
# SOFIA2

# CONCEPTS

MAY 2014

Version 3

indra

# 1 INDEX

# 2 INTRODUCTION

## 2.1 Current document goals and scope

The current document describes the SOFIA Platform basic concepts.

This guide should be a user's first contact point with the SOFIA2 Platform.

It describes the first steps needed to install it and to get a first glance at it.

# 3 WHAT'S SOFIA?

SOFIA is an acronym for **SMART OBJECTS FOR INTELLIGENT APPLICATIONS**. SOFIA is a platform, the result of a three-year ARTEMIS R&D project that ended in March 2012. Nineteen partners from four EU countries, including Nokia, Philips, Fiat, Acciona and Indra, were involved in the project.

SOFIA is a middleware architecture allowing for the interoperability of several systems and devices. It allows making real information available for intelligent applications (**Internet of Things**).

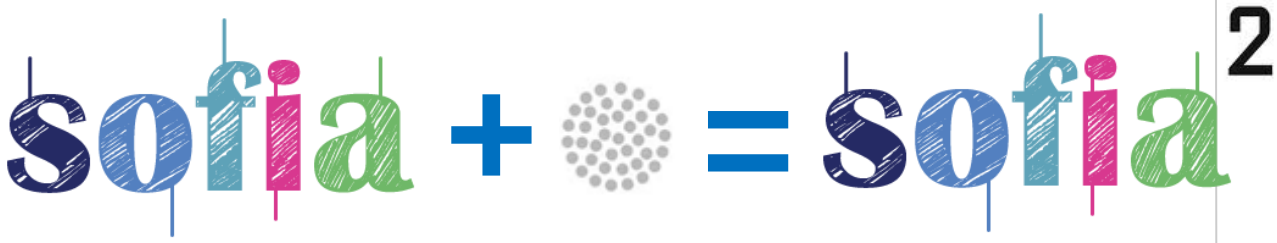
SOFIA is:

- Open-source
- Multi-platform: Available for MS Windows, Android, Linux, iOS,
- Multi-language: It has portings to Java, JavaScript, C++, Arduino
- Communication agnostic: With implementations for TCP, MQTT, HTTP (REST and WebServices), Ajax Push, …

Its goal is achieving interoperability among different applications that share **semantic concepts.**

After the end of the ARTEMIS Project, Indra kept evolving the original SOFIA project, creating a platform that focuses on enterprise use.

The current version of the Platform is called SOFIA2 (sometimes written SOFIA$^2$ ).




SOFIA2 is focused on the following areas:

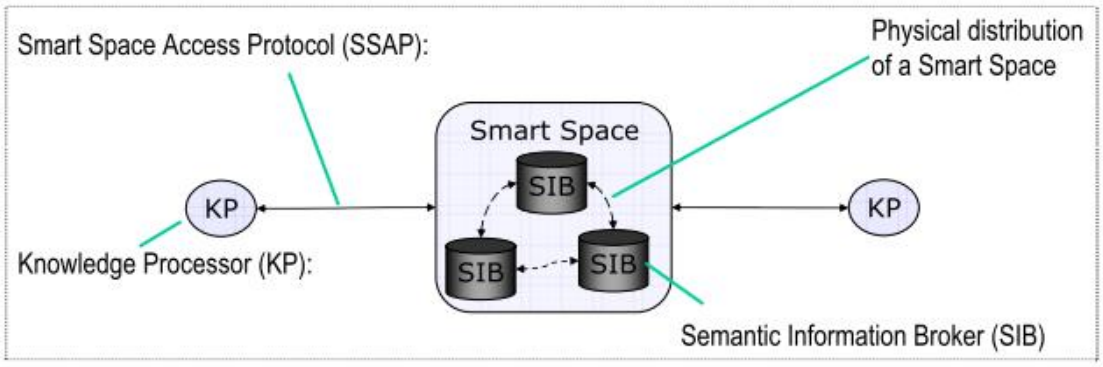
- Adapting it to the enterprise environment: High availability operation with distributed data centres …
- Work with the Platform was simplified, particularly in the following areas:
  - Ontology development (ontologies became lightweight)
  - Query language

