Liberation SISTRA: Login module

Code: DSI-LIBSISTRA-LOGIN

DOCUMENT CONTROL SHEET

DOCUMENT / FILE

Title: Liberation SISTRA: Login module	File Name / s:**SISTRA-PLUGINLOGIN-002.doc**
Code: **SISTRA-PLUGINLOGIN**	Software**: Word**
Date: September 2010	
Version**: 2**	

RECORD OF CHANGES

Version	Pages	Reason for change
1	8	Document Creation
2	11	CAS login integration

DISTRIBUTION OF DOCUMENT

Name	Staff

DOCUMENT CONTROL

PREPARATION	REVISED / APPROVED	ACCEPTED	ACCEPTED
Rafael Sanz	José Vicente Juan Pérez	**Jerome Navarrete**	**Bernat Alberti**

Complete with the name, signature and date ***Customers only***

INDEX

1. Goal

The purpose of this document is to define actions to perform in order to allow SISTRA platform use without DGTIC customized login module.

2. DGTIC Login module

DGTIC defined a customized login module strongly linked to its SEYCON user repository with next characteristics:

- Enterprise Single Sign On within CAIB (coupled to DGTIC user management)
- It allows 3 access modes: anonymous, user/password authenticated and authenticated by certificate
- It uses DGTIC signature module
- It was developed for exclusive use with CAIB customized jboss

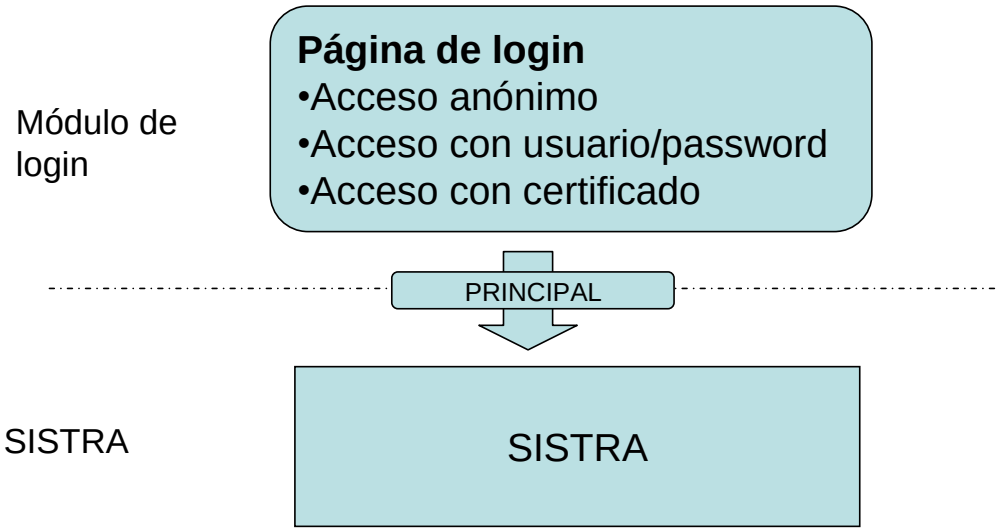
3. Required login module functionalities

SISTRA platform is based on the J2EE security standards, according to which an application expects to receive an identity (Main) having defined access permissions (Roles).

From platform point of view, the only requirement is that it should receive an identity (Main) with next properties:

- access method (anonymous, user/password authenticated and authenticated by certificate)
- nif/cif from authenticated user (if not anonymous)
- authenticated user name (if not anonymous)

Anonymous access is an authenticated access with a special user (nobody).



4. Proposed Solution

It is proposed to release this platform with a solution that doesn't force to adopt any Single Sign On in order to not compromise DGTIC login module nor any other SSO solution.

It is an organization task to decide which will be most appropriated solution for SSO corporative management and to study how to integrate it with SISTRA platform.

Single Sign On solution with which SISTRA will be released and it will serve as an example of required functionality is based on:

- JAAS LoginModule implementation for Jboss. This LoginModule will work against tables of test users. Source of this LoginModule will be provided enabling to modify them by the organization in order to make them work with other tables, ldap, etc.
- login pages located on each web module that will implement a similar access offered by DGTIC login page. @firma plug-in will be used for accessing with certificates (it's supposed that DGTIC will be who will use its signature api).
- JBoss will be used for Single Sign On management. JBoss has an embedded Apache Tomcat which provides a SSO mechanism using a valve (http://www.jboss.org/community/docs/DOC-12280)

5. Main Interface

Interface to be implemented by authenticated Main is next:

Method	Description	Parameters	Result
getMetodoAutenticacion	It gets authentication method	Main: authenticated Main	Authentication Method
getNif	It returns authenticated user nif/cif (if not anonymous)	Main: authenticated Main	Nif
getNombreCompleto	It gets full name of authenticated user (if not anonymous)	Main: authenticated Main	Full name

