



DEVELOPMENT OF A SOFIA2 CLIENT (KP) FOLLOWING THE MODEL KP ARCHITECTURE

May 2016

Version 1



INDEX

INDEX.....	2
STEP 1 – GETTING FAMILIAR WITH SOFIA2'S BASIC CONCEPTS	3
STEP 2 – GETTING SOFIA2 USE CREDENTIALS	4
REGISTERING IN SOFIA2 PLATFORM	4
LOGGING IN TO SOFIA2 ON CLOUD	5
STEP 3 – SELECTING DEVELOPMENT PLATFORM (WINDOWS, MAC, LINUX) AND DOWNLOAD THE SDK 7	
DOWNLOADING AND INSTALLING SOFIA2'S SDK.....	7
STARTING UP SOFIA2'S SDK.....	8
JAVA IDE.....	9
ARCHITECTURE CONSOLE	9
STEP 4 – CONFIGURATION	11
REGISTERING A KP IN THE PLATFORM	11
GENERATING A TOKEN	11
CREATING A NEW CONFIGURATION	13
ASSIGNING SOFTWARE IN KP	14
STEP 5 – DEVELOPING A KP FOR A GATEWAY FOLLOWING THE MODEL KP	16



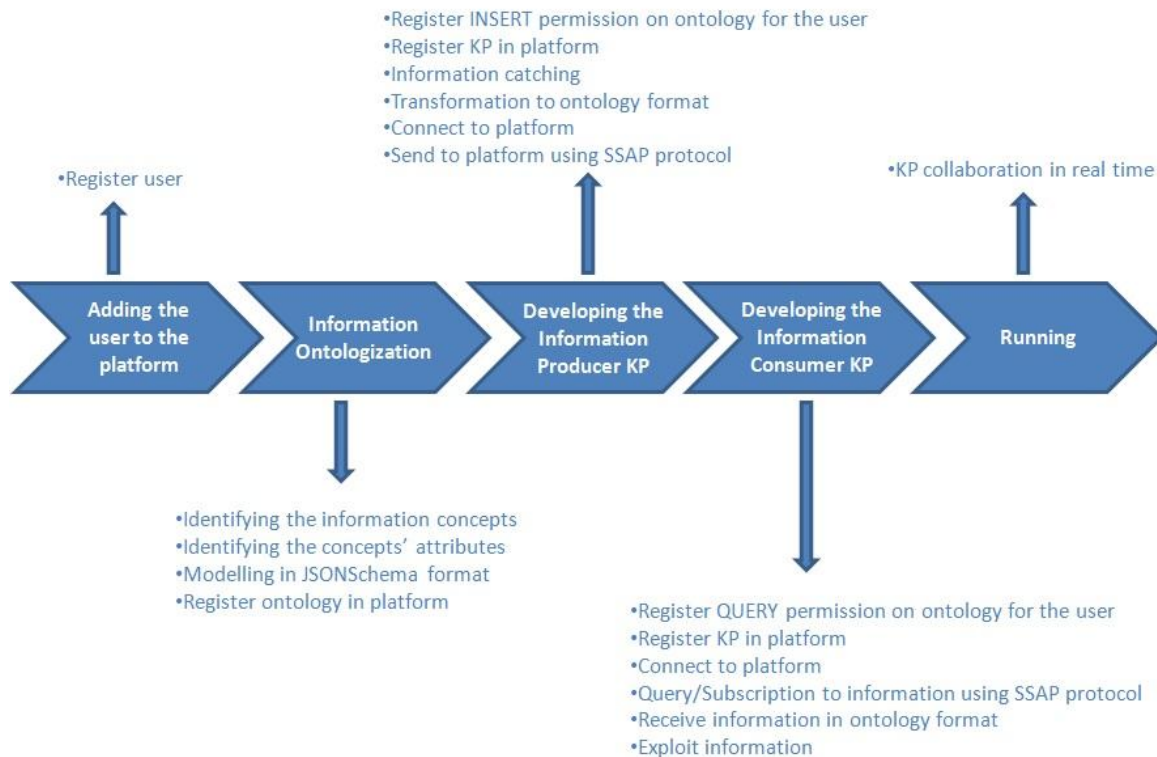
Step 1 – Getting familiar with SOFIA2's basic concepts

