BEMO-COFRA

Brazil-Europe Monitoring and Control Framework

(Project No. 288133)

D6.1 IoT-enabled legacy devices for production monitoring

Published by the BEMO-COFRA Consortium

Dissemination Level: Public

Project co-funded by the European Commission within the 7th Framework Programme
and
Conselho Nacional de Desenvolvimento Científico e Tecnológico
Objective ICT-2011-EU-Brazil

Document control page

Document file: D6.1_IoT-enabled_legacy_devices_for_production_monitoring.docx
Document version: 1.0
Document owner: Peeter Kool (CNet)

Work package: WP6 – Production Monitoring and Control Systems
Task: T6.1 – Integration of legacy monitoring and control devices
Deliverable type: P

Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of Changes made
0.1	Peeter Kool	2012-08-01	Initial ToC
0.5	Peeter Kool	2012-08-15	Initial content
0.7	Peeter Kool	2012-09-06	Added content
0.8	Peeter Kool	2012-09-26	Added new class documentation
1.0	Peeter Kool	2012-10-01	Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments

Index:

1. Executive summary

This document is delivered with the software deliverable D6.1 IoT-enabled legacy devices for production monitoring. This deliverable documents the prototype and the different software objects and classes used from the BEMO-COFRA platform and developed for this application. It also describes the M12 demo which has been built using the prototype.

2. Introduction

This document is delivered with the software deliverable D6.1 IoT-enabled legacy devices for production monitoring.

The prototype software is used in the M12 demo which is shortly explained in chapter 3 together with a component overview. The following chapter 4 contains documentation of some of the most important classes in the prototype deliverable.

The work has primarily been done in Task 6.1 Integration of legacy monitoring and control devices.

3. Component overview and M12 demo

The main components in this prototype are described in Figure 1. The hardware level contains the drivers for the different hardware devices that are integrated. In the gateway level we find the LinkSmart components such as the Event Manager and also newly developed LinkSmart proxies which interact with the different hardware drivers.

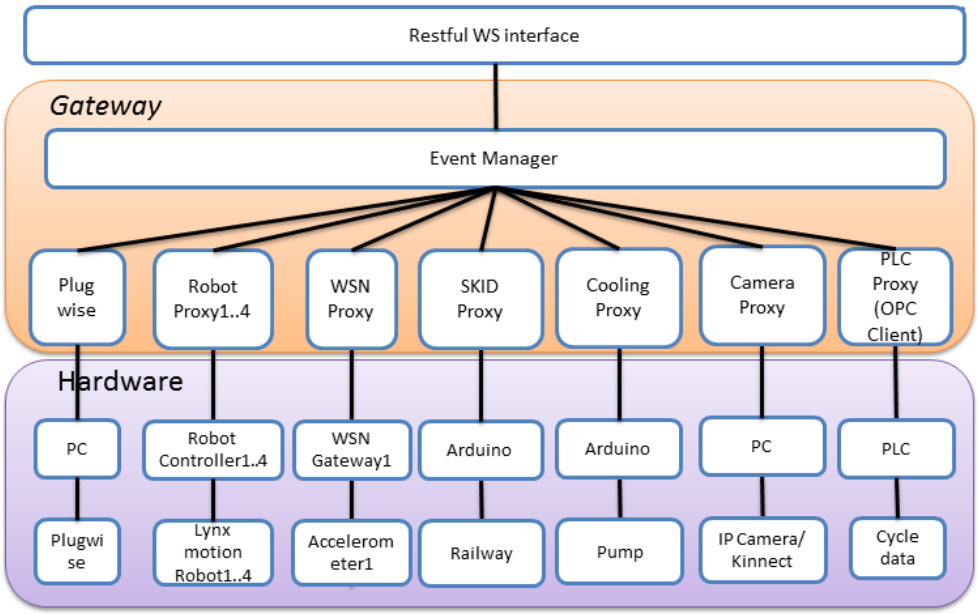


Figure 1: Component overview

The LinkSmart proxies make the devices known in the LinkSmart network and provide services that can be invoked from different applications. Furthermore LinkSmart provides basic communication in the prototype such as service discovery and eventing. The eventing of LinkSmart is a simple publish/subscribe model that is based on topics. More information regarding LinkSmart can be found in http://www.hydramiddleware.eu/news.php (LINKSMART, 2012) and http://sourceforge.net/projects/linksmart/ (LINKSMART2 , 2012).

In order to test these components they have been deployed in the month 12 demo. The M12 demo is based on the scenario in Figure 2.

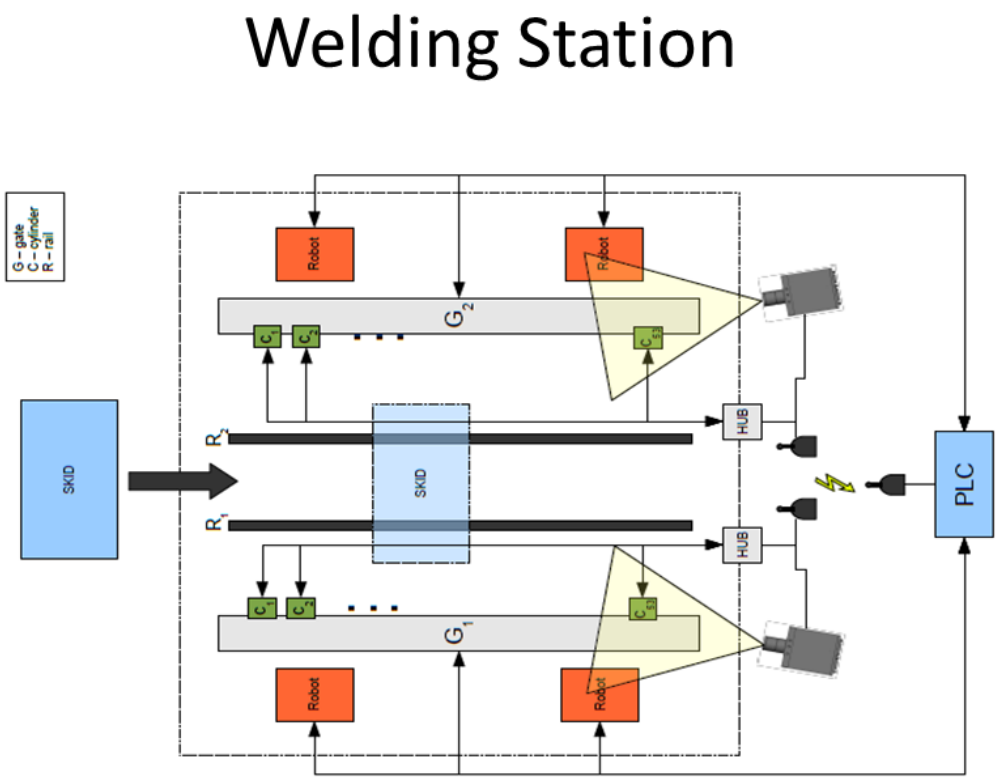


Figure 2: Month 12 Demo scenario

The scenario involves a welding station where car body parts arrive on a skid to be welded. This welding station contains a number of different devices:

- SKID : structure that carries the car body.
- Rail: The SKID moves under two rails that have a specific code for positioning. This code is read while the SKID is sliding in the Cell.
- Gate: A device that locks the SKID and the car body after it is correctly positioned into the Cell. There are two of them to lock the two lateral parts of the car body;
- Cylinder: A cylinder is a component of a clip. The clips are responsible to lock the car body before the robots start the welding task. Each cylinder has two sensors, indicating the status of that cylinder. For example, a cylinder has a sensor indicating an open state and a sensor indicating a closed state. There are 53 cylinders for each gate, or equivalently, 106 sensors
- Robot: The robots that do the actual welding.
- Hub: The state of each sensor is linked through a wire to a HUB composed of 8 devices with 16 inputs. The state of each sensor can be transmitted to a PLC through a wireless link.
- PLC: Device that commands all the operations in the cell, regarding, for example the gates, the movement of SKID and the clips.

