. assuranceLevel 31

3.5.15. acceptedTicketTypes 32

3.5.16. proxyGrantingProtocol 33

3.5.17. proxyUrl 34

3.5.18. renew 34

3.5.19. serverProtocol 35

3.5.20. serverPort 35

3.5.21. serverSSLPort 35

3.5.22. serverContextPath 36
3.5.23. ~~authorizedProxy~~ (obsolete) 36
3.5.24. authorizedProxies 36
3.5.25. proxyChainTrustHandler 37
3.5.26. baseCallbackUrl 38
3.5.27. proxyCallbackUrl 38
3.5.28. singleLogoutCallbackUrl 39
3.5.29. ~~applicationServer~~ (obsolete) 41
3.5.30. groups 41
3.5.31. acceptStrengths 42
3.5.32. maxConnections 43
3.5.33. connectionTimeout 43
3.5.34. strictSSLHostnameVerification 44
3.5.35. extraGroupHandler 44
3.5.36. authEventListeners 46
3.5.37. configurationOrder 47
3.5.38. initSignatureUrl 47
3.5.39. signatureUrl 48
3.5.40. retrieveSignatureUrl 48
3.5.41. transactionUrl 49
3.5.42. certificateRevocationUrl 49
3.5.43. requestingUserDetails 50
3.5.44. serviceResolver 50
3.5.45. loginDateValidator 52
3.5.46. reSubmitPosts 53
3.5.47. httpRedirector 53
3.5.48. trustNonEcasJEESubject 55
3.5.49. trustedCertificates 55
3.5.50. applicationSecurityLevel 57
3.5.51. negociatePrivateServiceTicket 59
3.5.52. advancedHttpSessionManagement 60
3.5.53. ticketResolver 61
3.5.54. redirectionInterceptors 62
3.5.55. extendedUserDetailsTypeMapper 64
3.5.56. Custom parameters 65
3.6. Reloadable configuration (Java 6) 67

4. THE WEB.XML CONFIGURATION 68

4.1. web.xml template 68
4.2. The security-constraint tag 69
4.3. The security-role tag 69

5. THE WEBLOGIC.XML CONFIGURATION ... **70**

6. THE ERROR PAGE ... **71**

7. LOGOUT.JSP ... **71**

8. CONFIGURING THE LOGGING OF THE ECAS IDENTITY ASSERTER V2 ... **73**

8.1. At a Glance ... 73

8.2. ECAS Client logging with WebLogic logging services ... 73

8.3. Logging of the ECAS Client using Apache log4j ... 75

8.3.1. Using a FilteringClassLoader ... 75

8.3.2. Using a log4j RepositorySelector ... 75

8.3.3. The log4j configuration file ... 76

9. SUBJECT AND USERPRINCIPAL ... **78**

10. CUD GROUPS ... **78**

11. CUSTOM APPLICATION GROUPS ... **79**

11.1. Configure a BEA WebLogic SSPI Authentication Provider ... 79

11.2. Configure a custom ECAS ExtraGroupHandler implementation ... 79

11.2.1. For ecas-config-mywebapp.properties ... 79

11.2.2. For ecas-config-mywebapp.xml ... 79

11.3. Configure a custom UserDetailsExtraGroupHandlerIntf implementation ... 79

12. MULTIFACTORAUTHORIZATIONFILTER ... **80**

12.1. MultiFactorAuthorizationFilter ... 80

12.2. TargetedUserMultiFactorAuthorizationFilter ... 81

13. ANONYMOUS ACCESS (GATEWAY) ... **81**

14. APPLICATION DESCRIPTION AND LOGO ... **82**

14.1. Where is it used? ... 82

14.2. How to register ... 82

14.3. About the logo ... 82

15. DOMAIN PROPAGATION ... **83**

16. LANGUAGE PROPAGATION ... **84**

16.1. Application to ECAS ... 84

16.2. ECAS to Application ... 84

17. ACCESSING THE WEBLOGIC CONSOLE ... **85**

18. SCRIPTCLIENT HELPER CLASS ... **87**

19. SINGLE LOGOUT ... **88**

20. PROGRAMMATIC COOKIE-PATH ... **90**

21. REFERENCE 91

22. APPENDIX I: HOW TO IMPORT SSL CERTIFICATES 92

22.1. SSL Configuration 92

22.1.1. Download the certificate 92

22.1.2. Client configuration 92

23. APPENDIX II: COMMISSIGN PKI CERTIFICATES 93

24. APPENDIX III: SECURITY-CONSTRAINTS 94

25. APPENDIX IV: FREQUENTLY ASKED QUESTIONS (F.A.Q.) 96

25.1. How to have my application groups returned by the ECAS ticket validation response? 96

25.2. Why do I get the exception: User null javax.security.auth.login.LoginException: [Security:090299]Username Not Supplied? 96

25.3. I keep getting a "Peer Not Authenticated" Exception 96

25.4. I have a FailedLoginException: javax.security.auth.login.LoginException: [Security:090300]Identity Assertion Failed: User N/A-30 does not exist 96

25.5. How to enable SSL debugging? 97

25.6. How to display SecurityProviders debug message for WebLogic Server? 97

25.7. When accessing my application, some users receive an error '*Invalid user: "userxxx" is an interinstitutional user*' 97

25.8. How come I am re-authenticated for every resource (pages, images, javascript, CSS, etc) on my site? 97

25.9. Why do I have to authenticate again when connecting to my application? 98

25.10. The ECAS page is quickly flashing or my browser goes into an infinite loop 98

25.11. My application is using Proxy Tickets and gets a PROXY_COMMUNICATION_ERROR error code 98

25.12. I'm using remote EJBs and receive a ClassNotFoundException on the EcasUser Principal 98

25.13. I'm using ECAS Proxy Tickets and I get the exception "The mandatory ECAS Server ProxyURL is not configured" 98

26. APPENDIX V: EXAMPLE CONFIGURATION FILES 100

26.1. Example 1 – relying on the configured ListenAddress 100

26.2. Example 2 – one CUD group and user details 100

26.3. Example 3 – interinstitutional users, custom groups and resubmitting forms 100

26.4. Example 4 – force using only one reverse proxy 101

26.5. Example 5 – server external to the Commission's network 101

26.6. Example 6 – using your own Ecas Mockup Server 101

26.7. Example 7 – using the ECAS Validation environment 102

26.8. Example 8 – using the ECAS Load environment 103

TABLE OF FIGURES

Figure 1 - The EcasIdentityAsserterV2 MBean Provider Specific screen in the WebLogic Server 9.2 Administration Console 2

Figure 2 - Excluded Context Paths MBean Attribute 3

Figure 3- ConfigFileReloadingCheckIntervalMillis MBean attribute in WebLogic Console 4

Figure 4 - ecas-config-*mywebapp*.properties template 13

Figure 5 - ecas-config-*mywebapp*.xml Template 20

Figure 6 - ECAS assurance levels per validation URL 32

Figure 7 - ECAS assurance levels per population 32

Figure 8 - TicketBased interface 33

Figure 9 - ProxyChainTrustHandlerIntf interface 37

Figure 10 - ExtraGroupHandlerIntf interface 44

Figure 11 - UserDetailsExtraGroupHandlerIntf interface 45

Figure 12 - DetailedAuthenticationSuccess interface 45

Figure 13 - AuthenticationSuccess interface 45

Figure 14 - UserDetailsAccessible interface 46

Figure 15 - ServiceResolver interface 51

Figure 16 - LoginDateValidatorIntf interface 52

Figure 17 - HttpRedirector interface 54

Figure 18 - trustedCertificates property (properties format) 56

Figure 19 - trustedCertificates property (XML format) 56

Figure 20 - ConfigurationDependent interface 65

Figure 21 - Example of use for custom parameters 66

Figure 22 - Reloading Configuration 67

Figure 23 - Configure WebLogic Logging 74

Figure 24 - log4j.properties Example 77

Figure 25 - Application Description and Logo on the ECAS Login Page 82

Figure 26 - Example using a script user and its related SSL certificate stored in a PKCS12 KeyStore 87

Document History

Version	*Author*	*Date*	*Comment*	*Modif.*
1.0	verfade	21/08/2009	Creation, taken from [ECAS-BASIC]	
1.1	ackxyyo	15/09/2009	Overall revision	All
1.2	ackxyyo	20/11/2009	Application registration – more info	ch. 10
1.3	verfade	15/12/2009	Changes for v. 1.9.1	
1.4	ackxyyo	17/03/2010	Changes for v. 1.10	ch. 11
1.5	verfade	10/04/2010	Changes for v. 1.11	
1.6	verfade	03/06/2010	Changes for v. 1.13	
1.7	lauredo	17/09/2010	Changes for v. 1.13.3	
1.8	verfade	24/09/2010	Added doc on ScriptClient	
1.9	lauredo	08/11/2010	Added examples for Ecas Validation (ecasv) and ECAS Load (ecasl for stress tests)	69,70
1.10	verfade	08/12/2010	Added information on domain propagation.	54
1.11	ackxyyo	15/12/2010	Limited scope of domain propagation (ECAS to application), typo correction	1, 54
1.12	lauredo	28/01/2010	Added information about the trustedCertificates property and the section about the reloadable configuration	39-40, 44
1.13	verfade	07/02/2011	Changes for v. 1.16.1	
1.14	lauredo	26/05/2011	Added FAQ entry on Proxy Tickets ProxyURL exception	69-70
1.15	verfade	01/06/2011	Changed CITnet URLs. Changes for v. 1.18.1.	
1.16	lauredo	21/06/2011	Added section about *ConfigFileReloadingCheckIntervalMillis* Added 5 sections about *ecasServerDirectHostName* Added the STRONG_SMS strength	3-4 21-24 34
1.17	verfade	19/07/2011	Changes for v. 1.20.1.	
1.18	donydgr	04/01/2012	Added that registration in ECAS is required for Language propagation Added that the security role must be compatible with *assuranceLevel* property	
1.19	peteror	02/03/2012	Changes for v. 2.5.0.	
3.1	donydgr	14/09/2012	Changes for v. 3.1.0, update ECAS configuration examples (fig. 3-4)	7-19
3.6	lauredo	13/12/2013	Changes for v. 3.6.3	13,48,56-57,70
3.7	catizmi	12/03/2014	Minor corrections	47
3.11.2	catizmi	19/12/2014	Changes for v.3.11.2, general improvements	All
4.3.0	catizmi	12/06/2015	Changes for v.4.3.0	38, 41
4.3.1	lauredo	17/06/2015	Changes for v.4.3.1	
4.3.2	catizmi	29/07/2015	Changes for v.4.3.2	
4.3.4	lauredo	13/10/2015	Changes for v.4.3.4	
4.4.1	lauredo	01/12/2015	Changes for v.4.4.1	
4.5.1	lauredo	07/01/2016	Changes for v.4.5.1	
4.5.2	lauredo	21/01/2016	Changes for v.4.5.2	
4.6.0	lauredo	08/02/2016	Changes for v.4.6.0	21, 38-40
4.8.0	lauredo	18/07/2016	Changes for v.4.8.0	
4.10.1	lauredo	25/10/2016	Changes for v.4.10.1	
4.12.0	lauredo	15/12/2016	Changes for v.4.12.0	
4.12.1	lauredo	19/01/2017	Changes for v.4.12.1	
4.13.0	lauredo	13/02/2017	Changes for v.4.13.0	
4.14.0	lauredo	08/03/2017	Changes for v.4.14.0	
4.15.0	lauredo	20/03/2017	Changes for v.4.15.0	
4.16.1	lauredo	11/05/2017	Changes for v.4.16.1	
4.26	catizmi	26/04/2018	Updated Single Logout documentation	88
4.26.4	donydgr	30/08/2018	Replaced ecasv (Validation) by ecasa (Acceptance)	102-104

Reference Documents

Code	***Title***
[ECAS-NEWS]	ECAS Client What's New
[ECAS-BASIC]	ECAS Client Installation and Configuration Guide – Basic
[ECAS-ADV]	ECAS Client Installation and Configuration Guide – Advanced (= this document)
[ECAS-TECH]	ECAS Technical Guide
[ECAS-PROXY]	ECAS Proxy
[ECAS-PROXY-JAVA]	ECAS Client Proxy Guide
[GATEWAY]	ECAS Gateway (Peek for SSO)
[SIGNATURE]	ECAS Signature
[ECAS-LANG-PROP]	ECAS Language Propagation
[ECAS-WS]	ECAS Webservices
[ECAS-FORGE]	https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/ECAS+Forge All the above-mentioned documents are available at this location.
[ECAS-FAQ]	https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/ECAS+FAQ
[ECAS-WIKI]	https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/ECAS+for+Developers

Contact:

EC-IAM-SERVICE-DESK@ec.europa.eu
DIGIT-ECAS-DEVELOPMENT@ec.europa.eu

1. INTRODUCTION

In [ECAS-BASIC], we walked you through a basic, no-fuss configuration of your application. We will now cover in details the alternative configuration mechanisms and more fine-grained settings.

2. THE CONFIGURATION OPTIONS

In [ECAS-BASIC], we have used a simple **configuration file** to configure the ECAS Client for WebLogic. Actually, ECAS Client can be configured in different places:

- The *EcasIdentityAsserterV2* MBean
- A configuration file in the classpath or on the file system

The order of precedence between these places is determined by a Provider Specific attribute of the *EcasIdentityAsserterV2* MBean called "*Configuration Order*".

The configuration order has the default value "`mbean << file`". This means that the configuration defined in the *EcasIdenityAsserterV2* MBean is used as the default and is overridden by the configuration defined in the external file.

This property can be used by WebLogic server administrators to provide a default configuration for a whole server where applications only need to override a minimal set of specific properties. Alternatively, the administrator could choose to reverse the configuration order into "`file << mbean`" so that the administrator of the server could override in the MBean some specific settings defined in the configurations of all deployed applications (e.g. the server name).

The configuration file is detailed in chapter 3.

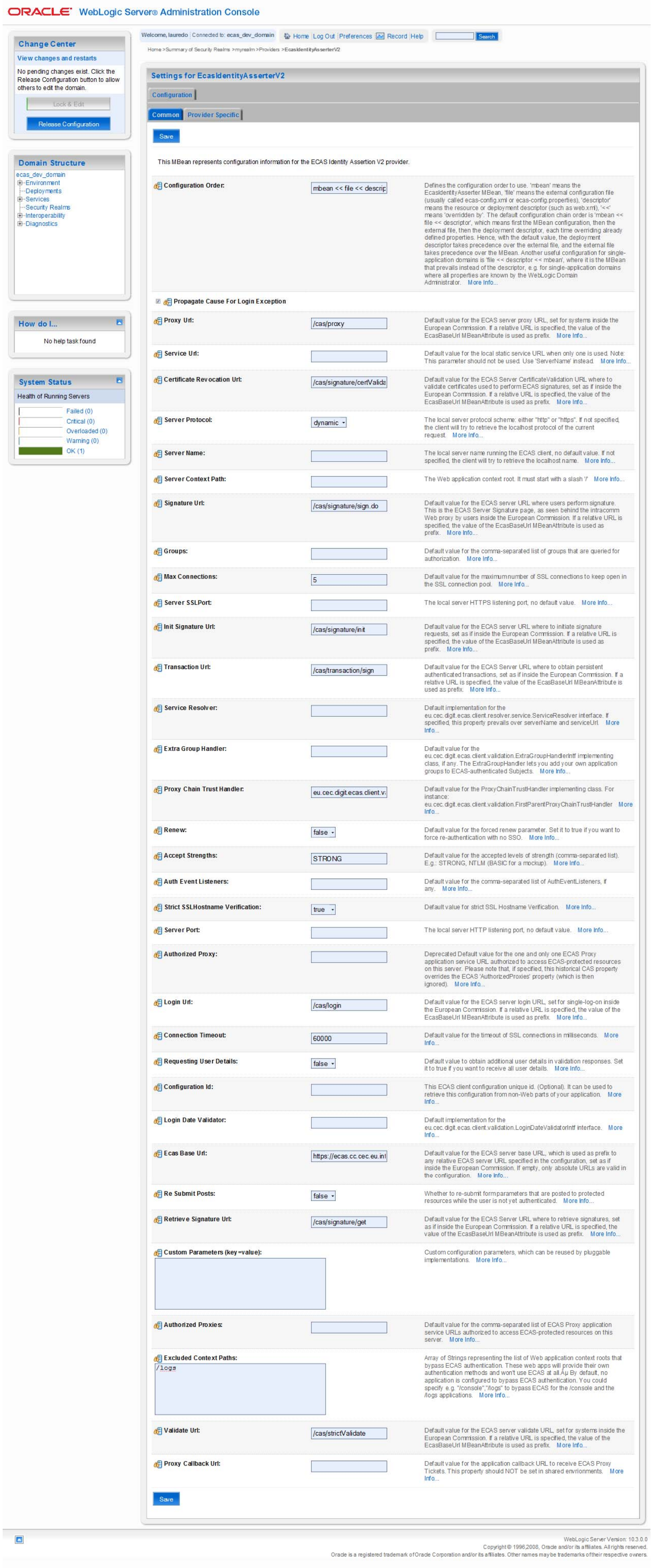
2.1. The EcasIdentityAsserterV2 MBean Configuration

The *EcasIdentityAsserterV2* MBean contains all the attributes needed to configure the ECAS Client. This is intended to provide default values to all the configuration properties for a whole application server. Every application on that server inherits from that base configuration and can choose to override the properties they want in their specific configuration file.

The *EcasIdentityAsserterV2* MBean is usually configured by the WebLogic Server administrator through WebLogic Administration Console or by modifying WebLogic *config.xml* or using WLST.

If you copied the provided "*security.properties*" file into your WebLogic Administration Console installation[1], you can see the contextual description for each MBean attribute on the Provider Specific tab.

[1] See [ECAS-BASIC], chapter 5 "Installing the EcasIdentityAsserterV2".



ORACLE WebLogic Server® Administration Console
Welcome, lauredo | Connected to: ecas_dev_domain
Home | Log Out | Preferences | Record | Help
Search
Home >Summary of Security Realms >myrealm >Providers >EcasIdentityAsserterV2
Change Center
View changes and restarts
Lock & Edit
Release Configuration
Domain Structure
ecas_dev_domain
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
System Status
Health of Running Servers
Failed (0)
Critical (0)
Overloaded (0)
Warning (0)
OK (1)
Settings for EcasIdentityAsserterV2
Configuration
Common
Provider Specific
Save

No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

No help task found

This MBean represents configuration information for the ECAS Identity Assertion V2 provider.

Configuration Order:	mbean << file << descrip	Defines the configuration order to use. 'mbean' means the EcasIdentityAsserter MBean, 'file' means the external configuration file (usually called ecas-config.xml or ecas-config.properties), 'descriptor' means the resource or deployment descriptor (such as web.xml), '<<' means 'overridden by'. The default configuration chain order is 'mbean << file << descriptor', which means first the MBean configuration, then the external file, then the deployment descriptor, each time overriding already defined properties. Hence, with the default value, the deployment descriptor takes precedence over the external file, and the external file takes precedence over the MBean. Another useful configuration for single-application domains is 'file << descriptor << mbean', where it is the MBean that prevails instead of the descriptor, e.g. for single-application domains where all properties are known by the WebLogic Domain Administrator. More Info...
☑ Propagate Cause For Login Exception		
Proxy Url:	/cas/proxy	Default value for the ECAS server proxy URL, set for systems inside the European Commission. If a relative URL is specified, the value of the EcasBaseUrl MBeanAttribute is used as prefix. More Info...
Service Url:		Default value for the local static service URL when only one is used. Note: This parameter should not be used. Use 'ServerName' instead. More Info...
Certificate Revocation Url:	/cas/signature/certValida	Default value for the ECAS Server CertificateValidation URL where to validate certificates used to perform ECAS signatures, set as if inside the European Commission. If a relative URL is specified, the value of the EcasBaseUrl MBeanAttribute is used as prefix. More Info...
Server Protocol:	dynamic	The local server protocol scheme: either "http" or "https". If not specified, the client will try to retrieve the localhost protocol of the current request. More Info...
Server Name:		The local server name running the ECAS client, no default value. If not specified, the client will try to retrieve the localhost name. More Info...
Server Context Path:		The Web application context root. It must start with a slash '/' More Info...
Signature Url:	/cas/signature/sign.do	Default value for the ECAS server URL where users perform signature. This is the ECAS Server Signature page, as seen behind the intracomm Web proxy by users inside the European Commission. If a relative URL is specified, the value of the EcasBaseUrl MBeanAttribute is used as prefix. More Info...
Groups:		Default value for the comma-separated list of groups that are queried for authorization. More Info...
Max Connections:	5	Default value for the maximumnumber of SSL connections to keep open in the SSL connection pool. More Info...
Server SSLPort:		The local server HTTPS listening port, no default value. More Info...
Init Signature Url:	/cas/signature/init	Default value for the ECAS server URL where to initiate signature requests, set as if inside the European Commission. If a relative URL is specified, the value of the EcasBaseUrl MBeanAttribute is used as prefix. More Info...
Transaction Url:	/cas/transaction/sign	Default value for the ECAS Server URL where to obtain persistent authenticated transactions, set as if inside the European Commission. If a relative URL is specified, the value of the EcasBaseUrl MBeanAttribute is used as prefix. More Info...
Service Resolver:		Default implementation for the eu.cec.digit.ecas.client.resolver.service.ServiceResolver interface. If specified, this property prevails over serverName and serviceUrl. More Info...
Extra Group Handler:		Default value for the eu.cec.digit.ecas.client.validation.ExtraGroupHandlerIntf implementing class, if any. The ExtraGroupHandler lets you add your own application groups to ECAS-authenticated Subjects. More Info...
Proxy Chain Trust Handler:	eu.cec.digit.ecas.client.v:	Default value for the ProxyChainTrustHandler implementing class. For instance: eu.cec.digit.ecas.client.validation.FirstParentProxyChainTrustHandler More Info...
Renew:	false	Default value for the forced renew parameter. Set it to true if you want to force re-authentication with no SSO. More Info...
Accept Strengths:	STRONG	Default value for the accepted levels of strength (comma-separated list). E.g.: STRONG, NTLM (BASIC for a mockup). More Info...
Auth Event Listeners:		Default value for the comma-separated list of AuthEventListeners, if any. More Info...
Strict SSLHostname Verification:	true	Default value for strict SSL Hostname Verification. More Info...
Server Port:		The local server HTTP listening port, no default value. More Info...
Authorized Proxy:		Deprecated Default value for the one and only one ECAS Proxy application service URL authorized to access ECAS-protected resources on this server. Please note that, if specified, this historical CAS property overrides the ECAS 'AuthorizedProxies' property (which is then ignored). More Info...
Login Url:	/cas/login	Default value for the ECAS server login URL, set for single-log-on inside the European Commission. If a relative URL is specified, the value of the EcasBaseUrl MBeanAttribute is used as prefix. More Info...
Connection Timeout:	60000	Default value for the timeout of SSL connections in milliseconds. More Info...
Requesting User Details:	false	Default value to obtain additional user details in validation responses. Set it to true if you want to receive all user details. More Info...
Configuration Id:		This ECAS client configuration unique id. (Optional). It can be used to retrieve this configuration from non-Web parts of your application. More Info...
Login Date Validator:		Default implementation for the eu.cec.digit.ecas.client.validation.LoginDateValidatorIntf interface. More Info...
Ecas Base Url:	https://ecas.cc.cec.eu.int	Default value for the ECAS server base URL, which is used as prefix to any relative ECAS server URL specified in the configuration, set as if inside the European Commission. If empty, only absolute URLs are valid in the configuration. More Info...
Re Submit Posts:	false	Whether to re-submit formparameters that are posted to protected resources while the user is not yet authenticated. More Info...
Retrieve Signature Url:	/cas/signature/get	Default value for the ECAS Server URL where to retrieve signatures, set as if inside the European Commission. If a relative URL is specified, the value of the EcasBaseUrl MBeanAttribute is used as prefix. More Info...
Custom Parameters (key=value):		Custom configuration parameters, which can be reused by pluggable implementations. More Info...
Authorized Proxies:		Default value for the comma-separated list of ECAS Proxy application service URLs authorized to access ECAS-protected resources on this server. More Info...
Excluded Context Paths:	/logs	Array of Strings representing the list of Web application context roots that bypass ECAS authentication. These web apps will provide their own authentication methods and won't use ECAS at all.Âµ By default, no application is configured to bypass ECAS authentication. You could specify e.g. "/console","/logs" to bypass ECAS for the /console and the /logs applications. More Info...
Validate Url:	/cas/strictValidate	Default value for the ECAS server validate URL, set for systems inside the European Commission. If a relative URL is specified, the value of the EcasBaseUrl MBeanAttribute is used as prefix. More Info...
Proxy Callback Url:		Default value for the application callback URL to receive ECAS Proxy Tickets. This property should NOT be set in shared envrionments. More Info...

WebLogic Server Version: 10.3.0.0
Copyright © 1996,2008, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Figure 1 - The EcasIdentityAsserterV2 MBean Provider Specific screen in the WebLogic Server 9.2 Administration Console

2.1.1. excludedContextPaths

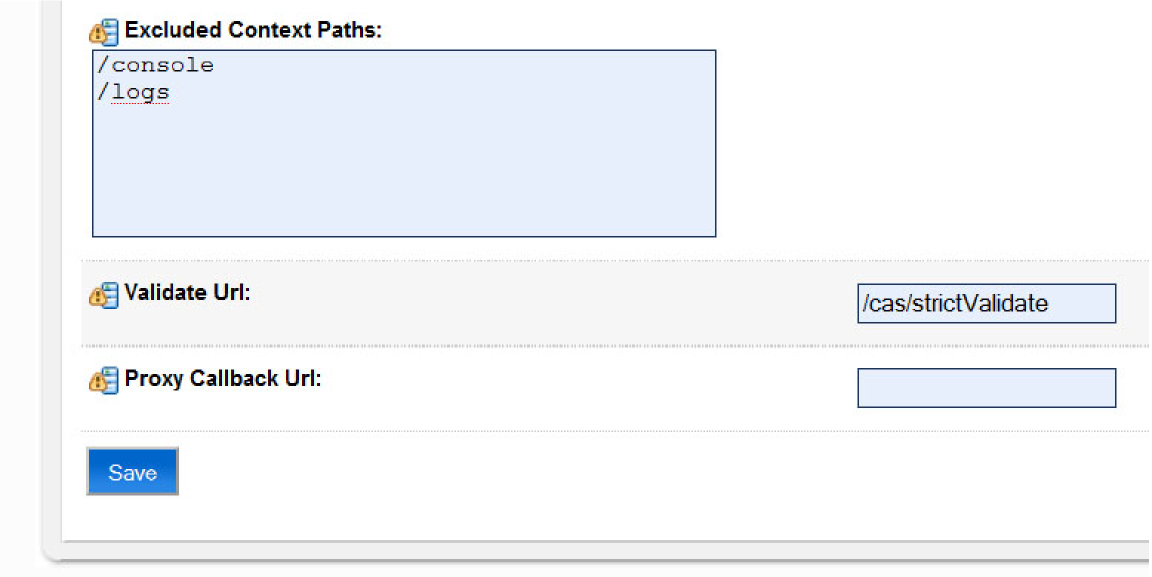
The ECAS Client 1.8.7 introduced an MBean attribute called "*excludedContextPaths*" that may be used by WebLogic Administrators to prevent a Web application from using ECAS-authentication.

This attribute is only available from the *EcasIdentityAsserterV2* MBean.

It allows your WebLogic server to bypass ECAS authentication for the specified context-paths. When a Web application context-path is excluded, the other configured Authentication Providers are called e.g. the DefaultAuthenticator (i.e. WebLogic-defined users) and the EcasIdentityAsserterV2 stays out of the loop.

The *excludedContextPaths* attribute takes a String[] value. In WebLogic Console, this means encoding one value per line.

For example, to bypass ECAS for the Console application (i.e. "/console") and a "/logs" application, specify in the "Provider-specific tab":



Excluded Context Paths:

```
/console
/logs
```

Validate Url: /cas/strictValidate

Proxy Callback Url:

Figure 2 - Excluded Context Paths MBean Attribute

2.1.2. ConfigFileReloadingCheckIntervalMillis

In the ECAS client for *WebLogic Server 10.3*, the configuration files are reloadable and any modification made to them triggers the reloading of the whole ECAS client configuration.

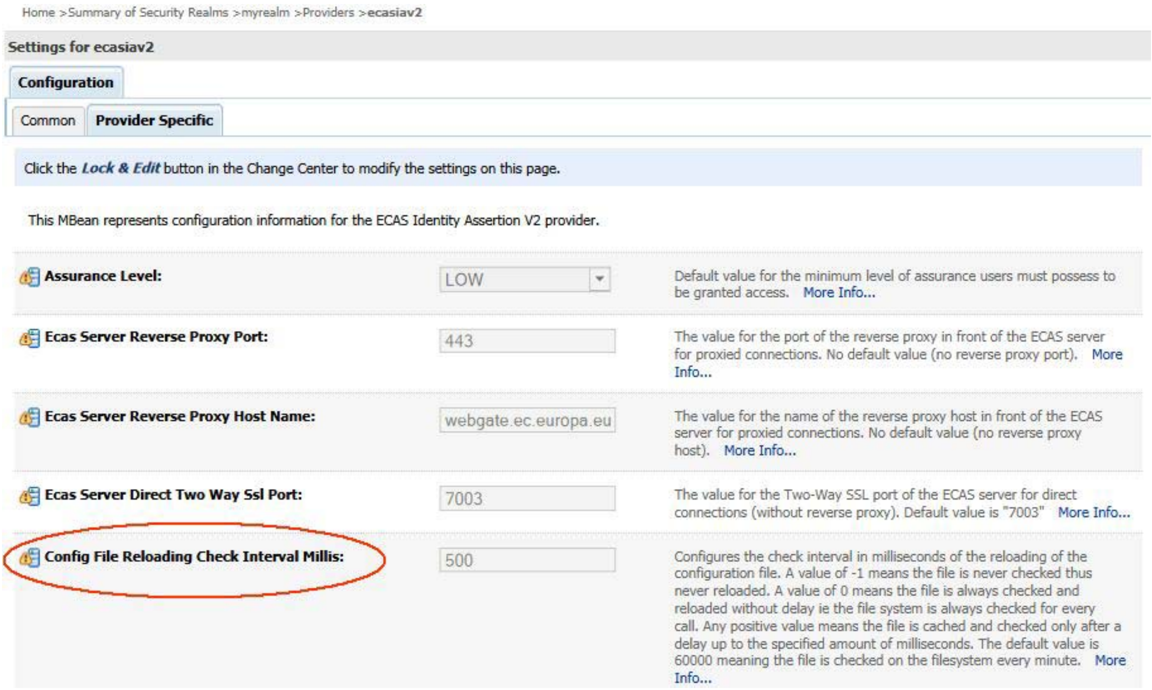
By default, the delay between two consecutive checks on the file system is one minute.

To change this behaviour, you may either use the following System property `eu.cec.digit.ecas.client.configFileReloading.checkIntervalMillis` or use an attribute in the "*Provider Specific*" tab of the Ecas Identity Asserter in WebLogic Server Administration Console called "`Config File Reloading Check Interval Millis`".

This property takes as value an amount of milliseconds.

- A value of `-1` means the file is never checked on the file system and thus never reloaded.
- A value of `0` means the file system is always checked and the files are reloaded without delay.
- Any positive value means the file is cached and checked only after a delay of at least the specify amount of milliseconds. The default value is `60000` meaning the file system is checked for changes every minute.

The ECAS Client 1.16.1 for Java 6 introduced an MBean attribute called "*configFileReloadingCheckIntervalMillis*" that may be used by WebLogic Administrators to control the reloading behaviour of the ECAS client configurations of all the ECAS-protected applications running on a given WebLogic server.



Home >Summary of Security Realms >myrealm >Providers >**ecasiav2**

Settings for ecasiav2

Configuration

Common | **Provider Specific**

Click the ***Lock & Edit*** button in the Change Center to modify the settings on this page.

This MBean represents configuration information for the ECAS Identity Assertion V2 provider.

Assurance Level:	LOW	Default value for the minimum level of assurance users must possess to be granted access. More Info...
Ecas Server Reverse Proxy Port:	443	The value for the port of the reverse proxy in front of the ECAS server for proxied connections. No default value (no reverse proxy port). More Info...
Ecas Server Reverse Proxy Host Name:	webgate.ec.europa.eu	The value for the name of the reverse proxy host in front of the ECAS server for proxied connections. No default value (no reverse proxy host). More Info...
Ecas Server Direct Two Way Ssl Port:	7003	The value for the Two-Way SSL port of the ECAS server for direct connections (without reverse proxy). Default value is "7003" More Info...
Config File Reloading Check Interval Millis:	500	Configures the check interval in milliseconds of the reloading of the configuration file. A value of -1 means the file is never checked thus never reloaded. A value of 0 means the file is always checked and reloaded without delay ie the file system is always checked for every call. Any positive value means the file is cached and checked only after a delay up to the specified amount of milliseconds. The default value is 60000 meaning the file is checked on the filesystem every minute. More Info...

Figure 3- ConfigFileReloadingCheckIntervalMillis MBean attribute in WebLogic Console

To summarize:

- If you want to disable reloading whatsoever, for example in PRODUCTION, specify as JVM argument:
  `-Deu.cec.digit.ecas.client.configFileReloading.checkIntervalMillis=-1`
- If you want to reload the configuration as soon as you modify the files, for example in DEVELOPMENT, specify as JVM argument:
  `-Deu.cec.digit.ecas.client.configFileReloading.checkIntervalMillis=0`
- Other (positive) values are for environments managed by developers but which are under more load and want to avoid stressing the file system

Or you use the same value but in the MBean attribute instead.

Note that this feature is only available to Java-6-based clients i.e. WebLogic Server 10.3 and above.

2.2. The configuration of the ECAS Client properties from within the web.xml deployment descriptor (phased out)

Configuring the ECAS Client by specifying properties in the *web.xml* deployment descriptor is not supported with the new V2 version. As the authentication mechanism is not done by a Servlet or Servlet Filter anymore, this way of configuring your Client does not work.

Now the web.xml deployment descriptor is only used to specify standard JEE security-constraints.

3. THE CONFIGURATION FILE

The ECAS Client can also be configured using a specific configuration file. It can be either a standard Java Properties file or an XML file.

3.1. Configuration file name and location

You can choose one of the two available naming conventions for the ECAS-client-configuration file name:

1. Using a static name: "*ecas-config.properties*" or "*ecas-config.xml*" (see 3.1.1).
2. Using a dynamic name specific to your application: "*ecas-config-<specific>.properties*" or "*ecas-config-<specific>.xml*" (see 3.1.2)

You also have to decide which deployment strategy to adopt regarding the configuration file location:

a) Either embed the configuration file *inside* your Web or enterprise archive (WAR, EAR)

b) Or deploy the configuration file *outside* of your archive, either in the classpath or on the file system.

Using a configuration file name specific to your application is especially important when you choose to deploy the file outside of your archive and are on a shared environment.

Alternatively, if you prefer to keep the configuration file inside your archive, it is perfectly valid to use the default static name ("*ecas-config.properties*" or "*ecas-config.xml*") as long as all other applications deployed on the same shared environment adopt the same policy.

3.1.1. The static configuration file (deprecated)

The default static configuration file is named "*ecas-config.properties*" or "*ecas-config.xml*" (according to the format you prefer).

Two deployment strategies are available for this file:

a) Embed it *inside* your application archive (WAR or EAR)

b) Put it *outside* of your archive.

While straightforward to implement, the first approach (configuration file embedded in the archive) will force you to deliver different binaries for different target environments.

Using an external configuration file has the advantage of letting you reuse the exact same application binary with different ECAS configurations for multiple environments (e.g. development, test, training, production) without the need to rebuild or repackage.