- **Smart Space:** A collaborative, virtual environment where devices and applications interoperate with each other to provide a complex functionality.
- **KP (Knowledge Processor):** It represents each client that produces and/or consumes information in a Smart Space.
- **SIB (Semantic Information Broker):** It is the Smart Space's core. It acts as the integration element for the exchanged semantic information, and also as information storage.
- **SSAP (Smart Space Access Protocol):** It is the standard messaging protocol between the KP's and the SIB.
- **Ontology:** It represents an Entity's definition within my System. They can be created, and there are pre-existing definitions (e.g. for a City environment).
- **Ontology Instance:** It represents a specific Entity in the system. It is a JSON structure.
- **KPModel:** Development model for KP managed by the SIB, based in Workers in a publish-subscribe model.



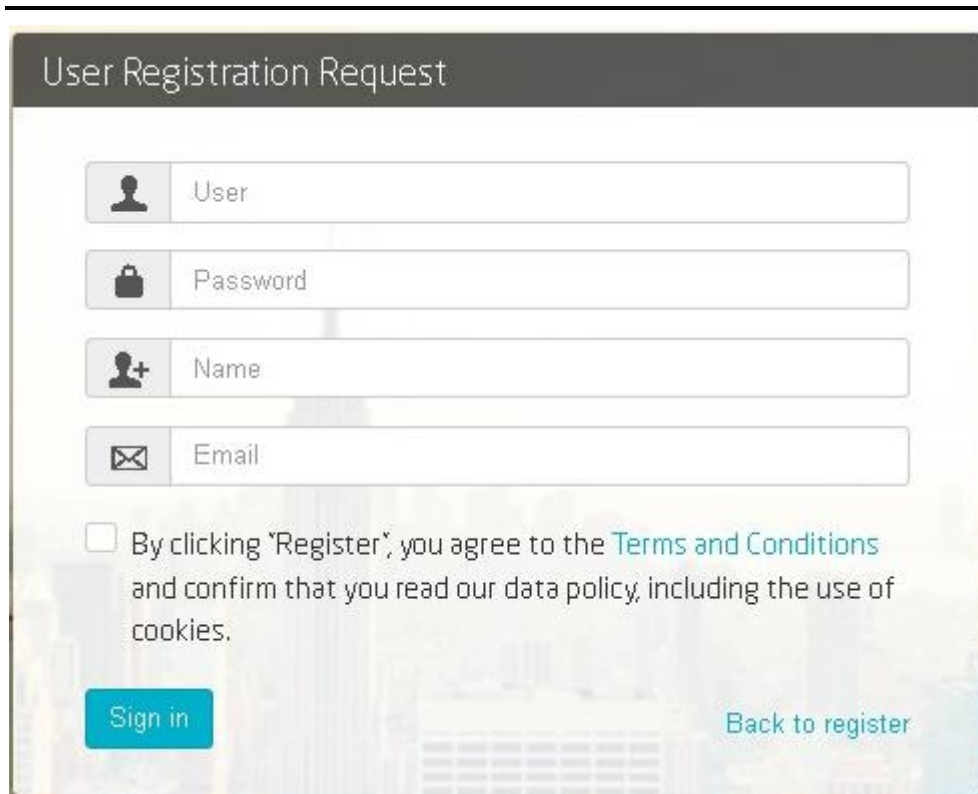
Step 2 – Getting SOFIA2 use credentials

The process to develop on the SOFIA2 platform is as follows:



Registering in SOFIA2 platform

We can register in the platform by accessing the page <http://sofia2.com/console/login> then select Register for this site. We will have the role USER, which will allow us to subscribe to public ontologies and perform queries on those.