The actual M12 demo is a simulation of a subset of the welding station scenario with some devices simulated, see Figure 3.

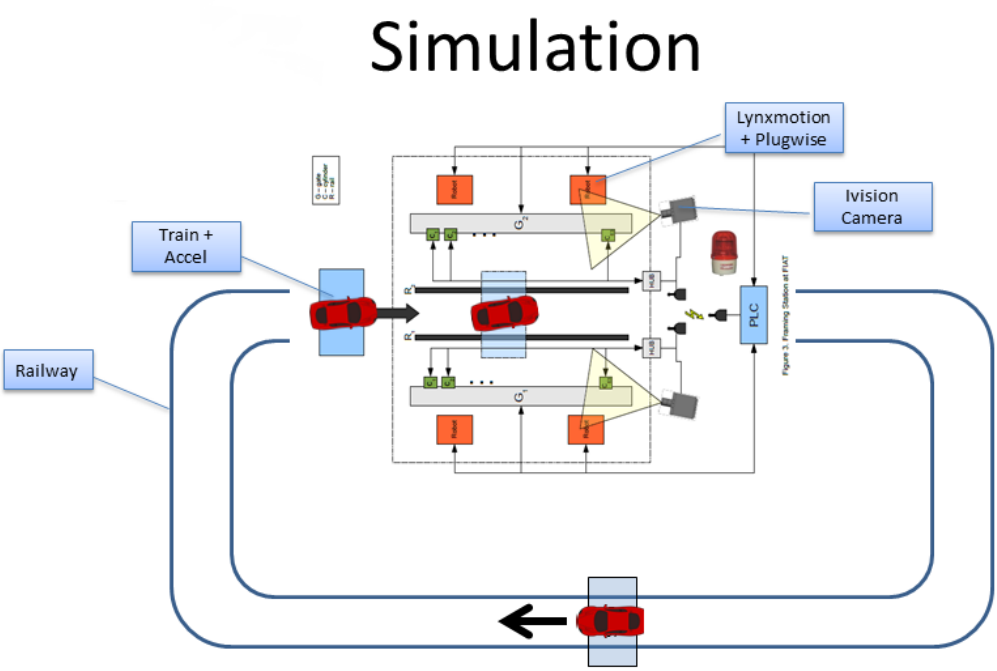


Figure 3: M12 Demo simulation

The SKID is simulated by a train that can be controlled through software. The robots are not actual welding robots but are controllable “hobby” robots. The PLC and the vision system are not simulated but are the same as in the scenario.

The M12 demo uses the following steps

- PLC runs a simplified framing station cycle
  - PLC waits until a car is in the station.
- A car is entering the framing station
  - Use magnetic sensor to identify a car is in the station
- Stop the SKID (train)
  - PLC cycle progresses & waits for position sensors to confirm.
- Sensor checks the position of the body
  - If the position is ok, publish ”event_frame_pos_ok“,
    - Set variable in OPC, PLC runs the cycle further (step progresses).
    - PLC proxy publish „event_start_welding_event” (or the steps)

    - If the position is not ok, publish “event_frame_pos_error”
        - variable in OPC, PLC runs the cycle further.
- Pneumatic gripper(s) lock the SKID
    - Set variable in OPC, Step cycle progresses
- Lynxmotion robots wait for „event_start_welding_event”
    - Robots move for a couple of seconds simulating a welding process
    - Show energy consumption of the robots
- Robots finish the welding process
    - Robot proxies raise an event “event_finished_welding_event” and PLC Proxy set variable in OPC
- Pneumatic gripper(s) Unlock the SKID
- Run the SKID (train) forward until it gets back into the station again.

4. Class Documentation

This section describes some of the more important classes and interfaces defined in the prototype.

4.1 eu.bemocofra.device.railproxy.ArduinoProxyListener Interface Reference

Public Member Functions

- void **updateRFIDReader** (int readerId, String tagId)

Member Function Documentation

void eu.bemocofra.device.railproxy.ArduinoProxyListener.updateRFIDReader (int *readerId*, String *tagId*)

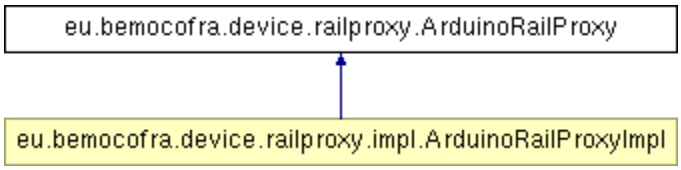
Called when new RFID Tag reading is received from the Arduino

Parameters:

readerId the id of the reader (1 = StartPosition; 2 = Welding Postion)
tagId the id of the tag

4.2 eu.bemocofra.device.railproxy.ArduinoRailProxy Interface Reference

Inheritance diagram for eu.bemocofra.device.railproxy.ArduinoRailProxy:



Public Member Functions

- void **setMotor** (int id, byte speed)
- void **updateRFIDReader** (int readerId, String tagId)
- void **addListener** (**ArduinoProxyListener** listener)

Member Function Documentation

void eu.bemocofra.device.railproxy.ArduinoRailProxy.setMotor (int *id*, byte *speed*)

Parameters:

id

speed (from 0..127)

Implemented in **eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl** (*p.13*).

void eu.bemocofra.device.railproxy.ArduinoRailProxy.updateRFIDReader (int *readerId*, String *tagId*)

Called when new RFID Tag reading is received from the Arduino

Parameters:

readerId the id of the reader ('1' = StartPosition; '2' = Welding Postion)

tagId the id of the tag

Implemented in **eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl** (*p.13*).

void eu.bemocofra.device.railproxy.ArduinoRailProxy.addListener (ArduinoProxyListener *listener*)

Add a **ArduinoProxyListener**

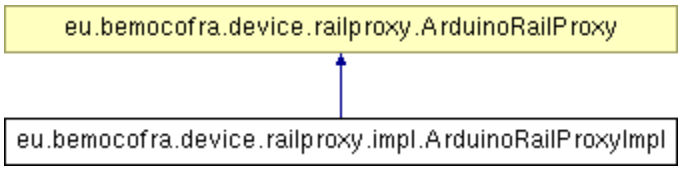
Parameters:

listener the Listener to add

Implemented in **eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl** (*p.13*).

