SECURITY

OCTOBER 2015

Version 1

1 INDICE

2 INTRODUCTION

2.1 Requirements

Before reading this guide, we recommend reading **Sofia2 Concepts.pdf**[1]

2.2 Goals and reach of the current document

This guide describes the security mechanisms applied to Sofia2.

The Sofia2 platform is a controlled interoperability environment implementing several security mechanisms that operate throughout the different layers to offer authorization, authentication, consistency and as a whole integral protection of the data.

Security in the Sofia2 platform is implemented through a mechanism of plug-ins that allow to totally customize Authentication and Authorization, by allowing for the implementation of mechanisms based on different standards (LDAP, BD, Oauth, etc).

The Security covers the different layers of the application:

- **Communications:** All the gateways implemented in the platform allow for the securization of the cannel using SSL certificates.
- **Clients:** The platform only allows the connection of clients that have been authorized in the platform.
- **Operations:** The platform only allows you to do the operations that you are authorized to do.
- **Information:** The platform only allows you to interact with information that you are authorized to access. The platform also validates that the information persisting on the platform fulfills with the expected information model.

[1] http://sofia2.com/docs/%28EN%29%20SOFIA2-SOFIA2%20Concepts.pdf

3 Security in Communications

Sofia2 uses the protocol https that creates a secure communication channel for all the communications with the outside world. This channel uses SSL/TSL protocol with a RSA standard cipher requiring a security certificate X.509. It is the most extended and accepted cryptosystem standard in the www world.

The private key is installed in Sofia2's SIB and its public part, known as certificate, must be installed in the browsers or incorporated in the apps that need to access the platform from the extranet. Security through certificates prevents any filter or manipulation of data between client and server through man-in-the-middle-type attacks.

It is worth noting that the authentication system is done using a server certificate, not a client-server one (such as the fnmt certificates). This kind of encryption ensures that the client is connecting to the required server and not a possible impersonator service, but it does not authenticate the client.

4 Clients

A user must register her KPs (Client) in the platform. Otherwise, the platform will reject the KPs' connections.

To register the KPs, the user will need an Authentication Token. The user can create a Token associated to the KP, depending on her role's restrictions. A user with the roles Colaborador (collaborator) or Usuario (user) can only create tokens for a KP if she is the owner of that KP.

The KPs have an encryption key, needed only if the encryption protocol XXTEA is selected to be used. This is used in devices that do not support https, such as Arduinos.

A KP can use one or more ontologies. The ontologies are the platform information produced or consumed by that KP.

Once a KP is registered in the platform, the KP can establish connections with the platform.

Once a KP is created, we can define an instance of the KP using the Web Console.

An instance identifies the client that is going to connect to the Sofia2 platform.

The connection of a KP with the platform must be seen as two types of connection:

- **Physical Connection:** Established by the transport protocol used for the connection by a KP (TCP/IP, MQTT, WebSockets, REST, WS, JMS, Ajax-Push…). The way to do this type of connection greatly depends in the API of the used KP (Java, JavaScript, Arduino, C++...).
- **Logical Connection:** Established by the SSAP (Smart Space Access Protocol) messaging protocol defined in Sofia2. It is common to all the KP's APIs.

We will focus on Logical Connection-level security, because the Physical level security is covered by the security in the communication layer. The logical connection must keep a KP with the platform.

For A KP to connect to the platform, produce/consume data, and interoperate with other KPs, the KP must first open a session with a SIB in the platform.

The SSAP protocol provides two operations in this sense:

- **JOIN**: Where a KP tells the platform its owner's login and password, along with its own instance data. Thus, if everything is correct, the platform authenticates the KP and opens a session with it, returning a session key that can be used through a customizable, pre-established time.
- **LEAVE**: Where a KP tells the platform that the later will close the session.

As long as there is a session between the KP and the platform, the KP can use the other operations in the SSAP protocol to produce or consume information.

5 Security in Access

The platform is divided in two areas, independent of each other in their security models.

