Technical and architectural documentation

CIMA

TI5 - 2015 / 2016

Written by
Frédéric DA SILVA

February 5, 2016

This documentation has been written to describe CIMA architecture. In this document, each bundle (and all the classes inside) will be described and bundle dependencies will be detailed. All classes have been comment using Javadoc tools.

fr.liris.cima.comm.protocol

---- **Protocol** : interface to define the use of a protocol, for example
- define protocol's name
- define protocol's parameter by setting up parameter's name and parameter's value
- send message using this protocol

---- **AbstractProtocol** : implementation of Protocol Class
---- **ProtocolResolver** : used to know the list of available protocols in the current instance of CIMA
- this class contains a list of all available protocols
- allows users to add or remove protocols

Used in :
fr.liris.cima.comm.http → this bundle uses **fr.liris.cima.comm.protocol** to create and configure (by setting up parameters like method, port number, uri, body) HTTP protocol, add it to CIMA

fr.liris.cima.gscl.commons → the **parser class** contained in this package use this bundle when creating protocol from xml element or creating protocol from obix element in order to insert it in the capability. This bundle is also used in **Activator class**, to print the list **of available protocol** in GSCL

fr.liris.cima.nscl.administration → the bundle is use in **AdministrationServer.java** and **Activator.java** to provide a protocol which will be used to handle request

fr.liris.cima.nscl.commons → the **parser class** contained in this package use this bundle when converting protocol in JSON or creating protocol from obix element in order to insert it in the capability. This bundle is also used in **Activator class**, to print the list **of available protocol** in GSCL

---

fr.liris.cima.nscl.commons

---- **Constants** : defines various constants for method such as RETRIEVE, UPDATE, CREATE, EXECUTE, for property like **sclBaseId**, **adminRequestingEntity** or **CimaAdress.** It also provide an enum to know if device's configuration is MANUAL or AUTOMATIC

---- **Encoder** : This class encode a given object to json format. It provide many method for example, encode contactInfo, device, list of devices, capability or list of capabilities to a json string.
---- **ContactInfo** : describe the deviceId and the port in the gateway that allow the client to communicate with the device.
---- **Device** :
APOCPATH : Application point of contact for the devices controller
TYPE : Default Device type
List of device capabilities
ContactInfo : Information for contacting the device in the cloudby-passing the OM2M layer
DeviceDescription: A simple device description (id, name, uri, modeConnection, dateConnection)
configuration: Configuration's type of the Device(automatic or manual)

---- **ClientSuscriber** : describe the client who subscribe to NSCL, using his IP and his port.
---- **Parameter** : described by an identifier idP, a type and a description.
---- **Parser** :
- used to parse **String JSON** to **String protocol, capability, device with OBIX structure**
- used to parse **JSON object or String** to **OBIX object capability, protocol, device** ...
- used to parse String **OBIX object or JSON object** to **OBIX object capability, protocol, device**...
- used to parse **String OBIX** to **contactinfo, protocol, device , capability**...
- used to parse **XML object** to a **clientSuscriber**.
- used to parse a **protocol** to a **String with JSON structure**.

---- **UID** : the name of the device, for example DEVICE_01, DEVICE_02
---- **Protocol** : to define a protocol configuration, for example
- protocol's name
- protocol's parameter by setting up parameter's name and parameter's value

---- **Result** : defines the result (a type and a description) send by a device, when get request by NSCL

Used in :
fr.liris.cima.comm.http : used in this bundle to define the reqEntity (org.eclipse.om2m.adminRequestingEntity) property insert in the request indication. This request will be sent, by using a METHOD, an URI, the reqEntity and a base.

fr.liris.cima.nscl.administration : used in this bundle to parse some obix object to JSON string

fr.liris.cima.nscl.core *:* (**infrastructureController**) used in this bundle, to help the infrastructure controller to perform request on devices and to manage client subscriptions,

capabilities... (fr.liris.cima.nscl.commons.Capability ,Device, Constants,Parser, ClientSubscriber,JsonEncoder)

fr.liris.cima.nscl.device.service : (ManagedDeviceService) used by this bundle to manage device service. (**DeviceManagerImpl**) which implements ManagedDeviceService (Device, Constants, ClientSubscriber)

---

fr.liris.cima.comm.http

---- **Http** : http protocol implementation with baseURL, method (GET, PUT, DELETE, POST), uri to contact, port to contact, http body content and http transport protocol (tcp for the moment)
---- **CIMARestHttpClient** : implements **RestClientService**, a REST service from the client side of view. RESTful web service can be accessed and request can be sent using GET, PUT, POST, DELETE...