4.3 eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl Class Reference

Inheritance diagram for eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl:



Public Member Functions

- **ArduinoRailProxyImpl** ()
- void **setMotor** (int id, byte speed)
- void **addListener** (**ArduinoProxyListener** listener)
- String **getHID** ()
- void **updateRFIDReader** (int readerId, String tagId)
- void **run** ()

Protected Member Functions

- void **activate** (ComponentContext context)
- void **deactivate** (ComponentContext context)

Package Attributes

- NetworkManagerApplication **networkManager**

Private Attributes

- **ArduinoSerial serial**
- LinkSmartRemoteServiceStore **remoteServiceStore**
- int **lastReader**
- String **lastId**
- **EventPublicationWrapper eventWrapper**
- List< **ArduinoProxyListener** > **listeners** = new LinkedList<**ArduinoProxyListener**>()
- BundleContext **bundleContext**
- String **myHID**

Static Private Attributes

- static final String **SID** = ArduinoRailProxy.class.getSimpleName()
- static final String **EVENT_MANAGER_PID** = "EventManager:BCDEMO"
- static final String **OSGI_SERVICE_HTTP_PORT**
- static final String **AXIS_SERVICES_PATH**
- static final String **ENDPOINT**
- static final String **SERVICE_ID**
- static Logger **LOG**

Detailed Description

Encapsulates communication with an Arduino sensor platform. Provides an interface to request the latest sensor values, set the LED status, and register as an event listener to receive updates about sensor values.

Author:
simon

Constructor & Destructor Documentation

eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.ArduinoRailProxyImpl ()

Creates a new representation of the pre-configured Arduino platform, connected through a serial port

Parameters:
portName the port name, as a String: on Windows, use the COM port name, e.g., "COM3"; on Mac OS X, use the **device** file name, e.g., "/dev/tty.usbserial-A600etcM"

Member Function Documentation

void eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.activate (ComponentContext *context*)
`[protected]`

void eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.deactivate (ComponentContext *context*)
`[protected]`

void eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.setMotor (int *id*, byte *speed*)

Sets the current rail speed.

Parameters:
id the ID of the pump actuator to set
speed the speed level (from 0..127)

Implements **eu.bemocofra.device.railproxy.ArduinoRailProxy** (*p.10*).

void eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.addListener (ArduinoProxyListener *listener*)

Add a ArduinoRailProxyListener

Parameters:
listener the Listener to add

Implements **eu.bemocofra.device.railproxy.ArduinoRailProxy** (*p.11*).

String eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.getHID ()

void eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.updateRFIDReader (int *readerId*, String *tagId*)

Called when new RFID Tag reading is received from the Arduino

Parameters:
readerId the id of the reader ('1' = StartPosition; '2' = Welding Postion)
tagId the id of the tag

Implements **eu.bemocofra.device.railproxy.ArduinoRailProxy** (*p.11*).

void eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.run ()

Member Data Documentation

ArduinoSerial eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.serial [private]

LinkSmartRemoteServiceStore eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.remoteServiceStore [private]

int eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.lastReader [private]

String eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.lastId [private]

EventPublicationWrapper eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.eventWrapper [private]

List<ArduinoProxyListener> eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.listeners = new LinkedList<ArduinoProxyListener>() [private]

final String eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.SID = ArduinoRailProxy.class.getSimpleName() [static, private]

the port name, as a String: on Windows, use the COM port name, e.g., "COM3"; on Mac OS X, use the **device** file name, e.g., "/dev/tty.usbserial-A600etcM"

final String eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.EVENT_MANAGER_PID = "EventManager:BCDEMO" [static, private]

final String eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.OSGI_SERVICE_HTTP_PORT [static, private]

Initial value:
```
System
                        .getProperty("org.osgi.service.http.port")
```

final String eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.AXIS_SERVICES_PATH [static, private]

Initial value:
```
"http://localhost:"
                        + OSGI_SERVICE_HTTP_PORT + "/axis/services/"
```

final String eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.ENDPOINT [static, private]

Initial value:
```
AXIS_SERVICES_PATH
                        + ArduinoRailProxy.class.getSimpleName()
```

final String eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.SERVICE_ID [static, private]

Initial value:
```
ArduinoRailProxyImpl.class
                        .getSimpleName()
```

NetworkManagerApplication eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.networkManager [package]

BundleContext eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.bundleContext [private]

Logger eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.LOG [static, private]

```
Initial value: Logger.getLogger(ArduinoRailProxyImpl.class
                        .getName())
```

String eu.bemocofra.device.railproxy.impl.ArduinoRailProxyImpl.myHID [private]

4.4 eu.bemocofra.device.railproxy.impl.ArduinoSerial Class Reference

Public Member Functions

- void **writeToSerial** (String key, byte value)
- synchronized void **serialEvent** (SerialPortEvent oEvent)
- void **addListener** (**ArduinoSerialListener** listener)

Static Public Member Functions

- static **ArduinoSerial getInstance** (String port)

Protected Member Functions

- void **stopSerialConnection** ()

Package Attributes

- SerialPort **serialPort**

Private Member Functions

- **ArduinoSerial** (String **portName**)
- void **flush** ()
- synchronized void **close** ()
- void **publishSensorEvent** (String key, String value)

Private Attributes

- InputStream **input**
- OutputStream **output**
- String **buffer** = ""
- Set< **ArduinoSerialListener** > **listeners**

Static Private Attributes

- static **ArduinoSerial arduinoSerial**
- static String **portName**
- static final int **TIME_OUT** = 2000
- static final int **DATA_RATE** = 9600

Detailed Description

Arduino sensor/actuator platform. The Arduino board is connected to the serial port, communication is using a text-based **Protocol**.

Constructor & Destructor Documentation

eu.bemocofra.device.railproxy.impl.ArduinoSerial.ArduinoSerial (String *portName*) `[private]`

Constructor. Creates a new **ArduinoSerial** instance using the given port name.

Parameters:

portName

Member Function Documentation

static ArduinoSerial eu.bemocofra.device.railproxy.impl.ArduinoSerial.getInstance (String *port*) `[static]`

void eu.bemocofra.device.railproxy.impl.ArduinoSerial.stopSerialConnection () `[protected]`

void eu.bemocofra.device.railproxy.impl.ArduinoSerial.writeToSerial (String *key*, byte *value*)

Sends a message to the sensor/actuator platform using the **Protocol**.

Parameters:

key the key to send, as a String
value the value to send, as a byte

void eu.bemocofra.device.railproxy.impl.ArduinoSerial.flush () `[private]`

synchronized void eu.bemocofra.device.railproxy.impl.ArduinoSerial.close () `[private]`

This should be called when you stop using the port. This will prevent port locking on platforms like Linux.

synchronized void eu.bemocofra.device.railproxy.impl.ArduinoSerial.serialEvent (SerialPortEvent *oEvent*)

Handle an event on the serial port. Read the data and print it.

void eu.bemocofra.device.railproxy.impl.ArduinoSerial.publishSensorEvent (String *key*, String *value*) `[private]`

Takes the key-value pair and handles it to notify the **listeners**. Called from **serialEvent(SerialPortEvent)**.

Parameters:

key the key of the event, as a String
value the value of the event, as a String

void eu.bemocofra.device.railproxy.impl.ArduinoSerial.addListener (ArduinoSerialListener *listener*)

Adds a new **ArduinoSerialListener** to the **listeners**.

Parameters:

listener the **ArduinoSerialListener** to add.

Member Data Documentation

ArduinoSerial eu.bemocofra.device.railproxy.impl.ArduinoSerial.arduinoSerial `[static, private]`

SerialPort eu.bemocofra.device.railproxy.impl.ArduinoSerial.serialPort `[package]`

String eu.bemocofra.device.railproxy.impl.ArduinoSerial.portName `[static, private]`

InputStream eu.bemocofra.device.railproxy.impl.ArduinoSerial.input `[private]`

Buffered input stream from the port

OutputStream eu.bemocofra.device.railproxy.impl.ArduinoSerial.output `[private]`

The output stream to the port

final int eu.bemocofra.device.railproxy.impl.ArduinoSerial.TIME_OUT = 2000 `[static, private]`

Milliseconds to block while waiting for port open

final int eu.bemocofra.device.railproxy.impl.ArduinoSerial.DATA_RATE = 9600 `[static, private]`

Default bits per second for COM port.