5.1 Administration

The platform has three defined Roles, which specify the functionalities available to users at an administrative level:

- **Administrador** (Administrator)
- **Colaborador** (Collaborator): This role allows user to operate with the platform's information, spilling and consuming information, and to create new information structures. It also allows users to perform information-processing tasks (through rules, scripts, reports, etc.)
- **Usuario** (User): This role allows users to operate with the platform's information, spilling and consuming information from existing information structures where that user has been authorized for these operations.

5.2 Operation

Permissions define the operative functionalities of a user on the information. These permissions apply at a level of Ontology – User.

Permissions also have three types.

Administrators have Total Permissions on all the ontologies.

Collaborators have Total Permissions on the ontologies they are Proprietary of.

- **Permiso de Lectura** (Reading Permission): Allows a user or collaborator to perform only Query-type operations on the ontologies where she has this permission.
- **Permiso de Inserción** (Insertion Permission): Allows a user or collaborator to perform only Insert-type operations on the ontologies where she has this permission.
- **Permiso Total** (Total Permission): Allows a user or collaborator to perform any operation on the ontologies where she has this permission.

6 Security in Data

All the operations are validated at Authentication level. To do this, the platform always checks whether the Client has already authenticated with the platform.

Once the Client Authentication has been checked, the platform confirms the client's authorization on two levels, sometimes three:

- Firstly, when the client tries to perform an operation on an ontology, the platform confirms whether the user can operate with the ontology (The user is authorized for that ontology).
- Secondly, the platform validates that the operation (Query, Insert, Delete, Update) that the user wants to perform can be done on that ontology (The user has the right permissions to perform that operation on that ontology).
- If all the previous steps are correctly checked and the operation is either Insert or Update, then there is a third validation to be made. This validate verifies that the information to be inserted scrupulously fulfills the defined Schema for the ontology. To do this, the JSON Schema is validated, comparing the information to be inserted with the ontology's structure.

7 Reference Implementation

The Security reference implementation is based on three plug-ins:

7.1 plugin-sofia-user

This plug-in, used only at Administration level, is in charge of recovering the user's information. In the reference implementation, the plug-in recovers this information from the configuration database.

This plug-in can work with encrypted or non-encrypted password, which allows for the customization of the encryption algorithm.

```
public void persist(Usuario user) throws NotImplementedException;

public void remove(Usuario user) throws NotImplementedException;

public void merge(Usuario user) throws NotImplementedException;

public long countUser() throws NotImplementedException;

public List<Usuario> findAllUser() throws NotImplementedException;

public Usuario findUser(String idUsuario) throws NotImplementedException;

public List<Usuario> findUsers(String qlString, List<Object> parametros)
throws NotImplementedException;

public Usuario findLoginUser(String identificador, String credential, String
sourceInfo) throws EmptyResultDataAccessException;

public List<Usuario> findUserByCriteria(Usuario usuario) throws
NotImplementedException;

public  List<Usuario>  findUserByIdentificacion(String  identificacion)  throws
EmptyResultDataAccessException;
```

7.2 plugin-console-security

This plug-in is in charge of managing the Authentication and Authorization in the Administration console, and it is based on Spring Security. In the reference implementation, this plug-in uses the configuration database.

This plug-in uses plugin-sofia-user to recover users' information.

7.3 plugin-sib-security

This plug-in is in charge of managing the Authentication and Authorization to the SIB operations and is based on a mechanism of Token – SessionKey.

It uses plugin-sofia-user to recover the users' information:

This plug-in must meet the following interface:

```
String authenticate(SSAPMessage message) throws AuthenticationException;

void checkSessionKeyActive(String sessionKey) throws AuthenticationException;

void closeSession(String sessionKey) throws AuthenticationException;

public void removeAuthenticationContextSessionkey(String sessionKey) throws
AuthenticationException;

void checkAuthorization(SSAPMessageTypes operationType, String ontologyName,
String sessionKey) throws AuthorizationServiceException;

void checkAuthorizationConfig(SSAPMessageTypes operationType, String
tableName, String sessionKey) throws AuthorizationServiceException;

void checkAuthorization(SSAPMessageTypes operationType, String kpName, String
instanceKpName, String token) throws AuthorizationServiceException;
```