A screenshot of a web form titled "User Registration Request". The form has a dark grey header with the title in white. Below the header, there are four input fields, each with a grey icon on the left: a person icon for "User", a padlock icon for "Password", a person with a plus icon for "Name", and an envelope icon for "Email". Below these fields is a checkbox with the text "By clicking 'Register', you agree to the [Terms and Conditions](#) and confirm that you read our data policy, including the use of cookies." At the bottom left is a blue "Sign in" button, and at the bottom right is a blue "Back to register" link. The background of the form is a blurred image of a city skyline.

User Registration Request

User

Password

Name

Email

☐ By clicking "Register", you agree to the [Terms and Conditions](#) and confirm that you read our data policy, including the use of cookies.

Sign in

[Back to register](#)

We will need to e-mail the platform's administrator to request the change to the role COLLABORATOR, because we will need to perform a number of operations we would not have access to.

Logging in to SOFIA2 On Cloud

We will log in at the page <http://sofia2.com/console/login> with the user we created in the previous point. This will give us access to the console.

Step 3 – Selecting development platform (Windows, Mac, Linux) and download the SDK

Downloading and installing SOFIA2's SDK

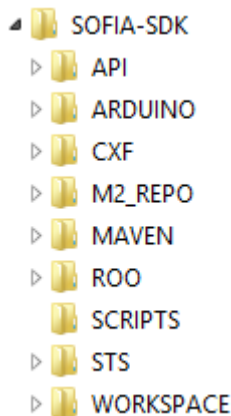
Download the environment's last version from the following URLs (available at http://sofia2.com/desarrollador_en.html#descargas):





- SOFIA2 SDK for Windows
http://sofia2.org/sdk/SOFIA2_SDK_WIN.zip
- SOFIA2 SDK for Mac
http://sofia2.org/sdk/SOFIA2_SDK_2.9_MAC.zip
- SOFIA2 SDK for Linux
http://sofia2.org/sdk/sofia2_sdk_linux.tar






Uncompress the downloaded ZIP file:







We can find all the components making up the SDK in the directory **SOFIA-SDK**




- The **development environments** for the C environments in Arduino and Java are in the  **ARDUINO** and  **STS** directories, respectively.
- The  **ROO** directory includes the Architecture console with the SOFIA commands to create Plugins, KP's and Gateways.
- The  **CXF** directory includes the productivity tools to work with WSDL and WADL.

- The  **MAVEN** and  **M2_REPO** directories include the Maven Tool and the dependencies it uses to develop on the SOFIA Platform.
- The  **API** directory includes Javascript and Java API's to develop KP's.
- The  **WORKSPACE** directory includes the IDE's' configuration directories.
- The  **SCRIPTS** directory provides the batch processes for the different components making up the SDK.

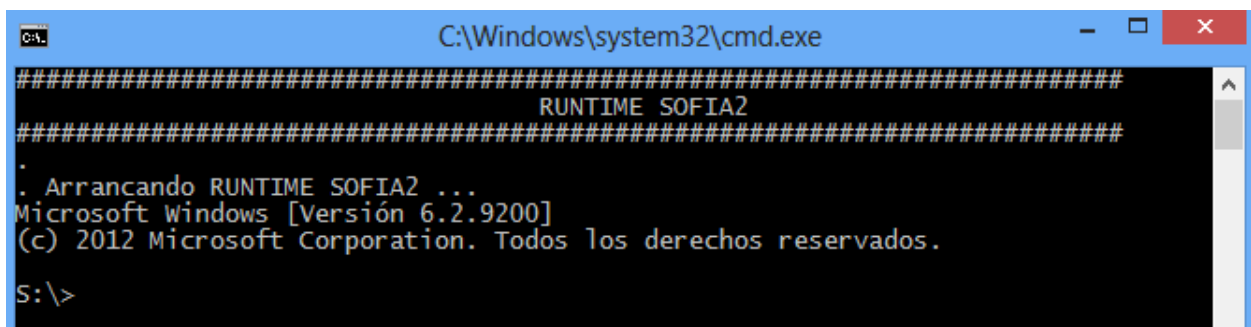
-  **ARDUINO**
-  **ARQSPRING**
-  **ARQSPRINGWSDL2JAVA**
-  **IDE**