  - SSAP: With a JSON implementation besides the XML one.
- Big Data Interfaces (Hadoop) to host huge amounts of data and data warehouse.
- Integration capacities with back-ends with standard protocols, e.g. Web Services, …
- Plug-in concept to expand the SIB
- Integrated storage and GIS queries
- Addition of pluggable security mechanisms.
- REST interfaces to connect easily from smart phones, devices, RIA applications, …

# 4 SOFIA2 PLATFORM CONCEPTS

The SOFIA Platform can be conceptualized through these four concepts:

- Smart Space
- SIB
- KP
- SSAP




## 4.1 Smart Space

- This is the virtual environment where different applications interoperate with each other to provide a complex functionality.
- The Smart Space's core is the **SIB**.
- Commonly, there is a single SIB (that may be a cluster of SIBs) in each Smart Space. In specific cases, however, a Smart Space may have a federation of SIBs.
- A Smart Space can communicate with other Smart Spaces by establishing trusted relationships with them.

## 4.2 SIB (Semantic Information Broker)

- This is the core of the Platform.
- It receives, processes and stores all the information of applications connected to the SOFIA Platform, thus acting as the Interoperability Bus.
- All the existing concepts in the domain (reflected in the ontologies) and their current states (specific instants of the ontologies) are reflected on it.
- SOFIA$^2$ proposes using JSON to exchange information (SSAP) and to define the ontologies.

```
{"body":
    "{"query":"{HumiditySensor.measurement:{$gt:18}}"}",
    "direction":"REQUEST",
    "ontology":"HumiditySensor",
```

```
    "messageType":"QUERY",
    "messageId":121,
    "sessionKey":"88bf5ee7-84d4-4956-98a3-ff290222fd64"
}
```

- There are implementations in several languages and platforms. Indra provides a JEE SIB that runs on any JEE Web Server (Tomcat, JBoss, …)
- Gateway supports TCP/IP, HTTP, REST, Bluetooth and Zigbee transport handlers.
- **Offering connectors** to communicate from different clients:
    - REST: from clients such as JavaScript, smart phones, ...
    - MQTT for bidirectional communications and limited devices.
    - Web Services/JMS for Enterprise applications.
    - Others including Bluetooth, Zigbee, ...
- SIB is now extensible via plug-ins.

## 4.3 KP (Knowledge Processor)

- This is each of the applications interoperating in the Smart Space through the SIB.
- Each application works with instances of the relevant concepts in the domain (ontology) for which it is designed.
- There are implementations in several languages including Java, JavaScript, Arduino, …
- There are three types of KP:
    - **Producer**: A KP that inserts information in the SIB, but never recovers any information from it.
    - **Consumer**: A KP that recovers information from the SIB, but never inserts any information in it.
    - **Prosumer**: A KP that both inserts information in the SIB and recovers information from the SIB.
- SOFIA$^2$ proposes using JSON to send SSAP messages, because messages will then be lighter and fitter for embedded devices.

## 4.4 SSAP (Smart Space Access Protocol)

- This is the standard messaging language to communicate between the SIBs and the KPs.
- This language is autonomous from the underlying network (GPRS, 3G, WIFI, Bluetooth, HFC, Zigbee)
- There are two implementations:

- **SSAP-XML**: In XML format (greater bandwidth)
- **SSAP-JSON**: Messages adapted to this protocol, designed for communications with mobile devices, web browsers…

