User documentation

CIMA

TI5 - 2015 / 2016

Written by
ABDELKADER Malik
DA SILVA Frédéric

Reviewed and corrected by:
DA SILVA Frédéric

Version 1.0

February 5, 2016

Table of contents

1. CIMA layers

The main components of CIMA are the **NSCL** (Network Service Capabilities Layer) layer, the layer **GSCL** (Gateway Service Capabilities Layer), **CIMA web configuration interface** and **device code** (**robot code** in our case) that enables the connected device by internet protocol to expose its capabilities.

a. OM2M layer

CIMA is based on OM2M framework, which has a set of Java and OSGi services easily extensible and customizable. These services are exposed by a RESTful API, providing the primary procedures for machines authentication, resources discovery, applications registration, synchronous and asynchronous communication, access rights permission etc …

b. GSCL layer

The GSCL layer handles device detection and management and expose their capacity in a RESTful way in /infos. Those capacities will be shared with the NSCL layer.

- **Http**: implementation of the http protocol in the gscl layer, in order to make requests
- **Commons** : It includes all utilities to convert the format (JSON-LD, XML, OBIX ...) of the data exchanged (capacities, devices ...)
- **DeviceService** : contains many interfaces which provides all the functions or methods to manage a device based on its ID, address to manage a capacity. Its also provides discovery devices methods.
- **PortForwarding** : This module implements the portforwarding, to manage the applications and allocations of port devices.
- **Core** : specify a controller to perform requests on devices (Execute, Retrieve, Update Delete, Create, Generate a resource on devices)
- **MgmtDevice** : use **DeviceService** to manage device
- **Discovery**: use **DeviceService** to perform discovery
- **ManualConfig** : To manually configure devices

- **HidDiscovery** : NOT USED

c. NSCL layer

Once capabilities are exchanged with the NSCL, its role is to make these devices and capabilities a available via the network to the web interface of CIMA. It also manages profiles, users, devices, capabilities, their persistence in mongo database, their availability and their accessibility via the network

- **Http**: implementation of the http protocol in the NSCL layer, in order to make requests

- **Commons** : It includes all utilities to convert the format (JSON-LD, XML, OBIX ...) of the data exchanged (capacities, devices ...)

- **DeviceService** : contains many interfaces which provides all the functions or methods to manage a device based on its ID, address, subscribers, to manage a capacity. Its also provides an infrastructure controller to control access to the network and the database, but also to manage requests responses.

- **MongoDAO** : This module manage the data persistence.

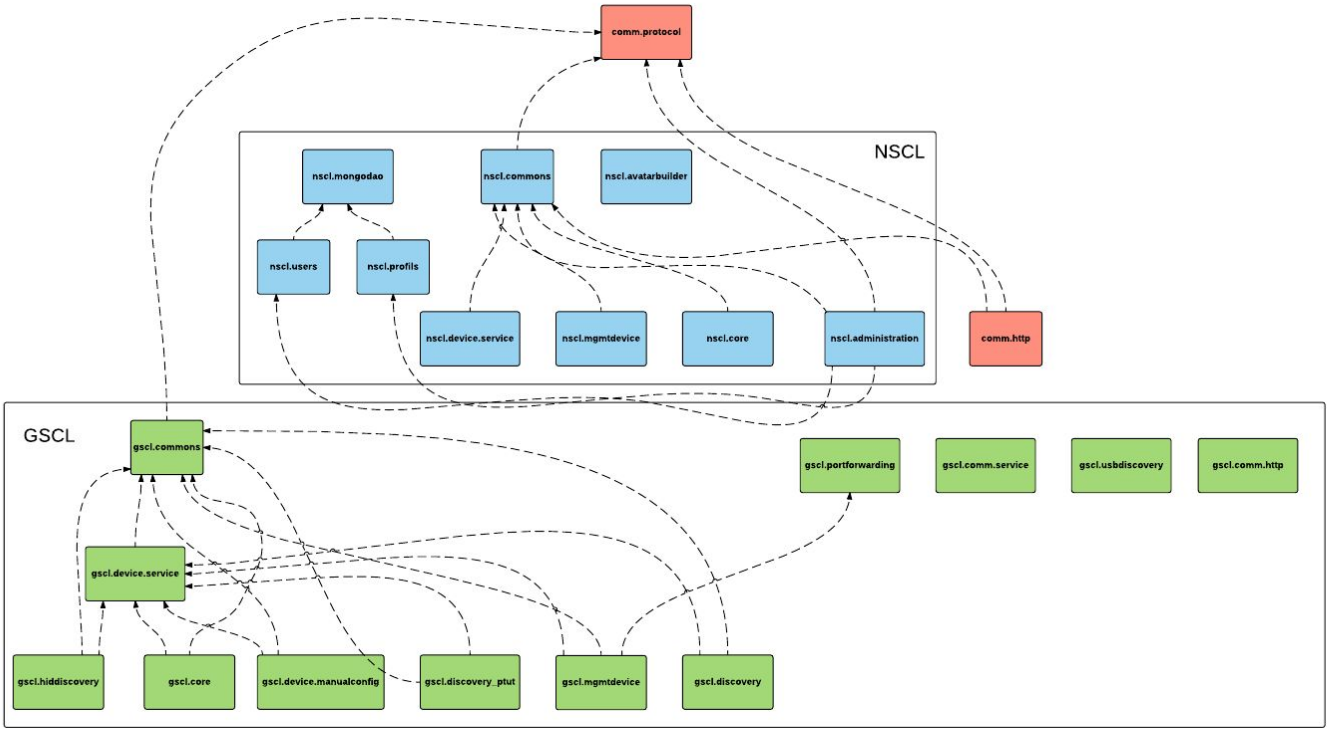
- **Users** : User who want to log into CIMA web interface.