Used in: Apparently nowhere.

---

fr.liris.cima.nscl.device.service

---- **ManagedDeviceService**: an interface that defines various way to manage device. For example:

- Get a device via id
- Get all devices
- Retrieve a device by address
- Add a device, remove a device
- Remove a device via Id
- Add a subscriber, remove a subscriber, get all subscribers

Used in:
fr.liris.cima.nscl.administration: (Activator)

fr.liris.cima.nscl.core:
(InfrastructureController) used to provide to this bundle the Manageddevice that will be request
(Activator) used to instantiate an InfrastuctureController

fr.liris.cima.nscl.mgmtdevice: used in this bundle, because DeviceManagerImpl implements ManagedDeviceService.
(Activator) used to instantiate an DeviceManagerImpl

---

fr.liris.cima.nscl.mongodao

---- **MongoDaoInterface** : DAO interface to persist object if this object is not persisted yet

---- **Persistable** : An persistable object
---- **PersistableData** : A persistable data
---- **MongoDao** : implements **MongoDaoInterface** to manage data persistence using a dao in mongo data base

Used in:
fr.liris.cima.nscl.profils: is used in this bundle to provide all necessary method to persiste profile data into mongo data base.

fr.liris.cima.nscl.users: is used in this bundle to provide all necessary method to persist profile data into mongo data base.

---

fr.liris.cima.nscl.profils

---- **Profil** : Define the profile class which implements persistable class
---- **ProfileMatching** : This class is responsible for matching profiles and devices
---- **ProfileMatchingManagerInterface** : Also responsible for matching profiles and devices
---- **ProfilManagerInterface** : Interface to manage profile
---- **ProfilManager** : Profile manager implements ProfilManagerInterface
---- **ProfileMatchingManager** : This manager implements ProfileMatchingManagerInterface

Used in:
fr.liris.cima.nscl.administration: is used in this bundle to helps managing profiles for any device and making match between profiles and device in NSCL administration server.

---

fr.liris.cima.nscl.users

---- **User** : Define an user of CIMA as a persistable object so it can be save in mongo data base
---- **UserManagerInterface** : Interface to manage CIMA user
---- **UserManager** : Implements UserManagerInterface in order to manage CIMA user.

Used in:
fr.liris.cima.nscl.administration: is used in this bundle to helps managing users in NSCL administration server.

---

fr.liris.cima.nscl.administration

---- **AdministrationServer:** implements IpuService from OM2M and defines the NSCL server behavior using a protocol resolver and a rest client service. It also defines the way to respond to GET, POST, DELETE method

Used in: ...

---

fr.liris.cima.nscl.core

---- **InfrastructureController** : Specific Infrastructure controller to perform requests on devices, manage clients subscriptions. This class implements ipuService from OM2M.

Used in...

---

fr.liris.cima.nscl.mgmtdevice

---- **DeviceManagerImpl** : this class is the implementation of ManagedDeviceService.

- Get a device via id
- Get all devices
- Retrieve a device by address
- Add a device, remove a device
- Remove a device via Id
- Add a subscriber, remove a subscriber, get all subscribers

Used in...

---

fr.liris.cima.gscl.commons

---- **Constants** : defines various constants, for method such as RETRIEVE, UPDATE, CREATE, EXECUTE, for property like **sclBaseId**,**RequestingEntity** **adminRequestingEntity** or **CimaAdress.** It also provide an URI for unknown device, a command to scan the wlan interface and an IP prefix.
---- **IPFinder** : The IPFinder TimerTask class look for all the available IP that can be found. It iterate in all network interfaces, get the local ip on the network and iterate all relevant ip to know if there is somebody behind. It uses many threads to let timeout on each irrelevant ips.
---- **IPChecker** : Just a class called in differents threads to call the “find” method IPFinder.
---- **IPFinderManager** : Class that manage search of new IPs. It maintain a list of accesibles IPs.
Usage : call startIfNotStarted to start the search : it will automatically search every 5 seconds. Call the accesiblesIP getter to have the list of accesibles IP at the time of calling.
---- **Parser** : for parsing exchanging data

- used to parse **string XML** to a **String OBIX**.
- used to parse **string XML** to a **Device Description, Result** or **Device**.
- used to parse **Element** to a **Capability, Parameter, Protocol**.
- used to parse **string OBIX** to a **Capability, Device**.