Note: You can also chain external configuration files if you want to have a base file from which the properties are inherited by all the environments and another localized file per environment. You can do this by using the `eu.cec.digit.ecas.client.configFile` configuration property inside your configuration properties file. See also *3.5.1 configFile*.

3.1.1.1. Configuration file inside the archive

Inside your WAR, create a file named *ecas-config.properties* (for the plain-text properties version) or *ecas-config.xml* (for the XML variant) in `WEB-INF/classes`.

For an EAR, you can put it in `APP-INF/classes` instead.

The properties file takes precedence over the XML file if both exist.

3.1.1.2. Configuration file outside of the archive

The mechanism described in this section still works but we highly recommend using the one described in section 3.1.2 instead.

Inside your WAR, create a file named *ecas-config.properties* in `WEB-INF/classes` containing a link to the actual configuration file you want to use:

```
eu.cec.digit.ecas.client.configFile=ecas-config-mywebapp.properties
```

Or, if you prefer the XML alternative, create an XML file named *ecas-config.xml* using the following template (***mywebapp*** stands for your application name):

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <!--
        Unique name of the external configuration file that contains
        the actual ECAS Client configuration.
    -->
    <configFile>ecas-config-mywebapp.xml</configFile>
</client-config>
```

For an EAR, you can put it in `APP-INF/classes` instead.

Let's say that the name of your application is *sysper2*, you would configure

either	*ecas-config-sysper2.properties*	in	`ecas-config.properties`
or	*ecas-config-sysper2.xml*	in	`ecas-config.xml`

and you would create the external file *ecas-config-sysper2.properties* (or *.xml*) containing your localized configuration.

<u>Remarks:</u>

- You should not name the external file `ecas-config.properties` or `ecas-config.xml` because it would create configuration **conflicts on shared servers** running multiple applications protected by ECAS.
- The file must be found in the classpath or on the file system. The ECAS Client first tries to locate the file in the classpath then if not found, tries to find it on the file system.
- The file extension must be either `.properties` or `.xml`.
- The **name** of your file must be **unique** per WebLogic domain, so using your application context root – which is also unique per domain – in the file name is a good way of avoiding conflicts.

3.1.2. <u>Context-specific configuration file (recommended)</u>

Instead of using the same file name for all Web applications deployed in a shared environment, you can use a naming convention to have each Web application use its own specific file, which the ECAS Client can retrieve automatically.

For this, simply make your configuration file observe the following naming convention:

"`ecas-config-`" + context-path + "`.properties`" (or "`.xml`").

where context-path means the context path of your Web application, without the starting slash.

If such a file is found, it will be used automatically instead of the default *`ecas-config.properties`* or *`ecas-config.xml`* files.

If the context-path of your application contains one or more slashes ('/'), you have to replace them with a dot ('.').

For example, if your application's context-path is "*/oib/f/budg-app*", the ECAS Client will automatically look for the files:

- *ecas-config-**oib.f.budg-app**.properties* and
- *ecas-config-**oib.f.budg-app**.xml*

If it can find one of these files in the classpath or on the file system, it will be loaded in priority and the default *ecas-config.properties* or *ecas-config.xml* files will be ignored.

Since you cannot deploy two applications on the same context-path in a domain, your configuration file should be unique per domain.

Using this naming convention, you don't need to create the default *ecas-config.properties* or *ecas-config.xml* file to link to your own file anymore.

Hence there is no need to use the mechanism described in section 3.1.1.2 anymore.

You may either put this file inside your WAR in `WEB-INF/classes` or directly in the classpath of your domain or server.

3.2. Configuration File Templates

Below you'll find the template files with all possible options for the properties and the xml versions of the configuration.

First, the template for the `ecas-config-`***`mywebapp`***`.properties` configuration file:

```
###########################################################################
# ECAS Client configuration properties                                    #
###########################################################################
#### Note: System property to use an alternative name: eu.cec.digit.ecas.client.configFile
#### E.g. -Deu.cec.digit.ecas.client.configFile=XXX.properties
####
#### Since version 1.9, conventional configuration file names can be used
#### instead of the "configFile" property.
####
#### Conventional name = "ecas-config-" + escaped(context-path) + (".xml"||".properties")
####
#### where escaped() means replacing all '/' by '.'
#### and   context-path is the context path of your web application WITHOUT the starting
slash '/'
####
####       e.g. "/dg-agri/ecas-demo" becomes "dg-agri.ecas-demo"
####       thus, the conventional file name would be: "ecas-config-dg-agri.ecas-
demo.properties"
####
#### The file using the conventional name prevails over the default file name ("ecas-
config.properties")
#### if both are available.
###########################################################################
## configFile:
### Unique name of the external configuration file that contains
### the actual ECAS Client configuration, if any.
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.configFile=ecas-config-myAppId.properties
###########################################################################
## configurationId:
### Unique id for this ECAS Client configuration.
### Can be used as key to retrieve this configuration in non-Web part of your application.
### [Optional] [Deprecated]
### [DefaultValue=null]
#eu.cec.digit.ecas.client.filter.configurationId=eu.cec.myDG.myApp.myConfig
###########################################################################
## serverName:
### Name of your host running the ECAS Client-protected application.
### If you don't specify either 'serverName' or 'serviceUrl' or a custom 'serviceResolver',
### the value is retrieved from the local server.
### [Optional]
### [DefaultValue=null]
#edu.yale.its.tp.cas.client.filter.serverName=myHost
###########################################################################
## serviceUrl:
### URL of your application.
### Either 'serviceUrl' or 'serverName' is required.
### Use this only if you have only one URL.
### [Optional] [NOT recommended]
#edu.yale.its.tp.cas.client.filter.serviceUrl=https://myHost:7002/myService
###########################################################################
## ecasServerDirectHostName:
### Name of the host of the ECAS server for direct connections (without reverse proxy)
### [Optional]
### [DefaultValue=ecas.cc.cec.eu.int]
#eu.cec.digit.ecas.client.filter.ecasServerDirectHostName=ecas.cc.cec.eu.int
###########################################################################
## ecasServerDirectOneWaySslPort:
### One-Way SSL port of the ECAS server for direct connections (without reverse proxy)
### [Optional]
### [DefaultValue=7002]
#eu.cec.digit.ecas.client.filter.ecasServerDirectOneWaySslPort=7002
###########################################################################
## ecasServerDirectTwoWaySslPort:
### Two-Way SSL port of the ECAS server for direct connections (without reverse proxy)
### [Optional]
### [DefaultValue=7003]
#eu.cec.digit.ecas.client.filter.ecasServerDirectTwoWaySslPort=7003
###########################################################################
## ecasServerReverseProxyHostName:
### Name of the reverse proxy host in front of the ECAS server for proxied connections
### [Optional]
### [DefaultValue=null]
#eu.cec.digit.ecas.client.filter.ecasServerReverseProxyHostName=webgate.ec.europa.eu
###########################################################################
## ecasServerReverseProxyPort:
### Port of the reverse proxy in front of the ECAS server for proxied connections
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.ecasServerReverseProxyPort=443
###########################################################################
## ecasBaseUrl:
### Base URL for all ECAS URLs when they are specified as relative URLs.
### If defined, prevails over the ECAS server URL properties named 'ecasServerDirect...' and
'ecasServerReverseProxy...'
### [Optional]
### [DefaultValue=https://ecas.cc.cec.eu.int:7002]
#eu.cec.digit.ecas.client.filter.ecasBaseUrl=https://ecas.cc.cec.eu.int:7002
###########################################################################
## initLoginUrl:
### ECAS Server init login URL.
### [Optional]
### [DefaultValue=/cas/login/init]
#eu.cec.digit.ecas.client.filter.initLoginUrl=/cas/login/init
###########################################################################
## loginUrl:
### ECAS Server login URL.
### [Optional]
### [DefaultValue=/cas/login]
#edu.yale.its.tp.cas.client.filter.loginUrl=/cas/login
###########################################################################
## validateUrl:
### ECAS Server Validation URL.
### [Optional]
### [DefaultValue=/cas/TicketValidationService]
### LegalValue:
#edu.yale.its.tp.cas.client.filter.validateUrl=/cas/TicketValidationService
###########################################################################
## proxyUrl:
### ECAS Server Proxy URL.
### [Optional] [For ECASProxies]
### [DefaultValue=/cas/proxy]
#edu.yale.its.tp.cas.client.filter.proxyUrl=/cas/proxy
###########################################################################
## renew:
### Use 'renew' to always force renew (i.e. force to re-authenticate
### by re-entering login and password).
### [Optional] [NOT recommended]
### [DefaultValue=false]
#edu.yale.its.tp.cas.client.filter.renew=false
###########################################################################
## authorizedProxy:
### The one and only one ECAS proxy authorized to access your application using
### proxy tickets.
### [Optional] [Deprecated]
### [NOT recommended] [For applications used by ECAS Proxy clients]
### [DefaultValue=none]
#edu.yale.its.tp.cas.client.filter.authorizedProxy=https://host.cec.eu.int/someService
###########################################################################
```

```
## serverProtocol:
### Protocol can be either http or https.
### This is only needed for applications behind a reverse proxy that want
### to overwrite the protocol of the local server (i.e. when the
### application is not accessible internally and when the reverse proxy
### protocol is different from the local server's protocol)
### [Optional]
### [DefaultValue=none]
### LegalValues:
#eu.cec.digit.ecas.client.filter.serverProtocol=http
#eu.cec.digit.ecas.client.filter.serverProtocol=https
#eu.cec.digit.ecas.client.filter.serverProtocol=dynamic
############################################################################
## serverPort:
### HTTP port of your host.
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.serverPort=7001
############################################################################
## serverSSLPort:
### HTTPS port of your host.
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.serverSSLPort=7002
############################################################################
## serverContextPath:
### Context root of the application.
### Must begin with a slash '/'.
### This is only needed for applications behind a reverse proxy that modifies
### the local application's context root and when the application is only
### accessible through that reverse proxy.
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.serverContextPath=/myProxiedApplicationContextPath
############################################################################
## authorizedProxies:
### The comma-separated list of ECAS proxies authorized to access your application
### using proxy tickets.
### [Optional] [For applications used by ECAS Proxy clients]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.authorizedProxies=https://host1.cec.eu.int/service1,\
#https://host2.cec.eu.int/service2,https://host3.cec.eu.int/service3,\
#https://host4.cec.eu.int/service4
############################################################################
## proxyChainTrustHandler:
### Class implementating proxyChainTrustHandlerIntf to be used.
### [Optional] [For applications used by ECAS Proxy clients]
### [DefaultValue=eu.cec.digit.ecas.client.validation.ProxyChainTrustHandler]
### ProvidedImplementations:
#eu.cec.digit.ecas.client.filter.proxyChainTrustHandler=eu.cec.digit.ecas.client.validation.
ProxyChainTrustHandler
#eu.cec.digit.ecas.client.filter.proxyChainTrustHandler=eu.cec.digit.ecas.client.validation.
FirstParentProxyChainTrustHandler
############################################################################
## proxyCallbackUrl:
### URL of your application used to receive ProxyTickets from the ECAS Server.
### [Optional] [For ECASProxies]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.proxyCallbackUrl=https://myHost:7002/myService/proxy
############################################################################
## applicationServer:
### Type of application server.
### [Optional] [Deprecated]
### [DefaultValue=weblogic]
#eu.cec.digit.ecas.client.filter.applicationServer=weblogic
############################################################################
## groups:
### Comma-separated list of groups to ask ECAS.
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.groups=MY_APPLICATION_CUD_GROUP1, MY_APPLICATION_CUD_GROUP2
############################################################################
## acceptStrengths:
### [Optional]
### [DefaultValue=PASSWORD,PASSWORD_SMS,PASSWORD_TOKEN,CLIENT_CERT]
### For the mock-up server, use BASIC strength.
### LegalValues:
#eu.cec.digit.ecas.client.filter.acceptStrengths=BASIC
#eu.cec.digit.ecas.client.filter.acceptStrengths=PASSWORD,PASSWORD_SMS,PASSWORD_TOKEN,CLIENT
_CERT
############################################################################
## maxConnections:
```

```
### [Optional]
### [DefaultValue=2]
#eu.cec.digit.ecas.client.filter.maxConnections=10
###########################################################################
## connectionTimeout:
### [Optional]
### [DefaultValue=180000] [unit=milliseconds]
#eu.cec.digit.ecas.client.filter.connectionTimeout=300000
###########################################################################
## strictSSLHostnameVerification:
### [Optional]
### [DefaultValue=true]
#eu.cec.digit.ecas.client.filter.strictSSLHostnameVerification=true
###########################################################################
## extraGroupHandler:
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.extraGroupHandler=eu.cec.digit.ecas.client.validation.JdbcE
xtraGroupHandler
###########################################################################
## authEventListeners:
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.authEventListeners=eu.cec.digit.ecas.client.event.StatsEven
tListener
###########################################################################
## configurationOrder:
### Defines the configuration order to use.
### 'mbean' means the EcasIdentityAsserter MBean,
### 'file' means the external configuration file (usually called ecas-config.xml or ecas-
config.properties),
### 'descriptor' means the resource or deployment descriptor (such as web.xml),
### '<<' means 'overridden by'.
### The default configuration chain order is 'mbean << file << descriptor',
### which means first the MBean configuration, then the external file,
### then the deployment descriptor, each time overriding already defined properties.
### Hence, with the default value, the deployment descriptor takes precedence over the
external file,
### and the external file takes precedence over the MBean.
### Another useful configuration for single-application domains is 'file << descriptor <<
mbean',
### where it is the MBean that prevails instead of the descriptor for application entirely
### configurable by the WebLogic Domain Administrator.
### [Optional]
### [DefaultValue=mbean << file << descriptor]
#eu.cec.digit.ecas.client.filter.configurationOrder=mbean << file << descriptor
###########################################################################
## initSignatureUrl:
### ECAS Server Signature init URL to negotiate a SignatureRequestId.
### [Optional]
### [DefaultValue=/cas/signature/init]
#eu.cec.digit.ecas.client.filter.initSignatureUrl=/cas/signature/init
###########################################################################
## signatureUrl:
### ECAS Server Signature page where the user performs the signature by re-authenticating.
### [Optional]
### [DefaultValue=/cas/signature/sign.do]
#eu.cec.digit.ecas.client.filter.signatureUrl=/cas/signature/sign.do
###########################################################################
## retrieveSignatureUrl:
### ECAS Server Signature get URL where applications retrieve signed XML documents.
### [Optional]
### [DefaultValue=/cas/signature/get]
#eu.cec.digit.ecas.client.filter.retrieveSignatureUrl=/cas/signature/get
###########################################################################
## transactionUrl:
### ECAS Server Signature transaction URL where applications sign XML documents based upon
ECAS Proxy Tickets.
### [Optional]
### [DefaultValue=/cas/transaction/sign]
#eu.cec.digit.ecas.client.filter.transactionUrl=/cas/transaction/sign
###########################################################################
## certificateRevocationUrl:
### ECAS Server Signature certificateRevocation URL where applications can ask ECAS whether
the signature certificate is valid.
### [Optional]
### [DefaultValue=/cas/signature/certValidate]
#eu.cec.digit.ecas.client.filter.certificateRevocationUrl=/cas/signature/certValidate
###########################################################################
## requestingUserDetails:
### Set "requestingUserDetails" on "true" to always request all additional user details such
as
```