String eu.bemocofra.device.railproxy.impl.ArduinoSerial.buffer = "" `[private]`

Set<ArduinoSerialListener> eu.bemocofra.device.railproxy.impl.ArduinoSerial.listeners `[private]`

The set of **ArduinoSerialListener**s that are notified of new events.

4.5 eu.bemocofra.device.railproxy.impl.ArduinoSerialListener Interface Reference

Public Member Functions

- void **updateRFIDReader** (char readerId, String tagId)

Detailed Description

Interface for sensor event subscribers

Author:

simon

Member Function Documentation

void eu.bemocofra.device.railproxy.impl.ArduinoSerialListener.updateRFIDReader (char *readerId*, String *tagId*)

Called when new RFID Tag reading is received from the Arduino

Parameters:

readerId the id of the reader ('s' = StartPosition; 'w' = Welding Postion)
tagId the id of the tag

4.6 eu.bemocofra.device.railproxy.ConstantSetting Class Reference

Static Public Attributes

- static String **PORT_NAME** = "COM49"

Member Data Documentation

String eu.bemocofra.device.railproxy.ConstantSetting.PORT_NAME = "COM49" `[static]`

4.7 de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo Class Reference

Denavit-Hartenberg Parameters describing one frame.

Public Member Functions

- **DHFrameInfo** (double a, double alpha, double d, double theta)
- double **getA** ()
  Length.
- double **getAlpha** ()
  Twist angle.
- double **getAlphaRadian** ()
  Twist angle in radians.
- double **getD** ()
  Offset.
- double **getTheta** ()
  theta
- double **getThetaRadian** ()
  theta in radians

Private Attributes

- double **mA**
- double **mAlpha**

- double **mAlphaRadian**
- double **mD**
- double **mTheta**
- double **mThetaRadian**

Detailed Description

Denavit-Hartenberg Parameters describing one frame.

Author:

Rainer Hessmer Ported from `http://code.google.com/p/robotic-tic-tac-toe-lynxmotion/`

Constructor & Destructor Documentation

de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.DHFrameInfo (double *a*, double *alpha*, double *d*, double *theta*)

Member Function Documentation

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.getA ()

Length.

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.getAlpha ()

Twist angle.

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.getAlphaRadian ()

Twist angle in radians.

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.getD ()

Offset.

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.getTheta ()

theta

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.getThetaRadian ()

theta in radians

Member Data Documentation

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.mA [private]

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.mAlpha [private]

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.mAlphaRadian [private]

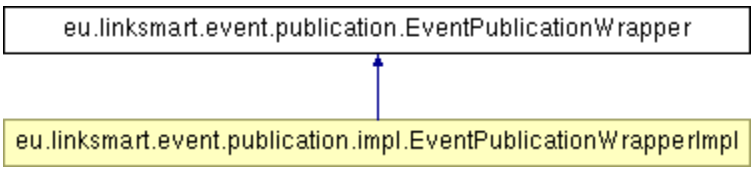
double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.mD [private]

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.mTheta [private]

double de.fraunhofer.fit.robot.kinematics.denavit_hardenberg.DHFrameInfo.mThetaRadian [private]

4.8 eu.linksmart.event.publication.EventPublicationWrapper Interface Reference

Inheritance diagram for eu.linksmart.event.publication.EventPublicationWrapper:



Public Member Functions

- void **findEventManager** (String serviceID, String eventManagerPID)
- boolean **publishEvent** (String serviceID, String topic, Part[] valueParts) throws RemoteException
- boolean **isEventManagerLocated** (String serviceID)

Static Public Attributes

- static final String **DESCRIPTION** = "EventPublisher"
- static final String **SID** = "EventPublisher"

Member Function Documentation

void eu.linksmart.event.publication.EventPublicationWrapper.findEventManager (String *serviceID*, String *eventManagerPID*)

Initiates a search for an EventManager

Parameters:

serviceID service ID of calling component
eventManagerPID name of EventManager to look for

Implemented in **eu.linksmart.event.publication.impl.EventPublicationWrapperImpl** (*p.23*).

boolean eu.linksmart.event.publication.EventPublicationWrapper.publishEvent (String *serviceID*, String *topic*, Part[] *valueParts*) throws RemoteException

Adds a timestamp and publishes an **event** via the EventManager that the calling component has triggered a search before

Parameters:

serviceID service ID of calling component
topic Topic of **event**
valueParts content of **event** without timestamp

Returns:

success of publishing

Exceptions:

RemoteException EventManager could not be found

Implemented in **eu.linksmart.event.publication.impl.EventPublicationWrapperImpl** (*p.22*).

boolean eu.linksmart.event.publication.EventPublicationWrapper.isEventManagerLocated (String *serviceID*)

Indicates if a particular EventManager was found

Parameters:

serviceID service ID of calling component

Returns:

true if the EventManager which was triggered to be looked for by a former call of **findEventManager(String serviceID, String eventManagerPID)** using serviceID was found false otherwise

Implemented in **eu.linksmart.event.publication.impl.EventPublicationWrapperImpl** (*p.23*).

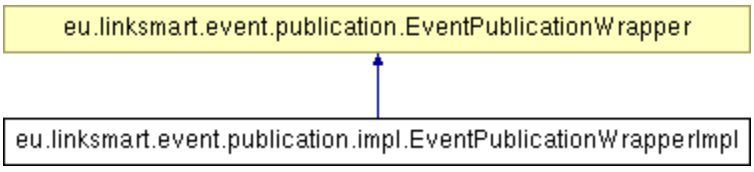
Member Data Documentation

final String eu.linksmart.event.publication.EventPublicationWrapper.DESCRIPTION = "EventPublisher" `[static]`

final String eu.linksmart.event.publication.EventPublicationWrapper.SID = "EventPublisher" `[static]`

4.9 eu.linksmart.event.publication.impl.EventPublicationWrapperImpl Class Reference

Inheritance diagram for eu.linksmart.event.publication.impl.EventPublicationWrapperImpl:



Public Member Functions

- boolean **publishEvent** (String serviceID, String topic, Part[] valueParts) throws RemoteException
- void **findEventManager** (String serviceID, String eventManagerPID)
- boolean **isEventManagerLocated** (String serviceID)

Static Public Attributes

- static final String **DESCRIPTION** = "EventPublisher"
- static final String **SID** = "EventPublisher"

Protected Member Functions

- void **activate** (ComponentContext context)
- void **deactivate** (ComponentContext context)
- EventManagerPort **getEventManager** (String serviceID)

Protected Attributes

- SimpleDateFormat **sdf** = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.S")

Private Attributes

- Logger **LOG** = Logger.getLogger(EventPublicationWrapperImpl.class.getName())
- LinkSmartRemoteServiceStore **remoteServiceStore**
- **EventManagerLocatorThread locatorThread**
- Map< String, EventManagerPort > **eventManagers**

Classes

- class **EventManagerLocatorThread**

Member Function Documentation

void eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.activate (ComponentContext *context*) `[protected]`

void eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.deactivate (ComponentContext *context*) `[protected]`

boolean eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.publishEvent (String *serviceID*, String *topic*, Part[] *valueParts*) throws RemoteException

Adds a timestamp and publishes an **event** via the EventManager that the calling component has triggered a search before

Parameters:

serviceID service ID of calling component
topic Topic of **event**
valueParts content of **event** without timestamp

Returns:

success of publishing

Exceptions:

RemoteException EventManager could not be found

Implements **eu.linksmart.event.publication.EventPublicationWrapper** (*p.21*).

void eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.findEventManager (String *serviceID*, String *eventManagerPID*)

Initiates a search for an EventManager

Parameters:

serviceID service ID of calling component
eventManagerPID name of EventManager to look for

Implements **eu.linksmart.event.publication.EventPublicationWrapper** (*p.20*).