---- **PortGenerator** : this class generate a port number between 1050 and 15000
---- **Capability** : this class represent a single capability for a device. A capability is defined by a name, the protocol used by the capability, a list of keywords, a list of parameters, a cloudPort, the type of configuration (AUTOMATIC or MANUAL) and the result returned by the device when it is retrieved or execute.
---- **ContactInfo** : describe the deviceId and the port in the gateway that allow the client to communicate with the device (contact information for the device)
---- **Encoder** : This class encode a given object into xml, obix and json format. It provide many method for example, encode contactInfo, device, list of devices, capability or list of capabilities to a xml, obix and json format...
---- DeviceDescription: A simple device description (id, name, uri, modeConnection, dateConnection)
---- **Device** : a device for CIMA GSCL
APOCPATH : Application point of contact for the devices controller
TYPE : Default Device type
List of device capabilities
ContactInfo : Information for contacting the device in the cloud by-passing the OM2M layer
DeviceDescription: A simple device description (id, name, uri, modeConnection, dateConnection)
configuration: Configuration's type of the Device(automatic or manual)
The device also have various way to say if the device is known or not, to get and set devices capabilities, to get and set contactInfo, to get and set deviceID. A device can be create using a simple deviceDescription.
---- **Parameter** : described by an identifier idP, a type and a description.
---- **UID** : the name of the device, for example DEVICE_01, DEVICE_02
---- **Protocol** : to define the use of a protocol, for example
- define protocol's name
- define protocol's parameter by setting up parameter's name and parameter's value

---- **Result** : defines the result (a type and a description) send by a device, when getting request by GSCL
---- **ExecuteShellComand**: Lookup a connected address in the local network by arp scan.

Used in :
fr.liris.cima.gscl.core : (**DeviceController**) used in this bundle, to help the DeviceController to perform request on devices (execute, retrieve , create, delete, update resources on devices) and to manage device capabilities...
(fr.liris.cima.gscl.commons.Capability ,Device, DeviceDescription, Constants,Parser, Encoder)

fr.liris.cima.gscl.device.service

- (**ConfigManager**) used to provide Capability Class
- (**ManagedDeviceService**) used in this bundle to provide Device Class to be managed.
- (**CapabilityManager**) an interface Capability manager which provide method to manage capabilities in the GSCL

fr.liris.cima.gscl.device.manualconfig (**ConfigManagerImpl** → implementation of ConfigManager) used in this bundle, to provide Capability class to be manually configured. (fr.liris.cima.gscl.commons.Capability ,Constants)

fr.liris.cima.gscl.discovery_ptut (**Discovery_ptut** → implementation of DiscoveryService) Device class and ExecuteShellCommand are used in this class to discover device in the local network. (fr.liris.cima.gscl.commons.Device ,ExecuteShellCommand, Constants, Parser) --- **Not used**

fr.liris.cima.gscl.discovery (DeviceDiscovery → implementation of DiscoveryService) used in this bundle to help discover device in the local network. (fr.liris.cima.gscl.commons.Parser ,Util, Constants)

fr.liris.cima.gscl.hiddiscovery (HidDeviceDiscovery → implementation of DiscoveryService) used in this bundle to help discover device using USB or bluetooth (fr.liris.cima.gscl.commons.Device ,ExecuteShellCommand, Constants, Parser) --- **Not used**

fr.liris.cima.gscl.mgmtdevice

- (ConfigManagerImpl) used to help this bundle in managing device capability.(fr.liris.cima.gscl.commons.Capability)
- (CapabilityManagerImpl → implementation of CapabilityManager) to keep in Map<> all capabilities and to manage it (add, remove, filter)
- (DeviceManagerImpl → implementation of ManagedDeviceService) to managed all device and capabilities

---

fr.liris.cima.gscl.device.service

---- **CapabilityManager** : provide methods to manage capabilities in the GSCL, for example Add an available Capability, Remove a Capability, Give the list of all available Capabilities, Give all capabilities which match with the filter.

---- **DiscoveryDevice** : Discovery a device in the local network.
---- **ConfigManager** : interface which provide method to add, remove or manage capabilities on a device.