```
### the domain, the username in this domain, first name, last name, email, department
number, etc.
### [Optional]
### [DefaultValue=false]
eu.cec.digit.ecas.client.filter.requestingUserDetails=true
###########################################################################
## serviceResolver:
### Set "serviceResolver" to replace the default implementation using the
### serverName or serviceUrl properties to construct the original service.
### [Optional]
### [DefaultValue=eu.cec.digit.ecas.client.resolver.service.DefaultServiceResolver]
### ProvidedImplementations:
#eu.cec.digit.ecas.client.filter.serviceResolver=eu.cec.digit.ecas.client.resolver.service.D
efaultServiceResolver
#eu.cec.digit.ecas.client.filter.serviceResolver=eu.cec.digit.ecas.client.resolver.service.R
everseProxyAwareServiceResolver
###########################################################################
### Custom configuration parameter name labelled "serviceResolverHeader":
### [Optional]
### Note that a custom parameter must have both a param.name and a param.value
#eu.cec.digit.ecas.client.filter.param.name.serviceResolverHeader=serviceResolverHeader
###########################################################################
### Custom configuration parameter value for "serviceResolverHeader":
### [Optional]
### Note that a custom parameter must have both a param.name and a param.value
#eu.cec.digit.ecas.client.filter.param.value.serviceResolverHeader=X-ori-url
###########################################################################
# loginDateValidator
### Strategy to override the expiration of the SSO session.
### Allows to refuse users who were authenticated too long ago.
#eu.cec.digit.ecas.client.filter.loginDateValidator=eu.cec.digit.ecas.client.configuration.D
ummyLoginDateValidator
#eu.cec.digit.ecas.client.filter.param.name.loginDateExpirationInMillis=loginDateExpirationI
nMillis
# one hour:
#eu.cec.digit.ecas.client.filter.param.value.loginDateExpirationInMillis=3600000
###########################################################################
# reSubmitPosts
### Whether to re-submit form parameters that are posted to protected
### resources while the user is not yet authenticated.
### [Optional]
### [DefaultValue=false]
eu.cec.digit.ecas.client.filter.reSubmitPosts=true
###########################################################################
## httpRedirector
### The implementation of HttpRedirector to use to redirect to the ECAS Server login URL.
### [Optional]
### [DefaultValue=eu.cec.digit.ecas.client.http.DefaultHttpRedirector]
### ProvidedImplementations:
#eu.cec.digit.ecas.client.filter.httpRedirector=eu.cec.digit.ecas.client.http.DefaultHttpRed
irector
#eu.cec.digit.ecas.client.filter.httpRedirector=eu.cec.digit.ecas.client.http.BrowserPostRed
irector
#eu.cec.digit.ecas.client.filter.httpRedirector=eu.cec.digit.ecas.client.http.LegacyHttpRedi
rector
#eu.cec.digit.ecas.client.filter.httpRedirector=eu.cec.digit.ecas.client.http.ajax.JsonHttpR
edirector
#eu.cec.digit.ecas.client.filter.httpRedirector=eu.cec.digit.ecas.client.http.JavascriptHttp
Redirector
#eu.cec.digit.ecas.client.filter.httpRedirector=eu.cec.digit.ecas.client.http.LegacyJavascri
ptHttpRedirector
###########################################################################
## trustNonEcasJEESubject
### Whether or not to trust users who are authenticated by the JEE container with another
mechanism than ECAS.
### If true, those users are not re-authenticated with ECAS but are granted immediate access
into the application.
### If false, these JEE-already-authenticated users are re-authenticated with ECAS for
requests which are filtered
### either by the GatewayFilter or the legacy EcasFilter.
### This property has no effect when using security-constraints or a WebLogic Identity
Assertion Provider.
### [Optional]
### [DefaultValue=false]
#eu.cec.digit.ecas.client.filter.trustNonEcasJEESubject=true
###########################################################################
## acceptedTicketTypes
### The "acceptedTicketTypes" property is the sequence of ECAS ticket-types accepted by the
application.
### If users try to access the application with other ticket types than the ones specified
here,
### an INVALID_TICKET error code is returned by ECAS.
### [Optional]
### [DefaultValue=SERVICE,PROXY]
#### Legal values: SERVICE,PROXY,DESKTOP or a combination
#eu.cec.digit.ecas.client.filter.acceptedTicketTypes=SERVICE
#eu.cec.digit.ecas.client.filter.acceptedTicketTypes=SERVICE,PROXY
#eu.cec.digit.ecas.client.filter.acceptedTicketTypes=SERVICE,PROXY,DESKTOP
###########################################################################
## assuranceLevel
### The "assuranceLevel" property is the level of assurance in the user's identity
### the application requires to grant access.
### If users with assurance levels lower than the one configured here try to access the
application,
### an INVALID_USER error code is returned by ECAS.
### [Optional]
### [DefaultValue=TOP]
#### Legal values: TOP,HIGH,MEDIUM,LOW,NO_ASSURANCE
#eu.cec.digit.ecas.client.filter.assuranceLevel=TOP
#eu.cec.digit.ecas.client.filter.assuranceLevel=HIGH
#eu.cec.digit.ecas.client.filter.assuranceLevel=MEDIUM
#eu.cec.digit.ecas.client.filter.assuranceLevel=LOW
###########################################################################
## proxyGrantingProtocol
### The "proxyGrantingProtocol" property is used to specify the protocol to be used
### to obtain ProxyGrantingTickets (PGT).
### [Optional]
### [DefaultValue=none]
#### Legal values: PGT_URL,CLIENT_CERT,DESKTOP
#eu.cec.digit.ecas.client.filter.proxyGrantingProtocol=PGT_URL
#eu.cec.digit.ecas.client.filter.proxyGrantingProtocol=CLIENT_CERT
#eu.cec.digit.ecas.client.filter.proxyGrantingProtocol=DESKTOP
###########################################################################
## trustedCertificates
### The "trustedCertificates" property is used to specify the comma-separated list of
base64-encoded X.509 certificates of the
### trusted Certificate Authorities used for SSL by the ECAS server or mockup server.
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.trustedCertificates=MIIDLTCCAhWgAwIBAgIBATANBgkqhkiG9w0BAQU
FADAmMSQwIgYDVQQDExtFdXJv\
#cGVhbiBDb21taXNzaW9uIFJvb3QgQ0EwHhcNMDMwMTIxMTgwMTM4WhcNMTIxMjMx\
#MTgwMTM4WjAmMSQwIgYDVQQDExtFdXJvcGVhbiBDb21taXNzaW9uIFJvb3QgQ0Ew\
#ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC2qIU7u75rPCUqzM0a0HT4\
#eSMa+bFzSWcIxqJU1dPY1WGkqsee8rom3waf3scuIXHdk6CL43+s2zMrd0W8gyLL\
#DBN46Z4BG9dIyhvHTlGTg7grVvHypbsvgC0lzb7xM/oFFs4AVUVqNgQPx1bELB3s\
#t3NZRLUvFWNHXWDzR6CC/JTznn7NYBB0OScX7oMjYPQFL6n7vgKIVaU7YcZ+tJ6r\
#a4oVt7zu3seiBzO0gijTcvlZ8PMIZUc21DnV2PtFgzaq5iem8mGdlVZXyL6MzbRx\
#d4GIODPnWpCKABHd8dUMbbkOtkp1HMEQmaEdYr4zFFs53Snq4YZzFFhxRrfZCZfj\
#AgMBAAGjZjBkMB8GA1UdIwQYMBaAFI+na6QQzxN0Z0ZdrF1wdDKuMkbbMB0GA1Ud\
#DgQWBBSPp2ukEM8TdGdGXaxdcHQyrjJG2zAOBgNVHQ8BAf8EBAMCAQYwEgYDVR0T\
#AQH/BAgwBgEB/wIBATANBgkqhkiG9w0BAQUFAAOCAQEAaPuj04oBLi6JExkkOzJX\
#yYf+x0/dXXEt8oknr2qlfyaM2R6PXVcqE6HKtRcvxzuDSrgEHBb8N9k21YuF6ftM\
#QeQTyRcJVJVuTW29Vw+vxj/NPqGSjpWTWA32jd2FqM8lcrw8JQ+cOtCMYxdjBb6j\
#AJ9yiZ6AScEWGlN6hUS/KFZByKEnQTLiJ9BHooB651e1+TYs8BA3LuSYi3xKYniT\
#sjde9nvePJAhTsxjs+oJklZiNR5yR6w385ah5Lgqyieb3+jAVfgExjc+h2hayOAl\
#0/y2h8gQOlDzNRPUUftrUo9dMKJqAAyZyH18HH3kYbJ+9iy/cmHXY8OU5AdqTS/F\
#hg\=\=\
#,...\
#,...\
#,...
###########################################################################
## applicationSecurityLevel
### The "applicationSecurityLevel" property is the level of security this application
requires
### [Optional]
### [DefaultValue=NO_SECURITY]
#### Legal values: TOP,HIGH,MEDIUM,LOW,NO_SECURITY
#eu.cec.digit.ecas.client.filter.applicationSecurityLevel=TOP
#eu.cec.digit.ecas.client.filter.applicationSecurityLevel=HIGH
#eu.cec.digit.ecas.client.filter.applicationSecurityLevel=MEDIUM
#eu.cec.digit.ecas.client.filter.applicationSecurityLevel=LOW
###########################################################################
## negotiatePrivateServiceTicket
### The "negotiatePrivateServiceTicket" property controls whether or not the ECAS ticket
must be
### sent in the service URL or can be pre-negotiated when LoginRequestTransactions are
### enabled (via a configured HttpRedirector equal to
eu.cec.digit.ecas.client.http.DefaultHttpRedirector
### or eu.cec.digit.ecas.client.http.BrowserPostRedirector).
### If "true", the ticket is not sent in the service URL but is only sent through the back-
channel
### between the application and the ECAS server over SSL/TLS.
### [Optional]
### [DefaultValue=false]
#### Legal values: false,true
#eu.cec.digit.ecas.client.filter.negotiatePrivateServiceTicket=false
#eu.cec.digit.ecas.client.filter.negotiatePrivateServiceTicket=true
###########################################################################
## advancedHttpSessionManagement
### The "advancedHttpSessionManagement" property controls whether HTTP state management
through the establishment of the HttpSession is mandatory.
### If "true", the ECAS Client will take appropriate care to enforce that a valid
HttpSession is created and maintained.
### Otherwise, these actions are not undertaken.
### You should let this value to "true" unless your end-users are only Web service clients
unable to maintain cookie-based HttpSessions.
### [Optional]
### [DefaultValue=true]
#### Legal values: true,false
#eu.cec.digit.ecas.client.filter.advancedHttpSessionManagement=true
#eu.cec.digit.ecas.client.filter.advancedHttpSessionManagement=false
###########################################################################
## ticketResolver:
### Set "ticketResolver" to replace the default implementation.
### [Optional]
### [DefaultValue=eu.cec.digit.ecas.client.resolver.ticket.DefaultTicketResolver]
### ProvidedImplementations:
#eu.cec.digit.ecas.client.filter.ticketResolver=eu.cec.digit.ecas.client.resolver.ticket.Def
aultTicketResolver
###########################################################################
## redirectionInterceptors:
### Set "redirectionInterceptors" to replace the default implementation.
### [Optional]
### [DefaultValue=eu.cec.digit.ecas.client.http.robot.DefaultRobotInterceptor]
### ProvidedImplementations:
#eu.cec.digit.ecas.client.filter.redirectionInterceptors=eu.cec.digit.ecas.client.http.robot
.DefaultRobotInterceptor
#eu.cec.digit.ecas.client.filter.redirectionInterceptors=eu.cec.digit.ecas.client.http.robot
.BlindRobotInterceptor
#eu.cec.digit.ecas.client.filter.redirectionInterceptors=eu.cec.digit.ecas.client.http.robot
.OnlyRobotInterceptor
#eu.cec.digit.ecas.client.filter.redirectionInterceptors=eu.cec.digit.ecas.client.http.robot
.DefaultRobotInterceptor,
eu.cec.digit.ecas.client.http.ajax.UnauthorizedAjaxRedirectionInterceptor
#eu.cec.digit.ecas.client.filter.redirectionInterceptors=eu.cec.digit.ecas.client.http.robot
.DefaultRobotInterceptor, eu.cec.digit.ecas.client.http.ajax.JsonAjaxRedirectionInterceptor
###########################################################################
## extendedUserDetailsTypeMapper:
### Set "extendedUserDetailsTypeMapper" to plugin your implementation of the
eu.cec.digit.ecas.client.validation.ExtendedUserDetailsTypeMapper interface,
### which is used to type extendedUserDetails instead of using a Map of Strings.
### [Optional]
### [DefaultValue=none]
### Example:
#eu.cec.digit.ecas.client.filter.extendedUserDetailsTypeMapper=eu.europa.ec.mydg.myapp.MyExt
endedUserDetailsTypeMapper
###########################################################################
## singleLogoutCallbackUrl:
### URL of your application used to receive LogoutRequests from the ECAS Server (for the
Single Sign-Out Protocol).
### [Optional] [For Clustered environments without HttpSession replication]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.singleLogoutCallbackUrl=https://myHost:7002/myService/singl
eLogout
###########################################################################
```

Figure 4 - ecas-config-*mywebapp*.properties template

Next, the template for the `ecas-config-`***`mywebapp`***`.xml` configuration file if you prefer the XML format. The W3C XML schema for it can be found at https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/.

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
        >
    <!--
    ECAS Client configuration properties

    Note: System property to use an alternative name: eu.cec.digit.ecas.client.configFile
    E.g. -Deu.cec.digit.ecas.client.configFile=XXX.properties

    Since version 1.9, conventional configuration file names can be used
    instead of the "configFile" property.

    Conventional name = "ecas-config-" + escaped(context-path) + (".xml"||".properties")

    where escaped() means replacing all '/' by '.'
    and   context-path is the context path of your web application WITHOUT the starting
slash '/'

          e.g. "/dg-agri/ecas-demo" becomes "dg-agri.ecas-demo"
          thus, the conventional file name would be: "ecas-config-dg-agri.ecas-
demo.properties"

    The file using the conventional name prevails over the default file name ("ecas-
config.properties")
    if both are available.
    -->

    <!-- Unique name of the external configuration file that contains
    the actual ECAS Client configuration, if any.
    [Optional]
    [DefaultValue=none] -->
    <!--<configFile>ecas-config-myAppId.properties</configFile>-->

    <!-- Unique id for this ECAS Client configuration.
    Can be used as key to retrieve this configuration in non-Web part of your application.
    [Optional] [Deprecated]
    [DefaultValue=null] -->
    <!--<configurationId>eu.cec.myDG.myApp.myConfig</configurationId>-->

    <!-- Name of your host running the ECAS Client-protected application.
    If you don't specify either 'serverName' or 'serviceUrl' or a custom 'serviceResolver',
    the value is retrieved from the local server.
    [Optional]
    [DefaultValue=null] -->
    <!--<cas:serverName>myHost</cas:serverName>-->

    <!-- URL of your application.
    Either 'serviceUrl' or 'serverName' is required.
    Use this only if you have only one URL.
    [Optional] [NOT recommended] -->
    <!--<cas:serviceUrl>https://myHost:7002/myService</cas:serviceUrl>-->

    <!-- Name of the host of the ECAS server for direct connections (without reverse proxy).
    [Optional]
    [DefaultValue=ecas.cc.cec.eu.int] -->
    <!--<ecasServerDirectHostName>ecas.cc.cec.eu.int</ecasServerDirectHostName>-->

    <!-- One-Way SSL port of the ECAS server for direct connections (without reverse proxy).
    [Optional]
    [DefaultValue=7002] -->
    <!--<ecasServerDirectOneWaySslPort>7002</ecasServerDirectOneWaySslPort>-->

    <!-- Two-Way SSL port of the ECAS server for direct connections (without reverse proxy).
    [Optional]
    [DefaultValue=7003]
    -->
    <!--<ecasServerDirectTwoWaySslPort>7003</ecasServerDirectTwoWaySslPort>-->

    <!-- Name of the reverse proxy host in front of the ECAS server for proxied connections
    [Optional]
    [DefaultValue=null]
    -->
    <!--
<ecasServerReverseProxyHostName>webgate.ec.europa.eu</ecasServerReverseProxyHostName>-->

    <!-- Port of the reverse proxy in front of the ECAS server for proxied connections
    [Optional]
    [DefaultValue=none]
    -->
    <!--<ecasServerReverseProxyPort>443</ecasServerReverseProxyPort>-->

    <!-- Base URL for all ECAS URLs when they are specified as relative URLs.
    If defined, prevails over the ECAS server URL properties named 'ecasServerDirect...' and
'ecasServerReverseProxy...'
    [Optional]
    [DefaultValue=https://ecas.cc.cec.eu.int:7002] -->
    <!--<ecasBaseUrl>https://ecas.cc.cec.eu.int:7002</ecasBaseUrl>-->

    <!-- ECAS Server init login URL.
    [Optional]
    [DefaultValue=/cas/login/init] -->
    <!--<initLoginUrl>/cas/login/init</initLoginUrl>-->

    <!-- ECAS Server login URL.
    [Optional]
    [DefaultValue=/cas/login] -->
    <!--<cas:loginUrl>/cas/login</cas:loginUrl>-->

    <!-- ECAS Server Validation URL.
    [Optional]
    [DefaultValue=/cas/TicketValidationService]
    LegalValue: -->
    <!--<cas:validateUrl>/cas/TicketValidationService</cas:validateUrl>-->

    <!-- ECAS Server Proxy URL.
    [Optional] [For ECASProxies]
    [DefaultValue=/cas/proxy] -->
    <!--<cas:proxyUrl>/cas/proxy</cas:proxyUrl>-->

    <!-- Use 'renew' to always force renew (i.e. force to re-authenticate
    by re-entering login and password).
    [Optional] [NOT recommended]
    [DefaultValue=false] -->
    <!--<cas:renew>false</cas:renew>-->

    <!-- The one and only one ECAS proxy authorized to access your application using
    proxy tickets.
    [Optional] [Deprecated]
    [NOT recommended] [For applications used by ECAS Proxy clients]
    [DefaultValue=none] -->
    <!--<cas:authorizedProxy>https://host.cec.eu.int/someService</cas:authorizedProxy>-->

    <!-- Protocol can be either http or https.
    This is only needed for applications behind a reverse proxy that want
    to overwrite the protocol of the local server (i.e. when the
    application is not accessible internally and when the reverse proxy
    protocol is different from the local server's protocol)
    [Optional]
    [DefaultValue=none]
    LegalValues: -->
    <!--<serverProtocol>http</serverProtocol>-->
    <!--<serverProtocol>https</serverProtocol>-->
    <!--<serverProtocol>dynamic</serverProtocol>-->

    <!-- HTTP port of your host.
    [Optional]
    [DefaultValue=none] -->
    <!--<serverPort>7001</serverPort>-->

    <!-- HTTPS port of your host.
    [Optional]
    [DefaultValue=none] -->
    <!--<serverSSLPort>7002</serverSSLPort>-->

    <!-- Context root of the application.
    Must begin with a slash '/'.
    This is only needed for applications behind a reverse proxy that modifies
    the local application's context root and when the application is only
    accessible through that reverse proxy.
    [Optional]
    [DefaultValue=none] -->
    <!--<serverContextPath>/myProxiedApplicationContextPath</serverContextPath>-->

    <!-- The tag list of ECAS proxies authorized to access your application
    using proxy tickets.
    [Optional] [For applications used by ECAS Proxy clients]
    [DefaultValue=none] -->
    <!--<authorizedProxies>-->
    <!--    <proxy>https://host1.cec.eu.int/service1</proxy>-->
    <!--    <proxy>https://host2.cec.eu.int/service2</proxy>-->
    <!--    <proxy>https://host3.cec.eu.int/service3</proxy>-->
    <!--    <proxy>https://host4.cec.eu.int/service4</proxy>-->
    <!--</authorizedProxies>-->

    <!-- Class implementating proxyChainTrustHandlerIntf to be used.
    [Optional] [For applications used by ECAS Proxy clients]
    [DefaultValue=eu.cec.digit.ecas.client.validation.ProxyChainTrustHandler]
    ProvidedImplementations: -->
    <!--
<proxyChainTrustHandler>eu.cec.digit.ecas.client.validation.ProxyChainTrustHandler</proxyCha
inTrustHandler>-->
    <!--
<proxyChainTrustHandler>eu.cec.digit.ecas.client.validation.FirstParentProxyChainTrustHandle
r</proxyChainTrustHandler>-->

    <!-- URL of your application used to receive ProxyTickets from the ECAS Server.
    [Optional] [For ECASProxies]
    [DefaultValue=none] -->
    <!--<proxyCallbackUrl>https://myHost:7002/myService/proxy</proxyCallbackUrl>-->

    <!-- Type of application server.
    [Optional] [Deprecated]
    [DefaultValue=weblogic] -->
    <!--<applicationServer>weblogic</applicationServer>-->

    <!-- Comma-separated list of groups to ask ECAS.
    [Optional]
    [DefaultValue=none] -->
    <!--<groups>-->
    <!--    <group>MY_APPLICATION_CUD_GROUP1</group>-->
    <!--    <group>MY_APPLICATION_CUD_GROUP2</group>-->
    <!--</groups>-->

    <!-- [Optional]
    [DefaultValue=PASSWORD,PASSWORD_SMS,PASSWORD_TOKEN,CLIENT_CERT]
    For the mock-up server, use BASIC strength.
    LegalValues: -->
    <!--<acceptStrengths>-->
    <!--    <strength>BASIC</strength>-->
    <!--    <strength>PASSWORD</strength>-->
    <!--    <strength>PASSWORD_SMS</strength>-->
    <!--    <strength>PASSWORD_TOKEN</strength>-->
    <!--    <strength>CLIENT_CERT</strength>-->
    <!--</acceptStrengths>-->

    <!-- [Optional]
    [DefaultValue=2] -->
    <!--<maxConnections>10</maxConnections>-->

    <!-- [Optional]
    [DefaultValue=180000] [unit=milliseconds] -->
    <!--<connectionTimeout>300000</connectionTimeout>-->

    <!-- [Optional]
    [DefaultValue=true] -->
    <!--<strictSSLHostnameVerification>true</strictSSLHostnameVerification>-->

    <!-- [Optional]
    [DefaultValue=none] -->
    <!--
<extraGroupHandler>eu.cec.digit.ecas.client.validation.JdbcExtraGroupHandler</extraGroupHand
ler>-->

    <!-- [Optional]
    [DefaultValue=none] -->
    <!--<authEventListeners>-->
    <!--    <listener>eu.cec.digit.ecas.client.event.StatsEventListener</listener>-->
    <!--</authEventListeners>-->

    <!-- Defines the configuration order to use.
    'mbean' means the EcasIdentityAsserter MBean,
    'file' means the external configuration file (usually called ecas-config.xml or ecas-
config.properties),
    'descriptor' means the resource or deployment descriptor (such as web.xml),
    '<<' means 'overridden by'.
    The default configuration chain order is 'mbean << file << descriptor',
    which means first the MBean configuration, then the external file,
    then the deployment descriptor, each time overriding already defined properties.
    Hence, with the default value, the deployment descriptor takes precedence over the
external file,
    and the external file takes precedence over the MBean.
    Another useful configuration for single-application domains is 'file << descriptor <<
mbean',
    where it is the MBean that prevails instead of the descriptor for application entirely
    configurable by the WebLogic Domain Administrator.
    [Optional]
    [DefaultValue=mbean << file << descriptor] -->
    <!--<configurationOrder>mbean &lt;&lt; file &lt;&lt; descriptor</configurationOrder>-->

    <!-- [Optional]
    [DefaultValue=/cas/signature/init] -->
    <!--<initSignatureUrl>/cas/signature/init</initSignatureUrl>-->

    <!-- [Optional]
    [DefaultValue=/cas/signature/sign.do] -->
    <!--<signatureUrl>/cas/signature/sign.do</signatureUrl>-->

    <!-- [Optional]
    [DefaultValue=/cas/signature/get] -->
    <!--<retrieveSignatureUrl>/cas/signature/get</retrieveSignatureUrl>-->

    <!-- [Optional]
    [DefaultValue=/cas/transaction/sign] -->
    <!--<transactionUrl>/cas/transaction/sign</transactionUrl>-->

    <!-- [Optional]
    [DefaultValue=/cas/signature/certValidate] -->
    <!--<certificateRevocationUrl>/cas/signature/certValidate</certificateRevocationUrl>-->

    <!-- Set "requestingUserDetails" on "true" to always request all additional user details
such as
    the domain, the username in this domain, first name, last name, email, department
number, etc.
    [Optional]
    [DefaultValue=false] -->
    <requestingUserDetails>true</requestingUserDetails>

    <!-- Set "serviceResolver" to replace the default implementation using the
    serverName or serviceUrl properties to construct the original service.
    [Optional]
    [DefaultValue=eu.cec.digit.ecas.client.resolver.service.DefaultServiceResolver]
    ProvidedImplementations: -->
    <!--
<serviceResolver>eu.cec.digit.ecas.client.resolver.service.DefaultServiceResolver</serviceRe
solver>-->
    <!--
<serviceResolver>eu.cec.digit.ecas.client.resolver.service.ReverseProxyAwareServiceResolver<
/serviceResolver>-->

    <!-- Strategy to override the expiration of the SSO session.
    Allows to refuse users who were authenticated too long ago. -->
    <!--
<loginDateValidator>eu.cec.digit.ecas.client.configuration.DummyLoginDateValidator</loginDat
eValidator>-->

    <!-- Whether to re-submit form parameters that are posted to protected
    resources while the user is not yet authenticated.
    [Optional]
    [DefaultValue=false] -->
    <reSubmitPosts>true</reSubmitPosts>

    <!-- The implementation of HttpRedirector to use to redirect to the ECAS Server login
URL.
    [Optional]
    [DefaultValue=eu.cec.digit.ecas.client.http.DefaultHttpRedirector]
    ProvidedImplementations: -->
    <!--
<httpRedirector>eu.cec.digit.ecas.client.http.DefaultHttpRedirector</httpRedirector>-->
    <!--
<httpRedirector>eu.cec.digit.ecas.client.http.BrowserPostRedirector</httpRedirector>-->
    <!--<httpRedirector>eu.cec.digit.ecas.client.http.LegacyHttpRedirector</httpRedirector>-
->
    <!--
<httpRedirector>eu.cec.digit.ecas.client.http.ajax.JsonHttpRedirector</httpRedirector>-->
    <!--
<httpRedirector>eu.cec.digit.ecas.client.http.JavascriptHttpRedirector</httpRedirector>-->

    <!--<params>-->
    <!--    <param>-->
    <!-- Custom configuration parameter name labelled "serviceResolverHeader":
        [Optional]
        Note that a custom parameter must have both a param.name and a param.value -->
    <!--        <name>serviceResolverHeader</name>-->

    <!-- Custom configuration parameter value for "serviceResolverHeader":
        [Optional]
        Note that a custom parameter must have both a param.name and a param.value -->
    <!--        <value>X-ori-url</value>-->
    <!--    </param>-->

    <!--    <param>-->
    <!-- Custom configuration parameter name labelled "loginDateExpirationInMillis":
        [Optional]
        Note that a custom parameter must have both a param.name and a param.value -->
    <!--        <name>loginDateExpirationInMillis</name>-->

    <!-- Custom configuration parameter value for "loginDateExpirationInMillis":
        [Optional]
        Note that a custom parameter must have both a param.name and a param.value -->
    <!--        one hour -->
    <!--        <value>3600000</value>-->
    <!--    </param>-->
    <!--</params>-->

    <!-- Whether or not to trust users who are authenticated by the JEE container with
another mechanism than ECAS.
         If true, those users are not re-authenticated with ECAS but are granted immediate
access into the application.
         If false, these JEE-already-authenticated users are re-authenticated with ECAS for
requests which are filtered
         either by the GatewayFilter or the legacy EcasFilter.
         This property has no effect when using security-constraints or a WebLogic Identity
Assertion Provider.
         [Optional]
         [DefaultValue=false] -->
    <!--<trustNonEcasJEESubject>true</trustNonEcasJEESubject>-->

    <!-- The "acceptedTicketTypes" property is the sequence of ECAS ticket-types accepted by
the application.
         If users try to access the application with other ticket types than the ones
specified here,
         an INVALID_TICKET error code is returned by ECAS.
         [Optional]
         [DefaultValue=SERVICE,PROXY]
         Legal values: SERVICE,PROXY,DESKTOP -->
    <!--<acceptedTicketTypes>
            <ticketType>SERVICE</ticketType>
            <ticketType>PROXY</ticketType>
            <ticketType>DESKTOP</ticketType>
         </acceptedTicketTypes>-->

    <!-- The "assuranceLevel" property is the level of assurance in the user's identity
         the application requires to grant access.
         If users with assurance levels lower than the one configured here try to access the
application,
         an INVALID_USER error code is returned by ECAS.
         [Optional]
         [DefaultValue=TOP]
         Legal values: TOP,HIGH,MEDIUM,LOW,NO_ASSURANCE -->
    <!--<assuranceLevel>TOP</assuranceLevel>-->
    <!--<assuranceLevel>HIGH</assuranceLevel>-->
    <!--<assuranceLevel>MEDIUM</assuranceLevel>-->
    <!--<assuranceLevel>LOW</assuranceLevel>-->

    <!-- The "proxyGrantingProtocol" property is used to specify the protocol to be used
         to obtain ProxyGrantingTickets (PGT).
         [Optional]
         [DefaultValue=none]
         Legal values: PGT_URL,CLIENT_CERT,DESKTOP -->
    <!--<proxyGrantingProtocol>PGT_URL</proxyGrantingProtocol>-->
    <!--<proxyGrantingProtocol>CLIENT_CERT</proxyGrantingProtocol>-->
    <!--<proxyGrantingProtocol>DESKTOP</proxyGrantingProtocol>-->

    <!-- The "trustedCertificates" property is used to specify the list of
         base64-encoded X.509 certificates of the trusted Certificate Authorities
         used for SSL by the ECAS server or mockup server each within a "trustedCertificate"
tag.
    -->
    <!--
    <trustedCertificates>
        <trustedCertificate>
            MIIDLTCCAhWgAwIBAgIBATANBgkqhkiG9w0BAQUFADAmMSQwIgYDVQQDExtFdXJv
            cGVhbiBDb21taXNzaW9uIFJvb3QgQ0EwHhcNMDMwMTIxMTgwMTM4WhcNMTIxMjMx
            MTgwMTM4WjAmMSQwIgYDVQQDExtFdXJvcGVhbiBDb21taXNzaW9uIFJvb3QgQ0Ew
            ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC2qIU7u75rPCUqzM0a0HT4
            eSMa+bFzSWcIxqJU1dPY1WGkqsee8rom3waf3scuIXHdk6CL43+s2zMrd0W8gyLL
            DBN46Z4BG9dIyhvHTlGTg7grVvHypbsvgC0lzb7xM/oFFs4AVUVqNgQPx1bELB3s
            t3NZRLUvFWNHXWDzR6CC/JTznn7NYBB0OScX7oMjYPQFL6n7vgKIVaU7YcZ+tJ6r
            a4oVt7zu3seiBzO0gijTcvlZ8PMIZUc21DnV2PtFgzaq5iem8mGdlVZXyL6MzbRx
            d4GIODPnWpCKABHd8dUMbbkOtkp1HMEQmaEdYr4zFFs53Snq4YZzFFhxRrfZCZfj
            AgMBAAGjZjBkMB8GA1UdIwQYMBaAFI+na6QQzxN0Z0ZdrF1wdDKuMkbbMB0GA1Ud
            DgQWBBSPp2ukEM8TdGdGXaxdcHQyrjJG2zAOBgNVHQ8BAf8EBAMCAQYwEgYDVR0T
            AQH/BAgwBgEB/wIBATANBgkqhkiG9w0BAQUFAAOCAQEAaPuj04oBLi6JExkkOzJX
            yYf+x0/dXXEt8oknr2qlfyaM2R6PXVcqE6HKtRcvxzuDSrgEHBb8N9k21YuF6ftM
            QeQTyRcJVJVuTW29Vw+vxj/NPqGSjpWTWA32jd2FqM8lcrw8JQ+cOtCMYxdjBb6j
            AJ9yiZ6AScEWGlN6hUS/KFZByKEnQTLiJ9BHooB651e1+TYs8BA3LuSYi3xKYniT
            sjde9nvePJAhTsxjs+oJklZiNR5yR6w385ah5Lgqyieb3+jAVfgExjc+h2hayOAl
            0/y2h8gQOlDzNRPUUftrUo9dMKJqAAyZyH18HH3kYbJ+9iy/cmHXY8OU5AdqTS/F
            hg==
        </trustedCertificate>
        <trustedCertificate>
            MIIDQjCCAiqgAwIBAgIBDzANBgkqhkiG9w0BAQUFADAmMSQwIgYDVQQDExtFdXJv
            cGVhbiBDb21taXNzaW9uIFJvb3QgQ0EwHhcNMDMwMjE3MTcyMzAwWhcNMTIxMjE3
            MTcyMzAwWjA7MRwwGgYDVQQKExNFdXJvcGVhbiBDb21taXNzaW9uMRswGQYDVQQD
            ExJDb21taXNTaWduIENsYXNzIEEwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
            AoIBAQDQc+nelotZJ6V0FYnoeXQqV5aeBydbcJHToLqBMxhJfiJv2FL/vRO151r9
            YZf2etCopiRey2av0AvSebYODIDTAX+ObFYZFXYVTdaCTob6s57vCuUzW4vdCOWA
            NTJ6Wk6nRJ8rJVJ1R35Jt458flFKGDGSGEk6CzkqTkOcrA+As4wURJKJCpjo8pOr
            LBc7GXvAwJ4ZgqOULM5r2+YnePNfKK2Ebg1AOjcvqF9MwbYxPZXwOMQtx7k//Hpa
            WK7Y2JqB8nxS36HkcLus/tBBOdzXNhq5G2rELCGoUd4HIKz1Q6BlQp7kx9vXcWGn
            x0MyXKtCJ052IHhIRHxvhFmBOeqxAgMBAAGjZjBkMB8GA1UdIwQYMBaAFI+na6QQ
            zxN0Z0ZdrF1wdDKuMkbbMB0GA1UdDgQWBBSfqRbgyf+Skzv2/mC99RNJPbI7sTAO
            BgNVHQ8BAf8EBAMCAQYwEgYDVR0TAQH/BAgwBgEB/wIBATANBgkqhkiG9w0BAQUF
            AAOCAQEAFTo9qGVx8W1O4N0XnXP46v1Qn5NXLprom1MUFGgftiNe5af9MJDnXw7+
            H2rOt3m9R0mlbVb1ePLuBLNCOFVFBdh1eoAcJJM+mYtWsJRGG+DzMgTwIVhRXJ60
            e2+pXUluUv4BbHSUu8ItZH4F+NxKQPb1AyjUiTHh7KY11B14zU1t+E6zVcHUoAfX
            1OsHevzAJuN7gFID1LK4hEuDfKg2ZT7IFlXjZ3oMNfKz4afVBhniUsS1+F123spK
            vzQT3iKQ4mI+s2Xdw00HDUGRkxVZ78YLZDTNQxvUzSmFhqyqO16Tq1Uyb2Bd3dLg
            P8AVVbA0sOpobbQQIyq1icd37q4yww==
        </trustedCertificate>
        <trustedCertificate>
            MIIETTCCAzWgAwIBAgIBATANBgkqhkiG9w0BAQUFADB2MQswCQYDVQQGEwJCRTERMA8GA1UECBMI
            QnJ1c3NlbHMxETAPBgNVBAcTCEJydXNzZWxzMQwwCgYDVQQKEwNDRUMxDjAMBgNVBAsTBURJR0lU
            MSMwIQYDVQQDExpFQ0FTIENlcnRpZmljYXRlIEF1dGhvcml0eTAeFw0wNjA0MTExNzUzMzZaFw0x
            MTA0MTIxNzUzMzZaMHYxCzAJBgNVBAYTAkJFMREwDwYDVQQIEwhCcnVzc2VsczERMA8GA1UEBxMI
            QnJ1c3NlbHMxDDAKBgNVBAoTA0NFQzEOMAwGA1UECxMFRElHSVQxIzAhBgNVBAMTGkVDQVMgQ2Vy
            dGlmaWNhdGUgQXV0aG9yaXR5MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAotKTdKad
            m5tzmr59Tofq25hcY1jfhxlrcL5NnfsLc6j+nvdUdDdN2soJbbOBCMcc8BeXW3b4ib+1WYmrE8B0
            wx/6dNonNf/4NFTYm+Xf80O81lsWO/E6279SBTvFw5bTfaPme+YLvpLiUa65XYpLkQK/LF4EEL7u
            Q8Di2nlpSqug27Dkp5Z3Jeh76aHvro1Ul+0qGEHyiJuYT7DSQSjc8XEXPwLpWdrc8izy0e6OsZIW
            jnrdqsBcIqylNriz7kC1Woft77aumgwiMVYMG2yCno4AIIrceI4j/VWYFMi60cX3zXGSfvLGA/uF
            gVoyDze2FKMskBBCT72G/XZ5NvDVcQIDAQABo4HlMIHiMBEGCWCGSAGG+EIBAQQEAwIA9zAOBgNV
            HQ8BAf8EBAMCAf4wHQYDVR0OBBYEFC4ZiCMBXEEqyHrAdhhn9nbAtv6gMB8GA1UdIwQYMBaAFC4Z
            iCMBXEEqyHrAdhhn9nbAtv6gMBIGA1UdEwEB/wQIMAYBAf8CAQEwaQYDVR0lBGIwYAYEVR0lAAYI
            KwYBBQUHAwEGCCsGAQUFBwMCBggrBgEFBQcDAwYIKwYBBQUHAwQGCCsGAQUFBwMFBggrBgEFBQcD
            BgYIKwYBBQUHAwcGCCsGAQUFBwMIBggrBgEFBQcDCTANBgkqhkiG9w0BAQUFAAOCAQEAWtsqmsus
            +8U4p/qZTxTGnmAm07ekim68nYJpJpif8Rg5OOBPv9RXfQABIrKHHkxyiZiG46xuHGbwNPVeXpZF
            PeYQfrKC1YcuuWXUOwDUNaKCSVg7CLLUE6Io0j8aEGK8/c2rEIcmlHWb6NSxOixgsdEoHFEyK2I/
            hL76/Bp9hAeuYFZTS8lwIU87ZUFGGSBZqfxn1f4NNFXjwjxn/sRGdnh5OJvu+0o1047haOmCbY1X
            Xio0JRekMDvoq7G30E+WsdTivkvrZoyjiX5GZ1HPRK0JFrtp+KJwawx2uG2hC0RweUo9iTbiWch3
            8axT7TmJf5/6Dqn43ro7STpwLpJOrw==
        </trustedCertificate>
    </trustedCertificates>
    -->

    <!-- The "applicationSecurityLevel" property is the level of security this application
requires.
         [Optional]
         [DefaultValue=NO_SECURITY]
         Legal values: TOP,HIGH,MEDIUM,LOW,NO_SECURITY -->
    <!--<applicationSecurityLevel>TOP</applicationSecurityLevel>-->
    <!--<applicationSecurityLevel>HIGH</applicationSecurityLevel>-->
    <!--<applicationSecurityLevel>MEDIUM</applicationSecurityLevel>-->
    <!--<applicationSecurityLevel>LOW</applicationSecurityLevel>-->

    <!-- The "negotiatePrivateServiceTicket" property controls whether or not the ECAS
ticket must be
         sent in the service URL or can be pre-negotiated when LoginRequestTransactions are
         enabled (via a configured HttpRedirector equal to
eu.cec.digit.ecas.client.http.DefaultHttpRedirector
         or eu.cec.digit.ecas.client.http.BrowserPostRedirector).

         If "true", the ticket is not sent in the service URL but is only sent through the
back-channel
         between the application and the ECAS server over SSL/TLS.
         [Optional]
         [DefaultValue=false]
         Legal values: false,true-->
    <!--<negotiatePrivateServiceTicket>false</negotiatePrivateServiceTicket>-->
    <!--<negotiatePrivateServiceTicket>true</negotiatePrivateServiceTicket>-->

    <!-- The "advancedHttpSessionManagement" property controls whether HTTP state management
through the establishment of the HttpSession is mandatory.
         If "true", the ECAS Client will take appropriate care to enforce that a valid
HttpSession is created and maintained.
         Otherwise, these actions are not undertaken.
         You should let this value to "true" unless your end-users are only Web service
clients unable to maintain cookie-based HttpSessions.
         [Optional]
         [DefaultValue=true]
         Legal values: true,false -->
    <!--<advancedHttpSessionManagement>true</advancedHttpSessionManagement>-->
    <!--<advancedHttpSessionManagement>false</advancedHttpSessionManagement>-->
    <!-- Set "ticketResolver" to replace the default implementation.
    [Optional]
    [DefaultValue=eu.cec.digit.ecas.client.resolver.ticket.DefaultTicketResolver]
    ProvidedImplementations: -->
    <!--
<ticketResolver>eu.cec.digit.ecas.client.resolver.ticket.DefaultTicketResolver</ticketResolv
er>-->

    <!-- Set "redirectionInterceptors" to replace the default implementation.
    [Optional]
    [DefaultValue=eu.cec.digit.ecas.client.http.robot.DefaultRobotInterceptor]
    ProvidedImplementations: -->
    <!--<redirectionInterceptors>-->
    <!--
<redirectionInterceptor>eu.cec.digit.ecas.client.http.robot.DefaultRobotInterceptor</redirec
tionInterceptor>-->
    <!--
<redirectionInterceptor>eu.cec.digit.ecas.client.http.robot.BlindRobotInterceptor</redirecti
onInterceptor>-->
    <!--
<redirectionInterceptor>eu.cec.digit.ecas.client.http.robot.OnlyRobotInterceptor</redirectio
nInterceptor>-->
    <!--
<redirectionInterceptor>eu.cec.digit.ecas.client.http.ajax.UnauthorizedAjaxRedirectionInterc
eptor</redirectionInterceptor>-->
    <!--
<redirectionInterceptor>eu.cec.digit.ecas.client.http.ajax.JsonAjaxRedirectionInterceptor</r
edirectionInterceptor>-->
    <!--</redirectionInterceptors>-->

    <!-- Set "extendedUserDetailsTypeMapper" to plugin your implementation of the
eu.cec.digit.ecas.client.validation.ExtendedUserDetailsTypeMapper interface,
         which is used to type extendedUserDetails instead of using a Map of Strings.
    [Optional]
    [DefaultValue=none]
    -->
    <!--
<extendedUserDetailsTypeMapper>eu.europa.ec.mydg.myapp.MyExtendedUserDetailsTypeMapper</exte
ndedUserDetailsTypeMapper>-->

    <!-- singleLogoutCallbackUrl:
    URL of your application used to receive LogoutRequests from the ECAS Server (for the
Single Sign-Out Protocol).
    [Optional] [For Clustered environments without HttpSession replication]
    [DefaultValue=none]-->
    <!--
<singleLogoutCallbackUrl>https://myHost:7002/myService/singleLogout</singleLogoutCallbackUrl
>-->

</client-config>
```

Figure 5 - ecas-config-*mywebapp*.xml Template

You will find the list of all properties and their usage in the following sections.

3.3. System Properties in configuration files

Java System Properties can be used in the ECAS Client configuration files.

The convention is to use the following syntax: **${*propertyName*}**

where *propertyName* can be any System Property available to the Java Runtime for which your application has read permission.

For example:

```
########################################################################
## singleLogoutCallbackUrl:
### URL of your application used to receive LogoutRequests from the ECAS Server (for the
Single Sign-Out Protocol).
### [Optional] [For Clustered environments without HttpSession replication]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.singleLogoutCallbackUrl=/context-
${weblogic.Name}/singleLogout
########################################################################
```

In this example, the System Property ${*weblogic.Name*} is replaced by the value of the corresponding System Property.

Note that System Properties are automatically expanded in the ECAS Client configuration files. To avoid such behaviour, specify the System Property

```
-Deu.cec.digit.ecas.client.configFile.expandProperties=false
```

in your server start-up script(s).

3.4. Setting the right configuration property

Depending on your needs, there are several configuration properties that you may want to define.

In order to…	**Set property…**
Fine-tune performance when connecting to ECAS	maxConnections connectionTimeout
Customize configuration	configFile configurationOrder configurationId (obsolete)

In order to…	**Set property…**
Tell my application server how to connect to ECAS (and its components)	ecasServerDirectHostName ecasServerDirectOneWaySslPort ecasServerDirectTwoWaySslPort ecasServerReverseProxyHostName ecasServerReverseProxyPort ecasBaseUrl initLoginUrl loginUrl validateUrl proxyUrl initSignatureUrl signatureUrl retrieveSignatureUrl transactionUrl certificateRevocationUrl
Tell ECAS how to connect to my application server	serverProtocol serverName serverPort serverSSLPort serverContextPath serviceUrl (not recommended) baseCallbackUrl proxyCallbackUrl singleLogoutCallbackUrl
Choose the information I want from ECAS	assuranceLevel groups requestingUserDetails
Fine-tune ECAS authentication	acceptStrengths acceptedTicketTypes renew proxyGrantingProtocol authorizedProxies authorizedProxy (obsolete)

In order to…	Set property…
Adjust ECAS security options	strictSSLHostnameVerification trustNonEcasJEESubject trustedCertificates applicationSecurityLevel negociatePrivateServiceTicket
Plug in specific behaviour through interfaces	proxyChainTrustHandler extraGroupHandler authEventListeners serviceResolver loginDateValidator httpRedirector ticketResolver redirectionInterceptors extendedUserDetailsTypeMapper Custom parameters
Configure my Web application's behaviour	reSubmitPosts advancedHttpSessionManagement

3.5. Configuration properties

3.5.1. *configFile*

Properties:	eu.cec.digit.ecas.client.configFile
XML:	configFile

This optional property is used to load the configuration properties from an external configuration file as described in section *3.1 - Configuration file name and location*.

3.5.2. *~~configurationId~~ (obsolete)*

Properties:	eu.cec.digit.ecas.client.filter.configurationId
XML:	configurationId

This optional property was used to define a unique identifier for your configuration. It is now obsolete and will be ignored by the ECAS Client.

3.5.3. *serverName*

Properties:	edu.yale.its.tp.cas.client.filter.serverName
XML:	serverName

This optional property specifies the hostname of the server running your application. It is used to construct the service parameter representing the URL where the user must be redirected to after successful authentication on the ECAS login screen. It can be used to specify a different hostname than the name of the local host, for instance, when the local server possesses more than one DNS name (multihome machine) or when behind a reverse proxy.

This property is ignored if serviceUrl is specified (see 3.5.4).

This property can be used together with the other properties: serverProtocol (see 3.5.19), serverPort (see 3.5.20), serverSSLPort (see 3.5.21) and serverContextPath (see 3.5.22) to construct the service URL that is passed to the ECAS Server for redirection after authentication.

Leave this property empty if you are using the serviceUrl or if you want to retrieve the HTTPS port at runtime from the local request.

For instance, if you do not want to construct the service URL using the values from your local server (e.g. because you are always accessing through the same reverse proxy), you can configure the following properties:

Properties format:

```
#############################################################################
## serverName:
### Name of your host running the ECAS Client-protected application.
### If you don't specify either 'serverName' or 'serviceUrl' or a custom 'serviceResolver',
### the value is retrieved from the local server.
### [Optional]
### [DefaultValue=null]
edu.yale.its.tp.cas.client.filter.serverName=webgate.ec.europa.eu
#############################################################################
## serverProtocol:
### Protocol can be either http or https.
### This is only needed for applications behind a reverse proxy that want
### to overwrite the protocol of the local server (i.e. when the
### application is not accessible internally and when the reverse proxy
### protocol is different from the local server's protocol)
### [Optional]
### [DefaultValue=none]
### LegalValues:
#eu.cec.digit.ecas.client.filter.serverProtocol=http
eu.cec.digit.ecas.client.filter.serverProtocol=https
###########################################################################
## serverPort:
### HTTP port of your host.
### [Optional]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.serverPort=443
###########################################################################
## serverSSLPort:
### HTTPS port of your host.
### [Optional]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.serverSSLPort=443
###########################################################################
## serverContextPath:
### Context root of the application.
### Must begin with a slash '/'.
### This is only needed for applications behind a reverse proxy that modifies
### the local application's context root and when the application is only
### accessible through that reverse proxy.
### [Optional]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.serverContextPath=/myWebAppContextPath
###########################################################################
```

XML format:

```
<!-- Name of your host running the ECAS Client-protected application.
If you don't specify either 'serverName' or 'serviceUrl' or a custom 'serviceResolver',
the value is retrieved from the local server.
[Optional]
[DefaultValue=null] -->
<cas:serverName>webgate.ec.europa.eu</cas:serverName>

<!-- Protocol can be either http or https.
This is only needed for applications behind a reverse proxy that want
to overwrite the protocol of the local server (i.e. when the
application is not accessible internally and when the reverse proxy
protocol is different from the local server's protocol)
[Optional]
[DefaultValue=none]
LegalValues: -->
<!--<serverProtocol>http</serverProtocol>-->
<serverProtocol>https</serverProtocol>

<!-- HTTP port of your host.
[Optional]
[DefaultValue=none] -->
<serverPort>443</serverPort>

<!-- HTTPS port of your host.
[Optional]
[DefaultValue=none] -->
<serverSSLPort>443</serverSSLPort>

<!-- Context root of the application.
Must begin with a slash '/'.
This is only needed for applications behind a reverse proxy that modifies
the local application's context root and when the application is only
accessible through that reverse proxy.
[Optional]
[DefaultValue=none] -->
<serverContextPath>/myWebAppContextPath</serverContextPath>
```

In this example, the service URL is constructed using https://webgate.ec.europa.eu:443/myWebAppContextPath as prefix. The rest of the URL is taken at runtime from the current request (e.g. the current page and query string).

When your application is always accessed via the same reverse proxy, these properties let you overwrite all the parts of the service URL instead of using the ones from the local server.

The properties you leave empty will be taken at runtime from the local request.

In most cases, you do not need to specify the serverName property and can let it be retrieved dynamically.

Leave this property empty if the service URL can use the local name of your server.

When the serverName property is not specified, the following process is followed:

1. The *Frontend Host* value of the local WebLogic server is retrieved. If it's not empty, it is the value that's used as serverName.
Note: the *Frontend Host* value can be configured with WebLogic Administration Console by following:
Home > Environment > Servers > (Admin)Server > Protocols > HTTP > Frontend Host

2. If the *Frontend Host* is empty, the *Listen Address* of the local WebLogic server is retrieved. If the *Listen Address* is not empty, it is used as serverName.
Note: the *Listen Address* value can be configured with WebLogic Administration Console by following:
Home > Environment > Servers > (Admin)Server > General > Listen Address

3. If both the *Frontend Host* and the *Listen Address* are empty, the local serverName is taken from the current `HttpServletRequest` by using its `getLocalName()` method.

3.5.4. serviceUrl (not recommended)

Properties:	edu.yale.its.tp.cas.client.filter.serviceUrl
XML:	serviceUrl

This optional property specifies a complete static service URL that will be passed to the ECAS Server for redirections after successful authentications. This property is only useful if your application possesses a central controller or a unique welcome page that can receive users after authentication whatever the resource they requested.

The serviceUrl property prevails over serverName (and serverProtocol, serverPort, serverSSLPort, serverContextPath) if both are configured.

Leave this property empty if you use serverName or if you want to let the service URL to be retrieved dynamically at runtime based upon the current request.

We do not recommend using this property.

3.5.5. ecasServerDirectHostName

Properties:	eu.cec.digit.ecas.client.filter.ecasServerDirectHostName
XML:	ecasServerDirectHostName

This optional property specifies the name of the host of the ECAS server for direct connections (without reverse proxy).

This is one out of five new properties to help customize which ECAS server environment is being accessed and through which URL.

The other 4 properties are:

- ecasServerDirectOneWaySslPort
- ecasServerDirectTwoWaySslPort
- ecasServerReverseProxyHostName
- ecasServerReverseProxyPort

These 5 properties let you configure the minimal changes needed to customize the ECAS server URLs while still benefitting from the default values existing for all relative URLs.

They offer more granularity than the *ecasBaseUrl* property.

The *ecasBaseUrl* property is applied indiscriminately to all ECAS server URLs.

As such, the *ecasBaseUrl* property is useful to configure the use of a different ECAS server which possesses only one possible URL.

On the contrary, these 5 properties are divided in 2 categories:

1. the direct access properties:
    a. ecasServerDirectHostName
    b. ecasServerDirectOneWaySslPort
    c. ecasServerDirectTwoWaySslPort
2. the reverse proxy properties:
    a. ecasServerReverseProxyHostName
    b. ecasServerReverseProxyPort

So this set of properties discriminates between direct-access and reverse-proxy-access ECAS server URLs i.e. between back-end URLs (used behind the scene in application-to-application communications) and front-end URLs (used by human end-users to access the ECAS Web site) which are typically accessed through a reverse proxy.

Thus these properties allow you to configure at once all the front-end URLs and/or all the back-end URLs independently.

For example, if you specify:

```
###########################################################################
## ecasServerReverseProxyHostName:
### Name of the reverse proxy host in front of the ECAS server for proxied connections
### [Optional]
### [DefaultValue=null]
eu.cec.digit.ecas.client.filter.ecasServerReverseProxyHostName=webgate.ec.europa.eu
###########################################################################
## ecasServerReverseProxyPort:
### Port of the reverse proxy in front of the ECAS server for proxied connections
### [Optional]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.ecasServerReverseProxyPort=443
###########################################################################
```

you would modify all the reverse-proxy-access URLs such as the login URL and the signature URL but would leave untouched the default values for the back-end URLs such as the *initLoginUrl* property or the *validateUrl* property.

Equivalent in XML format:

```
<!-- Name of the reverse proxy host in front of the ECAS server for proxied
connections
 [Optional]
 [DefaultValue=null]
-->
<ecasServerReverseProxyHostName>webgate.ec.europa.eu</ecasServerReverseProxyHostName>

<!-- Port of the reverse proxy in front of the ECAS server for proxied connections
 [Optional]
 [DefaultValue=none]
-->
<ecasServerReverseProxyPort>443</ecasServerReverseProxyPort>
```

For backward compatibility reasons, these properties are only taken into account when no conflicting property is configured (for instance, *ecasBaseUrl* is not set, *loginUrl* is not set or is not an absolute URL, etc).

These properties let you rely on the existing default values for all ECAS server URLs (which are all relative URLs) and let you change of ECAS server environment by modifying only a few hostname

and port properties, which are then reused to construct all ECAS server absolute URLs.
These properties are divided into two groups: the properties to be used for direct connections and the properties to be used for connections through reverse proxies.

On the contrary of the *ecasBaseUrl* property which applies for all (relative) ECAS server URLs, these two groups allow you to maintain two different network routes to connect to the ECAS server.

The properties for direct connections are used by all the backend network traffic (the communications between your application servers and the ECAS server happening behind the scene, such as Ticket validations).

The properties for reverse proxied connections are used by all the frontend network traffic (the communication between the end-user using her Web browser and the ECAS server, such as opening the ECAS login page URL).
Using these new properties simplifies your maintenance of the ECAS client configuration files and we recommend switching to this convention as soon as possible.

Example of valid values for *ecasServerDirectHostName*:

• ecas.cc.cec.eu.int	(default (production))
• ecast.cc.cec.eu.int	(for the test environment)
• ecasl.cc.cec.eu.int	(for the load-test environment)

3.5.6. ecasServerDirectOneWaySslPort

Properties:	eu.cec.digit.ecas.client.filter.ecasServerDirectOneWaySslPort
XML:	ecasServerDirectOneWaySslPort

This optional property specifies the One-Way SSL port of the ECAS server for direct connections (without reverse proxy).

See *3.5.5 ecasServerDirectHostName* above for the complete explanation.

Example of valid values for *ecasServerDirectOneWaySslPort*:

• 7002	(default (production))
• 17002	(for the dev2 environment)

3.5.7. ecasServerDirectTwoWaySslPort

Properties:	eu.cec.digit.ecas.client.filter.ecasServerDirectTwoWaySslPort
XML:	ecasServerDirectTwoWaySslPort

This optional property specifies the Two-Way SSL port of the ECAS server for direct connections (without reverse proxy).