Starting up SOFIA2's SDK

Once SOFIA2's SDK has been downloaded, we can find the following contents in the  **SOFIA** root directory:

-  **SOFIA2_API_JAVA**
-  **SOFIA2-SDK**
-  **SOFIA-RUNTIME**
-  **SOFIA2_API_JAVA.zip**
-  **Sofia2_IDE.bat**
-  **Sofia2_SDK-START.bat**
-  **Sofia2_SDK-STOP.bat**
-  **Sofia2_VariablesEntorno.bat**

By double-clicking the batch process  **Sofia2_SDK-START.bat** we will create the virtual drive **S:** and open the command console.




```


C:\Windows\system32\cmd.exe
#####
                        RUNTIME SOFIA2
#####
. Arrancando RUNTIME SOFIA2 ...
Microsoft Windows [Versión 6.2.9200]
(c) 2012 Microsoft Corporation. Todos los derechos reservados.
S:\>

```

This command console's session is configured to run with SOFIA's SDK and Runtime.

Installing SOFIA2's SDK modifies the  Sofia2_SDK-START.bat file, created by the Runtime. Due to this, installation directories must be observed.

IMPORTANT

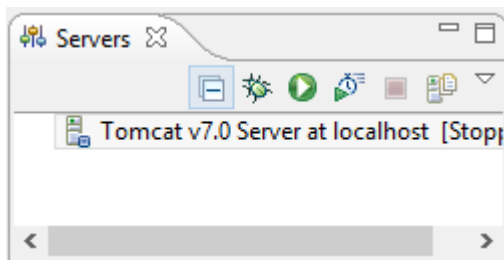
If the command console is closed, we must re-run the batch process  Sofia2_SDK-START.bat to re-open the console with the predefined configuration.

Java IDE

Run the command **S:\>Sofia2_IDE** in the console.

```
S:\>Sofia2_IDE
#####
                        SDK SOFIA2
#####
. Arrancando IDE Sofia2...
```

The iTR Architecture Eclipse IDE to develop SOFIA2 applications will open.



Architecture Console

It offers productivity tools to easily create SOFIA2 Projects.

Run the command **S:\>arqspring** in the console.

```
S:\>arqspring
#####
                        SDK SOFIA2
#####
. Arrancando ArqSpring...
```

The iTR Architecture console will open.

1.2.4.RELEASE [rev 75337cf]

```
sofia2> █
```

Step 4 – Configuration

Registering a KP in the platform

Users must register their KP's in the Platform. Otherwise, the Platform will reject the connection of that KP's.

To register a KP, the Platform provides an area of **KP Management**, where a user can create a new KP or manage those KP's that have been created previously:

Sofia 2

KP's/APPS SOFIA2 / New KP

CREATE NEW KP

KP

Identification

Encryption Key

89dd770d-820d-4363-8b1c-a8ff584d36d

Description

Ontology

CompostSensor
meteoStat13
SensorHumedad
SensorTemperatura

Group Ontologies

Meta-Inf

Cancel New


As we can see, a KP can make use of one or several ontologies. This is the Platform's information that the KP will produce or consume.

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

Generating a token

Once registered in the platform, we need to generate a token to connect with the KP.

To do this, we will go to the section **kp's/apps sofia2 > My Tokens**, where a user can generate one or more tokens associated to a KP.



[REST API Doc](#)
[User](#)
[Language](#)
[Log Off](#)

KP's/APPS SOFIA2 / My Tokens

MY TOKENS

KP Identification

Search Cancel

EXISTING KPS LIST

KP Identification

MonitorTempHumed

KPepito

KpScada

meteoemplokp

MonitorTempHumed

KP Owner

Showing 1 to 5 of 5 entries

First Previous 1 Next Last


TOKENS LIST

No data available in table

1

Generate Tokens

First Previous Next Last

To generate new tokens, we will select the KP for which we want to generate tokens and the number of tokens we want to generate, then we will press the button .

Then we will be shown a new table with the tokens we have generated and which the KP's can use to connect to the platform.