- **Profiles** : The devices profiles.

- **Administration** : The administration Server manages users, profiles. It also defines the behavior of the server and responses to queries GET, POST, DELETE using a well-defined protocol.

- **Core** : specify a controller to perform requests on devices (Execute, Retrieve, Update Delete, Create, Generate a resource on devices)

- **MgmtDevice** : use **DeviceService** to manage device



2. Use cases and integration test

Use case 1 : Authentication

Basic flow of events :

1. Connecting to localhost:8080/ManualConfiguration.html#/login
2. Authentication user with pseudo: “admin” and password: “admin”
3. The interface sends the informations to the NSCL on the URL :
4. The NSCL send back a response to the interface
5. The interface redirect the user to the main panel

Result : **OK**

Use case 2.1 : Connecting intelligent device

Basic flow of events :

1. The user connect his device
2. The GSCL request on /infos to get the XML from the device
3. The GSCL parse the XML to instantiate the class Device in the GSCL
4. The GSCL transform the Device instance to Obix and expose it
5. The NSCL retrieve the Obix from the GSCL and parse it to JSON-LD
6. The NSCL send the JSON-LD to the interface
7. The interface display the device on the device list
8. The user can use the section filters and the differents inputs to search for his device in the device list
9. Click on view button in front of his device

10. The user is redirected to the device page

Result : **OK**

Use case 2.2 : Connecting not intelligent device

Basic flow of events :

1. The GSCL request on /infos to get the XML from the device
2. The GSCL don't find the /infos so it instantiate the Device class from the GSCL with an unknow device
3. The GSCL transform the Device instance to Obix and expose it
4. The NSCL retrieve the Obix from the GSCL
5. The NSCL check in the database if the unknow device has already a profile and match it if it has one
6. The NSCL parse the Obix to JSON-LD
7. The NSCL send the JSON-LD to the interface
8. The interface display the device on the device list

Result : **NOT OK**

TODO - 5 : The interface match the profile with the device by checking the database in the NSCL if it has one, not the NSCL

Use case 2.3 : Apply profil for not intelligent device

Basic flow of events :

1. Click on profil list on the device page
2. Select a profile for the device and click apply
3. The interface send the device id and the profile to the NSCL
4. The NSCL persist the couple in the database
5. The NSCL expose the new device with the new capabilities

Result : **OK**

Use case 2.4 : Disconnecting device

Precondition: Use case 1 is done

Basic flow of events :

1. The user disconnect his device
2. Not verified yet
3. The device is no longer display on the device list

Result : **NOT OK**

Use case 3.1 : Testing capabilities motor

Precondition: Use case 1 and Use case 2 is done

Basic flow of events :

1. Click on show in front of the capability
2. The interface give the detail of the capability and a configure button
3. Click on the configure button and set the different parameters if needed
4. A dialogue will be display for user to enter the value of each parameters (speed, angle)

5. The interface request the NSCL, the NSCL send the request to the GSCL, the GSCL do the test and send back the response to the NSCL which send back the response to the interface

Result : **NOT OK**

TODO - 5 : The interface reach directly the robot capability address and retrieve the response

Use case 3.2 : Testing capabilities sensor

Precondition: Use case 1 and Use case 2 is done

Basic flow of events :

1. Click on show in front of the capability
2. The interface give the detail of the capability and a configure button
3. Click on the configure button
4. Interface ask for test to NSCL, NSCL returns the request to GSCL, GSCL made the test, GSCL send back the answer to NSCL, the NSCL returns the response to the interface.
5. A dialog will be displayed with the value returned by the sensor

Result : **NOT OK**

TODO - 4 : The interface reach directly the robot capability address and retrieve the response

Use case 4 : Portforwarding

Basic flow of events :

1. The GSCL detect the device and parse it to device
2. The port forwarding bundle on the GSCL start a port forwarding for each port in the capabilities
3. The port is add in the capabilities by the GSCL
4. The user retrieve the port of the portforwarding on the device page for each capability
5. The user use the port to contact one or many capabilities of the device

Result : **NOT OK** :

TODO - 2 : one port forwarding is open for each object

TODO - the port is not on the device page yet, it's just on the console

Use case 5.1 : Adding capability

Basic flow of events :

1. The user add a capability from the device
2. Not verified yet

Result : **NOT OK**

Use case 5.2 : Removing capability

Basic flow of events :

1. The user remove a capability from the device
2. Not verified yet

Result : **NOT OK**