---- **ManagedDeviceService** : interface which provide method to add, remove or manage capabilities on a device.
Provide method to

- Get a device via id
- Get a device via address
- Add a device
- Remove a device via Id
- Remove a device
- getDeviceCapabilities(String deviceId);
- getDevices(boolean all);
- getCapabilityDevice(String deviceId, String capabilityName);
- addKnownDevice(Device device);
- removeKnownDevice(Device device);
- getKnownDevice(String deviceId);
- removeUnknownDevice(Device device);
- addUnknownDevice(Device device);
- removeUnknownDeviceById(String id);
- updateUnknonwDevice(String deviceId, Device newDevice);
- updateDevice(String deviceId, Device newDevice);
- switchUnknownToKnownDevice(Device device);
- sendDeviceToNSCL(Device device, RestClientService clientService);
- devicesToObixFormat();
- getUnknownDevice(String deviceId);
- invokeCapability(String deviceId, Capability capability, RestClientService clientService);
- getUnknownDeviceCapabilities(String deviceId);
- capabilitiesToObixFormat(List<Capability> capabilities);
- devicesToObixFormat(List<Device> devices);
- getCapabilityDevice(String deviceId, String capabilityId);
- removeCapabilityDevice(String deviceId, String capabilityId);
- Capability updateDeviceCapability(String deviceId,Capability capability);

Used in :
fr.liris.cima.gscl.core :
(**Activator**) to instantiate a capability manager, a ManagedDeviceService and a DiscoveryService
(**DeviceController**) used in this bundle, to help the DeviceController to manage device capabilities… (fr.liris.cima.gscl.commons.Capability ,Device, DeviceDescription, Constants,Parser, Encoder)

fr.liris.cima.gscl.device.manualconfig:
Used in this bundle, more specifically in **Activator.java** and in **ConfigManagerImpl.java** which is an implementation of ConfigManager interface class.

fr.liris.cima.gscl.discovery_ptut:

Used in this bundle, because of **discovery_ptut.java** which is an implementation of ConfigManager interface class. It helps specifying the discovery service for discovering a device in the local network. Also used in **Activator.java.**

fr.liris.cima.gscl.discovery:
Used in **DeviceDiscovery.java**, to link a DiscoveryService to DeviceDiscovery object.

fr.liris.cima.gscl.mgmtdevice:
CapabilityManagerImpl.java is an implementation of ConfigManager interface class. It helps managing the capabilities of a device. Also used in **Activator.java.**

DeviceManagerImpl.java is an implementation of **ManagedDeviceService** interface class. It helps managing the capabilities of a device. Also used in **Activator.java.**

fr.liris.cima.gscl.hiddiscovery:
Used in **HidDeviceDiscovery.java**, to link a DiscoveryService to HidDeviceDiscovery object.

---

fr.liris.cima.gscl.comm.http

---- **CIMARestHttpClient** : implements RestClientService class, to provide to CIMA, methods to send REST request using HTTP protocol. This class helps converting a standard HTTP status code into a protocol-independent object and also converts a protocol independent parameters into a standard HTTP parameters.

Used in: Nowhere else.

---

fr.liris.cima.gscl.portforwarding

---- **PortForwardingException** : throws an exception if an error occurs during new port forwarding creation.
---- **PortForwardingInterface** : an interface to ask a new portForwarding, to get portForwarding for a device or to add a portForwarding to a device.
---- **PortForwardManager** : this class implements **PortForwardingInterface.java**, and store allocated port and deviceId into a Map<String, Integer>.
---- **PortForwardingProcessLauncher** : instantiate a PortForwardingManager and execute the C++ PortForwarding program specifying the address, the object port and the device ID and start listening in order get the port number. An instance of this class can be initiate by giving the following parameters objectPort, deviceId, address, protocol (0 for TCP and 1 for UDP), portForwardManager.

Used in:
fr.liris.cima.gscl.mgmtdevice: used in **DeviceManagerImpl.java**, to provide a PortForwardingInterface to the Device Manager, so it can allocate a port to any device, which is currently managed.

---

fr.liris.cima.gscl.mgmtdevice

---- **CapabilityManagerImpl.java:** manage in GSCL all the capability avalaible.
---- **DeviceManagerImpl:** manage device. This class can:

- filter devices using their addresses
- add, remove a device
- ask port forwarding for a specified device
- change status (known, unknown) for a device
- get, remove, invoke, update a capability for a device
- convert capability or device into obix format
- send device to NSCL layer using a REST client service

---- **ConfigManagerImpl:** set, update, remove, get capability and store configuration in map.