MY TOKENS

KP Identification

MonitorTempHumed

Search Cancel

EXISTING KPS LIST

KP Identification

MonitorTempHumed

KPepito

KpScada

meteoemplokp




MonitorTempHumed

KP Owner

Showing 1 to 5 of 5 entries

First Previous 1 Next Last

TOKENS LIST

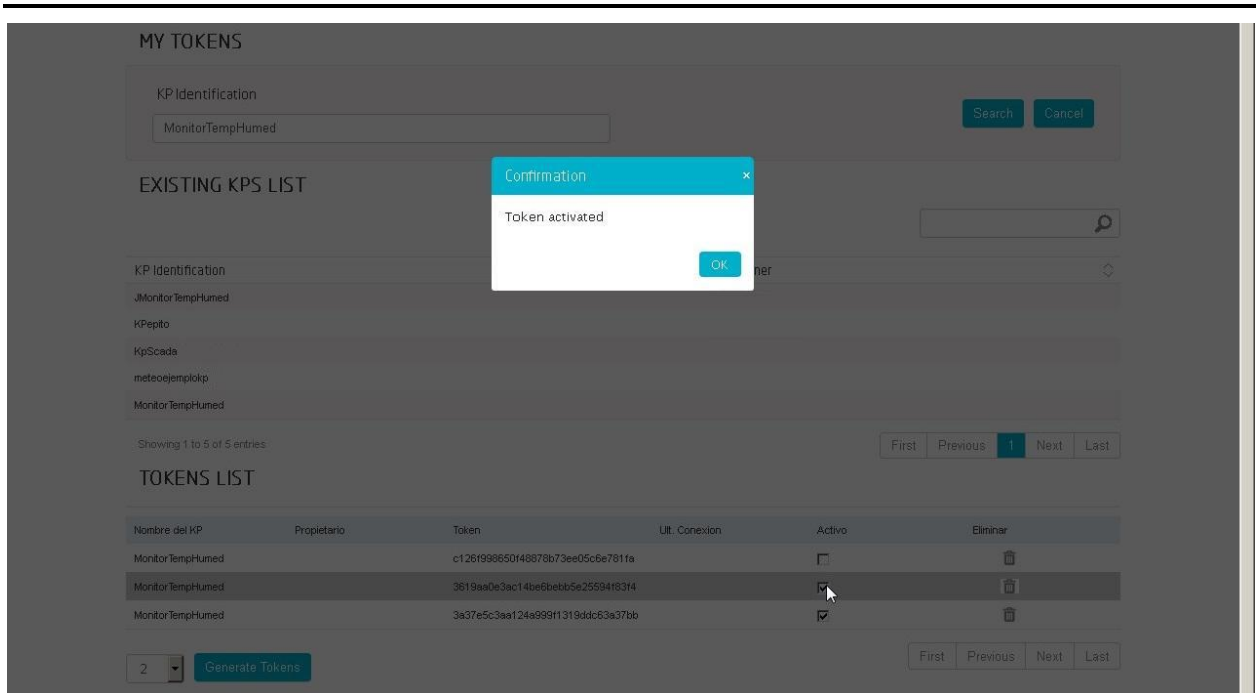
Nombre del KP	Propietario	Token	Ult. Conexion	Activo	Eliminar
MonitorTempHumed		c1261998650f48878b73ee05c6e781fa		<input type="checkbox"/>	
MonitorTempHumed		3619aa0e3ac14be6bebb5e25594f83f4		<input type="checkbox"/>	
MonitorTempHumed		3a37e5c3aa124a999f11319ddc63a37bb		<input checked="" type="checkbox"/>	

2

Generate Tokens

First Previous Next Last

The tokens are deactivated when generated. We can activate them by clicking on the checkbox Activo.



Creating a new configuration

The model based on Model KP delegates the KP's configuration to the SIB. TO do this, the user can create a new configuration that she can associate with her Model KP.

The user creates a new configuration using the option **SW Management > My SW Configurations > New Configuration**.

This screen is made up of two main sections, "**SW Management Inf.**" and "**Configuration Information**".