EventManagerPort eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.getEventManager (String *serviceID*) `[protected]`

boolean eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.isEventManagerLocated (String *serviceID*)

Indicates if a particular EventManager was found

Parameters:

serviceID service ID of calling component

Returns:

true if the EventManager which was triggered to be looked for by a former call of **findEventManager(String serviceID, String eventManagerPID)** using serviceID was found false otherwise

Implements **eu.linksmart.event.publication.EventPublicationWrapper** (*p.21*).

Member Data Documentation

Logger eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.LOG = Logger.getLogger(EventPublicationWrapperImpl.class.getName()) [private]

SimpleDateFormat eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.S") [protected]

LinkSmartRemoteServiceStore eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.remoteServiceStore [private]

EventManagerLocatorThread eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.locatorThread [private]

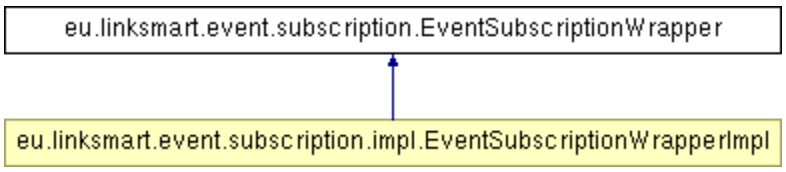
Map<String, EventManagerPort> eu.linksmart.event.publication.impl.EventPublicationWrapperImpl.eventManagers [private]

final String eu.linksmart.event.publication.EventPublicationWrapper.DESCRIPTION = "EventPublisher" [static, inherited]

final String eu.linksmart.event.publication.EventPublicationWrapper.SID = "EventPublisher" [static, inherited]

4.10 eu.linksmart.event.subscription.EventSubscriptionWrapper Interface Reference

Inheritance diagram for eu.linksmart.event.subscription.EventSubscriptionWrapper:



Public Member Functions

- void **findEventManager** (String serviceID, String eventManagerPID)
- void **registerCallback** (EventSubscriber subscriber, String serviceID)
- void **deregisterCallback** (String serviceID)
- void **subscribeWithTopic** (String serviceID, String topic)
- void **unsubscribeTopic** (String serviceID, String topic)
- void **unsubscribeAllTopics** (String serviceID)

Static Public Attributes

- static final String **DESCRIPTION** = "EventSubscriber"
- static final String **SID** = "EventSubscriber"

Member Function Documentation

void eu.linksmart.event.subscription.EventSubscriptionWrapper.findEventManager (String *serviceID*, String *eventManagerPID*)

Initiates a search for an EventManager

Parameters:

serviceID service ID of calling component
eventManagerPID name of EventManager to look for

Implemented in **eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl** (*p.28*).

void eu.linksmart.event.subscription.EventSubscriptionWrapper.registerCallback (EventSubscriber *subscriber*, String *serviceID*)

Registers a subscriber as service at the NetworkManager so that this component can be notified by the EventManager

Parameters:

subscriber Component which shall be notified
serviceID service ID of calling component

Implemented in **eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl** (*p.27*).

void eu.linksmart.event.subscription.EventSubscriptionWrapper.deregisterCallback (String *serviceID*)

Deregisters service from NetworkManager

Parameters:

serviceID service ID of calling component

Implemented in **eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl** (*p.27*).

void eu.linksmart.event.subscription.EventSubscriptionWrapper.subscribeWithTopic (String *serviceID*, String *topic*)

Subscribe to a certain topic at the EventManager which was triggered to be looked for by a former call of **findEventManager(String serviceID, String eventManagerPID)** The EventManager notifies the component which was register by a former call of **registerCallback(EventSubscriber subscriber, String serviceID)**

Parameters:

serviceID service ID of calling component
topic topic to subscribe to

Implemented in **eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl** (*p.28*).

void eu.linksmart.event.subscription.EventSubscriptionWrapper.unsubscribeTopic (String *serviceID*, String *topic*)

Un-Subscribe a certain topic at the EventManager which was subscribed by a former call ofsubscribeWithTopic(String serviceID, String topic) The EventManager will delete the **subscription** and will not notify the serviceID

Parameters:

serviceID service ID of subscriber
topic topic to un-subscribe to

Implemented in **eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl** (*p.28*).

void eu.linksmart.event.subscription.EventSubscriptionWrapper.unsubscribeAllTopics (String *serviceID*)

Un-Subscribe all topics at the EventManager which was subscribed by a former call ofsubscribeWithTopic(String serviceID, String topic) with the same serviceID The EventManager will delete all subscriptions and will not notify the serviceID

Parameters:

serviceID service ID of calling component

Implemented in **eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl** (*p.28*).

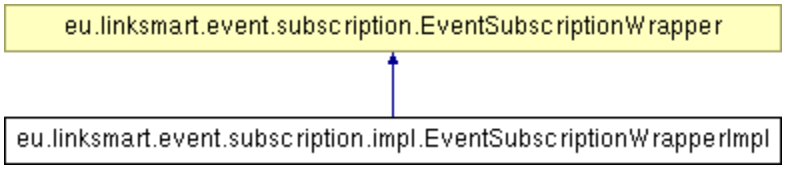
Member Data Documentation

final String eu.linksmart.event.subscription.EventSubscriptionWrapper.DESCRIPTION = "EventSubscriber" [static]

final String eu.linksmart.event.subscription.EventSubscriptionWrapper.SID = "EventSubscriber" [static]

4.11 eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl Class Reference

Inheritance diagram for eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl:



Public Member Functions

- void **registerCallback** (EventSubscriber subscriber, String serviceID)
- void **deregisterCallback** (String serviceID)
- void **findEventManager** (String serviceID, String eventManagerPID)
- synchronized void **subscribeWithTopic** (String serviceID, String topic)
- void **unsubscribeAllTopics** (String serviceID)
- void **unsubscribeTopic** (String serviceID, String topic)

Static Public Attributes

- static final String **DESCRIPTION** = "EventSubscriber"
- static final String **SID** = "EventSubscriber"

Protected Member Functions

- void **activate** (ComponentContext **context**)
- void **deactivate** (ComponentContext **context**)

Private Attributes

- Logger **LOG** = Logger.getLogger(EventSubscriptionWrapperImpl.class.getName())
- ComponentContext **context**
- LinkSmartRemoteServiceStore **remoteServiceStore**
- **EventManagerQueuedSubscriberThread subscriberThread**
- **EventManagerLocatorThread locatorThread**
- NetworkManagerApplication **networkManager**
- Map< String, EventManagerPort > **eventManagers**
- Map< String, String > **subscriberHIDs**
- Map< String, String > **topicIDs**
- LinkedList< String > **queuedTopicToSubscribe** = new LinkedList<String>()
- List< String > **subscribedTopics** = new ArrayList<String>()