See *3.5.5 ecasServerDirectHostName* above for the complete explanation.

Example of valid values for *ecasServerDirectTwoWaySslPort*:

7003	(default (production))
17003	(for the dev2 environment)

3.5.8. *ecasServerReverseProxyHostName*

Properties:	eu.cec.digit.ecas.client.filter.ecasServerReverseProxyHostName
XML:	ecasServerReverseProxyHostName

This optional property specifies the name of the reverse proxy host in front of the ECAS server for proxied connections.

See *3.5.5 ecasServerDirectHostName* above for the complete explanation.

Example of valid values for *ecasServerReverseProxyHostName*:

- ecas.ec.europa.eu
- webgate.ec.testa.eu

There is no default value, because there is no default reverse proxy at the Commission.

3.5.9. *ecasServerReverseProxyPort*

Properties:	eu.cec.digit.ecas.client.filter.ecasServerReverseProxyPort
XML:	ecasServerReverseProxyPort

This optional property specifies the port of the reverse proxy in front of the ECAS server for proxied connections.

See *3.5.5 ecasServerDirectHostName* above for the complete explanation.

Example of valid values for *ecasServerReverseProxyPort*:

- 443

There is no default value, because there is no default reverse proxy at the Commission.

3.5.10. *ecasBaseUrl*

Properties:	eu.cec.digit.ecas.client.filter.ecasBaseUrl
XML:	ecasBaseUrl

This optional property specifies a prefix for all ECAS Server URLs (such as *init login*, *login*, *signature* and *validate* URLs) which then allows using relative URLs for all ECAS Server URLs.

See also *3.5.5 ecasServerDirectHostName* above for an alternative way of configuring ECAS server URLs especially when behind reverse proxies.

Example of *ecasBaseUrl* valid values:

https://ecas.cc.cec.eu.int:7002	for internal deployment (within the E.C.) (default)
https://webgate.ec.europa.eu	for external deployment (not within the E.C.)

Leave this property empty if the default value https://ecas.cc.cec.eu.int:7002 (ECAS production) is suitable for your environment.

For applications deployed at the Data Centre, you should always leave this property empty and let the default value be used.

Override this property if you are configuring your application to use a *mock-up* server, a *test* or *load test* ECAS Server or if your application is not deployed at the Data Centre and the ECAS Server is unreachable through its direct (internal) URL (https://ecas.cc.cec.eu.int:7002/cas).

3.5.11. initLoginUrl

Properties:	eu.cec.digit.ecas.client.filter.initLoginUrl
XML:	initLoginUrl

This optional property specifies the ECAS Server URL where an ECAS login request transaction is initiated by the ECAS Client to configure all its authentication parameters such as the service, strength, renew, gateway…

This property is not used when the *LegacyHttpRedirector* is set to:

- `eu.cec.digit.ecas.client.http.BrowserPostRedirector` or
- `eu.cec.digit.ecas.client.http.LegacyHttpRedirector`.

See also 3.5.47 httpRedirector.

Example of valid values:

/cas/login/init	relative value (default)
https://ecas.cc.cec.eu.int:7002/cas/login/init	absolute value

If the value is a relative URL, the ecasBaseUrl will be used to prefix the specified value.

Leave this property empty if the default value `/cas/login/init` is suitable for your environment.

Note: the initLoginUrl is a "back-end" URL used by ECAS Client to connect to the ECAS Server and obtain a loginRequestId. It's an application-to-application URL that is never accessed by the end-users so if your application can reach the internal ECAS Server URL, ***never*** configure it to go through a reverse proxy.

The only reason why you would configure a proxied URL for this would be if your application isn't deployed in the Commission's network and cannot reach the internal ECAS Server URL. In this case we recommend that you keep a relative URL for this property and that you configure instead a proxied URL for the *ecasBaseUrl* property, which applies to all relative URLs.

3.5.12. loginUrl

Properties:	edu.yale.its.tp.cas.client.filter.loginUrl
XML:	loginUrl

This optional property specifies the ECAS Server login URL where the user will be redirected to perform her authentication.

Example of valid values:

/cas/login	relative value (default)
https://ecas.cc.cec.eu.int:7002/cas/login	internal to the Commission's network
https://www.cc.cec/cas/login	behind the IntraComm reverse proxy
https://webgate.ec.europa.eu/cas/login	behind the Webgate reverse proxy

If the value is a relative URL, the ecasBaseUrl will be used as prefix to the specified value.

You should use an absolute value when the ecasBaseUrl is not the correct prefix. For example when your application is deployed at the Data Centre (thus ecasBaseUrl is left empty) and your application must be accessed through a reverse proxy (thus loginUrl is set to an absolute value with the reverse proxy address).

Leave this property empty if the default value `/cas/login` is suitable for your environment.

3.5.13. validateUrl

Properties:	edu.yale.its.tp.cas.client.filter.validateUrl
XML:	validateUrl

This optional property specifies the ECAS Server validation URL where the ECAS client asks for validation of a service ticket.

The new ticket validation URL `/cas/TicketValidationService` was introduced to accommodate for current (since v. 1.11) and future changes.

This new URL accepts the following request parameters to define its validation behaviour:

- assuranceLevel
- ticketTypes
- proxyGrantingProtocol

These parameters are also defined in the ECAS Client configuration file and are explained in the following sections.

User populations are defined by the assuranceLevel property.

Leave this property empty if the default value `/cas/TicketValidationService` is suitable for your environment.

If the value is a relative URL, the ecasBaseUrl will be used as prefix to the specified value.

Note: the validateUrl is a “back-end” URL used by the ECAS Client to connect to the ECAS Server and obtain ticket validation XML responses. It’s an application-to-application URL that is never accessed by the end-users so if your application can reach the internal ECAS Server URL, ***never*** configure it to go through a reverse proxy.

The only reason why you would configure a proxied URL for this would be if your application isn’t deployed in the Commission’s network and cannot reach the internal ECAS Server URL. In this case we recommend that you keep a relative URL for this property and that you configure instead a proxied URL for the *ecasBaseUrl* property, which applies to all relative URLs.

3.5.14. assuranceLevel

Properties:	eu.cec.digit.ecas.client.filter.assuranceLevel
XML:	assuranceLevel

This optional ECAS Client configuration property specifies the user's identity assurance level the application requires.

If users with an assurance level lower than the one configured here try to access the application, an INVALID_USER error code is returned by ECAS.

Valid values:

- TOP (default)
- HIGH
- MEDIUM
- LOW

Leave this property empty if the default value is suitable for your environment.

Applications can retrieve the user's assurance level via the *UserDetailsAccessible* mixin interface. Method `UserDetailsAccessible#getAssuranceLevel()` returns a type-safe enum of type `eu.cec.digit.ecas.client.constants.AssuranceLevel`.

The assurance levels correspond to the following (deprecated) validation URLs:

Assurance level	Assurance level alias (= populations)	Old validation URL (deprecated)
TOP	INTERNAL	`/cas/strictValidate`
HIGH	INTERINSTITUTIONAL	`/cas/interinstitutionalValidate`
MEDIUM	SPONSORED	`/cas/sponsorValidate`
LOW	SELF_REGISTERED	`/cas/laxValidate`

Figure 6 - ECAS assurance levels per validation URL

The assurance levels accept or deny the following user populations. If the authenticated user belongs to a rejected population, the ECAS Server replies with an INVALID_USER error code.

Assurance level	Commission users (including generic accounts)	Inter-institutional users	Sponsored external users	Self-registered external users
TOP	☑ Yes	☒ No	☒ No	☒ No
HIGH	☑ Yes	☑ Yes	☒ No	☒ No
MEDIUM	☑ Yes	☑ Yes	☑ Yes	☒ No
LOW	☑ Yes	☑ Yes	☑ Yes	☑ Yes

Figure 7 - ECAS assurance levels per population

3.5.15. acceptedTicketTypes

Properties:	eu.cec.digit.ecas.client.filter.acceptedTicketTypes
XML:	acceptedTicketTypes

This optional ECAS Client configuration property specifies the list of ticket types your application accepts. More specifically, it was created for desktop applications which want to access remote ECAS protected resources.

If users try to access the application with other ticket types than the ones specified here, an INVALID_TICKET error code is returned by ECAS.

Valid values:

• Properties	a comma-separated list
• XML	a sequence of *ticketType* tags

Where each ticket type is one of the following:

- SERVICE
- PROXY
- DESKTOP

The default value is *SERVICE,PROXY*.

Leave this property empty unless you need to accept Desktop Proxy Tickets from an ECAS-enabled desktop application or want to prevent one type of tickets to be used to access your application.

Applications can retrieve the user's ticket type via the new *TicketBased* mixin interface. Method `TicketBased#getTicketType()` returns a type-safe enum of type `eu.cec.digit.ecas.client.constants.TicketType`.

The `TicketBased` interface:

```
package eu.cec.digit.ecas.client.jaas;

import eu.cec.digit.ecas.client.constants.TicketType;

public interface TicketBased {

    TicketType getTicketType();
}
```

Figure 8 - TicketBased interface

3.5.16. *proxyGrantingProtocol*

Properties:	eu.cec.digit.ecas.client.filter.proxyGrantingProtocol
XML:	proxyGrantingProtocol

This optional ECAS Client configuration property specifies the protocol to be used to obtain ProxyGrantingTickets (PGT).

Valid values:

- PGT_URL
- CLIENT_CERT
- DESKTOP

There is no default value.

Leave this property empty if you are not an ECAS Proxy application.

Applications can retrieve the proxy granting protocol it used via the *ProxiedUser* mixin interface. The *ProxiedUser* interface is implemented by the ECAS User Principals and indicates that the user possesses a Proxy Granting Ticket. Method `ProxiedUser#getProxyGrantingProtocol()` returns a type-safe enum of type `eu.cec.digit.ecas.client.constants.ProxyGrantingProtocol`.

3.5.17. proxyUrl

Properties:	edu.yale.its.tp.cas.client.filter.proxyUrl
XML:	proxyUrl

This optional property specifies the ECAS Server URL where the ECAS Client requests Proxy Granting Tickets (PGT).

Example valid value:

- /cas/proxy (default)

If the value is a relative URL, the ecasBaseUrl will be used as prefix to the specified value.

Leave this property empty if the default value `/cas/proxy` is suitable for your environment.

Note: the proxyUrl is a “back-end” URL used by the ECAS Client to connect to the ECAS Server and requests Proxy Granting Tickets. It’s an application-to-application URL that is never accessed by the end-users so if your application can reach the internal ECAS Server URL, ***never*** configure it to go through a reverse proxy.

The only reason why you would configure a proxied URL for this would be if your application isn’t deployed in the Commission’s network and cannot reach the internal ECAS Server URL. In this case we recommend that you keep a relative URL for this property and that you configure instead a proxied URL for the *ecasBaseUrl* property, which applies to all relative URLs.

3.5.18. renew

Properties:	edu.yale.its.tp.cas.client.filter.renew
XML:	Renew

Use this optional property to always force renewal of authentication for your application, i.e. never use Single Sign-On and to force users to re-authenticate by re-entering their login and password.

This property should never be used except for sensitive applications used in public kiosks such as eVoting.

Valid values:

- true
- false (default)

Leave this property empty if the default value `false` is suitable for your environment.

3.5.19. serverProtocol

Properties:	eu.cec.digit.ecas.client.filter.serverProtocol
XML:	serverProtocol

This optional property specifies the protocol scheme to be used while constructing the service URL to be passed to the ECAS Server for redirection after authentication.

This is only needed for applications behind a reverse proxy that want to overwrite the protocol scheme of the local server (i.e. when the application is not accessible internally and when the reverse proxy protocol scheme is different from the local server's protocol scheme).

Valid values:

http	when all requests use HTTP
https	when all requests use HTTPS
dynamic	when the local application server is able to decide dynamically at runtime whether the current request is secure or not

Leave this property empty if you are using the serviceUrl or if you want to retrieve the protocol scheme at runtime from the local request.

3.5.20. serverPort

Properties:	eu.cec.digit.ecas.client.filter.serverPort
XML:	serverPort

This optional property specifies the HTTP port to be used while constructing the service URL to be passed to the ECAS Server for redirection after authentication.

This is only needed for applications behind a reverse proxy that want to overwrite the HTTP port of the local server (i.e. when the application is not accessible internally and when the reverse proxy port is different from the local server's port). See also *3.5.3 serverName*.

Leave this property empty if you are using the serviceUrl or if you want to retrieve the HTTP port at runtime from the local request.

3.5.21. serverSSLPort

Properties:	eu.cec.digit.ecas.client.filter.serverSSLPort
XML:	serverSSLPort

This optional property specifies the HTTPS port to be used while constructing the service URL to be passed to the ECAS Server for redirection after authentication.

This is only needed for applications behind a reverse proxy that want to overwrite the HTTPS port of the local server (i.e. when the application is not accessible internally and when the reverse proxy SSL port is different from the local server's SSL port). See also *3.5.3 serverName*.

Leave this property empty if you are using the serviceUrl or if you want to retrieve the HTTPS port at runtime from the local request.

3.5.22. serverContextPath

Properties:	eu.cec.digit.ecas.client.filter.serverContextPath
XML:	serverContextPath

This optional property specifies the web application context root to be used while constructing the service URL to be passed to the ECAS Server for redirection after authentication.

This is only needed for applications behind a reverse proxy that wants to overwrite the context root of your application in the local server (i.e. when the application is not accessible internally and when the reverse proxy rewrites the context root of the application).

It must begin with a slash '/'.

Leave this property empty if you are using the serviceUrl or if you want to retrieve the context root at runtime from the local request.

3.5.23. ~~authorizedProxy~~ (obsolete)

Properties:	edu.yale.its.tp.cas.client.filter.authorizedProxy
XML:	authorizedProxy

This optional property is deprecated. Please use 3.5.24 *authorizedProxies* instead.

However if you specify both *authorizedProxy* and *authorizedProxies*, it is *authorizedProxy* that prevails for compatibility reasons.

3.5.24. authorizedProxies

Properties:	eu.cec.digit.ecas.client.filter.authorizedProxies
XML:	authorizedProxies

This optional property specifies a list of ECAS proxies authorized to access your application using proxy tickets.

Valid values:

• Properties	a comma-separated list
• XML	a sequence of *proxy* tags

Leave this property empty if you are not an ECAS Proxy application.

Example in properties format:

```
eu.cec.digit.ecas.client.filter.authorizedProxies=https://host1.cec.eu.int/service1,\
https://host2.cec.eu.int/service2,https://host3.cec.eu.int/service3,\
https://host4.cec.eu.int/service4
```

Example in XML format:

```
<authorizedProxies>
    <proxy>https://host1.cec.eu.int/service1</proxy>
    <proxy>https://host1.cec.eu.int/service2</proxy>
    <proxy>https://host1.cec.eu.int/service3</proxy>
    <proxy>https://host1.cec.eu.int/service4</proxy>
</authorizedProxies>
```

The complete ProxyChain (list of ECAS proxy applications) is now available programmatically through the *eu.cec.digit.ecas.client.jaas.ProxiedUser* mixin interface so that protected applications can know which their calling client application is and which ECAS proxy applications were in the proxy chain.

The accessor is *eu.cec.digit.ecas.client.jaas.ProxiedUser#getProxies()*

3.5.25. proxyChainTrustHandler

Properties:	eu.cec.digit.ecas.client.filter.proxyChainTrustHandler
XML:	proxyChainTrustHandler

This optional property specifies the class which implements `eu.cec.digit.ecas.client.validation.ProxyChainTrustHandlerIntf` to be used.

The `ProxyChainTrustHandlerIntf` interface lets you specify the strategy you want to use to validate the whole list of ECAS proxies between your application and the real authentication.

Leave this property empty if you are not an ECAS Proxy application.

The `ProxyChainTrustHandlerIntf` interface skeleton:

```
package eu.cec.digit.ecas.client.validation;

import eu.cec.digit.ecas.client.jaas.InvalidProxyChainException;

public interface ProxyChainTrustHandlerIntf {

    String[] getTrustedProxies();

    void setTrustedProxies(String[] trustedProxies);

    void validate(String[] proxyChain) throws InvalidProxyChainException;
}
```

Figure 9 - ProxyChainTrustHandlerIntf interface

Two implementations are provided:

1. `eu.cec.digit.ecas.client.validation.ProxyChainTrustHandler`: this is the default implementation. It checks that all the ECAS proxies in the chain are trusted (e.g. are mentioned in the authorizedProxies list). If one of the ECAS proxies in the chain is not in your configured list, the proxy ticket is not accepted and the user is not authenticated in your service.
2. `eu.cec.digit.ecas.client.validation.FirstParentProxyChainTrustHandler`: this implementation only checks that the application in front of you is trusted (is present in your authorizedProxies list). Using this implementation, you only have to trust the application calling your service and you delegate the responsibility of trusting your caller's callers to each application in the chain. If all the applications in the ECAS proxy chain follow this strategy, a lot more applications could use your service as long as they go through one application that you directly trust.

The first implementation is for secure services that need to keep control over who is allowed to use them. The second implementation is there to offer more flexibility and delegate the trust among "good citizen" applications.

Note: Implementations of this interface MUST also implement `java.io.Serializable`!

Note: If you specify an incorrect class name or if the class you specify cannot be found in your application classpath, an exception of type `eu.cec.digit.ecas.client.configuration.ConfigurationException` is thrown by the ECAS Client (the cause of which is going to be `java.lang.ClassNotFoundException`).

Note: If this implementation needs to rely upon the `eu.cec.digit.ecas.client.configuration.EcasConfigurationIntf`, make it also implement `eu.cec.digit.ecas.client.configuration.ConfigurationDependent`.

By default, the ECAS Client will call the `setTrustedProxies(String[])` method with the authorizedProxies list in the current `eu.cec.digit.ecas.client.configuration.EcasConfiguration` instance.

Refer to the javadoc for more details.

3.5.26. baseCallbackUrl

Properties:	eu.cec.digit.ecas.client.filter.baseCallbackUrl
XML:	baseCallbackUrl

This optional property can be set to define a prefix to be appended to other callback URL properties such as the *proxyCallbackUrl* and the *singleLogoutCallbackUrl* properties when they are configured as relative URLs.

For example, consider the following existing configuration:

```
###########################################################################
## proxyCallbackUrl:
### URL of your application used to receive ProxyTickets from the ECAS Server.
### [Optional] [For ECASProxies]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.proxyCallbackUrl=https://my.application.server:1042/myContex
tPath/proxyReceptor
###########################################################################
## singleLogoutCallbackUrl:
### URL of your application used to receive LogoutRequests from the ECAS Server (for the
Single Sign-Out Protocol).
### [Optional] [For Clustered environments without HttpSession replication]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.singleLogoutCallbackUrl=https://my.application.server:1042/m
yContextPath/singleLogout
###########################################################################
```

With the *baseCallbackUrl* property, the same configuration can be rewritten as follows:

```
###########################################################################
## baseCallbackUrl:
### The prefix to append to other callback URL properties when they are set as relative
URLs.
### [Optional]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.baseCallbackUrl=https://my.application.server:1042/myContext
Path
###########################################################################
## proxyCallbackUrl:
### URL of your application used to receive ProxyTickets from the ECAS Server.
### [Optional] [For ECASProxies]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.proxyCallbackUrl=/proxyReceptor
###########################################################################
## singleLogoutCallbackUrl:
### URL of your application used to receive LogoutRequests from the ECAS Server (for the
Single Sign-Out Protocol).
### [Optional] [For Clustered environments without HttpSession replication]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.singleLogoutCallbackUrl=/singleLogout
###########################################################################
```

Note that the *proxyCallbackUrl* property must be an HTTPS URL. Therefore if *proxyCallbackUrl* is a relative URL, the *baseCallbackUrl* property must start with "https://".

3.5.27. proxyCallbackUrl

Properties:	eu.cec.digit.ecas.client.filter.proxyCallbackUrl
XML:	proxyCallbackUrl

This optional property specifies the URL of your application used to receive Proxy Tickets (PT) from the ECAS Server.

Leave this property empty if you are not an ECAS Proxy application.

Note that this property can be defined as relative URL. In this case:

- if the *baseCallbackUrl* property is defined, it is used as prefix to construct an absolute URL for the *proxyCallbackUrl*;
- if no *baseCallbackUrl* property is configured, the *proxyCallbackUrl* property is automatically converted into an absolute URL by retrieving information from the deployed application and its local application server.

> Since the missing information needs to be retrieved from the local application server in a proprietary manner, this feature is only supported on Oracle WebLogic Server, Apache Tomcat and JBoss AS.
>
> However the feature works with any application server if the application is only deployed over HTTPS.

For example, consider the following configuration:

```
###########################################################################
## baseCallbackUrl:
### The prefix to append to other callback URL properties when they are set as relative
URLs.
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.baseCallbackUrl=
###########################################################################
## proxyCallbackUrl:
### URL of your application used to receive ProxyTickets from the ECAS Server.
### [Optional] [For ECASProxies]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.proxyCallbackUrl=/proxyReceptor
###########################################################################
```

The ECAS Client automatically completes the *proxyCallbackUrl* configuration property with values taken from the local application server running the application.

At runtime, assuming that the hostname retrieved in the local application server is **my.application.server**, the SSL/TLS port is **1042** and the application context-path is **/myContextPath**, the values of these properties are computed as follows:

```
eu.cec.digit.ecas.client.filter.proxyCallbackUrl=https://my.application.server:1042/myContextPath/proxyReceptor
```

Note that the values are taken from the running application server, not from the ECAS Client configuration.

3.5.28. *singleLogoutCallbackUrl*

Properties:	eu.cec.digit.ecas.client.filter.singleLogoutCallbackUrl
XML:	singleLogoutCallbackUrl

This optional property specifies the callback URL in the application to which the ECAS server will send Single Logout requests when users log out of the ECAS Server.

Previously, using the Single Logout Protocol in clustered environments required that *HttpSession* replication was enabled. Without *HttpSession* replication, a Single Logout request sent by the ECAS server could be received by the wrong cluster node, which would not be able to propagate the session invalidation to the right node.

While the obvious solution is to activate session replication in the cluster, some customers who only had session stickiness enabled but not session replication asked for an alternate solution.

Therefore we introduced a new parameter in the protocol to indicate on which node the Single Logout Request must be sent by the ECAS server.

This is configured in the ECAS client through this new configuration property, which is the location of the SAML *SingleLogoutService* where the ECAS server sends back SAMLP *LogoutRequests* to invalidate user sessions.

It is the URL of the application where the ECAS server must send back Single Logout requests to, so it can be used to force using a specific cluster node. And each cluster node may have this property configured to its own URL.

> This property is only useful for clustered applications without HttpSession *replication* but with session *stickiness*. If session replication is enabled, you don't need to configure this property.

Note that, if this property is not specified, the ECAS server sends Single Logout requests back to the originating service URL used at authentication time.

Example:

```
###########################################################################
## singleLogoutCallbackUrl:
### URL of your application used to receive LogoutRequests from the ECAS Server (for the
Single Sign-Out Protocol).
### [Optional] [For Clustered environments without HttpSession replication]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.singleLogoutCallbackUrl=https://clusterNode01:7002/app/singl
eLogout
###########################################################################
```

Please note that every URL protected by the ECAS client is able to process a Single Logout Request sent by the ECAS server (as long as the ECAS client is configured to opt-in the Single Logout Protocol, see 19 Single Logout), so there is no need to configure any additional Servlet mapped to the configured value, any ECAS-protected page will do.

This property can be defined as relative URL. In this case:

- if the *baseCallbackUrl* property is defined, it is used as prefix to construct an absolute URL for the *singleLogoutCallbackUrl*;
- if no *baseCallbackUrl* property is configured, the *singleLogoutCallbackUrl* property is automatically converted into an absolute URL by retrieving information from the deployed application and its local application server.

For example, consider the following configuration:

```
###########################################################################
## baseCallbackUrl:
### The prefix to append to other callback URL properties when they are set as relative
URLs.
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.baseCallbackUrl=
###########################################################################
## singleLogoutCallbackUrl:
### URL of your application used to receive LogoutRequests from the ECAS Server (for the
Single Sign-Out Protocol).
### [Optional] [For Clustered environments without HttpSession replication]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.singleLogoutCallbackUrl=/singleLogout
###########################################################################
```

The ECAS Client automatically completes the *singleLogoutCallbackUrl* configuration property with values taken from the local application server running the application.

At runtime, assuming that the hostname retrieved in the local application server is **my.application.server**, the SSL/TLS port is **1042** and the application context-path is **/myContextPath**, the values of these properties are computed as follows:

```
eu.cec.digit.ecas.client.filter.singleLogoutCallbackUrl=https://my.application.server:1042/myContextPath/singleLogout
```

Note that the values are taken from the running application server, not from the ECAS Client configuration.

3.5.29. ~~applicationServer~~ (obsolete)

Properties:	eu.cec.digit.ecas.client.filter.applicationServer
XML:	applicationServer

This optional property is used to specify the Application Server running your application.

The default value is “weblogic”.

Leave this property empty, it is not used anymore.

3.5.30. groups

Properties:	eu.cec.digit.ecas.client.filter.groups
XML:	Groups

This optional property specifies the list of CUD groups that your application wants ECAS Server to return in the validation response when the user belongs to one of these groups.

Valid values:

Properties	a comma-separated list
XML	a sequence of *group* tags

If the user belongs to one of these groups, it will be added in the user's Subject as Group Principal.

Leave this property empty if your application does not use any CUD groups.

Example in properties format:

```
eu.cec.digit.ecas.client.filter.groups=MY_CUD_GROUP_1,MY_CUD_GROUP_2,\
MY_CUD_GROUP_3,MY_CUD_GROUP_4
```

Example in XML format:

```
<groups>
    <group>MY_CUD_GROUP_1</group>
    <group>MY_CUD_GROUP_2</group>
    <group>MY_CUD_GROUP_3</group>
</groups>
```

This is equivalent as asking to ECAS the following four questions:

- Is the user a member of the group “MY_CUD_GROUP_1”?
- Is the user a member of the group “MY_CUD_GROUP_2”?
- Is the user a member of the group “MY_CUD_GROUP_3”?

You can also use the wildcard ‘*’ such as in

```
eu.cec.digit.ecas.client.filter.groups=MY_CUD_GROUP_*
```

or

```
<groups>
    <group>MY_CUD_GROUP_*</group>
</groups>
```

which will return all groups starting with “MY_CUD_GROUP_”, e.g. “MY_CUD_GROUP_1”, “MY_CUD_GROUP_2” and “MY_CUD_GROUP_3”.

These groups will have to be mapped to WebLogic roles in *weblogic.xml*. If you do not map them to WebLogic roles, they will be taken literally as global role names[2].

The groups specified here will be the only ones returned in ECAS ticket validation responses. A given ticket validation response contains a group only if the authenticated user belongs to that CUD group and only if that group was asked in the validation request (i.e. by specifying it in this property).

3.5.31. acceptStrengths

Properties:	eu.cec.digit.ecas.client.filter.acceptStrengths
XML:	acceptStrengths

This optional property specifies the list of authentication strength levels your application trusts.

Valid values:

Properties	a comma-separated list, ordered by preference
XML	a sequence of *strength* tags, ordered by preference

Where each strength is one of the following:

- PASSWORD the default level in production (previously known as STRONG)
- MOBILE_APP Single factor authentication using a PIN code. This strength is usable only on mobile devices using the ECAS mobile application.
- PASSWORD _SMS the two-factor authentication using the password plus an off-band SMS challenge (only for registered applications)
- PASSWORD_TOKEN the two-factor authentication using the password plus a security token challenge
- PASSWORD_TOKEN_CRAM the two-factor authentication using the password plus a Challenge-Response Authentication Mechanism (CRAM) performed via a hardware token
- PASSWORD_SOFTWARE_TOKEN Two-factor authentication using the password plus a Challenge-Response Authentication Mechanism using a QR code and the ECAS mobile application.
- PASSWORD_MOBILE_APP Two-factor authentication using the password and the ECAS mobile application. Please note that the mobile device has to have an active internet connection and the ECAS mobile application installed to perform an authentication using this strength.
- CLIENT_CERT the level used in production by the end-to-end monitoring tools and script accounts which authenticate using client X.509 certificates.
- BASIC only used by the ECAS mock-up server

In the future, ECAS could use different levels of authentication such as "STORK" for identities federated with Member States.

This property accepts a list of strengths (respectively a sequence of *strength* tags) ordered by preference.

For example, if you want to accept Single Sign-On with STRENGTH-1, STRENGTH-2 and STRENGTH-3 and you want to have unauthenticated users to authenticate the first time using

[2] And WebLogic will print a warning message about it, but in *weblogic.xml* you can specify them as externally-defined roles (global roles).

STRENGTH-1, specify "STRENGTH-1,STRENGTH-2,STRENGTH-3" (or the corresponding sequence of *strength* tags if using XML).

Otherwise, if you want users to authenticate using STRENGTH-2 first but also want to trust STRENGTH-1 and STRENGTH-3 specify "STRENGTH-2,STRENGTH-1,STRENGTH-3" (or the corresponding sequence of *strength* tags).

Leave this property empty if you only want to trust the default PASSWORD, PASSWORD_SMS, PASSWORD_TOKEN and CLIENT_CERT strengths.

3.5.32. maxConnections

Properties:	eu.cec.digit.ecas.client.filter.maxConnections
XML:	maxConnections

This optional property specifies the maximum number of HTTPS connections to the ECAS Server the ECAS Client will keep alive in the pool.

Valid values:

- 2 (default)
- an integer

Leave this property empty unless you specially need to change it or if you have specific performance issues with the ECAS Client.

3.5.33. connectionTimeout

Properties:	eu.cec.digit.ecas.client.filter.connectionTimeout
XML:	connectionTimeout

This optional property specifies the connection timeout in milliseconds for HTTPS connections to the ECAS Server.

Valid values:

- 180000 (default)
- an integer

Leave this property empty unless you specially need to change it or if you have specific performance issues with the ECAS Client.

3.5.34. strictSSLHostnameVerification

Properties:	eu.cec.digit.ecas.client.filter.strictSSLHostnameVerification
XML:	strictSSLHostnameVerification

This optional property specifies whether SSL hostname verification is performed strictly or not. In production, this property should be set to *true* to make sure that the hostnames used in URLs are compliant with the hostnames specified in the SSL certificates.

Valid values:

- true (default)
- false

3.5.35. extraGroupHandler

Properties:	eu.cec.digit.ecas.client.filter.extraGroupHandler
XML:	extraGroupHandler

This optional property specifies the class which implements `eu.cec.digit.ecas.client.validation.ExtraGroupHandlerIntf` to be used.

The extraGroupHandler extension is used to populate authenticated users' Subject with custom application groups.

Leave this property empty if you are not using a custom ExtraGroupHandler implementation.

The `ExtraGroupHandlerIntf` interface skeleton:

```
package eu.cec.digit.ecas.client.validation;

import java.util.List;

public interface ExtraGroupHandlerIntf {

    List getGroups(String username) throws ExtraGroupHandlingException;
}
```

Figure 10 - ExtraGroupHandlerIntf interface

Remarks:

- Implementations of this interface MUST also implement `java.io.Serializable`!
- The `java.util.List` returned by the `getGroups(String)` method is a list of `java.lang.Strings` representing the names of the group the user is a member of.
  The argument “username” is the Commission’s unique user id (uid).
- If you specify an incorrect class name or if the class you specify cannot be found in your application classpath, an exception of type `eu.cec.digit.ecas.client.configuration.ConfigurationException` is thrown by the ECAS Client (the cause of which is going to be `java.lang.ClassNotFoundException`).
- If this implementation needs to rely upon the `eu.cec.digit.ecas.client.configuration.EcasConfigurationIntf`, make it also implement `eu.cec.digit.ecas.client.configuration.ConfigurationDependent`.

Since version 1.8.7, if your application is configured to request user details (see 3.5.43 – requestingUserDetails), you may choose to implement another `ExtraGroupHandler` interface, the `eu.cec.digit.ecas.client.validation.UserDetailsExtraGroupHandlerIntf`, which allows to have access to all the user details (i.e. the user attributes) and hence to retrieve groups based upon the specific user attributes you need.

```
package eu.cec.digit.ecas.client.validation;

import java.util.List;

public interface UserDetailsExtraGroupHandlerIntf extends ExtraGroupHandlerIntf {

    List getGroups(DetailedAuthenticationSuccess detailedAuthenticationSuccess)
        throws ExtraGroupHandlingException;
}
```

Figure 11 - UserDetailsExtraGroupHandlerIntf interface

In this interface, the argument of the `getGroups()` method is a `eu.cec.digit.ecas.client.validation.DetailedAuthenticationSuccess` instance instead of the user's uid.

The `DetailedAuthenticationSuccess` interface skeleton:

```
package eu.cec.digit.ecas.client.validation;

import eu.cec.digit.ecas.client.jaas.UserDetailsAccessible;

public interface DetailedAuthenticationSuccess
        extends AuthenticationSuccess, UserDetailsAccessible {
}
```

Figure 12 - DetailedAuthenticationSuccess interface

The `DetailedAuthenticationSuccess` interface represents a successful authentication response and gives access to all the attributes of such a response.

To perform your group membership lookups, you may use any accessor available in `eu.cec.digit.ecas.client.validation.AuthenticationSuccess` and in `eu.cec.digit.ecas.client.jaas.UserDetailsAccessible`:

```
package eu.cec.digit.ecas.client.validation;

import java.util.Date;
import java.util.List;

public interface AuthenticationSuccess extends ValidationResult {

    String getUser();

    String getPgtIou();

    String getPgtId();

    List getGroups();

    Date getLoginDate();

    String getStrength();

    List getProxies();

    TicketType getTicketType();

    ProxyGrantingProtocol getProxyGrantingProtocol();
}
```

Figure 13 - AuthenticationSuccess interface

```
package eu.cec.digit.ecas.client.jaas;

public interface UserDetailsAccessible {
    AssuranceLevel getAssuranceLevel();

    String getDomain();

    String getDomainUsername();

    String getEmail();

    String getEmployeeNumber();

    String getEmployeeType();

    String getFirstName();

    String getLastName();

    String getLocale();

    String getMobilePhoneNumber();

    String getOrgId();

    Map/*<RegistrationLevel, String>*/ getRegistrationLevelVersions();

    String getStorkId();

    String getTelephoneNumber();

    String getTimeZone();

    String getTokenCramId();

    String getTokenId();

    String getUid();

    String getUnversionedUid();

    String getUserManager();
}
```

Figure 14 - UserDetailsAccessible interface

For example, the `UserDetailsExtraGroupHandlerIntf` lets you create and configure a custom `ExtraGroupHandler` retrieving groups in your application based upon let's say the email, the telephone number or the department number or any combination of available user attributes.

More details can be found in section 11.2 – "Configure a custom ECAS ExtraGroupHandler" and in the javadoc.

3.5.36. *authEventListeners*

Properties:	eu.cec.digit.ecas.client.filter.authEventListeners
XML:	authEventListeners

This optional property specifies the list of authentication event listeners, if any. These listeners must implement the `eu.cec.digit.ecas.client.event.AuthenticationEventListener` interface.

Valid values:

• Properties	a comma-separated list
• XML	a sequence of *listener* tags

Leave this property empty if you do not want to process authentication events.

Note: Implementations of this interface MUST also implement `java.io.Serializable`!

Note: If you specify an incorrect class name or if the class you specify cannot be found in your application classpath, an exception of type `eu.cec.digit.ecas.client.configuration.ConfigurationException` is thrown by the ECAS Client (the cause of which is going to be `java.lang.ClassNotFoundException`).

3.5.37. configurationOrder

Properties:	eu.cec.digit.ecas.client.filter.configurationOrder
XML:	configurationOrder

This optional property defines the configuration order to use.

The property should not be used in the configuration file; it is only useful in the MBean properties.