In the first section we will specify the software management data such as:

- **Application Name:** The system will tell us if we try to generate a software application with a pre-existing name.
- **Application:** It allows us to select the .war with the software that the clients will use.
- **Description:** Description to be included on the software application.

In the section on configuration data, we will specify the software configuration properties (such as: KP, KP instance, connection token, IP, port, etc.)

SW MANAGEMENT / New Configuration

NEW SW CONFIGURATION

SW Management Inf.

Application Name Version Application

Description

Configuration Information

Description

Properties

Property	Value	Options
<input type="text"/>	<input type="text"/>	<input type="text"/>

Property Information

Property

Description



Value



Once we create a software application, we can see the detail with all the data.

Assigning software in KP

We must establish a link between the newly created application configuration and the previously created KP and KP instance.

To do so, we must go to the menu option SW Management > SW Configuration Assignment. We will see a screen like the following one, where we will have to insert the values and press the button Version Assignment:

Sofia 2  

REST API Doc User  Language  Log Off

SW MANAGEMENT / SW Configuration Assignment

SW CONFIGURATION ASSIGNMENT

Managed APP

Managed APP Version

Kp

KP Instance

LIST OF ASSIGNMENT

Kp	Kp Instance	Managed APP	Managed APP Version	User	Dps.
Showing 0 to 0 of 0 entries					

First Previous Next Last

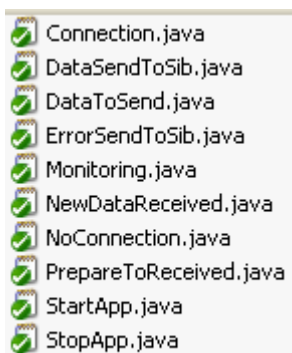
The newly created assignment will appear in the assignment list.



Step 5 – Developing a KP for a gateway following the model KP

- 1) From the platform's console we write ***sofia2 crearAppModelo -id [NAME OF MODEL APP] -paquetebase [APPLICATION PACKAGE]***
- 2) We implement the method `readDataSensor` of the class `PrepareToReceived`, where we must establish the connectivity with the sensors and convert the information we get from them in a `SensorMessage` object.
- 3) In the `PrepareToReceived` class we can read from all the sensors or we can create new classes, extending from `PrepareToReceivedWorkerImpl` to manage each of the connected sensors.
- 4) We implement the method `generateSSAPMessage` of the class `NewDataReceived`, where we must transform the data we have gathered from the sensors into a SSAP message.
- 5) Through the implementation of the class `DataSendToSib`, we get the SIB's response message, along with the original message we had sent to the SIB.
- 6) If we want to subscribe to the SIB, we must implement the class `SubscriptionListener` in the method `init`. We can subscribe to the SIB using the method `subscribe(ontology, query, SSAPQueryType)`; and, through the method `onEvent`, we can define the operations to be performed whenever we receive from the SIB a notification associated to that subscription.
- 7) Additionally, we can implement the other classes that allow us to control aspects such as:

- 8.1. Starting and stopping the KP Application server. StartApp and StopApp (The later must implement the desired behaviour when the KP container requests the stopping).
- 8.2. Connecting to the SIB and loss of the connectivity.
- 8.3. Pre-processing and post-processing the messages sent to the SIB.
- 8.4. Error when sending messages to the SIB.



We achieve this by implementing each of the Workers defined in the previous diagram.

- 8) The developers may need to register new Workers. Those must be registered using the bean KPWorkerCfg.

- 8.1. Getting a reference to this object:

```
@Autowired
```

```
protected KPWorkerCfg kPWorkerCfg;
```

- 8.2. Invoking the method `public void addEvent(String event, Class message, Subscriber worker)`:

```
kPWorkerCfg.addEvent("MY_EVENT", LifeCicleMessage.class, new StartAppWorker())
```

- 9) To publish an existing method, or one method created by the developers, we must do so through the method `publish (String topic, Object message)` which we find in the bean KPWorkerCfg.

The events that are de facto available in the platform are those that we find in the following list of classes:

- SibEvents
- SensorEvents
- LifeCicleEvents
- InfraestructureEvents