Used in....

---

fr.liris.cima.gscl.discovery

---- **CIMACommunicationHandler.java:** handle ommunication in CIMA using client socket, data to send, response to get.
---- **CIMAInternalCommunication.java:** This class allows to send information to C server by opening socket connection between gscl and port forwarding part, sending connection data and reading server response.

---- **CIMAReceiver.java:** receive data on specified socket.
---- **DeviceDiscovery.java:** Specific device discover, for discovering a device in the local network. This class uses:

- A rest client service
- A device managed service
- CIMAInternalCommunication class
- a map mapKnownAddresses to store known device
- a map mapConnectedAddresses to store the connected device
- a map mapConfiguredAddresses to store the unknown address for configuration
- a map mapConnectionPortForwarding

- a map mapDisconnectionPortForwarding to store pair port number and a status

This class can:

- Lookup a connected address in the local network by arp scan
- Send a notification to the NSCL for device disconnecting.
- Send notification to Infrastructure controller
- handle new device connection by sending a request for discovery of potential device
- Generates a list of id for the port forwarding section

Used in: ...

---

fr.liris.cima.gscl.core

---- **DeviceController.java:** Specific device controller to perform requests on devices.
getAPOCPath()

Method:

- getAPOCPath() - Returns the implemented Application Point of Contact id
- Executes a resource on devices
- Retrieves a resource on devices.
- Updates a resource on devices.
- Deletes a resource on devices.
- Create a resource on devices
- Generates a list of id for the port forwarding section

Used in ...

---

fr.liris.cima.gscl.hiddiscovery

---- **DeviceFinder.java:** process to find any new device on the local network
---- **HidDeviceDiscovery.java:** Specific device discover, for discovering a device in the local network

---- **HIDFetcher.java:** Retrieve any HID information and delete useless information
---- **Parser.java:** Manages the "depth" of the information, prepare work for XMLizer
---- **XMLizer.java:** Receives the pre-structure code and transforms it into XML
---- **PIUX.java:** ...

Used in ...

---

fr.liris.cima.gscl.comm.service

---- **ICIMAClientService.java:** Rest client interface.

Method to:

- Sends a generic request to handle and receives a generic returned response.
- Returns the communication protocol name

---

fr.liris.cima.gscl.device.manualconfig

---- **ConfigManagerImpl.java:** (**ConfigManagerImpl** → implementation of ConfigManager) used in this bundle, to provide Capability class to be manually configured. (fr.liris.cima.gscl.commons.Capability ,Constants)

Used in : ...