Valid values:

• `mbean`	the EcasIdentityAsserter MBean
• `file`	the external configuration file (usually called `ecas-config.xml` or `ecas-config.properties`)
• `descriptor`	the resource or deployment descriptor (such as `web.xml`)
• `<<`	overridden by

- `mbean << file << descriptor`(default)
- `file << descriptor << mbean`

The default configuration chain order is `mbean << file << descriptor`, which means first the MBean configuration, then the external file, then the deployment descriptor, each time overriding already defined properties.

Hence, with the default value, the deployment descriptor takes precedence over the external file, and the external file takes precedence over the MBean.

Another useful configuration for single-application domains is `file << descriptor << mbean`, where it is the MBean that prevails instead of the descriptor for application entirely configurable by the WebLogic Domain Administrator.

3.5.38. initSignatureUrl

Properties:	eu.cec.digit.ecas.client.filter.initSignatureUrl
XML:	initSignatureUrl

This optional property specifies the ECAS Server URL where an ECAS signature transaction is initiated by the ECAS Client to configure all its signature parameters and to send the document to sign.

Example of valid values:

• /cas/signature/init	relative value	(default)
• https://ecas.cc.cec.eu.int:7002/cas/signature/init	absolute value	

If the value is a relative URL, the ecasBaseUrl will be used as prefix to the specified value.

Leave this property empty if the default value `/cas/signature/init` is suitable for your environment.

Note: the initSignatureUrl is a “back-end” URL used by ECAS Client to connect to the ECAS Server and obtain a signatureRequestId. It’s an application-to-application URL that is never accessed by the end-users so if your application can reach the internal ECAS Server URL, ***never*** configure it to go through a reverse proxy.

The only reason why you would configure a proxied URL for this would be if your application isn’t deployed in the Commission’s network and cannot reach the internal ECAS Server URL. In this case we recommend that you keep a relative URL for this property and that you configure instead a proxied URL for the *ecasBaseUrl* property, which applies to all relative URLs.

3.5.39. signatureUrl

Properties:	eu.cec.digit.ecas.client.filter.signatureUrl
XML:	signatureUrl

This optional property specifies the ECAS Server signature URL where the user will be redirected to sign the document.

Example of valid values:

/cas/signature/sign.do	relative value (default)
https://ecas.cc.cec.eu.int:7002/cas/signature/sign.do	internal to the Commission's network
https://www.cc.cec/cas/signature/sign.do	behind the IntraComm reverse proxy
https://webgate.ec.europa.eu/cas/signature/sign.do	behind the Webgate reverse proxy

If the value is a relative URL, the ecasBaseUrl will be used as prefix to the specified value.

You should use an absolute value when the ecasBaseUrl is not the correct prefix. For example when your application is deployed at the Data Centre (thus ecasBaseUrl is left empty) and your application must be accessed through a reverse proxy (thus signatureUrl is set to an absolute value with the reverse proxy address).

Leave this property empty if the default value `/cas/signature/sign.do` is suitable for your environment.

3.5.40. retrieveSignatureUrl

Properties:	eu.cec.digit.ecas.client.filter.retrieveSignatureUrl
XML:	retrieveSignatureUrl

This optional property specifies the ECAS Server URL where the ECAS Client retrieves the signature once the document has been signed.

Example of valid values:

/cas/signature/get	relative value	(default)
https://ecas.cc.cec.eu.int:7002/cas/signature/get	absolute value	

If the value is a relative URL, the ecasBaseUrl will be used as prefix to the specified value.

Leave this property empty if the default value `/cas/signature/get` is suitable for your environment.

Note: the retrieveSignatureUrl is a “back-end” URL used by ECAS Client to connect to the ECAS Server and obtain the signature. It’s an application-to-application URL that is never accessed by the end-users so if your application can reach the internal ECAS Server URL, ***never*** configure it to go through a reverse proxy.

The only reason why you would configure a proxied URL for this would be if your application isn't deployed in the Commission's network and cannot reach the internal ECAS Server URL. In this case we recommend that you keep a relative URL for this property and that you configure instead a proxied URL for the *ecasBaseUrl* property, which applies to all relative URLs.

3.5.41. transactionUrl

Properties:	eu.cec.digit.ecas.client.filter.transactionUrl
XML:	transactionUrl

This optional property specifies the ECAS Server URL where the ECAS Client requests signatures for asynchronous transactions.

Example of valid values:

• /cas/transaction/sign	relative value	(default)
• https://ecas.cc.cec.eu.int:7002/cas/transaction/sign	absolute value	

If the value is a relative URL, the ecasBaseUrl will be used as prefix to the specified value.

Leave this property empty if the default value `/cas/transaction/sign` is suitable for your environment.

Note: the transactionUrl is a "back-end" URL used by ECAS Client to connect to the ECAS Server and request signatures for asynchronous transactions. It's an application-to-application URL that is never accessed by the end-users so if your application can reach the internal ECAS Server URL, ***never*** configure it to go through a reverse proxy.

The only reason why you would configure a proxied URL for this would be if your application isn't deployed in the Commission's network and cannot reach the internal ECAS Server URL. In this case we recommend that you keep a relative URL for this property and that you configure instead a proxied URL for the *ecasBaseUrl* property, which applies to all relative URLs.

3.5.42. certificateRevocationUrl

Properties:	eu.cec.digit.ecas.client.filter.certificateRevocationUrl
XML:	certificateRevocationUrl

This optional property specifies the ECAS Server URL where the ECAS Client can check whether an ECAS Server signature certificate is valid or not.

Example of valid values:

• /cas/signature/certValidate	relative value	(default)
• https://ecas.cc.cec.eu.int:7002/cas/signature/certValidate	absolute value	

If the value is a relative URL, the ecasBaseUrl will be used as prefix to the specified value.

Leave this property empty if the default value `/cas/signature/certValidate` is suitable for your environment.

Note: the certificateRevocationUrl is a "back-end" URL used by ECAS Client to connect to the ECAS Server and obtain check whether an ECAS Server signature certificate is valid or not. It's an application-to-application URL that is never accessed by the end-users so if your application can reach the internal ECAS Server URL, ***never*** configure it to go through a reverse proxy.

The only reason why you would configure a proxied URL for this would be if your application isn't deployed in the Commission's network and cannot reach the internal ECAS Server URL. In this case we recommend that you keep a relative URL for this property and that you configure instead a proxied URL for the *ecasBaseUrl* property, which applies to all relative URLs.

3.5.43. requestingUserDetails

Properties:	eu.cec.digit.ecas.client.filter.requestingUserDetails
XML:	requestingUserDetails

Use this optional property to always request all additional user details such as the domain, domain's username, first name, last name, email, department number, etc.

Valid values:

- true
- false (default)

Leave this property empty if you do not want to receive user attributes such as the first name, the last name, email, etc.

If you set this property to "`true`", the `UserPrincipal` returned by ECAS for authenticated users will implement the `eu.cec.digit.ecas.client.jaas.UserDetailsAccessible` mixin interface (see Figure 14 - UserDetailsAccessible interface).

If you let this property on the default value "`false`", the user details are not populated (i.e. they are always `null`).

You can use this property together with your custom `UserDetailsExtraGroupHandlerIntf` implementation (see above).

3.5.44. serviceResolver

Properties:	eu.cec.digit.ecas.client.filter.serviceResolver
XML:	serviceResolver

This optional property specifies the class which implements `eu.cec.digit.ecas.client.resolver.service.ServiceResolver` to be used.

The `ServiceResolver` interface lets you specify the strategy you want to use to construct the service passed to the ECAS Server. This replaces the default implementation which uses the serverName or serviceUrl properties to construct the service. For instance some implementations could retrieve custom X-Headers depending on the reverse proxy.

Leave this property empty if you are not using a custom ServiceResolver implementation.

The `ServiceResolver` interface skeleton:

```
package eu.cec.digit.ecas.client.resolver.service;

import eu.cec.digit.ecas.client.authentication.EcasServletAuthentication;
import eu.cec.digit.ecas.client.configuration.EcasConfigurationIntf;
import javax.servlet.http.HttpServletRequest;

public interface ServiceResolver {

    String getService(HttpServletRequest request,
                      EcasConfigurationIntf configuration,
                      EcasServletAuthentication ecasServletAuthentication);
}
```

Figure 15 - ServiceResolver interface

Three implementations are provided:

1. `eu.cec.digit.ecas.client.resolver.service.DefaultServiceResolver`: this is the default implementation. It constructs the service as follows:

   - If a serviceUrl is configured, it is used;
   - Otherwise it tries to use all the configured service-related properties such as serverName, serverPort, serverSSLPort, serverProtocol and serverContextPath.

   Note: for WebLogic Server, if the serverName is left `null`, the configured `ListenAddress` or `FrontEndHost` is used.

2. `eu.cec.digit.ecas.client.resolver.service.ReverseProxyAwareServiceResolver`: this implementation constructs the service upon what the user-agent sees as URL in the address bar rather than what is configured at the local server side. For this, an intermediate page is used which will get the `javascript:window.location.href` and send it back to the ECAS client.

3. `eu.cec.digit.ecas.client.resolver.service.RegExpServiceResolver`: obtains the service value using the same logic as the DefaultServiceResolver but filters the result using a given regular expression and a replacement string. This implementation is useful when a hostname or domain name needs to be rewritten, when some query string parameters need to be removed, when the session id in rewritten URLs needs to be removed or masked, etc.
   Both the regular expression and the replacement string have to be specified as custom parameters within the ECAS Client configuration.
   The RegExpServiceResolver requires two additional operation parameters defined in properties: the regexp and replacement values.

   Sample properties configuration:

```
eu.cec.digit.ecas.client.filter.serviceResolver=eu.cec.digit.ecas.client.resolver.service.RegExpServiceResolver
##########################################################################
# Custom configuration parameter name labelled "regExpServiceResolver.regexp":
eu.cec.digit.ecas.client.filter.param.name.regExpServiceResolver.regexp=regExpServiceResolver.regexp
# Custom configuration parameter value for "regExpServiceResolver.regexp":
eu.cec.digit.ecas.client.filter.param.value.regExpServiceResolver.regexp=[MY REGEXP]
# Custom configuration parameter name labelled "regExpServiceResolver.replacement":
eu.cec.digit.ecas.client.filter.param.name.regExpServiceResolver.replacement=regExpServiceResolver.replacement
# Custom configuration parameter value for "regExpServiceResolver.replacement":
eu.cec.digit.ecas.client.filter.param.value.regExpServiceResolver.replacement=[MY REPLACEMENT VALUE]
```

Sample XML configuration:

```
<serviceResolver>eu.cec.digit.ecas.client.resolver.service.RegExpServiceResolver</
serviceResolver>
 <params>
   <param>
       <!-- Custom configuration parameter name labelled
"regExpServiceResolver.regexp": -->
       <name>regExpServiceResolver.regexp</name>
       <value>[MY REGEXP</value>
   </param>
   <param>
       <!-- Custom configuration parameter name labelled
"regExpServiceResolver.replacement": -->
       <name>regExpServiceResolver.replacement</name>
       <value>[MY REPLACEMENT VALUE]</value>
   </param>
  </params>
```

Remarks:

- Implementations of this interface MUST also implement `java.io.Serializable`!
- If you specify an incorrect class name or if the class you specify cannot be found in your application classpath, an exception of type `eu.cec.digit.ecas.client.configuration.ConfigurationException` is thrown by the ECAS Client (the cause of which is going to be `java.lang.ClassNotFoundException`).
- If this implementation needs to rely upon the `eu.cec.digit.ecas.client.configuration.EcasConfigurationIntf`, make it also implement `eu.cec.digit.ecas.client.configuration.ConfigurationDependent`.

Refer to the javadoc for more details.

3.5.45. loginDateValidator

Properties:	eu.cec.digit.ecas.client.filter.loginDateValidator
XML:	loginDateValidator

This optional property specifies the class which implements `eu.cec.digit.ecas.client.validation.LoginDateValidatorIntf` to be used.

The `LoginDateValidatorIntf` interface lets you specify the strategy you want to use to reject login dates based upon specific application security policies. Typically if the timestamp of the authentication is too old, a renewal of authentication is to be performed, else SSO continues.

Leave this property empty if you are not using a custom LoginDateValidator implementation.

The `LoginDateValidatorIntf` interface skeleton:

```java
package eu.cec.digit.ecas.client.validation;

import eu.cec.digit.ecas.client.jaas.InvalidLoginDateException;
import java.util.Date;

public interface LoginDateValidatorIntf {

    boolean validate(Date loginDate) throws InvalidLoginDateException;
}
```

Figure 16 - LoginDateValidatorIntf interface

Use a custom implementation when you need to ensure that the authentication occurred within a given time frame. When the `validate()` method returns `false`, the client will trigger a renewal of the authentication, otherwise SSO will continue.

Remarks:

- Implementations of this interface MUST also implement `java.io.Serializable`!
- If you specify an incorrect class name or if the class you specify cannot be found in your application classpath, an exception of type `eu.cec.digit.ecas.client.configuration.ConfigurationException` is thrown by the ECAS Client (the cause of which is going to be `java.lang.ClassNotFoundException`).
- If this implementation needs to rely upon the `eu.cec.digit.ecas.client.configuration.EcasConfigurationIntf`, make it also implement `eu.cec.digit.ecas.client.configuration.ConfigurationDependent`.

Refer to the javadoc for more details.

3.5.46. reSubmitPosts

Properties:	eu.cec.digit.ecas.client.filter.reSubmitPosts
XML:	reSubmitPosts

Use this optional property to allow re-submitting a form that is posted to a protected resource before the user is authenticated.

This is useful when users submit a form and their Web session has expired, or when one application needs to post parameters to another application on a protected URL (the user is already authenticated in the first application but is not yet authenticated in the second one so SSO has still to occur). The posted parameters are stored in the user's Web session before redirecting to ECAS and then they are restored back in the request after successful ECAS authentication.

Valid values:

- true
- false (default)

Performance consideration: an application that uploads documents would have them stored in the user’s Web session while waiting for authentication. This could cause, for example, out-of-memory errors. This is the reason the feature is disabled by default.

3.5.47. httpRedirector

Properties:	eu.cec.digit.ecas.client.filter.httpRedirector
XML:	httpRedirector

This optional property specifies the class which implements `eu.cec.digit.ecas.client.http.HttpRedirector` to be used.

This `HttpRedirector` interface lets you specify the strategy that is used to redirect the user to the ECAS server login page. This happens when the user is not yet authenticated and when no valid ticket can be found in the request URL. Leave this property empty if you are not using a custom HttpRedirector implementation.

The `HttpRedirector` interface skeleton:

```
package eu.cec.digit.ecas.client.http;

import eu.cec.digit.ecas.client.configuration.EcasConfigurationIntf;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public interface HttpRedirector {

    void setConfiguration(EcasConfigurationIntf configuration);

    void sendRedirect(HttpServletRequest request,
                      HttpServletResponse response,
                      LoginParameters loginParameters)
        throws IOException, ServletException;
}
```

Figure 17 - HttpRedirector interface

Convenient implementations are provided:

1. `eu.cec.digit.ecas.client.http.DefaultHttpRedirector`:
   This is the default implementation, the one that is used when you leave this property empty. It uses HTTP redirections with a 302 (for HTTP/1.0) or 303 (for HTTP/1.1) status code as redirection mechanism.
   On top of that, it uses the new loginRequestId mechanism to prevent tampering of the login parameters and the client fingerprint mechanism to enhance the protocol security.
   This implementation does not work with the legacy Mock-up server.

2. `eu.cec.digit.ecas.client.http.BrowserPostRedirector`:
   This is a proof-of-concept implementation that renders an HTML page and uses JavaScript to automatically submit a form posting the login parameters to the Ecas server login URL.
   Since this implementation needs JavaScript to be enabled, it may cause accessibility issues to some users.
   This implementation does not work with the legacy Mock-up server.

3. `eu.cec.digit.ecas.client.http.LegacyHttpRedirector`:
   This is the implementation that was used in previous releases. It is fully backward-compatible with prior versions of the ECAS server.
   This is the value you have to specify when you are using the legacy Mock-up server.

4. `eu.cec.digit.ecas.client.http.JavascriptHttpRedirector:`
   This *HttpRedirector* performs the redirection using a client-side JavaScript redirect rather than a server-side HTTP 302/303.
   This HttpRedirector can also be used to block non-browser User-Agents incapable of executing JavaScript while genuine browsers capable of executing JavaScript are allowed.

Remarks:

- Implementations of this interface MUST also implement `java.io.Serializable`!
- If you specify an incorrect class name or if the class you specify cannot be found in your application classpath, an exception of type `eu.cec.digit.ecas.client.configuration.ConfigurationException` is thrown by the ECAS Client (the cause of which is going to be `java.lang.ClassNotFoundException`).
- If this implementation needs to rely upon the `eu.cec.digit.ecas.client.configuration.EcasConfigurationIntf`, make it also implement `eu.cec.digit.ecas.client.configuration.ConfigurationDependent`.
- The `eu.cec.digit.ecas.client.http.ajax.JsonHttpRedirector` and `eu.cec.digit.ecas.client.http.ajax.UnauthorizedHttpRedirector` AJAX-compliant HTTP Redirectors are now deprecated in favour of redirectionInterceptors (see 3.5.54).

Refer to the javadoc for more details.

3.5.48. trustNonEcasJEESubject

Properties:	eu.cec.digit.ecas.client.filter.trustNonEcasJEESubject
XML:	trustNonEcasJEESubject

Optional property to specify whether or not to trust users who are authenticated by the JEE container with another mechanism than ECAS.

If `true`, those users are not re-authenticated with ECAS but are granted immediate access into the application. If `false`, these JEE-already-authenticated users are re-authenticated with ECAS for requests which are filtered either by the `GatewayFilter` or the legacy `EcasFilter`.

This property has no effect when using security-constraints or a WebLogic Identity Assertion Provider.

Valid values:

- true
- false (default)

3.5.49. trustedCertificates

Properties:	eu.cec.digit.ecas.client.filter.trustedCertificates
XML:	trustedCertificates

This optional ECAS Client configuration property specifies the list of additional SSL certificate authorities the application is willing to trust for SSL connections. It constitutes an addition to the existing JVM trustStore (= the Java standard trust) and the embedded trustStore inside the ECAS client (which contains the certificate authorities of the production ECAS servers and the Commission's reverse proxies).

Valid values:

- Properties a comma-separated list of X.509 certificates as base64 strings
- XML a sequence of *trustedCertificate* tags each of which contains the base64 string of one X.509 certificate

Leave this property empty if you do not need to trust any additional ECAS server certificate.

Note that the format of each X.509 certificate is the base64 PEM encoding without the delimiters "`-----BEGIN CERTIFICATE-----`" and "`-----END CERTIFICATE-----`".

Example in properties format:

```
###########################################################################
## trustedCertificates
### The "trustedCertificates" property is used to specify the comma-separated list of
base64-encoded X.509 certificates of the
### trusted Certificate Authorities used for SSL by the ECAS server or mockup server.
### [Optional]
### [DefaultValue=none]
#eu.cec.digit.ecas.client.filter.trustedCertificates=MIIDLTCCAhWgAwIBAgIBATANBgkqhkiG9w0BAQU
FADAmMSQwIgYDVQQDExtFdXJv\
cGVhbiBDb21taXNzaW9uIFJvb3QgQ0EwHhcNMDMwMTIxMTgwMTM4WhcNMTIxMjMx\
MTgwMTM4WjAmMSQwIgYDVQQDExtFdXJvcGVhbiBDb21taXNzaW9uIFJvb3QgQ0Ew\
ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC2qIU7u75rPCUqzM0a0HT4\
eSMa+bFzSWcIxqJU1dPY1WGkqsee8rom3waf3scuIXHdk6CL43+s2zMrd0W8gyLL\
DBN46Z4BG9dIyhvHTlGTg7grVvHypbsvgC0lzb7xM/oFFs4AVUVqNgQPx1bELB3s\
t3NZRLUvFWNHXWDzR6CC/JTznn7NYBB0OScX7oMjYPQFL6n7vgKIVaU7YcZ+tJ6r\
a4oVt7zu3seiBzO0gijTcvlZ8PMIZUc21DnV2PtFgzaq5iem8mGdlVZXyL6MzbRx\
d4GIODPnWpCKABHd8dUMbbkOtkp1HMEQmaEdYr4zFFs53Snq4YZzFFhxRrfZCZfj\
AgMBAAGjZjBkMB8GA1UdIwQYMBaAFI+na6QQzxN0Z0ZdrF1wdDKuMkbbMB0GA1Ud\
DgQWBBSPp2ukEM8TdGdGXaxdcHQyrjJG2zAOBgNVHQ8BAf8EBAMCAQYwEgYDVR0T\
AQH/BAgwBgEB/wIBATANBgkqhkiG9w0BAQUFAAOCAQEAaPuj04oBLi6JExkkOzJX\
yYf+x0/dXXEt8oknr2qlfyaM2R6PXVcqE6HKtRcvxzuDSrgEHBb8N9k21YuF6ftM\
QeQTyRcJVJVuTW29Vw+vxj/NPqGSjpWTWA32jd2FqM8lcrw8JQ+cOtCMYxdjBb6j\
AJ9yiZ6AScEWGlN6hUS/KFZByKEnQTLiJ9BHooB651e1+TYs8BA3LuSYi3xKYniT\
sjde9nvePJAhTsxjs+oJklZiNR5yR6w385ah5Lgqyieb3+jAVfgExjc+h2hayOAl\
0/y2h8gQOlDzNRPUUftrUo9dMKJqAAyZyH18HH3kYbJ+9iy/cmHXY8OU5AdqTS/F\
hg\=\=\
,...\
,...\
,...
###########################################################################
```

Figure 18 - trustedCertificates property (properties format)

Example in XML format:

```
<trustedCertificates>
    <trustedCertificate>
            MIIDLTCCAhWgAwIBAgIBATANBgkqhkiG9w0BAQUFADAmMSQwIgYDVQQDExtFdXJv
            cGVhbiBDb21taXNzaW9uIFJvb3QgQ0EwHhcNMDMwMTIxMTgwMTM4WhcNMTIxMjMx
            MTgwMTM4WjAmMSQwIgYDVQQDExtFdXJvcGVhbiBDb21taXNzaW9uIFJvb3QgQ0Ew
            ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC2qIU7u75rPCUqzM0a0HT4
            eSMa+bFzSWcIxqJU1dPY1WGkqsee8rom3waf3scuIXHdk6CL43+s2zMrd0W8gyLL
            DBN46Z4BG9dIyhvHTlGTg7grVvHypbsvgC0lzb7xM/oFFs4AVUVqNgQPx1bELB3s
            t3NZRLUvFWNHXWDzR6CC/JTznn7NYBB0OScX7oMjYPQFL6n7vgKIVaU7YcZ+tJ6r
            a4oVt7zu3seiBzO0gijTcvlZ8PMIZUc21DnV2PtFgzaq5iem8mGdlVZXyL6MzbRx
            d4GIODPnWpCKABHd8dUMbbkOtkp1HMEQmaEdYr4zFFs53Snq4YZzFFhxRrfZCZfj
            AgMBAAGjZjBkMB8GA1UdIwQYMBaAFI+na6QQzxN0Z0ZdrF1wdDKuMkbbMB0GA1Ud
            DgQWBBSPp2ukEM8TdGdGXaxdcHQyrjJG2zAOBgNVHQ8BAf8EBAMCAQYwEgYDVR0T
            AQH/BAgwBgEB/wIBATANBgkqhkiG9w0BAQUFAAOCAQEAaPuj04oBLi6JExkkOzJX
            yYf+x0/dXXEt8oknr2qlfyaM2R6PXVcqE6HKtRcvxzuDSrgEHBb8N9k21YuF6ftM
            QeQTyRcJVJVuTW29Vw+vxj/NPqGSjpWTWA32jd2FqM8lcrw8JQ+cOtCMYxdjBb6j
            AJ9yiZ6AScEWGlN6hUS/KFZByKEnQTLiJ9BHooB651e1+TYs8BA3LuSYi3xKYniT
            sjde9nvePJAhTsxjs+oJklZiNR5yR6w385ah5Lgqyieb3+jAVfgExjc+h2hayOAl
            0/y2h8gQOlDzNRPUUftrUo9dMKJqAAyZyH18HH3kYbJ+9iy/cmHXY8OU5AdqTS/F
            hg==
    </trustedCertificate>
    <trustedCertificate>...</trustedCertificate>
    <trustedCertificate>...</trustedCertificate>
    <trustedCertificate>...</trustedCertificate>
</trustedCertificates>
```

Figure 19 - trustedCertificates property (XML format)

3.5.50. applicationSecurityLevel

Properties:	eu.cec.digit.ecas.client.filter.applicationSecurityLevel
XML:	applicationSecurityLevel

This optional ECAS Client configuration property specifies the sensitivity of the application. The sensitivity level determines how the user ID (uid) is transmitted to the ECAS client:

- *unversioned uid* – In the default behaviour, an ECAS user ID is a 7 or 8 character alphanumeric string (e.g. ndongreg). This attribute cannot ever change for a given user.

- *versioned uid* – In higher application security level settings, an ECAS user ID is a 47 or 48 character alphanumeric string composed of the unversioned uid, a numerical representation of the application security level and a version suffix in the UUID format.
  Example: **ndongreg-30-a61ea9b6-29a1-4830-9796-0d27dda13981**
  Here, ndongreg is the unversioned uid, 30 the numeric code for the application security level "HIGH" and a61ea9b6-29a1-4830-9796-0d27dda13981 the version. The default version is 0, encoded in the UUID format. The version changes, if the user performs an action which is considered to reduce the level of assurance into his identity. This is currently the case for certain changes to 2-factor authentication elements. For instance, if an external user deletes all his mobile phone numbers and then adds a new mobile phone number to his account, one cannot guarantee that the second factor (the mobile phone) is still owned by the same physical person. Thus, this action would lessen the level of assurance into this user's identity and create a new uid version for that user. For an application which identifies their users by uid only, this user would appear as a new user, since his uid has changed.

Valid values:

- NO_SECURITY (default)
- TOP
- HIGH
- MEDIUM
- LOW

Leave this property empty if the default value is suitable for your environment.

Example in properties format:

```
############################################################################
## applicationSecurityLevel
### The "applicationSecurityLevel" property is the level of security this application requires
### [Optional]
### [DefaultValue=NO_SECURITY]
#### Legal values: TOP,HIGH,MEDIUM,LOW,NO_SECURITY
#eu.cec.digit.ecas.client.filter.applicationSecurityLevel=TOP
#eu.cec.digit.ecas.client.filter.applicationSecurityLevel=HIGH
#eu.cec.digit.ecas.client.filter.applicationSecurityLevel=MEDIUM
#eu.cec.digit.ecas.client.filter.applicationSecurityLevel=LOW
############################################################################
```

Example in XML format:

```
    <!-- The "applicationSecurityLevel" property is the level of security this application requires.
         [Optional]
         [DefaultValue=NO_SECURITY]
         Legal values: TOP,HIGH,MEDIUM,LOW,NO_SECURITY -->
    <!--<applicationSecurityLevel>TOP</applicationSecurityLevel>-->
    <!--<applicationSecurityLevel>HIGH</applicationSecurityLevel>-->
    <!--<applicationSecurityLevel>MEDIUM</applicationSecurityLevel>-->
    <!--<applicationSecurityLevel>LOW</applicationSecurityLevel>-->
```

When NO_SECURITY is configured, the *applicationSecurityLevel* is not sent by the ECAS Client to the ECAS server (same behaviour as pre 1.23.0) so that the user authenticates with her unversioned uid. Therefore *eu.cec.digit.ecas.client.jaas.UserDetailAccessible#getUid()* returns the unversioned uid (i.e. same value as *eu.cec.digit.ecas.client.jaas.UserDetailAccessible#getUnversionedUid()*)

When TOP, HIGH, MEDIUM or LOW is configured the *applicationSecurityLevel* is sent by the ECAS Client to the ECAS server and the user authenticates with a versioned uid corresponding to the security level. Therefore the method *eu.cec.digit.ecas.client.jaas.UserDetailAccessible#getUid()* returns the versioned uid corresponding to the security level.

Extract of a log with *applicationSecurityLevel=HIGH*

```
    Subject
        Principals:
            -eu.cec.digit.ecas.client.j2ee.weblogic.EcasUser: eu.cec.digit.ecas.client.j2ee.weblogic.EcasUser@1dc9ea7{
        user="ndongreg-30-a61ea9b6-29a1-4830-9796-0d27dda13981"
        ticketType="SERVICE"
        -pgtId: "available"
        -pgtIou: "available"
        -proxies: "[]"
        -proxyGrantingProtocol: "PGT_URL"
        -uid: "ndongreg-30-a61ea9b6-29a1-4830-9796-0d27dda13981"
        -email: "Gregory.DONY@ext.ec.europa.eu"
        -employeeType: "n"
        -firstName: "gregory"
        -lastName: "DONY"
        -domain: "external"
        -domainUsername: "donydgr"
        -locale: "en"
        -assuranceLevel: "LOW"
        -registrationLevelVersions: "{HIGH=a61ea9b6-29a1-4830-9796-0d27dda13981}
```

When TOP, HIGH, MEDIUM or LOW is configured the Signature Client also sends the *applicationSecurityLevel* so that the signature contains the versioned uid.

Example of a signature with *applicationSecurityLevel=HIGH*:

```
<message>...
<ECAS:SignatureId>...</ECAS:SignatureId>
<ECAS:SignatureRequestInfo>
<ECAS:SignatureRequestId>...</ECAS:SignatureRequestId>
<ECAS:Service>…</ECAS:Service>
<ECAS:DisplayedDescription>Validation of the user
name</ECAS:DisplayedDescription>
<ECAS:Reason>For approval</ECAS:Reason>
<ECAS:AcceptedStrengths>
<ECAS:AcceptedStrength>…</ECAS:AcceptedStrength>
</ECAS:AcceptedStrengths>
<ECAS:ApplicationSecurityLevel>HIGH</ECAS:ApplicationSecurityLevel>
</ECAS:SignatureRequestInfo>
<ECAS:SignerInfo>
<ECAS:Username>ndongreg-30-61cf3141-6850-4a2a-b981-10f52db24a6a</ECAS:Username>
<ECAS:Uid>ndongreg</ECAS:Uid>
<ECAS:Domain>external</ECAS:Domain>
<ECAS:Moniker>donydgr</ECAS:Moniker>
<ECAS:FirstName>gregory</ECAS:FirstName>
<ECAS:LastName>DONY</ECAS:LastName>
<ECAS:Email>Gregory.DONY@ext.ec.europa.eu</ECAS:Email>
<ECAS:LoginStrength>STRONG</ECAS:LoginStrength>
<ECAS:LoginTimeStamp>2012-08-29T18:07:03.204+02:00</ECAS:LoginTimeStamp>
<ECAS:SignStrength>STRONG</ECAS:SignStrength>
<ECAS:SignTimeStamp>2012-08-29T18:07:33.703+02:00</ECAS:SignTimeStamp>
</ECAS:SignerInfo>
</ECAS:SignatureInfo></ds:Object></ds:Signature></message>
```

The concept of the ECAS XML signature is explained in detail in [SIGNATURE].

3.5.51. *negociatePrivateServiceTicket*

Properties:	eu.cec.digit.ecas.client.filter.negociatePrivateServiceTicket
XML:	negociatePrivateServiceTicket

This optional ECAS Client configuration property (de)activates the "Private Service Ticket" protocol.

In the Private Service Ticket protocol the service ticket (ST) is negotiated beforehand and does not need to be exchanged via the end-user's browser anymore.

It enhances security as the ticket is not sent in the service URL but is only sent through the back-channel between the application and the ECAS server over SSL/TLS.

This protocol can only be used when the HttpRedirector negotiates LoginRequestTransaction's (i.e. `eu.cec.digit.ecas.client.filter.httpRedirector=eu.cec.digit.ecas.client.http.DefaultHttpRedirector` or

`eu.cec.digit.ecas.client.filter.httpRedirector=eu.cec.digit.ecas.client.http.BrowserPostRedirector`)

See [ECAS-PRIVATE-ST] for a detailed description of the Private ST protocol.

Valid values:

- true
- false (default)

Leave this property empty if the default value is suitable for your environment.

Example in properties format:

```
###########################################################################
## negotiatePrivateServiceTicket
### The "negotiatePrivateServiceTicket" property controls whether or not the ECAS ticket must be
### sent in the service URL or can be pre-negotiated when LoginRequestTransactions are
### enabled (via a configured HttpRedirector equal to eu.cec.digit.ecas.client.http.DefaultHttpRedirector
### or eu.cec.digit.ecas.client.http.BrowserPostRedirector).
### If "true", the ticket is not sent in the service URL but is only sent through the back-channel
### between the application and the ECAS server over SSL/TLS.
### [Optional]
### [DefaultValue=false]
#### Legal values: false,true
#eu.cec.digit.ecas.client.filter.negotiatePrivateServiceTicket=false
#eu.cec.digit.ecas.client.filter.negotiatePrivateServiceTicket=true
###########################################################################
```

Example in XML format:

```
    <!-- The "negotiatePrivateServiceTicket" property controls whether or not the ECAS ticket must be
         sent in the service URL or can be pre-negotiated when LoginRequestTransactions are
         enabled (via a configured HttpRedirector equal to eu.cec.digit.ecas.client.http.DefaultHttpRedirector
         or eu.cec.digit.ecas.client.http.BrowserPostRedirector).

         If "true", the ticket is not sent in the service URL but is only sent through the back-channel
         between the application and the ECAS server over SSL/TLS.
         [Optional]
         [DefaultValue=false]
         Legal values: false, true-->
    <!--<negotiatePrivateServiceTicket>false</negotiatePrivateServiceTicket>-->
    <!--<negotiatePrivateServiceTicket>true</negotiatePrivateServiceTicket>-->
```

3.5.52. *advancedHttpSessionManagement*

The advanced HTTP Session Management system prevents the possibility of race conditions in the ECAS authentication negotiation when the user has multiple concurrent threads running for her on the server. Authentication could fail when starting a browser with multiple tabs opened on protected pages of the protected application.

A new ECAS configuration property is available to control the new advanced HttpSession management mechanisms.

Properties:	eu.cec.digit.ecas.client.filter.advancedHttpSessionManagement
XML:	advancedHttpSessionManagement

This optional ECAS Client configuration property controls whether HTTP state management through the establishment of the HttpSession is mandatory.

Valid values:

- true (default)
- false

You should let this value empty unless your end-users are only Web service clients unable to maintain cookie-based HttpSessions.

Example in properties format:

```
#############################################################################
## advancedHttpSessionManagement
### The "advancedHttpSessionManagement" property controls whether HTTP state management through the establishment of the HttpSession is mandatory.
### If "true", the ECAS Client will take appropriate care to enforce that a valid HttpSession is created and maintained.
### Otherwise, these actions are not undertaken.
### You should let this value to "true" unless your end-users are only Web service clients unable to maintain cookie-based HttpSessions.
### [Optional]
### [DefaultValue=true]
#### Legal values: true,false
#eu.cec.digit.ecas.client.filter.advancedHttpSessionManagement=true
#eu.cec.digit.ecas.client.filter.advancedHttpSessionManagement=false
#############################################################################
```

Example in XML format:

```
    <!-- The "advancedHttpSessionManagement" property controls whether HTTP state management through the establishment of the HttpSession is mandatory.
         If "true", the ECAS Client will take appropriate care to enforce that a valid HttpSession is created and maintained.
         Otherwise, these actions are not undertaken.
         You should let this value to "true" unless your end-users are only Web service clients unable to maintain cookie-based HttpSessions.
         [Optional]
         [DefaultValue=true]
         Legal values: true,false -->
    <!--<advancedHttpSessionManagement>true</advancedHttpSessionManagement>-->
    <!--<advancedHttpSessionManagement>false</advancedHttpSessionManagement>-->
```

3.5.53. *ticketResolver*

This optional property specifies the class which implements the `eu.cec.digit.ecas.client.resolver.ticket.TicketResolver` interface to be used.
The responsibility of this interface is to resolve ECAS tickets (such as the Service Ticket) i.e. to retrieve them from incoming requests by any appropriate means.