For more information consult javadoc

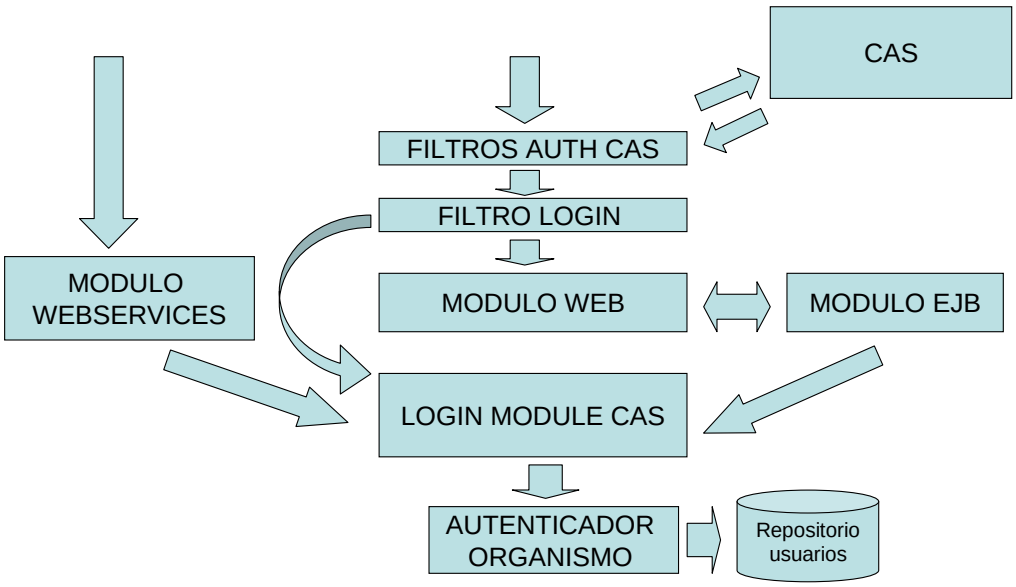
6. ANNEX I: CAS integration

SISTRA offers a login plug-in that allows to integrate with CAS (Central Authentication Service) single sign on service from JASIG.

CAS is based on specific web filters that implement CAS customized authentication. This authentication is not passed to web container (getPrincipal and isUserInRole functions from HttpServletRequest), but they are simulated through a request wrapper that intercepts the original request. By not moving to the web container, security information isn't spread to EJB container when calling from web layer to EJB one.

Next solution was designed for implementing this integration:

- web filters of CAS authentication were included in all web modules
- a customized web filter performing login in web container was included in order to achieve that information is passed to EJB container.
- Login module receives validated main by CAS in web layer and it authenticates it. This login module can receive two request types:
    - User reference validated by CAS, for which it trusts on its authentication
    - user / password (fi for webservices layer) for which it authenticates against user data source. This authentication was externalized in a class to be implemented by each organization in order to be able of customizing user data source.



On the other hand, CAS must be customized in order to achieve that SISTRA can receive next information in CAS authenticated user attributes:

- Nif: authenticated user Nif
- Name: full name of authenticated user
- Authentication method: C (Certificate) / U (User) / A (Anonymous)

Next steps are necessary for integrating SISTRA with CAS:

- Compile the project indicating that you work with CAS:
  - Set next properties in config.properties file (it sets project compilation options):

```
# Configuration login: JAAS or CAS
login-config =CAS


Login CAS #: info needed for CAS filters
# - Url where the CAS
cas.urlCas =https:// HostnameCAS / cas
#     - Url where this sistra
cas.urlSistra =https://hostnameSistra
#     - Url where this sistra (separate fronts
(sistrafront, formfront, zonaperfront, redosefront) and
backs (rest)
cas.urlSistra.front =https://hostnameSistraFronts
cas.urlSistra.back =https://hostnameSistraBacks
```

- Compile integration module with SISTRA: /integracio/llibreria-casClient from build.xml located in this directory. A jar will be generated: /integracio/llibreria-casClient/output/product/sistra-casClient.jar

- Include following libraries in the default /lib of Jboss:

  cas-client-core-3.1.10.jar

  saaj-api.jar

  saaj-impl.jar

  loginModuleCIM.jar

  OpenSAML-1.1b.jar

  sistra-casClient.jar

  xmlsec-1.3.0.jar

- Remove JBoss saaj implementation from default /lib of JBoss because it is too old to be used with CAS:

  jboss-saaj.jar

- Ensure that JBoss xml stack was updated. To do it copy next libraries to /lib/endorsed:

  resolver.jar

  xercesImpl.jar

  xml-apis.jar

- Modify JBoss login configuration to add login module for CAS:

```
<Application-policy name = "seycon">
  <Authentication>
 <Login-module code = "es.caib.sistra.casClient.loginModule.CasInternalLoginModule"
       flag = "sufficient">
            <Module-option name = "unauthenticatedIdentity"> nobody </ module-option>
```

```
</ Login-module>
</ Authentication>
</ Application-policy>
```

- Finally, it would be necessary to set CAS plug-in properties file and indicate class performing user authentication for users not coming from CAS. This property file must be located in <dir_config>\ sistra\plugins\plugin-login.properties

```
# Attributes where data are collected authenticated user
cas.nifAttribute = nif
cas.nombreAttribute = nombreApellidos
cas.metodoAutenticacionAttribute = metodoAutenticacion
# Roles: if they recover from an attribute information
CAS # gathered by the property name is indicated. If the roles
# Are set by the body's own authenticator class will be established
# As INTERNAL
cas.rolesAttribute = INTERNAL
# Cas.rolesAttribute = roles

# Authenticator class
cas.loginModule.autenticador = es.xxx.xxx.xxx
```

Authenticate class to be implemented by the organization must meet next interface es.caib.sistra.casClient.loginModule.AutenticadorInt:

```
/ **
 * Authenticate the user against power users
 *param Plugin propsPlugin Properties
 *param User Username
 *param Pass Password
 *return Returns the correct case in user information. In case incorrect returns null.
 * /
public UserInfo authenticate (propsPlugin Properties, String user, String pass);

/ **
 * Get list of roles for a user. It is used when the list of roleit is not provided by CAS
 *param Plugin propsPlugin Properties
 *param User Username
 *return In case it returns the correct list of user roles. In case incorrect returns null.
 * /
public List <String> obtenerRoles (propsPlugin Properties, String user);
```