---

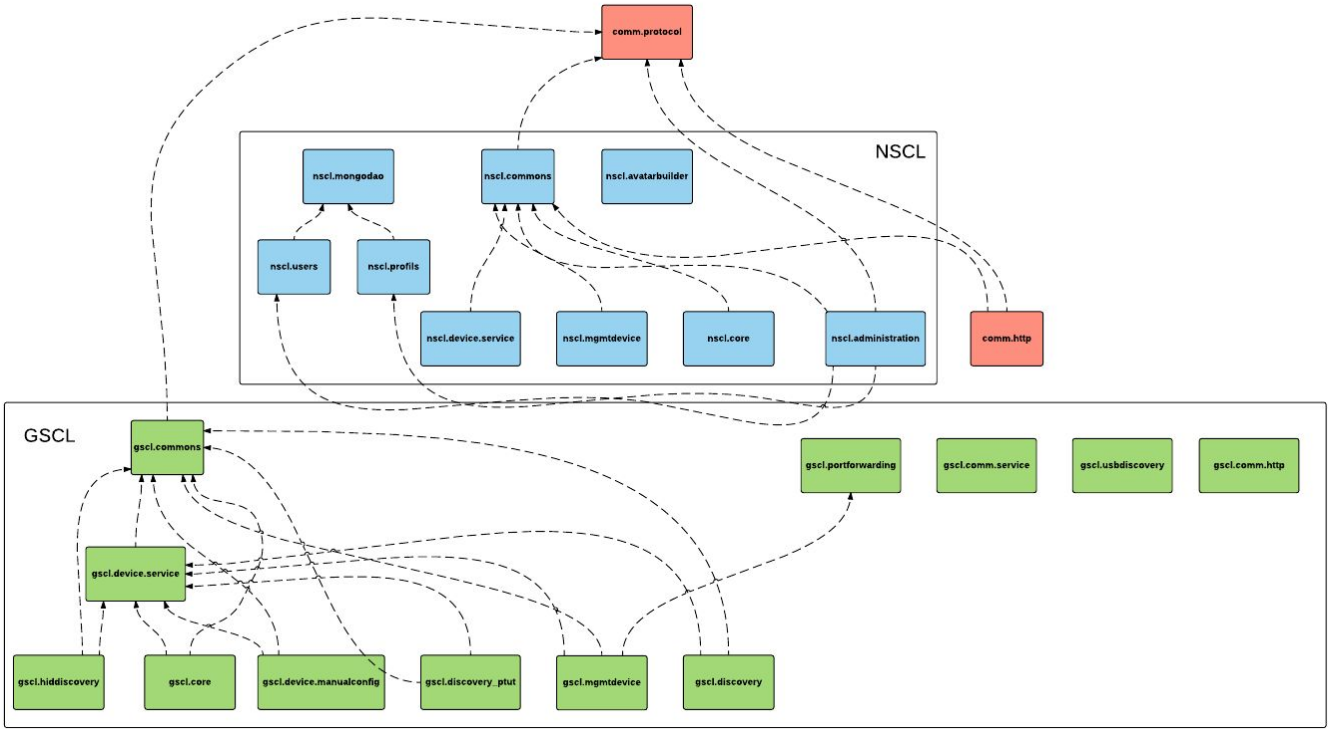
fr.liris.cima.gscl.discovery_ptut

---- **Discovery_ptut.java:** Specific device discover, for discovering a device in the local network

---

fr.liris.cima.gscl.usbdiscovery

Not develop yet.



[INFO] fr.liris.cima :: comm protocol ..... SUCCESS [0.632 s]
[INFO] fr.liris.cima.nscl :: cima nscl nscl ..... SUCCESS [0.021 s]
[INFO] fr.liris.cima.nscl :: nscl commons ..... SUCCESS [0.525 s]
[INFO] fr.liris.cima :: comm http ..... SUCCESS [0.416 s]
[INFO] fr.liris.cima.nscl :: nscl device service ..... SUCCESS [0.252 s]
[INFO] fr.liris.cima.nscl.mongodao :: cima nscl mongodao .. SUCCESS [0.311 s]
[INFO] fr.liris.cima.nscl.profils :: cima nscl profils .... SUCCESS [0.284 s]
[INFO] fr.liris.cima.nscl.users :: cima nscl users ..... SUCCESS [0.299 s]
[INFO] fr.liris.cima.nscl.administration :: cima nscl administration SUCCESS [0.354 s]
[INFO] fr.liris.cima.nscl.avatarbuilder :: cima nscl avatarbuilder SUCCESS [0.462 s]
[INFO] fr.liris.cima.nscl.core :: cima nscl core ..... SUCCESS [0.379 s]
[INFO] fr.liris.cima.nscl.core :: cima nscl mgmtdevice .... SUCCESS [0.354 s]
[INFO] fr.liris.cima.gscl :: cima gscl parent ..... SUCCESS [0.009 s]
[INFO] fr.liris.cima :: gscl commons ..... SUCCESS [0.478 s]
[INFO] fr.liris.cima.gscl :: gscl device service ..... SUCCESS [0.297 s]
[INFO] fr.liris.cima.gscl.core :: cima gscl comm http ..... SUCCESS [0.280 s]
[INFO] fr.liris.cima.gscl.core :: cima gscl portforwarding SUCCESS [0.275 s]
[INFO] fr.liris.cima.gscl.core :: cima gscl mgmtdevice .... SUCCESS [0.258 s]
[INFO] fr.liris.cima.gscl.core :: cima gscl discovery ..... SUCCESS [0.281 s]
[INFO] fr.liris.cima.gscl.core :: cima gscl core ..... SUCCESS [0.371 s]
[INFO] fr.liris.cima.gscl.core :: cima gscl hiddiscovery .. SUCCESS [0.237 s]
[INFO] fr.liris.cima.gscl.core :: cima gscl usbdiscovery .. SUCCESS [0.206 s]
[INFO] fr.liris.cima.gscl :: comm service ..... SUCCESS [0.151 s]
[INFO] fr.liris.cima.gscl.core :: cima gscl discovery_ptut SUCCESS [0.255 s]
[INFO] fr.liris.cima.gscl.core :: cima gscl manualconfig .. SUCCESS [0.222 s]