The default implmentation is `eu.cec.digit.ecas.client.resolver.ticket.DefaultTicketResolver`.

Properties:	eu.cec.digit.ecas.client.filter.ticketResolver
XML:	ticketResolver

Example in properties format:

```
#############################################################################
## ticketResolver:
### Set "ticketResolver" to replace the default implementation.
### [Optional]
### [DefaultValue=eu.cec.digit.ecas.client.resolver.ticket.DefaultTicketResolver]
### ProvidedImplementations:
#eu.cec.digit.ecas.client.filter.ticketResolver=eu.cec.digit.ecas.client.resolver.ticket.DefaultTicketResolver
#############################################################################
```

Example in XML format:

```
    <!-- Set "ticketResolver" to replace the default implementation.
         [Optional]
         [DefaultValue=
eu.cec.digit.ecas.client.resolver.ticket.DefaultTicketResolver]
         ProvidedImplementations: -->
    <!--<ticketResolver>
eu.cec.digit.ecas.client.resolver.ticket.DefaultTicketResolver
</ticketResolver>-->
```

This is a technical property and you should let this configuration property empty to keep the default value.

3.5.54. redirectionInterceptors

Properties:	eu.cec.digit.ecas.client.filter.redirectionInterceptors
XML:	redirectionInterceptors

This optional property specifies the class which implements the `eu.cec.digit.ecas.client.http.RedirectionInterceptor` interface to be used.

This `RedirectionInterceptor` interface lets you specify if a request should be redirected to the ECAS server.

In some cases it is not desirable to redirect the User-Agent to the ECAS login page at all, because some incoming requests will never be able to perform a successful ECAS authentication.

Let's take a few examples:

- If an AJAX framework is used, the AJAX requests must not be redirected to the ECAS login page for authentication since the AJAX script is never going to perform a successful authentication on the ECAS login page.
  Therefore intercepting these AJAX requests and preventing them to be redirected to the ECAS login page makes perfect sense.
- Search engines or robots trying to index a Web application will never perform a successful authentication on the ECAS login page because they do not know any user's password therefore it is useless to redirect them to ECAS.
  Intercepting such requests constitutes a gain of resources both for the application and for the ECAS service.
- The *Microsoft Office Existence Discovery Protocol* is performed behind the scene and without the interaction of the end-user. Thus it is not able to login and it can be intercepted before any redirection to ECAS occurs.

The ECAS client accepts a collection of *RedirectionInterceptors* which are chained and performed in the specified order.

For instance, it is perfectly valid to intercept both robot and AJAX requests by configuring both the *RobotInterceptor* and an *AjaxInterceptor* at the same time.

```
package eu.cec.digit.ecas.client.http;

import java.io.IOException;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public interface RedirectionInterceptor {
          HttpServletResponse enrichResponseOnRedirection(
          HttpServletRequest request, HttpServletResponse response)
               throws IOException, ServletException;

       boolean interceptMandatoryRedirection(
          HttpServletRequest request, HttpServletResponse response, FilterChain fc)
               throws IOException, ServletException;

       boolean interceptOptionalRedirection(
          HttpServletRequest request, HttpServletResponse response, FilterChain fc)
               throws IOException, ServletException;
}
```

It contains 2 *intercept* methods, one exists for the *gateway* mode (*interceptOptionalRedirection*) and the other one is for the usual, non-*gateway* mode (*interceptMandatoryRedirection*, see chapter 13 Anonymous access (Gateway) for details).
Additional information is available in the javadoc of the interface.

Provided implementations:

- *eu.cec.digit.ecas.client.http.robot.DefaultRobotInterceptor*
- *eu.cec.digit.ecas.client.http.ajax.UnauthorizedAjaxRedirectionInterceptor*
- *eu.cec.digit.ecas.client.http.ajax.JsonAjaxRedirectionInterceptor*

Example in properties format:

```
############################################################################
## redirectionInterceptors:
### Set "redirectionInterceptors" to replace the default implementation.
### [Optional]
### [DefaultValue=eu.cec.digit.ecas.client.http.robot.DefaultRobotInterceptor]
### ProvidedImplementations:
#eu.cec.digit.ecas.client.filter.redirectionInterceptors=eu.cec.digit.ecas.client.http.robot
.DefaultRobotInterceptor
#eu.cec.digit.ecas.client.filter.redirectionInterceptors=eu.cec.digit.ecas.client.http.robot
.DefaultRobotInterceptor,
eu.cec.digit.ecas.client.http.ajax.UnauthorizedAjaxRedirectionInterceptor
#eu.cec.digit.ecas.client.filter.redirectionInterceptors=eu.cec.digit.ecas.client.http.robot
.DefaultRobotInterceptor, eu.cec.digit.ecas.client.http.ajax.JsonAjaxRedirectionInterceptor
############################################################################
```

Example in XML format:

```
<!-- Set "redirectionInterceptors" to replace the default implementation.
 [Optional]
 [DefaultValue=eu.cec.digit.ecas.client.http.robot.DefaultRobotInterceptor]
ProvidedImplementations: -->
<!--<redirectionInterceptors>-->
<!--
<redirectionInterceptor>eu.cec.digit.ecas.client.http.robot.DefaultRobotInterceptor</redirectionInterceptor>-->
<!--
<redirectionInterceptor>eu.cec.digit.ecas.client.http.ajax.UnauthorizedAjaxRedirectionInterceptor</redirectionInterceptor>-->
<!--
<redirectionInterceptor>eu.cec.digit.ecas.client.http.ajax.JsonAjaxRedirectionInterceptor</redirectionInterceptor>-->
<!--</redirectionInterceptors>-->
```

3.5.55. *extendedUserDetailsTypeMapper*

Properties:	eu.cec.digit.ecas.client.filter.extendedUserDetailsTypeMapper
XML:	extendedUserDetailsTypeMapper

This optional property specifies the class which implements the `eu.cec.digit.ecas.client.validation.ExtendedUserDetailsTypeMapper` interface to be used.

This interface is taking care of the concept of “extended” attributes.

Some projects receive more attributes than the standard ones returned by ECAS.

These additional attributes are called “extended” attributes and are returned by ECAS in a generic way, not to bring business-specific concepts to every application using the ECAS client.

Because of this, the ECAS client exposes a new pluggable interface, *eu.cec.digit.ecas.client.validation.ExtendedUserDetailsTypeMapper*, that developers can implement to type the extended attributes into an Object of the type of their choice.

The configuration to plug such an implementation using the properties configuration is shown in the following snippet:

```
################################################################################
## extendedUserDetailsTypeMapper:
### Set "extendedUserDetailsTypeMapper" to plugin your implementation of the
eu.cec.digit.ecas.client.validation.ExtendedUserDetailsTypeMapper interface,
### which is used to type extendedUserDetails instead of using a Map of Strings.
### [Optional]
### [DefaultValue=none]
### Example:
eu.cec.digit.ecas.client.filter.extendedUserDetailsTypeMapper=eu.europa.ec.mydg.myapp.MyExtendedUserDetailsTypeMapper
################################################################################
```

The skeleton of the interface to implement is:

```
public interface ExtendedUserDetailsTypeMapper {

   /**
    * Maps a {@link eu.cec.digit.ecas.client.jaas.UserDetailsAccessible} instance into a higher
    * abstraction easier to work with at the level of the protected application.
    * <p/>
    * The returned type is a {@link eu.cec.digit.ecas.client.jaas.ExtendedUserDetails}
    * which must correctly implement {@link java.io.Serializable}.
    *
```

```
     * @param userDetailsAccessible the instance containing the user attributes to
                                    convert, never null.
     * @return another abstraction based on the given user attributes
     * which must correctly implement {@link java.io.Serializable}.
     * @throws TypeMappingException if an error occurs when converting the types.
     */
    ExtendedUserDetails map(UserDetailsAccessible userDetailsAccessible)
        throws TypeMappingException;
}
```

Extended attributes are present in the new mixin interface *eu.cec.digit.ecas.client.jaas.ExtendedUser* implemented by all ECAS User Principals and are converted by the configured *ExtendedUserDetailsTypeMapper* into *eu.cec.digit.ecas.client.jaas.ExtendedUserDetails*.

3.5.56. Custom parameters

The configuration lets you add your own custom parameters.

Those parameters can be retrieved from the `EcasConfigurationIntf` instance that is passed to custom implementations that also implement `eu.cec.digit.ecas.client.configuration.ConfigurationDependent`.

The `ConfigurationDependent` interface skeleton:

```
package eu.cec.digit.ecas.client.configuration;

public interface ConfigurationDependent {

    void setConfiguration(EcasConfigurationIntf configuration);

    EcasConfigurationIntf getConfiguration();
}
```

Figure 20 - ConfigurationDependent interface

`eu.cec.digit.ecas.client.configuration.ConfigurationDependent` is a mixin interface you can choose to implement when you are implementing one of the following interfaces:

- `ProxyChainTrustHandlerIntf`
- `ExtraGroupHandlerIntf`
- `ServiceResolver`
- `LoginDateValidatorIntf`

If you do so, the `EcasConfigurationIntf` instance will be injected into your implementation at configuration time. This means your implementation can take decision depending on the configuration and its custom parameters.

To retrieve the custom parameters, use `EcasConfigurationIntf#getParams()` which returns the `java.util.Map` of configured custom parameters.

Hereunder is an example of a `LoginDateValidatorIntf` implementation that also implements `ConfigurationDependent` and uses its own custom parameter (`loginDateExpirationInMillis`) to configure its behaviour.

```
public class DummyLoginDateValidator
       implements LoginDateValidatorIntf, ConfigurationDependent {

    private EcasConfigurationIntf configuration;

    public boolean validate(Date loginDate) {
       String millisAsString =
               (String)configuration.getParams().get("loginDateExpirationInMillis");
       long millis = Long.parseLong(millisAsString);
       return System.currentTimeMillis() - loginDate.getTime() <= millis;
    }

    public EcasConfigurationIntf getConfiguration() {
        return configuration;
    }

    public void setConfiguration(EcasConfigurationIntf config) {
        this.configuration = config;
    }
}
```

Figure 21 - Example of use for custom parameters

The custom parameters are specified using the following convention.

Example in properties format:

```
###########################################################################
### Custom configuration parameter name labelled "loginDateExpirationInMillis":
### Note that a custom parameter must have both a param.name and a param.value
eu.cec.digit.ecas.client.filter.param.name.loginDateExpirationInMillis=loginDateExpirationIn
Millis
###########################################################################
### Custom configuration parameter value for "loginDateExpirationInMillis":
### Note that a custom parameter must have both a param.name and a param.value
# one hour:
eu.cec.digit.ecas.client.filter.param.value.loginDateExpirationInMillis=3600000
###########################################################################
```

Example in XML format:

```
    <params>
    <!-- Note that a custom parameter must have both a name and a value -->
        <param>
    <!-- Custom configuration parameter name labelled "loginDateExpirationInMillis": -->
            <name>loginDateExpirationInMillis</name>

    <!-- Custom configuration parameter value for "loginDateExpirationInMillis": -->
    <!--        one hour -->
            <value>3600000</value>
        </param>
    </params>
```

With these examples, you are able retrieve the custom parameter value using `(String) configuration.getParams().get("loginDateExpirationInMillis")`.

3.6. Reloadable configuration (Java 6)

Since Client 1.16.1 for *WebLogic Server 10.3* or *Tomcat 7*, the configuration files are reloadable and any modification made to them triggers the reloading of the whole ECAS client configuration.

By default, the delay between two consecutive checks on the file system is one minute.

To change this behaviour, you may either use the following System property `eu.cec.digit.ecas.client.configFileReloading.checkIntervalMillis` or use a new property in the "*Provider Specific*" tab of the Ecas Identity Asserter in WebLogic Server Administration Console called "`ConfigFileReloadingCheckIntervalMillis`".

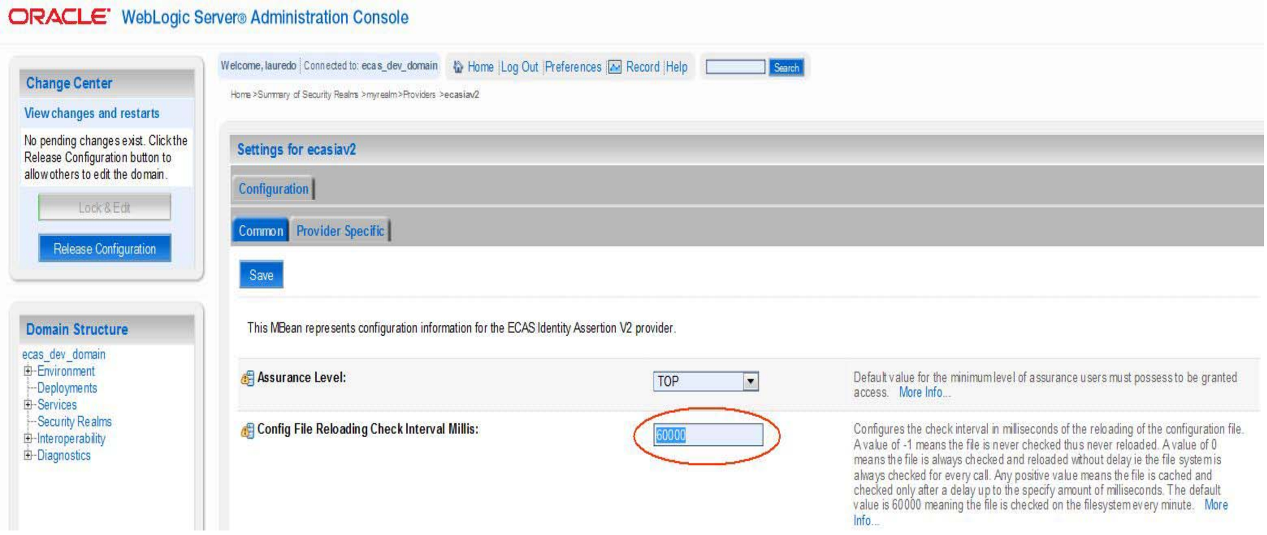
This property takes as value an amount of milliseconds.

- A value of `-1` means the file is never checked on the file system and thus never reloaded.
- A value of `0` means the file system is always checked and the files are reloaded without delay.
- Any positive value means the file is cached and checked only after a delay up to the specify amount of milliseconds. The default value is `60000` meaning the file system is checked for changes every minute.

To summarize:

- If you want to disable reloading whatsoever, for example in PRODUCTION, specify as JVM argument:
  `-Deu.cec.digit.ecas.client.configFileReloading.checkIntervalMillis=-1`
- If you want to reload the configuration as soon as you modify the files, for example in DEVELOPMENT, specify as JVM argument:
  `-Deu.cec.digit.ecas.client.configFileReloading.checkIntervalMillis=0`
- Other (positive) values are for environments managed by developers but which are under more load and want to avoid stressing the file system

Note that this feature is only available to Java-6-based clients i.e. WebLogic Server 10.3 and Tomcat 7.



No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

This MBean represents configuration information for the ECAS Identity Assertion V2 provider.

Assurance Level: TOP — Default value for the minimum level of assurance users must possess to be granted access. More Info...

Config File Reloading Check Interval Millis: 60000 — Configures the check interval in milliseconds of the reloading of the configuration file. A value of -1 means the file is never checked thus never reloaded. A value of 0 means the file is always checked and reloaded without delay ie the file system is always checked for every call. Any positive value means the file is cached and checked only after a delay up to the specify amount of milliseconds. The default value is 60000 meaning the file is checked on the filesystem every minute. More Info...

Figure 22 - Reloading Configuration

4. THE WEB.XML CONFIGURATION

4.1. web.xml template

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
                             http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
         version="2.4">

    <description>ecas-demo</description>
    <display-name>ecas-demo</display-name>

    <context-param>
        <description>
            When dispatching to the error page because a LoginException
            occurred, should we use a redirection (true) or a forward
            (false) ?
        </description>
        <param-name>eu.cec.digit.ecas.client.dispatch.useRedirect</param-name>
        <param-value>false</param-value>
    </context-param>
    <context-param>
        <description>
            Shoud all URLs specified for the error pages be considered
            relative to the Web app context root ?
        </description>
        <param-name>
            eu.cec.digit.ecas.formAuth.contextRelativeURLs
        </param-name>
        <param-value>true</param-value>
    </context-param>
    <context-param>
        <description>
            The default error page to be forwarded to when a LoginException
            occurs.
        </description>
        <param-name>
            eu.cec.digit.ecas.client.dispatch.forward.defaultError
        </param-name>
        <param-value>/errors/defaultError.jsp</param-value>
    </context-param>
    <context-param>
        <description>
            The error page to be forwarded to when an InvalidUserException
            occurs (e.g. the user is a Guest or an external user and the
            application only accepts internal Commission users).
        </description>
        <param-name>
            eu.cec.digit.ecas.client.dispatch.forward.invalidUser
        </param-name>
        <param-value>/errors/invalidUser.jsp</param-value>
    </context-param>
    <context-param>
        <description>
            The error page to be forwarded to when a
            FailedTicketValidationException occurs
            (e.g. problem validating the Service or Proxy Ticket).
        </description>
        <param-name>
            eu.cec.digit.ecas.client.dispatch.forward.failedTicketValidation
        </param-name>
        <param-value>/errors/failedTicketValidation.jsp</param-value>
    </context-param>
    <context-param>
        <description>
            The error page to be forwarded to when an UnexpectedLoginException
            occurs.
        </description>
        <param-name>
            eu.cec.digit.ecas.client.dispatch.forward.unexpectedLoginException
        </param-name>
        <param-value>/errors/unexpectedLoginException.jsp</param-value>
    </context-param>
    <context-param>
        <description>
            The error page to be forwarded to when a generic LoginException
            occurs.
        </description>
        <param-name>
            eu.cec.digit.ecas.client.dispatch.forward.loginException
        </param-name>
        <param-value>/errors/loginException.jsp</param-value>
    </context-param>

    <error-page>
        <error-code>403</error-code>
        <location>/errors/unauthorisedAccess.jsp</location>
    </error-page>

    <!-- example security constraint: -->
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>ecas-demo</web-resource-name>
            <description>
            Require users to authenticate in the 'protected/*' context
            </description>
            <url-pattern>/protected/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <description>allow users with role 'myRole'</description>
            <!-- The role name is mapped from CUD groups
            to the role defined in weblogic.xml
            -->
            <role-name>myRole</role-name>
        </auth-constraint>
        <user-data-constraint>
            <description>
            Encryption is not required for the application in general.
            </description>
            <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
    </security-constraint>

    <!-- role: myRole -->
    <security-role>
        <description>Role needed to access the protected area</description>
        <role-name>myRole</role-name>
    </security-role>

</web-app>
```

4.2. The security-constraint tag

Authorization is handled by WebLogic via the “`security-constraint`” tag.

The resources that need to be protected by a given **role** are specified within the “`url-pattern`” tags of the “`web-resource-collection`” tag.

The roles are specified by a “`role-name`” tag within the “`auth-constraint`” tag.

These roles are virtual roles that are mapped to “real” roles by WebLogic RoleMapper using the **weblogic.xml** deployment descriptor.

Here we chose to only authorize users having the role “**myRole**” to enter the “**/protected/***” path[3].

For more details about security-constraints, see APPENDIX III: Security-constraints

4.3. The security-role tag

This tag is used by WebLogic for the mapping of roles using **weblogic.xml**

```
    <security-role>
        <description>Role needed to access the protected area</description>
        <role-name>myRole</role-name>
    </security-role>
```

[3] The role must be compatible with the *assuranceLevel* property.

5. The weblogic.xml Configuration

```
<weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90"
                  xmlns:j2ee="http://java.sun.com/xml/ns/j2ee"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:schemaLocation="http://www.bea.com/ns/weblogic/90
                  http://www.bea.com/ns/weblogic/90/weblogic-web-app.xsd">

    <security-role-assignment>
        <role-name>myRole</role-name>
        <!-- insert your own role or CUD group here -->
        <principal-name>MY_GROUP</principal-name>
    </security-role-assignment>

    <session-descriptor>
        <!--
        The cookie-path parameter is required to prevent WebLogic SSO
        and to prevent cookie collisions when behind the same reverse proxy.
        If this parameter is not set, users won't be able to switch from one
        application to another without their session cookie being overwritten
        and their Web session lost.
        -->
        <cookie-path>/mypath</cookie-path>
    </session-descriptor>

    <context-root>/mypath</context-root>

</weblogic-web-app>
```

This deployment descriptor maps the name of the required role used inside the sample application ("**myRole**" role name) to the name of the groups you requested (i.e. one or more CUD groups). Here it is the fictive CUD group "**MY_GROUP**" the user must belong to in order to be authorized to enter the protected resources.

You can map a role (such as "**myRole**") to any principal i.e. one or more user names, one or more group names[4] (user names and group names being the ones returned by ECAS).

The `cookie-path` element in the `session-descriptor` element is needed to prevent session cookie collision when behind a reverse proxy as well as to avoid the SSO mechanism WebLogic uses by default. You need to set the `cookie-path` parameter value to the **context root** of your application ("**/mypath**" in this case). This tag is required whenever you want to use two different applications protected by ECAS on the same WebLogic server.

It is a good security practice to use it anyway because otherwise, in WebLogic server, the user session is shared across Web applications and a user authenticated in one application is therefore authenticated for any other application on the same server.

It is also a good deployment practice as otherwise any application on the same domain (e.g. accessed through the same reverse proxy) using the same cookie name would have its session cookies collide with yours and would prevent your Web session to be maintained by the user's browser.

Important note:

If you specify a `cookie-path` parameter with a wrong value (i.e. a misspelled or non-existing path), WebLogic won't be able to find your users' sessions anymore and your application will stop working (possibly by endlessly redirecting to the "**/protected/login**" URL).

6. The Error Page

As in any secure web application, you need to provide an error page to be displayed to unauthorized users. Such a page is given in the sample application as an example. You should of course write your own one adapted to your specific needs.

[4] Specifying more than one value will mean a logical OR (union) between the specified values.

```
<%@ page isErrorPage="true"
             import="java.io.PrintWriter"
%><html>
<head>
    <title>Authentication Error</title>
</head>

<body>
<div>
    An error occurred during the authentication process: <code><%= exception.getMessage()
%></code>
    <br />
    <pre>
        <% exception.printStackTrace(new PrintWriter(out)); %>
    </pre>
</div>
</body>
</html>
```

7. LOGOUT.JSP

The sample application comes with a `logout.jsp` example page:

```
<%@ page session="false" %>
<html>
        <head>
                <title>Log Out</title>
        </head>
        <body>
        <% if (null != request.getSession(false))
                        request.getSession(false).invalidate();
        %>
                <h3>
                        You are now <font color="red"><b>logged out</b></font> from
application <font color="green">”sampleA”</font>.
                </h3>
                <br/>
                <b>Note</b>: You will need to log out from ECAS if you want
                to be able to log in with another account using this browser.
        </body>
</html>
```

This page only performs a logout from the application but not from ECAS.

If the user does not log out of ECAS, she will automatically (single-sign on) be logged in when she enters again the application. She needs to log out of ECAS if she wants to enter the application with a different account using the same browser.

That is where the Single Logout protocol comes into play (see chapter 19. *Single Logout* for details).

Hereunder is a snippet showing how to perform the Single Logout protocol:

```
<%
    // Example snippet to perform an ECAS Single Logout
    // Note: Do not use this snippet as is in a JSP (ugly!)
    //       but put it into the appropriate component of your Web framework (e.g. a
controller, action or filter)

    // 1/ retrieve the client configuration and the validation configuration
    EcasConfigurationIntf ecasClientConfig = Client.getConfigFromContext(application);
    if (null != ecasClientConfig) {
        StatefulServiceResolver serviceResolver = (StatefulServiceResolver)
ecasClientConfig.getServiceResolver();
        // 2/ Find the service:
        String service;
        if (serviceResolver instanceof InteractiveServiceResolver) {
            InteractiveServiceResolver interactiveServiceResolver =
(InteractiveServiceResolver) serviceResolver;
            service = interactiveServiceResolver.getServiceForLogin(request, response);
            if (response.isCommitted()) return;
            interactiveServiceResolver.clearState(request, response);
        } else {
            service = serviceResolver.getService(request);
        }
        String ecasLogoutUrl = EcasUtil.replace(ecasClientConfig.getLoginUrl(), "login",
"logout", -1) + "?url="
                + RFC3986PercentCodec.UTF8_PERCENT_CODEC.encode(service);
%>
```

```
<a href="<%= ecasLogoutUrl%>"><em>Log out of ECAS</em></a>.
```

When users click on the `ecasLogoutUrl`, they will be taken to the ECAS Server that will ask them to confirm the "global" logout. If they accept, ECAS will asynchronously send a Single Logout request to all participating applications for which the users have a session. Then, they will be redirected to the origin application.

Note that because this Single Logout requests are asynchronous, the origin application MUST invalidate its own session before redirecting users to the ECAS logout page. There is indeed no guarantee that the Single Logout request will be processed before users come back to the origin application.

8. Configuring the Logging of the ECAS Identity Asserter V2

8.1. At a Glance

By default, the ECAS Identity Asserter V2 tries to locate Apache log4j. If log4j is found, it will log using it (see 8.3 – “Logging of the ECAS Client using Apache log4j”).

If log4j cannot be found, the ECAS Identity Asserter V2 logs using WebLogic logging services.

You may however override this behaviour by specifying the following system property: `eu.cec.digit.ecas.client.logging.log4jDisabled` in order to completely disable the use of log4j.

For instance, if you specify `-Deu.cec.digit.ecas.client.logging.log4jDisabled=true` even if log4j is available, the ECAS Identity Asserter V2 won’t use it and will use WebLogic logging services in every logging call.

8.2. ECAS Client logging with WebLogic logging services

Please refer to http://edocs.bea.com/wls/docs103/logging/config_logs.html for details about WebLogic logging services.

When the ECAS Identity Asserter V2 uses WebLogic logging services, it relies on WebLogic `weblogic.logging.NonCatalogLogger`, uses a logging subsystem called “ECAS” and by default, its debug statements are disabled. Only a specific system property can enable its debug logging.

If you don’t configure anything you only receive ECAS Info-severity in the domain and/or server logs.

If you want to log at different severity level (e.g. to obtain debug log statements) you need to configure two things:

1. Add the system property `eu.cec.digit.ecas.client.logging` with one of the legal value:

`all`	if you want to get all possible logs.
`debug`	if you want to log messages with a level equals or higher than `debug`
`info`	if you are interested in some important information from the ECAS Client.
`warn`	if you want to log warning or error messages.
`error`	if you are just interested in error messages.
`fatal`	if you want to see only fatal error messages.
`off`	if you do not want to log any messages at all, coming from the ECAS Client.

   For instance: `-Deu.cec.digit.ecas.client.logging=debug` to log a debug severity.

2. Configure WebLogic Server to display ECAS log messages where you want them to appear.
   For instance, if you want to see debug messages in the standard out or nohup file, you need to configure the standard out severity either using WebLogic Admin Console or using WLST or by specifying the system property `-Dweblogic.log.StdoutSeverity=Debug`.
   For WebLogic 10.3, you need also to specify the logger severity for the ECAS subsystem:
   `-Dweblogic.Log.LoggerSeverityProperties="ECAS=Debug"`

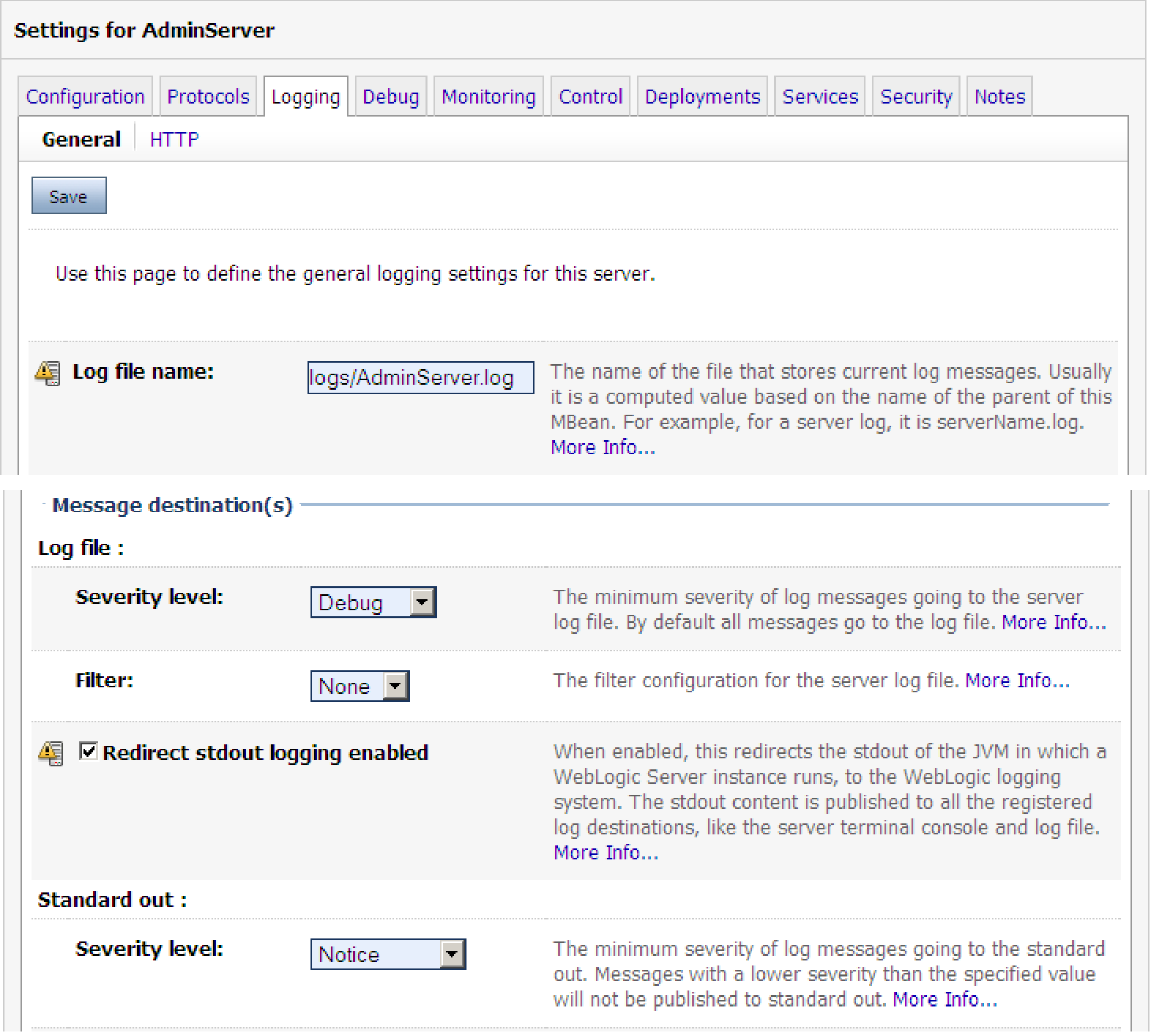
You may also edit your config.xml (at your own risk) and update the log element with

```
<logger-severity-properties>ECAS=Debug</logger-severity-properties>
```

Hereunder are screenshots of Weblogic Administration Console counter parts to achieve the same things.

By default, in WebLogic Server, the logs are sent to a file you can choose from the WebLogic Console.

In our example, the log file is called "*AdminServer.log*" under *logs*:



Settings for AdminServer

Use this page to define the general logging settings for this server.

Log file name: logs/AdminServer.log

The name of the file that stores current log messages. Usually it is a computed value based on the name of the parent of this MBean. For example, for a server log, it is serverName.log. More Info...

Message destination(s)

Log file :

Severity level: Debug

The minimum severity of log messages going to the server log file. By default all messages go to the log file. More Info...

Filter: None

The filter configuration for the server log file. More Info...

☑ Redirect stdout logging enabled

When enabled, this redirects the stdout of the JVM in which a WebLogic Server instance runs, to the WebLogic logging system. The stdout content is published to all the registered log destinations, like the server terminal console and log file. More Info...

Standard out :

Severity level: Notice

The minimum severity of log messages going to the standard out. Messages with a lower severity than the specified value will not be published to standard out. More Info...

Figure 23 - Configure WebLogic Logging

From the WebLogic Console, you can choose to log to the command prompt console ("stdout" checkbox) with the severity level you desire.

Depending on your WebLogic configuration, the log messages will appear in the specified log file, or in your server window. With managed server, it can be visible in Control > Remote Start Output.

The formatting of log messages looks like below:

```
<02-feb-05 16:34:58 CET> <Debug> <ECAS> <d02di0304149> <myserver> <ExecuteThread: '11' for
queue: 'default'> <kernel identity> <> <000000> <validate:
<cas:serviceResponse xmlns:cas="http://ecas.cc.cec.eu.int:7001/cas/schemas/1.2">
  <cas:authenticationSuccess>
    <cas:user>lauredo</cas:user>
    <cas:groups number="1">
      <cas:group>INTERNET</cas:group>
    </cas:groups>
    <cas:strength>STRONG</cas:strength>
    <cas:loginDate>2005-02-02T16:19:23.537+01:00</cas:loginDate>
  </cas:authenticationSuccess>
</cas:serviceResponse>
>
```

If you set the `severity level` on "`info`" for "`stdout`", the EcasIdentityAsserter will only log messages in the command prompt console such as:

```
<02-feb-05 16:34:58 CET> <Info> <ECAS> <d02di0304149> <myserver> <ExecuteThread: '11' for
queue: 'default'> <kernel identity> <> <000000> <EcasLoginModule: login successful for:
eu.cec.digit.ecas.client.j2ee.jaas.EcasSubject@3128a2 :
        -user: lauredo
        -loginDate: 2005-02-02T16:19:23.537+01:00
```

```
       -group (1): INTERNET
       -strength: STRONG
>
```

If you do not care about log messages in the command prompt console, leave the default `severity level` on “`error`” or above.

8.3. Logging of the ECAS Client using Apache log4j

By default, the ECAS Client tries to locate and use Apache log4j (http://logging.apache.org/log4j/) for logging – provided that you did not specify the `-Deu.cec.digit.ecas.client.logging.log4jDisabled=true` system property (see above).

If you need a centralized configuration for the logging of all the Web applications in the same application server, you can use only one *log4j.properties* (or *log4j.xml*) and one *log4j.jar* which you put in the server system classpath (e.g. by adding it in your WebLogic start script or setEnv script). But doing so, you will have to edit the central log4j configuration file each time one of the Web applications needs to change its logging. This is often not the desired configuration.

If you do not set the log4j JAR in the system classpath using a global log4j configuration for the whole server, you have two ways of configuring log4j.

8.3.1. Using a FilteringClassLoader

The first approach is to use a WebLogic FilteringClassLoader.

Please refer to http://edocs.bea.com/wls/docs100/programming/classloading.html#wp1097187

This method consists in using an EAR with a weblogic-application.xml descriptor containing