Static Private Attributes

- static final String **OSGI_SERVICE_HTTP_PORT**
- static final String **AXIS_SERVICES_PATH**

Classes

- class **EventManagerLocatorThread**
- class **EventManagerQueuedSubscriberThread**

Member Function Documentation

void eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.activate (ComponentContext *context*) `[protected]`

void eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.deactivate (ComponentContext *context*) `[protected]`

void eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.registerCallback (EventSubscriber *subscriber*, String *serviceID*)

Registers a subscriber as service at the NetworkManager so that this component can be notified by the EventManager

Parameters:

subscriber Component which shall be notified
serviceID service ID of calling component

Implements **eu.linksmart.event.subscription.EventSubscriptionWrapper** (*p.25*).

void eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.deregisterCallback (String *serviceID*)

Deregisters service from NetworkManager

Parameters:

serviceID service ID of calling component

Implements **eu.linksmart.event.subscription.EventSubscriptionWrapper** (*p.25*).

void eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.findEventManager (String *serviceID*, String *eventManagerPID*)

Initiates a search for an EventManager

Parameters:

serviceID service ID of calling component
eventManagerPID name of EventManager to look for

Implements **eu.linksmart.event.subscription.EventSubscriptionWrapper** (*p.25*).

synchronized void eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.subscribeWithTopic (String *serviceID*, String *topic*)

Subscribe to a certain topic at the EventManager which was triggered to be looked for by a former call of **findEventManager(String serviceID, String eventManagerPID)** The EventManager notifies the component which was register by a former call of **registerCallback(EventSubscriber subscriber, String serviceID)**

Parameters:

serviceID service ID of calling component
topic topic to subscribe to

Implements **eu.linksmart.event.subscription.EventSubscriptionWrapper** (*p.25*).

void eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.unsubscribeAllTopics (String *serviceID*)

Un-Subscribe all topics at the EventManager which was subscribed by a former call ofsubscribeWithTopic(String serviceID, String topic) with the same serviceID The EventManager will delete all subscriptions and will not notify the serviceID

Parameters:

serviceID service ID of calling component

Implements **eu.linksmart.event.subscription.EventSubscriptionWrapper** (*p.26*).

void eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.unsubscribeTopic (String *serviceID*, String *topic*)

Un-Subscribe a certain topic at the EventManager which was subscribed by a former call ofsubscribeWithTopic(String serviceID, String topic) The EventManager will delete the **subscription** and will not notify the serviceID

Parameters:

serviceID service ID of subscriber
topic topic to un-subscribe to

Implements **eu.linksmart.event.subscription.EventSubscriptionWrapper** (*p.25*).

Member Data Documentation

final String eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.OSGI_SERVICE_HTTP_PORT
`[static, private]`

```
Initial value: System
                        .getProperty("org.osgi.service.http.port")
```

final String eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.AXIS_SERVICES_PATH [static, private]

```
Initial value: "http://localhost:"
                        + OSGI_SERVICE_HTTP_PORT + "/axis/services/"
```

Logger eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.LOG = Logger.getLogger(EventSubscriptionWrapperImpl.class.getName()) [private]

ComponentContext eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.context [private]

LinkSmartRemoteServiceStore eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.remoteServiceStore [private]

EventManagerQueuedSubscriberThread eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.subscriberThread [private]

EventManagerLocatorThread eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.locatorThread [private]

NetworkManagerApplication eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.networkManager [private]

Map<String, EventManagerPort> eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.eventManagers [private]

Map<String, String> eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.subscriberHIDs [private]

Map<String, String> eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.topicIDs [private]

LinkedList<String> eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.queuedTopicToSubscribe = new LinkedList<String>() [private]

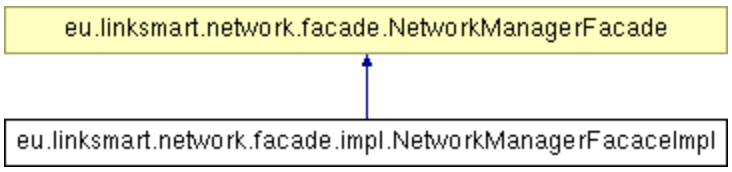
List<String> eu.linksmart.event.subscription.impl.EventSubscriptionWrapperImpl.subscribedTopics = new ArrayList<String>() [private]

final String eu.linksmart.event.subscription.EventSubscriptionWrapper.DESCRIPTION = "EventSubscriber" [static, inherited]

final String eu.linksmart.event.subscription.EventSubscriptionWrapper.SID = "EventSubscriber" [static, inherited]

4.12 eu.linksmart.network.facade.impl.NetworkManagerFacaceImpl Class Reference

Inheritance diagram for eu.linksmart.network.facade.impl.NetworkManagerFacaceImpl:



Public Member Functions

- String **createHIDForServiceID** (String serviceID) throws RemoteException

Protected Member Functions

- void **activate** (ComponentContext context)
- void **deactivate** (ComponentContext context)

Static Protected Attributes

- static final String **OSGI_SERVICE_HTTP_PORT**
- static final String **AXIS_SERVICES_PATH**

Private Attributes

- Logger **LOG** = Logger.getLogger(NetworkManagerFacaceImpl.class)
- NetworkManagerApplication **networkManager**

Member Function Documentation

void eu.linksmart.network.facade.impl.NetworkManagerFacaceImpl.activate (ComponentContext *context*)
`[protected]`

void eu.linksmart.network.facade.impl.NetworkManagerFacaceImpl.deactivate (ComponentContext *context*)
`[protected]`

String eu.linksmart.network.facade.impl.NetworkManagerFacaceImpl.createHIDForServiceID (String *serviceID*) throws RemoteException

Registers the service for the given serviceID at the local Network Manager. An HID for this service will be created.

Parameters:

serviceID ID of the service to be registered. E.g. MyServiceInterface.class.getSimpleName()

Returns:

The HID under which the service can be accessed.

Exceptions:

Exception

Implements **eu.linksmart.network.facade.NetworkManagerFacade**

Member Data Documentation

final String eu.linksmart.network.facade.impl.NetworkManagerFacaceImpl.OSGI_SERVICE_HTTP_PORT [static, protected]

```
Initial value: System
                        .getProperty("org.osgi.service.http.port")
```

final String eu.linksmart.network.facade.impl.NetworkManagerFacaceImpl.AXIS_SERVICES_PATH [static, protected]

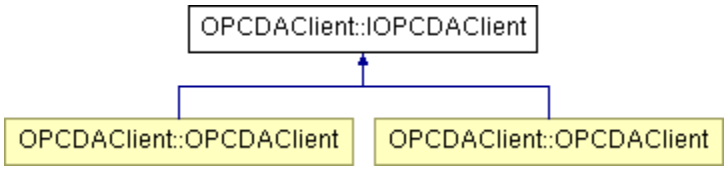
```
Initial value: "http://localhost:"
                        + OSGI_SERVICE_HTTP_PORT + "/axis/services/"
```

Logger eu.linksmart.network.facade.impl.NetworkManagerFacaceImpl.LOG = Logger.getLogger(NetworkManagerFacaceImpl.class) [private]

NetworkManagerApplication eu.linksmart.network.facade.impl.NetworkManagerFacaceImpl.networkManager [private]

4.13 OPCDAClient::IOPCDAClient Interface Reference

Inheritance diagram for OPCDAClient::IOPCDAClient:



Public Member Functions

- bool **connect** ()
  connect to server
- bool **disconnect** ()
  disconnect from server
- bool **addGroup** (string name, int updateRate, float percentDeadband)
  add a query group to a (connected) server
- bool **removeGroup** (string name)
  remove a group from connected server
- List< string > **getGroups** ()
  get added groups
- bool **getGroupStatus** (string name, out int updateRate, out float percentDeadband)
  get group status
- bool **addVariables** (string groupName, int[] varIds, string[] opcItemIds, out **OPCError**[] errors)
  add variables to a group
- bool **removeVariables** (string groupName, int[] varIds, string[] opcItemIds, out **OPCError**[] errors)
  remove variables from a group

- void **setCallback** (**IOPCDAClientCallback** cb)
  set a callback
- void **resetCallback** (**IOPCDAClientCallback** cb)
  remove a callback
- bool **read** (int varId, out **OPCValue** val)
  synchronous read
- bool **write** (int varId, object value)
  synchronous write
- bool **connect** ()
  connect to server
- bool **disconnect** ()
  disconnect from server
- bool **addGroup** (string name, int updateRate, float percentDeadband)
  add a query group to a (connected) server
- bool **removeGroup** (string name)
  remove a group from connected server
- List< string > **getGroups** ()
  get added groups
- bool **getGroupStatus** (string name, out int updateRate, out float percentDeadband)
  get group status
- bool **addVariables** (string groupName, int[] varIds, string[] opcItemIds, out **OPCError**[] errors)
  add variables to a group
- bool **removeVariables** (string groupName, int[] varIds, string[] opcItemIds, out **OPCError**[] errors)
  remove variables from a group
- void **setCallback** (**IOPCDAClientCallback** cb)
  set a callback
- void **resetCallback** (**IOPCDAClientCallback** cb)
  remove a callback
- bool **read** (int varId, out **OPCValue** val)
  synchronous read
- bool **write** (int varId, object value)
  synchronous write

Member Function Documentation

bool OPCDAClient::IOPCDAClient::connect ()

connect to server

Returns:

bool OPCDAClient::IOPCDAClient::disconnect ()

disconnect from server

Returns:

bool OPCDAClient::IOPCDAClient::addGroup (string *name*, int *updateRate*, float *percentDeadband*)

add a query group to a (connected) server

Parameters:

name unique group name
updateRate update rate in milliseconds
percentDeadband percent of dead band

Returns:

true: the group is added; false: failed (same group exists; server disconnected; ...)

bool OPCDAClient::IOPCDAClient::removeGroup (string *name*)

remove a group from connected server

Parameters:

name

Returns:

List<string> OPCDAClient::IOPCDAClient::getGroups ()

get added groups

Returns:

bool OPCDAClient::IOPCDAClient::getGroupStatus (string *name*, out int *updateRate*, out float *percentDeadband*)

get group status

Parameters:

name
updateRate
percentDeadband

Returns:

bool OPCDAClient::IOPCDAClient::addVariables (string *groupName*, int[] *varIds*, string[] *opcItemIds*, out OPCError[] *errors*)

add variables to a group

Parameters:

groupName
varIds unique id
opcItemIds **OPC** item id array
errors error array correspondent to each var id in varIds[]

Returns:

bool OPCDAClient::IOPCDAClient::removeVariables (string *groupName*, int[] *varIds*, string[] *opcItemIds*, out OPCError[] *errors*)

remove variables from a group

Parameters:

groupName
varIds
opcItemIds
errors

Returns:

void OPCDAClient::IOPCDAClient::setCallback (IOPCDAClientCallback *cb*)

set a callback

Parameters:

cb

void OPCDAClient::IOPCDAClient::resetCallback (IOPCDAClientCallback *cb*)

remove a callback

Parameters:

cb

bool OPCDAClient::IOPCDAClient::read (int *varId*, out OPCValue *val*)

synchronous read

Parameters:

varId

Returns:

bool OPCDAClient::IOPCDAClient::write (int *varId*, object *value*)

synchronous write

Parameters:

varId
value

Returns:

bool OPCDAClient::IOPCDAClient::connect ()

connect to server

Returns:

bool OPCDAClient::IOPCDAClient::disconnect ()

disconnect from server

Returns:

bool OPCDAClient::IOPCDAClient::addGroup (string *name*, int *updateRate*, float *percentDeadband*)

add a query group to a (connected) server

Parameters:

name unique group name
updateRate update rate in milliseconds
percentDeadband percent of dead band

Returns:

true: the group is added; false: failed (same group exists; server disconnected; ...)

bool OPCDAClient::IOPCDAClient::removeGroup (string *name*)

remove a group from connected server

Parameters:

name

Returns:

List<string> OPCDAClient::IOPCDAClient::getGroups ()

get added groups

Returns:

bool OPCDAClient::IOPCDAClient::getGroupStatus (string *name*, out int *updateRate*, out float *percentDeadband*)

get group status

Parameters:

name
updateRate
percentDeadband

Returns:

bool OPCDAClient::IOPCDAClient::addVariables (string *groupName*, int[] *varIds*, string[] *opcItemIds*, out OPCError[] *errors*)

add variables to a group

Parameters:

groupName
varIds unique id
opcItemIds **OPC** item id array
errors error array correspondent to each var id in varIds[]

Returns:

bool OPCDAClient::IOPCDAClient::removeVariables (string *groupName*, int[] *varIds*, string[] *opcItemIds*, out OPCError[] *errors*)

remove variables from a group

Parameters:

groupName
varIds
opcItemIds
errors

Returns:

void OPCDAClient::IOPCDAClient::setCallback (IOPCDAClientCallback *cb*)

set a callback

Parameters:

cb

void OPCDAClient::IOPCDAClient::resetCallback (IOPCDAClientCallback *cb*)

remove a callback

Parameters:

cb

bool OPCDAClient::IOPCDAClient::read (int *varId*, out OPCValue *val*)

synchronous read

Parameters:

varId

Returns:

bool OPCDAClient::IOPCDAClient::write (int *varId*, object *value*)

synchronous write

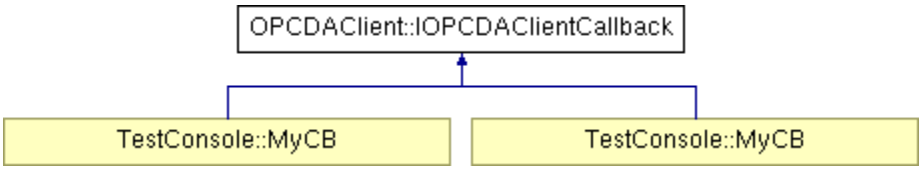
Parameters:

varId
value

Returns:

4.14 OPCDAClient::IOPCDAClientCallback Interface Reference

Inheritance diagram for OPCDAClient::IOPCDAClientCallback:



Public Member Functions

- void **onDataChange** (string groupName, **OPCValueQuality** masterQuality, **OPCError** masterError, **OPCValue**[] values, **OPCError**[] valueErrors)
  Data changed notification.
- void **onDisconnection** (**OPCDisconnectionReason** reason, string serverShutdownReason)
  Called when the server is unexpectedly disconnected: 1) by server shutdown; 2) by connection lost.
- void **onDataChange** (string groupName, **OPCValueQuality** masterQuality, **OPCError** masterError, **OPCValue**[] values, **OPCError**[] valueErrors)
  Data changed notification.
- void **onDisconnection** (**OPCDisconnectionReason** reason, string serverShutdownReason)
  Called when the server is unexpectedly disconnected: 1) by server shutdown; 2) by connection lost.