- There are three types of messages:
  - **REQUEST**: A request sent from the KP to the SIB.
  - **CONFIRM**: A response sent from the SIB to the KP to answer a REQUEST message.
  - **INDICATION**: A notification sent by the SIB to the KP on an event that the KP subscribed to.
- The operations made between the SIB and the KP are as follows:
  - **JOIN:** Connection of a KP to a SIB (implies: Authentication, authorization, creating a session in the Smart Space).
  - **LEAVE:** Disconnection of a KP from a SIB.
  - **INSERT/UPDATE/DELETE**: Allows a KP to insert/update/delete information that is on the SIB.
  - **QUERY:** It allows a KP to recover information from the SIB. It can go on the Historical Data Base or on the Real-Time Data Base.
  - **SUBSCRIBE**: It allows a KP to subscribe to the running of a query every X seconds, or to an event triggered on the SIB.
  - **INDICATION**: The result sent by the SIB to one or to several KPs to solve a subscription.
  - **UNSUBSCRIBE**: It ends an existing subscription.
  - **GET_CONFIG:** It allows the KP to ask for the configuration associated to its own instance.
  - Notify changes from the SIB to subscribers.

## 4.5 Ontologies

**Ontologies** are semantic descriptions of a set of classes. Applications sharing classes (commonly called *concepts*) from the same ontology can easily exchange information using specific instances of those common classes.

SOFIA represents ontologies in JSON format. For instance, this may be an ontology used by the KP representing a temperature sensor:

```
"TemperatureSensor": {
      "GpsCoordinates": {
        "altitude": 0,
        "latitude": 40.512274,
        "longitude": -3.675679
      },
```

```
      "identifier": "S_Temperature_00001",
      "measurement": 19,
      "timestamp": 1373887443001,
      "unit": "C"
    }
  },
```

These JSON ontologies are added to the platform. Each ontology has a JSON Schema that allows it to validate whether the semantic information sent by the KP satisfied that ontology:

The JSON Schema satisfying the TemperatureSensor ontology from the previous example is the following one:

```
{
    "$schema": "http://json-schema.org/draft-03/schema#",
    "title": "TemperatureSensor Schema",
    "type": "object",
    "properties": {
        "_id": {
            "type": "object",
            "$ref": "#/identifier"
        },
        "TemperatureSensor": {
            "type": "string",
            "$ref": "#/data"
        }
    },
    "identifier": {
        "title": "id",
        "description": "TemperatureSensor's inserted id",
        "type": "object",
        "properties": {
            "$oid": {
                "type": "string",
                "required": false
            }
        }
    },
    "data": {
        "title": "data",
        "description": "Info TemperatureSensor",
        "type": "object",
        "properties": {
            "identifier": {
                "type": "string",
                "required": true
            },
            "timestamp": {
                "type": "integer",
                "minimum": 0,
                "required": true
            },
            "measurement": {
                "type": "number",
                "required": true
            },
            "unit": {
                "type": "string",
```

```
                "required": true
            },
            "GpsCoordinates": {
                "required": true,
                "$ref": "#/gps"
            }
        }
    },
        "gps": {
            "title": "gps",
            "description": "TemperatureSensor Gps",
            "type": "object",
            "properties": {
                "altitude": {
                    "type": "number",
                    "required": false
                },
                "latitude": {
                    "type": "number",
                    "required": true
                },
                "longitude": {
                    "type": "number",
                    "required": true
                }
            }
        },
        "additionalItems": false
    }
```

When an ontology is stored in the Real-Time Data Base, the platform adds meta-information to it. This meta-information deals with that ontology's use context. We can see the information highlighted in yellow in the following example:

```
{
    "_id": {
      "$oid": "51e3dbd465701fd8e0f69828"
    },
    "contextData": {
      "session_key": "08bf50c8-6ea6-41dc-99ac-5d12a6f517a3",
      "user_id": 1,
      "kp_id": 9,
      "kp_identifier": "sensorsgateway",
      "timestamp": "1373887444356"
    },
    "TemperatureSensor": {
      "GpsCoordinate": {
        "altitude": 0,
        "latitude": 40.512274,
        "longitude": -3.675679
      },
      "identifier": "S_Temperature_00001",
      "measurement": 19,
      "timestamp": 1373887443001,
      "unit": "C"
    }
  },
```

As you can see, the contextData includes the key of the session that the KP established with the SIB; the user identifier used by the KP; the KP's identifier; the identifier of the KP's connected instance, and a time stamp of the moment when the information was added.