```
<prefer-application-packages>
  <package-name>org.apache.log4j.*</package-name>
</prefer-application-packages>
```

The FilteringClassLoader permits two things:

1. It prevents log4j from being loaded from a parent classLoader and ensures that it is the log4j jar from your application that is going to be loaded. This allows your application to have its own log4j loggers hierarchy.
2. It forces all resources such as log4j.properties or log4j.xml to be loaded first from the ContextClassLoader then from the parent classLoader. This allows to configure your application’s isolated log4j to load its own log4j configuration file.

This is the recommended way of configuring log4j in WebLogic 9+.

Hence, with the FilteringClassLoader, you can have one independent log4j configuration per Web application if you put a *log4j.properties* (or *log4j.xml*) in `WEB-INF/classes` and *log4j.jar* in `WEB-INF/lib` of each application.

8.3.2. Using a log4j RepositorySelector

If you do not want to use a FilteringClassLoader but want each application to be able to use its own log4j configuration, you may configure your WebLogic server to use the ECAS Client log4j RepositorySelector.

To activate it, you have to specify the following system property:

`-Deu.cec.digit.ecas.client.logging.log4jRepositorySelectorEnabled=true`

The ECAS log4j RepositorySelector allows each application to share the same log4j jar while having its own log4j configuration and logger hierarchy. It can be used with a central log4j jar in the system classpath or with a shared log4j library inside an application possessing multiple modules using different log4j configurations.

8.3.3. The log4j configuration file

Hereunder is an example log4j configuration file.

For more information on log4j, please consult http://logging.apache.org/log4j/docs/manual.html

The example *log4j.properties* file looks like below:

```
#Comment next line if you don’t want log4j debugging information
log4j.debug=true

####################
#                  #
#    appenders     #
#                  #
####################

#####################################
### direct log messages to stdout ###
#####################################
#This appender logs to the standard out
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d [%t] %5p %c{1}:%L - %m%n
#####################################

###########################
### merging file logger ###
###########################
log4j.appender.MERGE_FILE=org.apache.log4j.DailyRollingFileAppender
#Specify the path to your log file here:
log4j.appender.MERGE_FILE.File=D:\\bea1001\\user_projects\\domains\\base_domain\\logs\\ecas.log
log4j.appender.MERGE_FILE.datePattern='.'yyyy-MM-dd'.txt'
log4j.appender.MERGE_FILE.Append=true
log4j.appender.MERGE_FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.MERGE_FILE.layout.ConversionPattern=%d [%t] %-5p %c - (%F:%L) - %x - %m%n
###########################

#############################
### SOCKET_SERVER appender###
#############################
#This appender can be used to log to a Socket server, which is far better than logging to files
log4j.appender.SOCKET_SERVER=org.apache.log4j.net.SocketAppender
log4j.appender.SOCKET_SERVER.RemoteHost=127.0.0.1
log4j.appender.SOCKET_SERVER.Port=4445
log4j.appender.SOCKET_SERVER.LocationInfo=true
#############################

#############################
### LOG FACTOR 5 appender###
#############################
# This appender is used to scrutinize log messages in an AWT console
# launched by the same JVM as log4j
# USE ONLY FOR DEBUGGING IN DEVELOPMENT
# DO NOT USE IN PRODUCTION
log4j.appender.LF5=org.apache.log4j.lf5.LF5Appender
log4j.appender.LF5.MaxNumberOfRecords=2000

####################
#                  #
#   root logger    #
#                  #
####################

# Set root logger level to INFO and
# its only appender to the file (MERGE_FILE).
log4j.rootLogger=INFO, MERGE_FILE
# You can specify as many appenders as you want
# by separating them by commas: e.g.
#log4j.rootLogger=ERROR, APPENDER_1, APPENDER_2, APPENDER_3


##########################
#                        #
#   logger definitions   #
#                        #
##########################
```

```
# Specify the logging level (priority) you desire per logger:
log4j.logger.eu.cec.digit.ecas=DEBUG
log4j.logger.eu.cec.digit.ecas.util.dependencies=ERROR
log4j.logger.eu.cec.digit.ecas.util.commons.pool=WARN
log4j.logger.eu.cec.digit.ecas.util.httpclient=ERROR
log4j.logger.eu.cec.digit.ecas.client.filter.MetaFilter=ERROR
log4j.logger.org.apache=INFO
log4j.logger.httpclient.wire=ERROR

# Note that all loggers inherit their appenders from the rootLogger and # their parent
loggers.
# The '.' denotes a hierachical link between loggers exactly as
# Java packages do. For instance: eu is the parent logger of eu.cec and
# eu.cec.digit, which itself is the parent of eu.cec.digit.ecas
```

Figure 24 - log4j.properties Example

It is important that you set the loggers of the last section to avoid excessive logging. For instance, omitting to set httpclient.wire to a high threshold (ERROR) while the logging defaults to DEBUG will lead to a huge amount of logging, cluttering relevant messages. The proposed configuration is adequate for normal ECAS usage.

All the log4j loggers used by the ECAS Client inherit from eu.cec.digit.ecas. Hence you can disable or choose the granularity you want for the ECAS Client logging by specifying:

```
log4j.logger.eu.cec.digit.ecas=level
```

Where 'level' is one of:

`ALL`	if you want to get all possible logs.
`DEBUG`	if you want to log messages with a level equals or higher than '`DEBUG`'
`INFO`	if you are interested in some important information from the ECAS Client.
`WARN`	if you want to log warning or error messages.
`ERROR`	if you are just interested in error messages.
`FATAL`	if you want to see only fatal error messages.
`OFF`	if you do not want to log any messages at all coming from the ECAS Client.

You can also specify more granularity should you be interested only by the logging of some packages or classes.

If your application is already using `log4j` or *Apache Jakarta commons-logging* with `log4j`, you may need to add the ECAS Client loggers to your existing configuration file. This can be as simple as adding to your own *log4j.properties*, the line:

```
log4j.logger.eu.cec.digit.ecas=DEBUG
```

with the level (priority) you want.

If you do not want to merge the log messages from the ECAS Client with your own logs, simply specify a specific appender for the ECAS Client and set the ECAS Client loggers additivity to 'false':

```
log4j.logger.eu.cec.digit.ecas=DEBUG, MY_APPENDER_FOR_ECAS_CLIENT
log4j.additivity.eu.cec.digit.ecas=false
```

Note: *log4j.properties* is the plain old configuration format using a Java `ResourceBundle` *.properties* file. But if you prefer, you are free to use the XML format of it, *log4j.xml*. In the test application (/WEB-INF/classes of *ecas-demo.ear*), you will find an example of log4j XML configuration file.

More details about log4j can be found at http://logging.apache.org/log4j.

9. Subject and UserPrincipal

Once the user is authenticated, you can use the security of the container to retrieve the current `javax.security.auth.Subject` and the user's `java.security.Principal`:

```
Subject mySubject = weblogic.security.Security.getCurrentSubject();
```

Using the Servlet or EJB API, you can retrieve the current user's Principal.

In the Web parts of your application, you can use

```
javax.servlet.http.HttpServletRequest#getUserPrincipal()
```

If you use some EJBs, you can use

```
javax.ejb.EJBContext#getCallerPrincipal()
```

If you configured the client to retrieve user details (using requestingUserDetails), the `java.security.Principal` returned, which is of the type: `eu.cec.digit.ecas.client.j2ee.weblogic.EcasUser` which implements the `eu.cec.digit.ecas.client.jaas.UserDetailsAccessible` mixin interface, will be populated so that you can use it to retrieve all user's attributes. See also Figure 14 - UserDetailsAccessible interface.

10. CUD Groups

The CUD – Central User Database – is the authoritative user database that populates ECAS directories. Groups returned by the ECAS Server when your application is configured to request so (see 3.5.30 groups) come from the CUD.

Each application must possess at least one CUD group and must check that users trying to access it belong to that CUD group.

The CUD and CUD groups are managed by DIGIT USER ACCESS (DIGIT.A.1.UAA), the User Access Administration. It is them you have to contact to obtain new groups or to manage your existing group(s).

11. Custom Application Groups

If you need to use other groups in your application than the CUD groups, you have two options.

You may use an out-of-the-box BEA Authentication Provider which will play nicely with the ECAS Identity Asserter or implement the eu.cec.digit.ecas.client.validation.ExtraGroupHandlerIntf interface (see the extraGroupHandler configuration property).

How to choose between a BEA Authentication Provider and implementing the ExtraGroupHandler interface?

- If you are using an LDAP directory or a database, without logic coming from the application, it's probably easier to use a WebLogic Authentication Provider as they come out of the box and all you have to do is to configure them through WebLogic Server Administration Console. So you are done in a few clicks if the DataBase or the LDAP is up and running.
- If you want to re-use custom application logic (e.g. because you are using an ORM solution such as iBatis or Hibernate for your groups) that is defined only in your application part (and not accessible by WebLogic Server AuthenticationProvider classLoader), you probably have to implement the ExtraGroupHandler interface, which will be accessed by the Web application classLoader, allowing you to re-use any piece of logic defined in your application.

11.1. Configure a BEA WebLogic SSPI Authentication Provider

If you choose the Authentication Provider way, you need to follow the WebLogic Security Service Provider Interfaces as defined on BEA edocs. An AuthenticationProvider is configured using WebLogic Administration Console.

11.2. Configure a custom ECAS ExtraGroupHandler implementation

You should refer to 3.5.35 extraGroupHandler for details about the ExtraGroupHandler feature.

If you choose to implement the ECAS Client `ExtraGroupHandler` interface, you can configure it in the ECAS Client configuration file (e.g. in the external configuration file `ecas-config-`*`mywebapp`*`.properties` or `ecas-config-`*`mywebapp`*`.xml`).

11.2.1. *For ecas-config-*mywebapp*.properties*

```
##############################################################################
## extraGroupHandler:
### [Optional]
### [DefaultValue=none]
eu.cec.digit.ecas.client.filter.extraGroupHandler=eu.cec.mydg.myproject.MyCustomGroupHandler
Impl
##############################################################################
```

11.2.2. *For ecas-config-*mywebapp*.xml*

```
<extraGroupHandler>eu.cec.mydg.myproject.MyCustomGroupHandlerImpl</extraGroupHandler>
```

The `ExtraGroupHandler` interface skeleton can be found at Figure 10 - ExtraGroupHandlerIntf interface.

11.3. Configure a custom UserDetailsExtraGroupHandlerIntf implementation

The ECAS Client 1.8.7 introduced a new kind of ExtraGroupHandler, the *eu.cec.digit.ecas.client.validation.UserDetailsExtraGroupHandlerIntf* which receives all the user details as argument instead of only the user's uid.

When the ECAS Client is configured to return users' attributes, these attributes can be reused in your own logic to handle your application groups.

This *eu.cec.digit.ecas.client.validation.UserDetailsExtraGroupHandlerIntf* interface takes as argument of its *getGroups()* method a full *DetailedAuthenticationSuccess* object instead of only the user id.
DetailedAuthenticationSuccess contains all the elements present in service-ticket validation responses i.e. uid, strength, loginDate, groups, all user details, etc.
By using this interface, you can perform group-related queries using any user attribute.

An example of *UserDetailsExtraGroupHandlerIntf* implementation is available in the ecas-demo application. It creates extra groups based upon the value of the departmentNumber attribute. See 3.5.35 – "extraGroupHandler" for interface skeletons or the javadoc for details.

As you may have noticed, `UserDetailsExtraGroupHandlerIntf` extends `ExtraGroupHandlerIntf`. This is for backward compatibility with all configuration engines.
The drawback is that when you implement the new `UserDetailsExtraGroupHandlerIntf` interface, you also need to provide an implementation of the old `#getGroups(String)` method. Since the old method won't be used, you may just throw an `UnsupportedOperationException` in it.

12. MULTIFACTORAUTHORIZATIONFILTER

12.1. MultiFactorAuthorizationFilter

The *eu.cec.digit.ecas.client.filter.MultiFactorAuthorizationFilter* permits to protect one area of an application with stronger multi-factor strengths while the rest of the application may be configured to use merely a single-factor strength.

This is a typical use case for administration areas within Web applications.

This Filter reuses the existing ECAS client configuration but retains only multi-factor authentication strengths. Therefore your ECAS client configuration must contain at least one multi-factor authentication strength besides other single-factor authentication strengths which might be used for Web SSO.

For example:

```
eu.cec.digit.ecas.client.filter.acceptStrengths=PASSWORD,CLIENT_CERT,PASSWORD_SMS,PASSWORD_T
OKEN
```

The preferred strength in this example is "PASSWORD" but SSO is accepted with "PASSWORD_SMS" and "PASSWORD_TOKEN".

Since "PASSWORD_SMS" and "PASSWORD_TOKEN" are multi-factor strengths they will be used by this Filter, which would actually retain the equivalent configuration:

```
eu.cec.digit.ecas.client.filter.acceptStrengths=PASSWORD_SMS,PASSWORD_TOKEN
```

The single-factor strengths (PASSWORD, CLIENT_CERT) are simply ruled out.

Configuration example:

```
<filter-mapping>
   <filter-name>MultiAuthorization Filter</filter-name>
   <url-pattern>/admin.jsp</url-pattern>
</filter-mapping>

<filter>
   <filter-name>MultiAuthorization Filter</filter-name>
   <filter-class>eu.cec.digit.ecas.client.filter.MultiFactorAuthorizationFilter</filter-
class>
</filter>
```

Important notes:

- Only registered applications may be allowed to use multi-factor authentication strengths such as PASSWORD_SMS.
  To obtain multi-factor privileges for your application, you have to contact EC IAM SERVICE DESK.

- [Security] Be aware that re-authenticating the end-user using a new Authentication Strength triggers the invalidation of the current HttpSession. To avoid session-fixation attacks, the end-user is always authenticated in a new HttpSession.

12.2. TargetedUserMultiFactorAuthorizationFilter

A new Filter has been introduced to force only selected users to use multi-factor authentication: *eu.cec.digit.ecas.client.filter.TargetedUserMultiFactorAuthorizationFilter*.

Note that this filter causes a "re-authentication" of selected users using a multi-factor strength so these users may have to authenticate twice if they were not authenticated using a multi-factor strength in the first place.

Inheriting classes must implement the *isTargetedUser(String, HttpServletRequest)* method using their own business logic to decide whether the user must authenticate using a multi-factor strength.

Configuration example:

```
<filter-mapping>
   <filter-name>TargetedUserMultiFactorAuthorization Filter</filter-name>
   <url-pattern>/admin.jsp</url-pattern>
</filter-mapping>

<filter>
   <filter-name>TargetedUserMultiFactorAuthorization Filter</filter-name>
   <filter-class>mypackage.MyTargetedUserMultiFactorAuthorizationFilterImpl</filter-class>
</filter>
```

13. ANONYMOUS ACCESS (GATEWAY)

If your application needs to know beforehand whether the user is already authenticated in ECAS, you can use the *gateway* functionality.

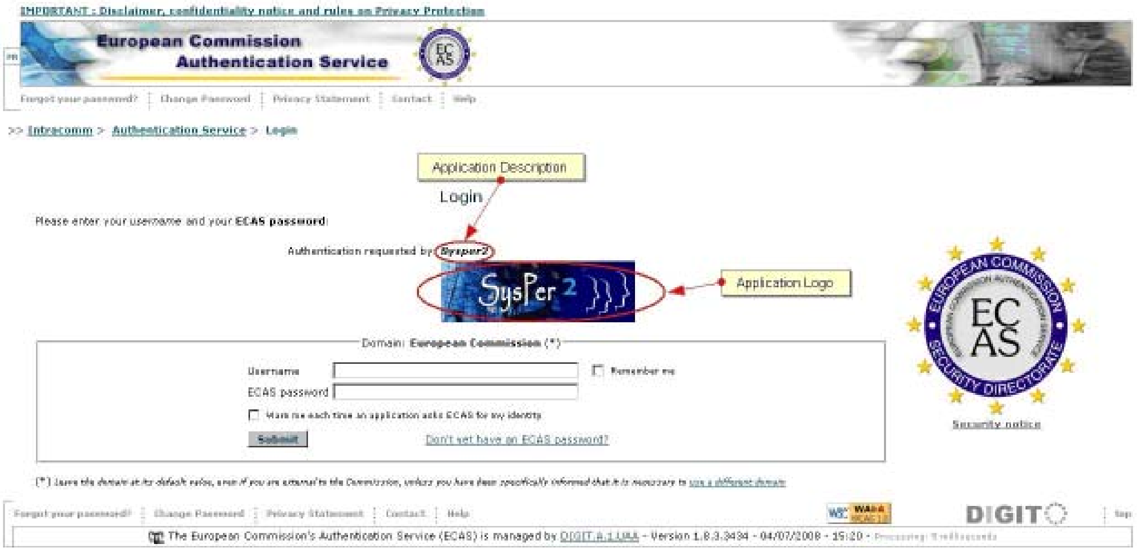
The Gateway functionality is also known as the “peek for Single Sign On”: the application redirects to ECAS to peek whether the user is already authenticated. If the user is, she returns to the application with a service ticket in the query string, if she isn’t, instead of blocking on the ECAS login screen, she is instantly redirected back to the application without a ticket and the application is made aware that she has no SSO session. In such a case, the application is browsed anonymously until the user decides to authenticate.

Refer to [GATEWAY] for more information.

14. APPLICATION DESCRIPTION AND LOGO

14.1. Where is it used?

The ECAS Server login page displays the calling application description and logo when the application is registered.



IMPORTANT : Disclaimer, confidentiality notice and rules on Privacy Protection

European Commission Authentication Service

Forgot your password? | Change Password | Privacy Statement | Contact | Help

>> Intracomm > Authentication Service > Login

Login

Please enter your username and your ECAS password:

Authentication requested by Sysper2

Domain: European Commission (*)

Username ▭ □ Remember me

ECAS password ▭

□ Warn me each time an application asks ECAS for my identity

Submit

Don't yet have an ECAS password?

Security notice

(*) Leave the domain at its default value, even if you are external to the Commission, unless you have been specifically informed that it is necessary to use a different domain

Forgot your password? | Change Password | Privacy Statement | Contact | Help | top

The European Commission's Authentication Service (ECAS) is managed by DIGIT.A.1.UAA - Version 1.8.3.3434 - 04/07/2008 - 15:20 - Processing: 0 milliseconds

Figure 25 - Application Description and Logo on the ECAS Login Page

14.2. How to register

To register your application, open an SMT ticket for the IAM SD group with the following information:

- **Description**: a short form that will be displayed on the ECAS page.
- **Application name**: won't be displayed, for informational purpose only.
- The different **URLs** it uses. You may have different URLs to provide either because your production environment is accessed by different URLs and/or because you want to register test, development and other environments on top of production.
- The **logo** you want to be displayed on the ECAS login page.

14.3. About the logo

There are no hard-and-fast rules regarding the logo.

- We accept usual formats (PNG, GIF, JPG). Avoid BMP and exotic formats.
- The largest dimension (height or width) should not exceed 250 pixels, but if you're slightly above, it will do.
- Mind the logo file name. It will appear unchanged to a user right-clicking it and selecting "properties" in their browser. If you have acquired your logo from a third party, the file name might be totally unrelated to your application, or even be unintentionally funny[5].
- Your logo will be displayed "as is", with no modification, so make sure it will look good.

[5] If the file name looks or sound funny, we will assume you meant it to be that way and we will upload it to ECAS unchanged.

15. Domain Propagation

The user’s domain can be retrieved in your application.

The `eu.cec.digit.ecas.client.jaas.UserDetailsAccessible` mixin interface implemented by the authenticated user's Principal contains a `getDomain()` method returning a String representation of the user's domain. Hence using:

```
((UserDetailsAccessible) request.getUserPrincipal()).getDomain();
```

would return the user's domain as a java.lang.String.

Note that this requires that you request user attributes from ECAS (See 3.5.43 – requestingUserDetails, p. 50). By default, user attributes are not requested from the ECAS server, so you need:

```
eu.cec.digit.ecas.client.filter.requestingUserDetails=true
```

or equivalent to be configured in your client file.

16. Language Propagation

For the user convenience, the language she selected can be propagated *to* and *from* ECAS.

Terminology: from a technical point of view, we will refer to the user's Locale, although from a practical point of view, we are talking about the user language, which is a subset of the locale.

16.1. Application to ECAS

If your application already knows the user language before triggering the authentication, it will be able to propagate the Locale to the ECAS server by setting an attribute in the HttpSession:

```
import eu.cec.digit.ecas.client.constants.SessionConstant;
import java.util.Locale;

// your code to retrieve the user's Locale:
Locale myUserLocale = retrieveTheUserLocale();
session.setAttribute(SessionConstant.LOCALE.toString(), myUserLocale);
```

If myUserLocale is not a valid java.util.Locale or does not match one of the supported languages in ECAS, it won't be taken into account.

If the ECAS client finds this property in the user's HttpSession before triggering the authentication, it will negotiate a LoginRequestId propagating this language to the ECAS user interface[6].

Note that this implementation requires that you use the provided ECAS client with the default HttpRedirector (see 3.5.47 – httpRedirector, p.53). Not configuring any HttpRedirector means using the default one.

16.2. ECAS to Application

The language the user selected in ECAS after authentication can be retrieved in your application, i.e. if you don't know the user's language beforehand and want to get this information from the ECAS server.

The `eu.cec.digit.ecas.client.jaas.UserDetailsAccessible` mixin interface implemented by the authenticated user's Principal contains a `getLocale()` method returning a String representation of the user's Locale as in use in the ECAS user interface while the authentication process occurred. Hence using:

```
((UserDetailsAccessible) request.getUserPrincipal()).getLocale();
```

would return the user's Locale as a java.lang.String.

Note that this requires that you request user attributes from ECAS (See 3.5.43 – requestingUserDetails, p. 50). By default, user attributes are not requested from the ECAS server, so you need:

```
eu.cec.digit.ecas.client.filter.requestingUserDetails=true
```

or equivalent to be configured in your client file.

[6] Your application must be registered in ECAS (or ECAST). Please contact ECAS support to perform the registration.

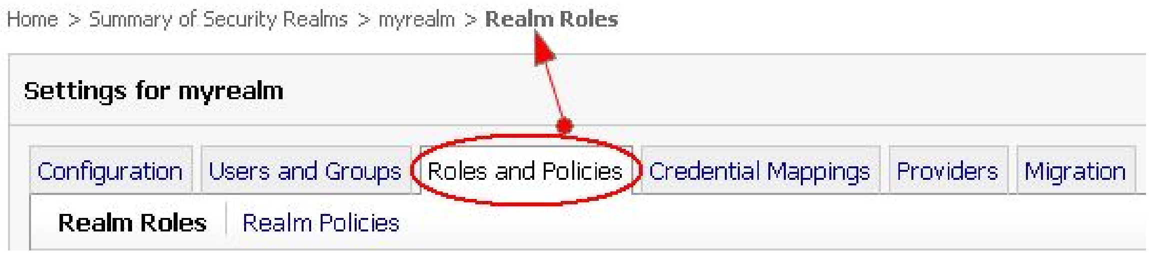
17. ACCESSING THE WEBLOGIC CONSOLE

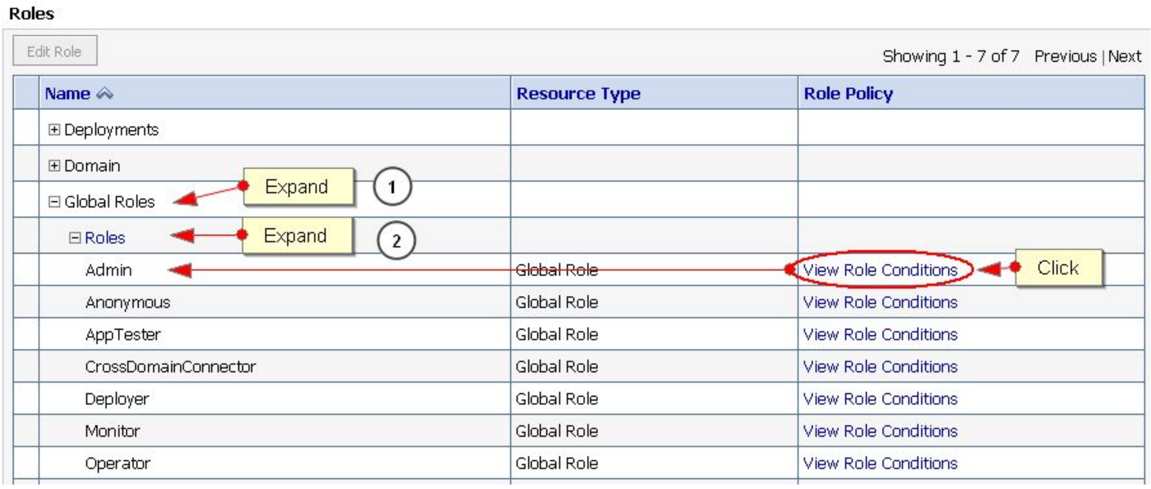
In [ECAS-BASIC], we stated that the WebLogic administration console would only be accessible via a direct URL (*/console/login/LoginForm.jsp*), unless you added your Commission username to the Administrators group e.g. by using the Admin Console prior to installing the ECAS Client.

In section see 2.1.1 – excludedContextPaths, we have also seen that you can configure the ECAS Identity Asserter MBean to exclude WebLogic Administration Console context-path from ECAS-protection and use WebLogic DefaultAuthenticator instead to protect the console.

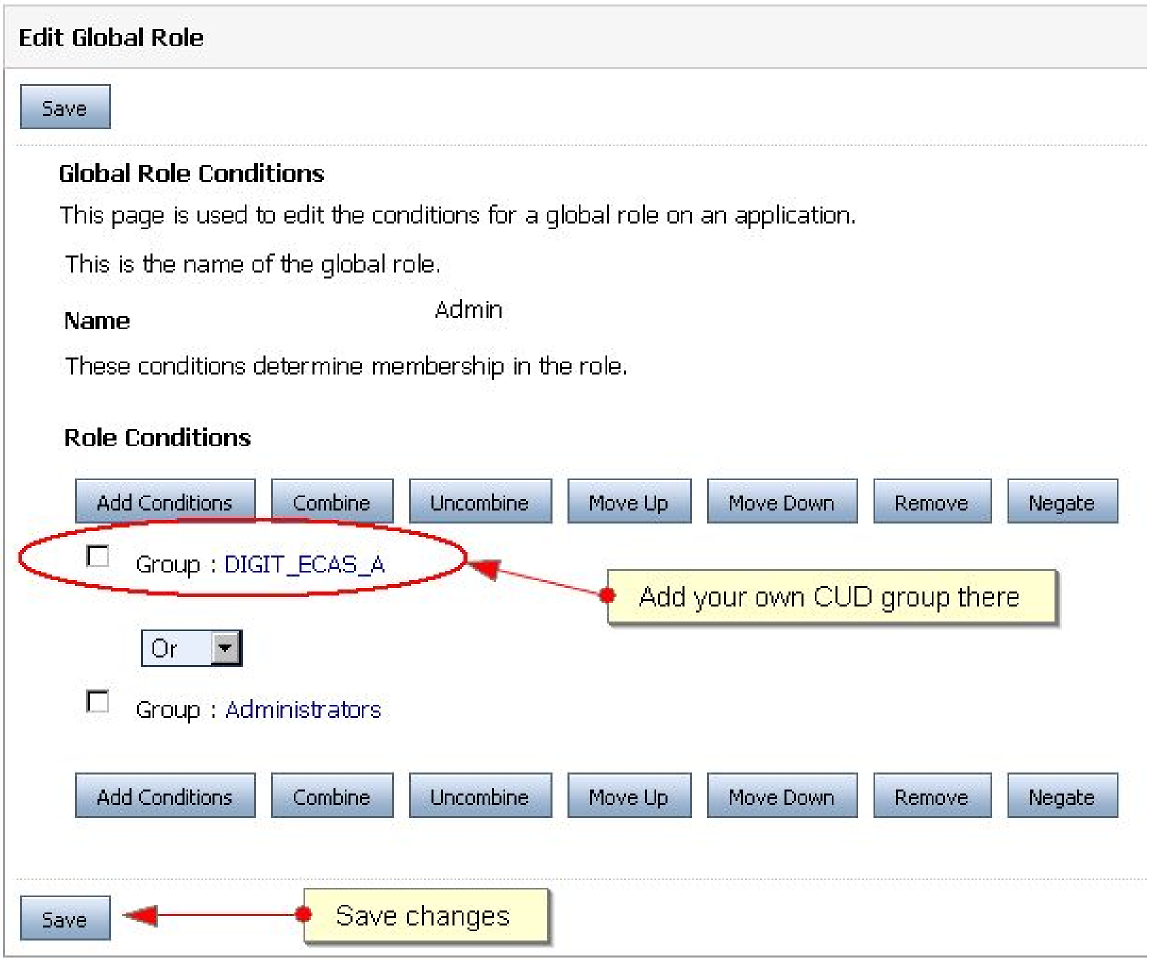
If you want to use ECAS to protect the console, there is still another way to do it if your team belongs to a given CUD group. You can ask ECAS to return any CUD group a user is member of, thus the only step to use that group instead of creating users in WebLogic, is to associate that group to the "**Admin**" role of WebLogic.

You can do that using the Admin console:





	Name	Resource Type	Role Policy
	⊞ Deployments		
	⊞ Domain		
	⊟ Global Roles		
	⊟ Roles		
	Admin	Global Role	View Role Conditions
	Anonymous	Global Role	View Role Conditions
	AppTester	Global Role	View Role Conditions
	CrossDomainConnector	Global Role	View Role Conditions
	Deployer	Global Role	View Role Conditions
	Monitor	Global Role	View Role Conditions
	Operator	Global Role	View Role Conditions



Edit Global Role

Global Role Conditions

This page is used to edit the conditions for a global role on an application.

This is the name of the global role.

Name Admin

These conditions determine membership in the role.

Role Conditions

□ Group : DIGIT_ECAS_A

Add your own CUD group there

Or

□ Group : Administrators

Save changes

This example adds the CUD group called "DIGIT_ECAS_A" to the Admin role of WebLogic. Of course, do use your own CUD group instead of DIGIT_ECAS_A.

The second part consists in telling the Admin Console to request that CUD group from ECAS at authentication time. This is done by adding an ECAS configuration file into %WL_INSTALL%/server/lib/consoleapp/webapp/WEB-INF/classes.

A minimal ecas-config.xml for the console looks like:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <cas:serverName>MY_SERVER_NAME</cas:serverName>
    <groups>
        <group>MY_CUD_GROUP</group>
    </groups>
</client-config>
```

In the previous example, it would be:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <groups>
        <group>DIGIT_ECAS_A</group>
    </groups>
</client-config>
```

Don't forget to activate your changes and to restart WebLogic Server.

All members of the configured CUD group can now access WebLogic Admin Console after being authenticated using ECAS.

18. SCRIPTCLIENT HELPER CLASS

This section applies only for Java 6 clients (i.e. WebLogic 10.3 and Tomcat 7).

The class "ScriptClient" helps with 2-way SSL for script/batch/job applications. This "ScriptClient" reuses the ECAS client facilities to trust the embedded trusted SSL certificates and may reuse the SSL certificate parsing logic for SSL hostname verifications.

Related issue: https://webgate.ec.europa.eu/CITnet/jira/browse/ECAS-911

```
// all the arguments:
boolean debugSslToSdtOut = true;
String ecasServerCertLoginUrl =
    "https://ecas.cc.cec.eu.int:7003/cas/ws/CertLoginService/http/post";
String targetServiceUrl = "http://your.service.org";
String keyStoreFileName = "some-scriptuser.p12";
String keyStoreType = "PKCS12";
String keyStoreProviderName = "SunJSSE";
String keyAlias = "YourAlias";
char[] keyPassPhrase = new char[]{'x', 'y', 'z'};
int connectTimeoutMillis = 15000;
int maxConnections = 2;
boolean verifyHostname = true;
int readTimeoutMillis = 15000;

// JSSE debugging:
if (debugSslToSdtOut) {
    // -Djavax.net.debug=all
    System.setProperty("javax.net.debug", "all");
}

// load the keyStore:
InputStream keyStoreInputStream = getClass().getResourceAsStream(keyStoreFileName);
KeyStore keyStore = KeyStore.getInstance(keyStoreType, keyStoreProviderName);
keyStore.load(keyStoreInputStream, keyPassPhrase);
keyStoreInputStream.close();

// build a script client:
ScriptClient scriptClient = new ScriptClient.Builder(keyStore, keyAlias, keyPassPhrase).
    connectTimeoutMillis(connectTimeoutMillis).
    ecasServerCertLoginUrl(ecasServerCertLoginUrl).
    maxConnections(maxConnections).
    verifyHostname(verifyHostname).
    readTimeoutMillis(readTimeoutMillis).build();

// retrieve a service ticket for the given service:
String serviceTicket = scriptClient.getServiceTicket(targetServiceUrl);
```

Figure 26 - Example using a script user and its related SSL certificate stored in a PKCS12 KeyStore

19. SINGLE LOGOUT

The CAS 3.1 Single Logout is implemented in the ECAS Client.

- What is Single Logout?

Single Logout means that end users can log out from *any* application participating in the Single Logout process and that they will be immediately logged out of *all* the applications participating in the Single Logout process. Without that, the user needs to log out of each application individually[7].

The CAS protocol is designed to support Single Logout. However, not all CAS-protected applications may take part in the Single Logout process.

- Where does Single Logout work?

For an application to successfully participate in Single Logout, first it must embed a CAS Client that supports the `logout` protocol.

This is the case for the ECAS Clients for Java.

Moreover, the CAS-protected application must rely on server-side managed sessions so that, when notified, it is able to terminate the user's session (e.g. delete it from a database, invalidate it, etc.), effectively logging them out.

- Where does Single Logout not work?

Single Logout is not available to clients / development languages that only rely on client-side sessions (e.g. cookies). CAS clients don't have access to the browser's cookies and therefore can't terminate a session. (However, closing a browser window should do that.)

Also, applications using CAS clients that do not support the `logout` protocol may notice extra requests in their access logs that appear not to do anything.

- How does Single Logout work in ECAS?

After successfully authenticating into the ECAS Server, the user is redirected back to the protected application with a Service Ticket (also known as "token").

The ECAS Client retrieves this ticket, validates it against the ECAS Server and, if successful, obtains the related user details. Then, the ECAS Client establishes a Security Context for the authenticated user in the Servlet Container and creates a new HTTP session for it.

If the protected application participates in the Single Logout process, the ECAS Client notifies the ECAS Server during ticket validation.

Later on, when the user logs out from any application participating in the Single Logout process (or logs out of the ECAS Server directly), the ECAS Server sends `logout requests` to all the applications participating in the Single Logout process, using back-end connections.

The back-end call made by the ECAS Server carries the same ticket used by the protected application to authenticate the user and create the HTTP session.

[7] An example of this behaviour can be seen in the latest ecas-demo Web application.

For example:

```
<LogoutRequest ID="_9ad62727-1834-41eb-8fa1-08515a68ba3c"
               IssueInstant="2011-06-22T18:05:31.160+02:00"
               Destination="http://myserver:7001/ecas-demo/protected/index.jsp"
               Version="2.0"
               xmlns="urn:oasis:names:tc:SAML:2.0:protocol"
               xmlns:saml2a="urn:oasis:names:tc:SAML:2.0:assertion">
    <saml2a:Issuer>ecas.cc.cec.eu.int:7002</saml2a:Issuer>
    <saml2a:NameID/>
    <SessionIndex>ST-12345-vHhXfKR9G88E1iJs</SessionIndex>
</LogoutRequest>
```

The protected application that receives the `logout request` must be able to retrieve the HTTP session corresponding to the given ticket in order to invalidate it, effectively logging the user out.

Therefore, when the ticket is validated and the corresponding HTTP session created, the protected application needs to record the fact that the ticket and the session correspond to each other.

The ECAS Client handles the whole Single Logout process for applications that request it. However, if your application has specific requirements that aren't supported by the existing mechanism, custom logic may be plugged in (see below).

- How do I activate Single Log Out for my application?

In the ECAS Client, the entry point for Single Logout support is an `HttpSessionListener`. Therefore, for your application to participate in the Single Logout process, just add the following snippet in your `web.xml` deployment descriptor:

```
<listener>
    <listener-class>
        eu.cec.digit.ecas.client.session.SingleSignOutHttpSessionListener
    </listener-class>
</listener>
```

Without this listener, `logout requests` sent by the ECAS Server will be ignored.

- What about clusters?

The `SingleSignOutHttpSessionListener` is cluster-compliant only when session replication is enabled. If the application only has session stickiness, you will also need to define the singleLogoutCallbackUrl property. Please refer to the property definition for detailed explanations.

- How do I customize Single Logout for my application?

Implementing the `eu.cec.digit.ecas.client.session.SingleLogOutSessionMapping` interface allows you to customize the ticket-session mapping logic, for example to use distributed caching like Coherence or HazelCast.

The `SingleLogOutSessionMapping` interface skeleton is:

```
package eu.cec.digit.ecas.client.session;

import import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

public interface SingleLogOutSessionMapping {

    void mapTokenAndSession(String token, HttpSession session);

    HttpSession removeMappingByToken(String token,
                                     HttpServletRequest backChannelLogOutRequest);

    boolean removeMappingBySessionId(String sessionId);

    void changeSessionId(String oldSessionId, HttpSession session);

}
```

When authentication occurs, the `mapTokenAndSession(String, HttpSession)` is invoked: the `token` argument is the ticket validated and the `session` argument is the corresponding authenticated session. The implementation must record the mapping between the ticket and the session, ideally in both directions.

When a session expires, its destruction results in a call to `removeMappingBySessionId(String)` where the mapping must be cleaned up, not only for consistency but also to avoid memory leaks.

Similarly, when Servlet 3.1-compliant Containers change a given session ID, the `changeSessionId(String, HttpSession)` implementation must align the mapping, based on the previous session ID.

Finally, when a `logout request` is handled, the `removeMappingByToken(String, HttpServletRequest)` is invoked. The implementation must delete the corresponding mapping and return the session, so that the caller can take care of invalidating it.

To configure your custom implementation of the `SingleLogOutSessionMapping` interface, add the following servlet context initialization parameter to your `web.xml`:

```
<context-param>
  <param-name>eu.cec.digit.ecas.client.filter.singleLogOutSessionMappingClass</param-name>
  <param-value>eu.europa.ec.mydg.myproject.MySingleLogOutSessionMapping</param-value>
</context-param>
```

where `eu.europa.ec.mydg.myproject.MySingleLogOutSessionMapping` is your implementation[8].

Customizing the session mapping logic probably meets the needs of most applications. However, if your platform uses specific session replication mechanisms, the default handling of the Single Logout process may not be suitable for you. In that case, you may want to implement your own flavour of the `eu.cec.digit.ecas.client.session.SingleLogOutHandler` interface. For more information, please refer to the Javadoc.

20. PROGRAMMATIC COOKIE-PATH

The *eu.cec.digit.ecas.client.j2ee.weblogic.ContextPathAsCookiePathListener* is a *ServletContextListener* for WebLogic that permits setting programmatically the cookie-path to be the same as the Web application context-path.

This is useful when deploying the same Web application archive on distinct context-paths without a need for repackaging.

To activate this feature, add the following snippet to your `web.xml`:

```
<listener>
    <listener-class>
        eu.cec.digit.ecas.client.j2ee.weblogic.ContextPathAsCookiePathListener
    </listener-class>
</listener>
```

[8] If you specify an incorrect class name or if the class you specify cannot be found in your application classpath, unsurprisingly a `java.lang.ClassNotFoundException` will occur.

21. Reference

CITnet:

- https://webgate.ec.europa.eu/CITnet/confluence/display/CITNET/Home

HTTP/1.1:

- http://www.w3.org/Protocols/rfc2616/rfc2616.html

web.xml:

- WebLogic 10.0: http://e-docs.bea.com/wls/docs100/webapp/web_xml.html
- WebLogic 10.3: http://download.oracle.com/docs/cd/E12840_01/wls/docs103/webapp/web_xml.html
- WebLogic 12.1: http://docs.oracle.com/cd/E24329_01/web.1211/e21049/web_xml.htm#WBAPP502

weblogic.xml:

- WebLogic 10.0: http://e-docs.bea.com/wls/docs100/webapp/weblogic_xml.html
- WebLogic 10.3: http://download.oracle.com/docs/cd/E12840_01/wls/docs103/webapp/weblogic_xml.html
- WebLogic 12.1: http://docs.oracle.com/cd/E24329_01/web.1211/e21049/weblogic_xml.htm#WBAPP571

Log4j:

- http://logging.apache.org/log4j/docs/

ECAS:

IAM project home page on CITnet: https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/

ECAS implementation releases: https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/Downloads-WebLogic

ECAS Forum: https://webgate.ec.europa.eu/CITnet/forums/viewforum.php?f=35

ECAS WIKI: https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/ECAS

ECAS JIRA: https://webgate.ec.europa.eu/CITnet/jira/browse/ECAS

ECAS SVN: https://webgate.ec.europa.eu/CITnet/svn/ecas-public/

ECAS FishEye: https://webgate.ec.europa.eu/CITnet/fisheye/browse/ECAS

22. APPENDIX I: HOW TO IMPORT SSL CERTIFICATES

22.1. SSL Configuration

The WebLogic server must be configured to be able to establish SSL connections with ECAS.

By default, the Commission certificates are included within the client JAR file. Therefore they do not need to be imported.

Nevertheless, if you are using a mockup, you will need to import the certificate of your mockup using the same kind of commands. As an example, this section describes how to import the Commission Certificate Authority certificates into the Java trustStore of the JVM and the one of WebLogic Server.

22.1.1. *Download the certificate*[9]

The certificates can be found under the names "**European Commission Root CA**" (Europeancommission.cer**)** and "**CommisSign Class A**" (CommisSign.cer**)** on the ECAS Forge:
https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/Downloads-Certificates

22.1.2. *Client configuration*

Import the certificates in the JVM trustStore:

```
%JAVA_HOME%\bin\keytool -import -v -alias "European Commission Root CA" -file
EuropeanCommission.cer -keystore %BEA_HOME%\jdk150_06\jre\lib\security\cacerts
Enter keystore password:  changeit
Trust this certificate? [no]:  yes

%JAVA_HOME%\bin\keytool -import -v -alias commissionCertificate -file CommisSign.cer -
keystore %BEA_HOME%\jdk150_06\jre\lib\security\cacerts
Enter keystore password:  changeit
Trust this certificate? [no]:  yes
```

Import the certificates in the WebLogic trustStore:

```
%JAVA_HOME%\bin\keytool -import -v -alias "European Commission Root CA" -file
EuropeanCommission.cer -keystore %BEA_HOME%\weblogic92\server\lib\cacerts
Enter keystore password:  changeit
Trust this certificate? [no]:  yes

%JAVA_HOME%\bin\keytool -import -v -alias commissionCertificate -file CommisSign.cer -
keystore %BEA_HOME%\weblogic92\server\lib\cacerts
Enter keystore password:  changeit
Trust this certificate? [no]:  yes
```

See also:
https://webgate.ec.europa.eu/CITnet/confluence/display/IAM/How+To+Import+SSL+Certificates

[9] If you don't want to or can't download these certificates, they are provided in the next Annex.

23. APPENDIX II: CommisSign PKI Certificates

In this section, we copied the two CommisSign PKI certificates in PEM format.

If you copy each one in a text file with a ".cer" or ".crt" extension under Windows, you can see their content with Windows Certificate Viewer by double-clicking them.

European Commission Root CA:

```
-----BEGIN CERTIFICATE-----
MIIDLTCCAhWgAwIBAgIBATANBgkqhkiG9w0BAQUFADAmMSQwIgYDVQQDExtFdXJv
cGVhbiBDb21taXNzaW9uIFJvb3QgQ0EwHhcNMDMwMTIxMTgwMTM4WhcNMTIxMjMx
MTgwMTM4WjAmMSQwIgYDVQQDExtFdXJvcGVhbiBDb21taXNzaW9uIFJvb3QgQ0Ew
ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC2qIU7u75rPCUqzM0a0HT4
eSMa+bFzSWcIxqJU1dPY1WGkqsee8rom3waf3scuIXHdk6CL43+s2zMrd0W8gyLL
DBN46Z4BG9dIyhvHTlGTg7grVvHypbsvgC0lzb7xM/oFFs4AVUVqNgQPx1bELB3s
t3NZRLUvFWNHXWDzR6CC/JTznn7NYBB0OScX7oMjYPQFL6n7vgKIVaU7YcZ+tJ6r
a4oVt7zu3seiBzO0gijTcvlZ8PMIZUc21DnV2PtFgzaq5iem8mGdlVZXyL6MzbRx
d4GIODPnWpCKABHd8dUMbbkOtkp1HMEQmaEdYr4zFFs53Snq4YZzFFhxRrfZCZfj
AgMBAAGjZjBkMB8GA1UdIwQYMBaAFI+na6QQzxN0Z0ZdrF1wdDKuMkbbMB0GA1Ud
DgQWBBSPp2ukEM8TdGdGXaxdcHQyrjJG2zAOBgNVHQ8BAf8EBAMCAQYwEgYDVR0T
AQH/BAgwBgEB/wIBATANBgkqhkiG9w0BAQUFAAOCAQEAaPuj04oBLi6JExkkOzJX
yYf+x0/dXXEt8oknr2qlfyaM2R6PXVcqE6HKtRcvxzuDSrgEHBb8N9k21YuF6ftM
QeQTyRcJVJVuTW29Vw+vxj/NPqGSjpWTWA32jd2FqM8lcrw8JQ+cOtCMYxdjBb6j
AJ9yiZ6AScEWGlN6hUS/KFZByKEnQTLiJ9BHooB651e1+TYs8BA3LuSYi3xKYniT
sjde9nvePJAhTsxjs+oJklZiNR5yR6w385ah5Lgqyieb3+jAVfgExjc+h2hayOAl
0/y2h8gQOlDzNRPUUftrUo9dMKJqAAyZyH18HH3kYbJ+9iy/cmHXY8OU5AdqTS/F
hg==
-----END CERTIFICATE-----
```

CommisSign Class A:

```
-----BEGIN CERTIFICATE-----
MIIDQjCCAiqgAwIBAgIBDzANBgkqhkiG9w0BAQUFADAmMSQwIgYDVQQDExtFdXJv
cGVhbiBDb21taXNzaW9uIFJvb3QgQ0EwHhcNMDMwMjE3MTcyMzAwWhcNMTIxMjE3
MTcyMzAwWjA7MRwwGgYDVQQKExNFdXJvcGVhbiBDb21taXNzaW9uMRswGQYDVQQD
ExJDb21taXNTaWduIENsYXNzIEEwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQDQc+nelotZJ6V0FYnoeXQqV5aeBydbcJHToLqBMxhJfiJv2FL/vRO151r9
YZf2etCopiRey2av0AvSebYODIDTAX+ObFYZFXYVTdaCTob6s57vCuUzW4vdCOWA
NTJ6Wk6nRJ8rJVJ1R35Jt458flFKGDGSGEk6CzkqTkOcrA+As4wURJKJCpjo8pOr
LBc7GXvAwJ4ZgqOULM5r2+YnePNfKK2Ebg1AOjcvqF9MwbYxPZXwOMQtx7k//Hpa
WK7Y2JqB8nxS36HkcLus/tBBOdzXNhq5G2rELCGoUd4HIKz1Q6BlQp7kx9vXcWGn
x0MyXKtCJ052IHhIRHxvhFmBOeqxAgMBAAGjZjBkMB8GA1UdIwQYMBaAFI+na6QQ
zxN0Z0ZdrF1wdDKuMkbbMB0GA1UdDgQWBBSfqRbgyf+Skzv2/mC99RNJPbI7sTAO
BgNVHQ8BAf8EBAMCAQYwEgYDVR0TAQH/BAgwBgEB/wIBATANBgkqhkiG9w0BAQUF
AAOCAQEAFTo9qGVx8W1O4N0XnXP46v1Qn5NXLprom1MUFGgftiNe5af9MJDnXw7+
H2rOt3m9R0mlbVb1ePLuBLNCOFVFBdh1eoAcJJM+mYtWsJRGG+DzMgTwIVhRXJ60
e2+pXUluUv4BbHSUu8ItZH4F+NxKQPb1AyjUiTHh7KY11B14zU1t+E6zVcHUoAfX
1OsHevzAJuN7gFID1LK4hEuDfKg2ZT7IFlXjZ3oMNfKz4afVBhniUsS1+F123spK
vzQT3iKQ4mI+s2Xdw00HDUGRkxVZ78YLZDTNQxvUzSmFhqyqO16Tq1Uyb2Bd3dLg
P8AVVbA0sOpobbQQIyq1icd37q4yww==
-----END CERTIFICATE-----
```

The CommisSign PKI is also available on Europa:
http://ec.europa.eu/dgs/personnel_administration/commissign/index_en.htm

24. APPENDIX III: SECURITY-CONSTRAINTS

The Servlet specification defines security constraints, how they apply and how they are combined. See in JSR-000154 Java Servlet 2.4 Specification (http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html), the chapter "SRV.12.8 Specifying Security Constraints" for the whole explanation.

Some example security constraints are available in the `web.xml` of the `ecas-demo` application, feel free to experiment with them.

If multiple security-constraints are defined, the most specific applies to a request.

Hereunder are some basic examples:

1. A security-constraint with an empty auth-constraint forbids all access by any user:

```
<security-constraint>
       <web-resource-collection>
              <web-resource-name>Forbidden</web-resource-name>
              <url-pattern>/noaccess/*</url-pattern>
       </web-resource-collection>
       <auth-constraint />
</security-constraint>
```

2. A security constraint with an `auth-constraint` with a role of "*" gives access to all authenticated users if weblogic.xml contains the directive `allow-all-roles` set on "true" otherwise authenticated users must possess one of the roles referenced in web.xml:

```
<security-constraint>
       <web-resource-collection>
              <web-resource-name>Authenticated Users Only</web-resource-name>
              <url-pattern>/authenticated/*</url-pattern>
       </web-resource-collection>
       <auth-constraint>
              <role-name>*</role-name>
       </auth-constraint>
</security-constraint>
```

3. A security-constraint with no `auth-constraint` and no `user-data-constraint` gives access to any request:

```
<security-constraint>
       <web-resource-collection>
              <web-resource-name>Access Granted for Everyone</web-resource-name>
              <url-pattern>/public/*</url-pattern>
       </web-resource-collection>
</security-constraint>
```

Security Recommendation

1. It is strongly recommended that secure Web applications take following approach. All access should be denied by default with

```
<security-constraint>
       <web-resource-collection>
              <web-resource-name>Default</web-resource-name>
              <url-pattern>/</url-pattern>
       </web-resource-collection>
       <auth-constraint />
</security-constraint>
```

2. Specific access should then be granted with constraints like:

```
<security-constraint>
       <web-resource-collection>
              <url-pattern>/public/*</url-pattern>
              <url-pattern>/images/*</url-pattern>
              <http-method>GET</http-method>
              <http-method>HEAD</http-method>
       </web-resource-collection>
       <web-resource-collection>
              <url-pattern>/servlet/*</url-pattern>
              <http-method>GET</http-method>
              <http-method>HEAD</http-method>
              <http-method>POST</http-method>
       </web-resource-collection>
       <auth-constraint>
              <role-name>MY_APP_ROLE</role-name>
       </auth-constraint>
</security-constraint>
```

Where `MY_APP_ROLE` is the name of the role used by your application and can be mapped to any Principal (uid or group) in your weblogic.xml deployment descriptor.

25. APPENDIX IV: FREQUENTLY ASKED QUESTIONS (F.A.Q.)

For more information, surf to [ECAS-FAQ].

25.1. How to have my application groups returned by the ECAS ticket validation response?

You need to ask the “User Access Administration” (UAA) to add your groups to Meta-Directory so that they are returned as CUD groups in ECAS validation response. See *10 CUD Groups*.

Functional mailbox: DIGIT USER ACCESS

25.2. Why do I get the exception: User null javax.security.auth.login.LoginException: [Security:090299]Username Not Supplied?

This exception is thrown by WebLogic **“Default Authentication Provider”** when:

- Either it is not the EcasIdentityAsserterV2 that is the **first** one to be used but another Authentication Provider, such as the **“Default Authentication Provider”,** is used before it. See [ECAS-BASIC] to know how to re-order the configured Authentication Providers so that the EcasIdentityAsserterV2 is used first.

  Having the EcasIdentityAsserterV2 used first won’t make the problem disappear but it will make WebLogic Server throw the EcasIdentityAsserterV2 Exception instead of the Exception of the “**Default Authentication Provider**”. Hence, you will be able to see the real cause of the issue and fix it or ask appropriate support for it.

- Or all the configured Authentication Providers’ Control Flag are not set on ‘`OPTIONAL`’, one of them is set on ‘`REQUIRED`’, which, not being able to authenticate ECAS tickets, throws this exception.

  See [ECAS-BASIC] to see where the Control Flags are configured using WebLogic Console.

25.3. I keep getting a "Peer Not Authenticated" Exception

This exception is thrown when the SSL handshake is refused. The solution is to install the *right* certificates in the *right* place.

- It may happen when you are using another JVM than the one you actually installed the certificates into (typically, BEA JRockit instead of Sun JDK or vice-versa).
- Or also when you are not connecting directly to the ECAS Server but going through a Web proxy. In that case, you need to install this proxy certificate instead.

25.4. I have a FailedLoginException: javax.security.auth.login.LoginException: [Security:090300]Identity Assertion Failed: User N/A-30 does not exist

This happens if you forgot to set the WebLogic DefaultAuthenticator Control Flag to 'OPTIONAL' and let it on 'REQUIRED'. Setting it to 'OPTIONAL' solves the problem.

The stacktrace looks as follows:

```
Error 500--Internal Server Error

javax.security.auth.login.FailedLoginException: [Security:090304]Authentication Failed: User
N/A-30 javax.security.auth.login.LoginException: [Security:090300]Identity Assertion Failed:
User N/A-30 does not exist
```

```
at
weblogic.security.providers.authentication.LDAPAtnLoginModuleImpl.login(LDAPAtnLoginModuleIm
pl.java:218)
at
weblogic.security.service.DelegateLoginModuleImpl$loginDelegateAction.run(DelegateLoginModul
eImpl.java:169)
at …
```

You can detect the error by looking at the name of the actual LoginModule throwing the exception. If it is the "**LDAPAtnLoginModuleImpl**" then you are in this case.

25.5. How to enable SSL debugging?

You can enable SSL debugging by using the following System properties, which you can set for example in your startup script:

```
set SSL_DEBUG_OPTIONS=-Djavax.net.debug=ssl,handshake,trustmanager
-Dssl.debug=true

set ECAS_OPTIONS=-Deu.cec.digit.ecas.client.logging=debug
-Dweblogic.StdoutDebugEnabled=true

set JAVA_OPTIONS=%ECAS_OPTIONS% %JAVA_OPTIONS% %SSL_DEBUG_OPTIONS
```

25.6. How to display SecurityProviders debug message for WebLogic Server?

You can add System properties to your startup script to activate WebLogic Security debug. This allows you to see all authentication debug message in the console or nohup.

```
-Dweblogic.log.StdoutSeverity=Debug
-Dweblogic.debug.DebugSecurityAtn=true
-Deu.cec.digit.ecas.client.logging=debug
-Dweblogic.Log.LoggerSeverityProperties="Security=Debug;ECAS=Debug"
```

25.7. When accessing my application, some users receive an error '*Invalid user: "userxxx" is an interinstitutional user*'

By default, the ECAS Client is using an assurance level that only accepts Commission's users.

If you want to accept more populations (such as interinstitutional users), you need to configure another assurance level, see *3.5.14 assuranceLevel.*

25.8. How come I am re-authenticated for every resource (pages, images, javascript, CSS, etc) on my site?

This may happen when your server is unable to maintain the Web session of the user.

It may be because the user is not sending the session cookie to your server.

This happens when you configured an incorrect cookie-path in your `weblogic.xml`.

It can also happen when your application is accessed through a reverse proxy.
If the user navigates to another application behind the same reverse proxy and if that application uses the same session cookie name as your application (by default all Java and ColdFusion applications use `jsessionid` as session cookie name), the other application's cookie collides with your own and the user's browser replaces your session cookie by the one of the other application. Your application is therefore unable anymore to identify the user and maintain her session.

In such a case, the solution is to specify the `cookie-path` directive in your `weblogic.xml` deployment descriptor. Alternatively, you can choose a unique cookie-name in your `weblogic.xml`.

25.9. Why do I have to authenticate again when connecting to my application?

The usual explanation is a web session loss due to cookie collisions. Please see [ECAS-FAQ].

25.10. The ECAS page is quickly flashing or my browser goes into an infinite loop

The usual explanation is a web session loss due to cookie collisions. Please see [ECAS-FAQ].

25.11. My application is using Proxy Tickets and gets a PROXY_COMMUNICATION_ERROR error code

Your application gets a PROXY_COMMUNICATION_ERROR error code when the ECAS Server cannot open an SSL connection to your server.

This may be the case if your server does not possess a trusted SSL certificate (i.e. a certificate signed by the CommisSign PKI) or if your server is not reachable from the ECAS Server.

25.12. I'm using remote EJBs and receive a ClassNotFoundException on the EcasUser Principal

When you use remote EJBs, the Principals of the Subject of the caller's thread are serialized over IIOP and deserialized at the remote EJB server.

Hence the remote EJB server must have the caller's Principal classes in its classpath.

For ECAS, this means that the remote EJB server must have the ECAS Identity Asserter JAR available in its system classpath.

25.13. I'm using ECAS Proxy Tickets and I get the exception "The mandatory ECAS Server ProxyURL is not configured"

The corresponding stacktrace looks like:

```
javax.servlet.ServletException: The mandatory ECAS Server ProxyURL is not configured.  Check
your ECAS client configuration file(s).
        at
eu.cec.digit.ecas.client.proxy.EcasProxyTicketReceptor.init(EcasProxyTicketReceptor.java:78)
        at
weblogic.servlet.internal.StubSecurityHelper$ServletInitAction.run(StubSecurityHelper.java:2
83)
```

or

```
javax.servlet.ServletException: The mandatory ECAS Server ProxyURL is not configured.  Check
your ECAS client configuration file(s).
        at
eu.cec.digit.ecas.client.proxy.EcasProxyTicketReceptor.init(EcasProxyTicketReceptor.java:78)
        at
weblogic.servlet.internal.StubSecurityHelper$ServletInitAction.run(StubSecurityHelper.java:2
78)
        at
weblogic.security.acl.internal.AuthenticatedSubject.doAs(AuthenticatedSubject.java:321)
        at weblogic.security.service.SecurityManager.runAs(SecurityManager.java:121)
        at
weblogic.servlet.internal.StubSecurityHelper.createServlet(StubSecurityHelper.java:64)
```

You get this exception because the ECAS Client configuration is only loaded on demand and if the EcasProxyTicketReceptor is initialized before any authentication request, the configuration is not loaded yet.

To solve the issue, in your *web.xml*, locate the following snippet

```
    <!--
        The ProxyTicketReceptor Servlet receives Proxy Granting Tickets from
        the ECAS server.
    -->
    <!-- ECAS uncomment to enable -->
    <servlet>
        <servlet-name>ProxyTicketReceptor</servlet-name>
        <servlet-class>
            eu.cec.digit.ecas.client.proxy.EcasProxyTicketReceptor
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <!---->
```

and remove any <load-on-startup> instruction.

The ProxyTicketReceptor Servlet snippet in your *web.xml* should then look like:

```
    <servlet>
        <servlet-name>ProxyTicketReceptor</servlet-name>
        <servlet-class>
            eu.cec.digit.ecas.client.proxy.EcasProxyTicketReceptor
        </servlet-class>
    </servlet>
```

Without the load-on-startup tag, the Servlet is not initialized too early and the configuration is going to exist when the first ProxyGrantingTicket is sent by the ECAS server.

26. APPENDIX V: EXAMPLE CONFIGURATION FILES

26.1. Example 1 – relying on the configured ListenAddress

The application is deployed inside the Commission’s network.

The application server is managed by the Data Centre, which configures WebLogic `ListenAddress` with the correct value.

The application only performs authentication (no need for groups nor `ExtraGroupHandler`).

The application context-path is `/mycontextpath`.

The configuration file `ecas-config-mycontextpath.xml` is:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
</client-config>
```

26.2. Example 2 – one CUD group and user details

The application is deployed inside the Commission’s network.

The application server is managed by the Data Centre, which configures WebLogic `ListenAddress` with the correct value.

The application performs authorization based upon a CUD group called `EXAMPLE_GRP`.

The application wants to receive all user details.

The application context-path is `/mycontextpath`.

The configuration file `ecas-config-mycontextpath.xml` is:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <groups>
        <group>EXAMPLE_GRP</group>
    </groups>
    <requestingUserDetails>true</requestingUserDetails>
</client-config>
```

26.3. Example 3 – interinstitutional users, custom groups and resubmitting forms

The application is deployed inside the Commission’s network.

The application server is managed by the Data Centre, which configures WebLogic `ListenAddress` with the correct value.

The application performs authorization based upon its own custom groups using an `ExtraGroupHandler` implementation: `eu.europa.ec.dg.ecas.demo.JpaExtraGroupHandler`.

The application also accepts interinstitutional users, who have an *assurance level* equal to *HIGH*.

The application wants its forms to be automatically resubmitted when user sessions time out.

The application context-path is `/mycontextpath`.

The configuration file `ecas-config-mycontextpath.xml` is:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <extraGroupHandler>eu.europa.ec.dg.ecas.demo.JpaExtraGroupHandler</extraGroupHandler>
    <reSubmitPosts>true</reSubmitPosts>
    <assuranceLevel>HIGH</assuranceLevel>
</client-config>
```

26.4. Example 4 – force using only one reverse proxy

The application is deployed inside the Commission’s network.

The application server must always be accessed through one reverse proxy.

This reverse proxy is IntraComm: `http://www.cc.cec`

The application only performs authentication (no need for groups nor `ExtraGroupHandler`).

The application internal context-path is `/mycontextpath`.

The reverse proxy transforms the internal context-path into `/proxiedpath`.

The configuration file `ecas-config-mycontextpath.xml` is:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <cas:serverName>www.cc.cec</cas:serverName>
    <cas:loginUrl>https://www.cc.cec/cas/login</cas:loginUrl>
    <serverProtocol>http</serverProtocol>
    <serverPort>80</serverPort>
    <serverContextPath>/proxiedpath</serverContextPath>
    <signatureUrl>https://www.cc.cec/cas/signature/sign.do</signatureUrl>
</client-config>
```

26.5. Example 5 – server external to the Commission’s network

The application is deployed outside of the Commission’s network.

The application is accessed through one reverse proxy.

This reverse proxy URL is: `https://externalgateway`

The application only performs authentication (no need for groups nor `ExtraGroupHandler`).

The application internal context-path is `/mycontextpath`.

The reverse proxy transforms the internal context-path into `/proxiedpath`.

The configuration file `ecas-config-mycontextpath.xml` is:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <cas:serverName>externalgateway</cas:serverName>
    <ecasBaseUrl>https://webgate.ec.europa.eu</ecasBaseUrl>
    <serverProtocol>https</serverProtocol>
    <serverPort>443</serverPort>
    <serverContextPath>/proxiedpath</serverContextPath>
</client-config>
```

Note: this is an extremely rare use case. In all other cases, applications should **not** modify the `ecasBaseUrl` property.

26.6. Example 6 – using your own Ecas Mockup Server

The application is configured to use your own installation of the Ecas Mockup Server.

Important configuration notes about the Mockup Server:

1. Your Mockup Server probably uses your own SSL certificate, which you have to trust (so import it into your JVM trustStore e.g. *$JRE_HOME/jre/lib/security/cacerts*)
2. The Mockup Server only accepts the **BASIC** strength

In the “.properties” configuration file, this translates as:

```
###########################################################################
# ECAS Client configuration properties                                    #
###########################################################################
## ecasBaseUrl:
### Base URL for all ECAS URLs when they are specified as relative URLs.
### [Optional]
### [DefaultValue=https://ecas.cc.cec.eu.int:7002]
eu.cec.digit.ecas.client.filter.ecasBaseUrl=https://@MY_MOCKUP_SERVER_NAME@:@MY_MOCKUP_SERVE
R_SSL_PORT@
###########################################################################
## acceptStrengths:
### [Optional]
### [DefaultValue=PASSWORD]
### For the mock-up server, use BASIC strength.
eu.cec.digit.ecas.client.filter.acceptStrengths=BASIC
###########################################################################
```

In the XML configuration file, this looks like:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <ecasBaseUrl>
        https://@MY_MOCKUP_SERVER_NAME@:@MY_MOCKUP_SERVER_SSL_PORT@
    </ecasBaseUrl>
    <acceptStrengths>
        <strength>BASIC</strength>
    </acceptStrengths>
</client-config>
```

Where you have to replace @MY_MOCKUP_SERVER_NAME@ by the hostname of your Mockup Server and @MY_MOCKUP_SERVER_SSL_PORT@ by its HTTPS port.

An important remark is that @MY_MOCKUP_SERVER_NAME@ must match the common name (CN) specified in the SSL certificate used by your server.

26.7. Example 7 – using the ECAS Acceptance environment

The application is configured to use the ECAS Acceptance environment, which is the acceptance or playground environment.

The ECAS Acceptance environment is accessible at https://ecasa.cc.cec.eu.int:7002/cas/login (only from inside the Commission's network) or https://ecas.acceptance.ec.europa.eu/cas/login (Public URL). It uses the same intermediate and root certificates as production (CommisSign – 2 and European Commission Root CA – 2). These certificates are embedded in the ECAS client trustStore so nothing has to be done about it.

You need to adapt your ECAS Client configuration file to point to the Acceptance environment because all default values use the production servers.

In the “.properties” configuration file, this translates as:

```
###########################################################################
# ECAS Client configuration properties                                    #
###########################################################################
## ecasBaseUrl:
### Base URL for all ECAS URLs when they are specified as relative URLs.
### [Optional]
### [DefaultValue=https://ecas.cc.cec.eu.int:7002]
eu.cec.digit.ecas.client.filter.ecasBaseUrl= https://ecasa.cc.cec.eu.int:7002
###########################################################################
```

In the XML configuration file, this looks like:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <ecasBaseUrl>
        https://ecasa.cc.cec.eu.int:7002
    </ecasBaseUrl>
</client-config>
```

The *ecasBaseUrl* property is the easiest way to modify all the URLs in the client configuration to point to the validation environment instead of production.

This only works if you haven't configured an absolute URL in one of the URL properties.

26.8. Example 8 – using the ECAS Load environment

The application is configured to use the ECAS Load environment for stress testing purposes.

The ECAS Load environment is accessible at https://ecasl.cc.cec.eu.int:7002/cas/login (only from inside the Commission's network and only during your stress test window as granted by DIGIT TESTCENTRE).

Important configuration notes about the Load environment:

1. You need to adapt your ECAS Client configuration file to point to the Load environment because all default values use the production servers.
2. Besides, same as a mockup server, the Load server only accepts the **BASIC** strength

In the ".properties" configuration file, this translates as:

```
############################################################################
# ECAS Client configuration properties                                     #
############################################################################
## ecasBaseUrl:
### Base URL for all ECAS URLs when they are specified as relative URLs.
### [Optional]
### [DefaultValue=https://ecas.cc.cec.eu.int:7002]
eu.cec.digit.ecas.client.filter.ecasBaseUrl= https://ecasl.cc.cec.eu.int:7002
############################################################################
## acceptStrengths:
### [Optional]
### [DefaultValue=PASSWORD]
### For the mock-up server, use BASIC strength.
eu.cec.digit.ecas.client.filter.acceptStrengths=BASIC
############################################################################
```

In the XML configuration file, this looks like:

```
<?xml version="1.0" encoding="utf-8"?>
<client-config xmlns="https://ecas.ec.europa.eu/cas/schemas/client-config/ecas/"
               xmlns:cas="https://ecas.ec.europa.eu/cas/schemas/client-config/cas/">
    <ecasBaseUrl>
        https://ecasl.cc.cec.eu.int:7002
    </ecasBaseUrl>
    <acceptStrengths>
        <strength>BASIC</strength>
    </acceptStrengths>
    <trustedCertificates>
        <trustedCertificate>
MIIETTCCAzWgAwIBAgIBATANBgkqhkiG9w0BAQUFADB2MQswCQYDVQQGEwJCRTER
MA8GA1UECBMIQnJ1c3NlbHMxETAPBgNVBAcTCEJydXNzZWxzMQwwCgYDVQQKEwND
RUMxDjAMBgNVBAsTBURJR0lUMSMwIQYDVQQDExpFQ0FTIENlcnRpZmljYXRlIEF1
dGhvcml0eTAeFw0wNjA0MTExNzUzMzZaFw0xMTA0MTIxNzUzMzZaMHYxCzAJBgNV
BAYTAkJFMREwDwYDVQQIEwhCcnVzc2VsczERMA8GA1UEBxMIQnJ1c3NlbHMxDDAK
BgNVBAoTA0NFQzEOMAwGA1UECxMFRElHSVQxIzAhBgNVBAMTGkVDQVMgQ2VydGlm
aWNhdGUgQXV0aG9yaXR5MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
otKTdKadm5tzmr59Tofq25hcY1jfhxlrcL5NnfsLc6j+nvdUdDdN2soJbbOBCMcc
8BeXW3b4ib+1WYmrE8B0wx/6dNonNf/4NFTYm+Xf80O81lsWO/E6279SBTvFw5bT
faPme+YLvpLiUa65XYpLkQK/LF4EEL7uQ8Di2nlpSqug27Dkp5Z3Jeh76aHvro1U
l+0qGEHyiJuYT7DSQSjc8XEXPwLpWdrc8izy0e6OsZIWjnrdqsBcIqylNriz7kC1
Woft77aumgwiMVYMG2yCno4AIIrceI4j/VWYFMi60cX3zXGSfvLGA/uFgVoyDze2
FKMskBBCT72G/XZ5NvDVcQIDAQABo4HlMIHiMBEGCWCGSAGG+EIBAQQEAwIA9zAO
BgNVHQ8BAf8EBAMCAf4wHQYDVR0OBBYEFC4ZiCMBXEEqyHrAdhhn9nbAtv6gMB8G
A1UdIwQYMBaAFC4ZiCMBXEEqyHrAdhhn9nbAtv6gMBIGA1UdEwEB/wQIMAYBAf8C
AQEwaQYDVR0lBGIwYAYEVR0lAAYIKwYBBQUHAwEGCCsGAQUFBwMCBggrBgEFBQcD
AwYIKwYBBQUHAwQGCCsGAQUFBwMFBggrBgEFBQcDBgYIKwYBBQUHAwcGCCsGAQUF
BwMIBggrBgEFBQcDCTANBgkqhkiG9w0BAQUFAAOCAQEAWtsqmsus+8U4p/qZTxTG
nmAm07ekim68nYJpJpif8Rg5OOBPv9RXfQABIrKHHkxyiZiG46xuHGbwNPVeXpZF
PeYQfrKC1YcuuWXUOwDUNaKCSVg7CLLUE6Io0j8aEGK8/c2rEIcmlHWb6NSxOixg
sdEoHFEyK2I/hL76/Bp9hAeuYFZTS8lwIU87ZUFGGSBZqfxn1f4NNFXjwjxn/sRG
dnh5OJvu+0o1047haOmCbY1XXio0JRekMDvoq7G30E+WsdTivkvrZoyjiX5GZ1HP
RK0JFrtp+KJwawx2uG2hC0RweUo9iTbiWch38axT7TmJf5/6Dqn43ro7STpwLpJO
rw==</trustedCertificate>
    </trustedCertificates>
</client-config>
```

The *ecasBaseUrl* property is the easiest way to modify all the URLs in the client configuration to point to the Load environment instead of production.

This only works if you haven't configured an absolute URL in one of the URL properties.

The new *trustedCertificates* property can be used as an alternative to importing the Certificate Authority into the JVM trustStore.