Member Function Documentation

void OPCDAClient::IOPCDAClientCallback::onDataChange (string *groupName*, OPCValueQuality *masterQuality*, OPCError *masterError*, OPCValue[] *values*, OPCError[] *valueErrors*)

Data changed notification.

Parameters:

groupName
masterQuality communication quality for this notification
masterError Error of this notification
values Values notified
valueErrors Error of each value notified

Implemented in **TestConsole::MyCB** (*p.40*), and **TestConsole::MyCB** (*p.40*).

void OPCDAClient::IOPCDAClientCallback::onDisconnection (OPCDisconnectionReason *reason*, string *serverShutdownReason*)

Called when the server is unexpectedly disconnected: 1) by server shutdown; 2) by connection lost.

Parameters:

reason

serverShutdownReason Server shutdown message

Implemented in **TestConsole::MyCB** (*p.40*), and **TestConsole::MyCB** (*p.40*).

void OPCDAClient::IOPCDAClientCallback::onDataChange (string *groupName*, OPCValueQuality *masterQuality*, OPCError *masterError*, OPCValue[] *values*, OPCError[] *valueErrors*)

Data changed notification.

Parameters:

groupName
masterQuality communication quality for this notification
masterError Error of this notification
values Values notified
valueErrors Error of each value notified

Implemented in **TestConsole::MyCB** (*p.40*), and **TestConsole::MyCB** (*p.40*).

void OPCDAClient::IOPCDAClientCallback::onDisconnection (OPCDisconnectionReason *reason*, string *serverShutdownReason*)

Called when the server is unexpectedly disconnected: 1) by server shutdown; 2) by connection lost.

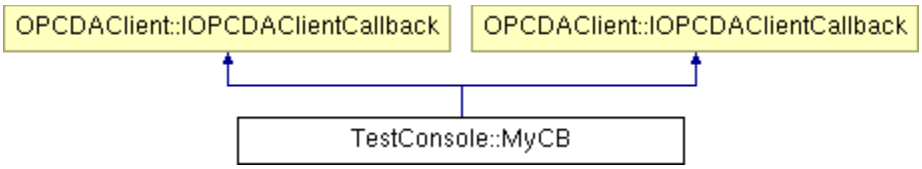
Parameters:

reason
serverShutdownReason Server shutdown message

Implemented in **TestConsole::MyCB** (*p.40*), and **TestConsole::MyCB** (*p.40*).

4.15 TestConsole::MyCB Class Reference

Inheritance diagram for TestConsole::MyCB:



Public Member Functions

- void **onDataChange** (string groupName, **OPCValueQuality** masterQuality, **OPCError** masterError, **OPCValue**[] values, **OPCError**[] valueErrors)
  Data changed notification.
- void **onDisconnection** (**OPCDisconnectionReason** reason, string serverShutdownReason)
  Called when the server is unexpectedly disconnected: 1) by server shutdown; 2) by connection lost.
- void **onDataChange** (string groupName, **OPCValueQuality** masterQuality, **OPCError** masterError, **OPCValue**[] values, **OPCError**[] valueErrors)
  Data changed notification.

- void **onDisconnection** (**OPCDisconnectionReason** reason, string serverShutdownReason)
  Called when the server is unexpectedly disconnected: 1) by server shutdown; 2) by connection lost.

Member Function Documentation

void TestConsole::MyCB::onDataChange (string *groupName*, OPCValueQuality *masterQuality*, OPCError *masterError*, OPCValue[] *values*, OPCError[] *valueErrors*) `[inline]`

Data changed notification.

Parameters:

groupName
masterQuality communication quality for this notification
masterError Error of this notification
values Values notified
valueErrors Error of each value notified

Implements **OPCDAClient::IOPCDAClientCallback** (*p.39*).

void TestConsole::MyCB::onDisconnection (OPCDisconnectionReason *reason*, string *serverShutdownReason*) `[inline]`

Called when the server is unexpectedly disconnected: 1) by server shutdown; 2) by connection lost.

Parameters:

reason
serverShutdownReason Server shutdown message

Implements **OPCDAClient::IOPCDAClientCallback** (*p.39*).

void TestConsole::MyCB::onDataChange (string *groupName*, OPCValueQuality *masterQuality*, OPCError *masterError*, OPCValue[] *values*, OPCError[] *valueErrors*) `[inline]`

Data changed notification.

Parameters:

groupName
masterQuality communication quality for this notification
masterError Error of this notification
values Values notified
valueErrors Error of each value notified

Implements **OPCDAClient::IOPCDAClientCallback** (*p.39*).

void TestConsole::MyCB::onDisconnection (OPCDisconnectionReason *reason*, string *serverShutdownReason*) `[inline]`

Called when the server is unexpectedly disconnected: 1) by server shutdown; 2) by connection lost.

Parameters:

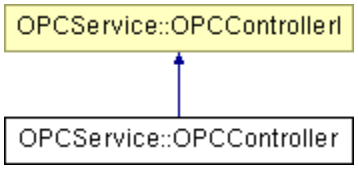
reason

serverShutdownReason Server shutdown message

Implements **OPCDAClient::IOPCDAClientCallback** (*p.39*).

4.16 OPCService::OPCController Class Reference

Inheritance diagram for OPCService::OPCController:



Public Member Functions

- int **addition** (int a, int b)
- string **getStartDemo** ()
  This method return if the demo scenario has been started.
- string **getStatusClamps** ()
  This method return the status of clamps in teh welding station.
- string **getStatusRobot** ()
  This method return the status of robots in teh welding station.
- string **getSKIDStartPosition** ()
  This method return if SKID is at the start position.
- string **getSKIDWeldingPosition** ()
  This method return if SKID is at the welding station.
- string **getActuatorSKID** ()
  This method return if SKID is at the welding station.
- string **getActuatorClamps** ()
  This method return the status of the Clamps motor.
- string **getActuatorRobot** ()
  This method return if Robot is at the welding station.
- string **getStationRest** ()
  This method return if the station is at the rest position.
- string **getStatusBodz** ()
- string **getAlarm1** ()
  This method return if Alarm station is turned On.
- string **getAlarm2** ()
  This method return if Alarm station is turned On.
- string **getAlarm3** ()
  This method return if Alarm station is turned On.

- string **getAlarm4** ()
  This method return if Alarm station is turned On.
- string **setStartDemo** (bool value)
  this method is used to start / stop the demo scenario
- string **setStatusClamps** (bool value)
  this method is used to register & unregister the clamps when they lock / unlock the SKID
- string **setStatusRobot** (bool value)
  this method is used to register & unregister the Robots when they start / finish the welding process
- string **setSKIDStartPosition** (bool value)
  this method is used to register & unregister the SKID when it arrives / leaves to the start position
- string **setSKIDWeldingPosition** (bool value)
  this method is used to register & unregister the SKID when it arrives / leaves to the welding station

Private Member Functions

- **OPCController** ()
- string **getPLCVariable** (string varName)
- string **setPLCVariable** (string varName, bool value)

Private Attributes

- **OPCDAClient.OPCDAClient opcClient**

Constructor & Destructor Documentation

OPCService::OPCController::OPCController () `[inline, private]`

Member Function Documentation

int OPCService::OPCController::addition (int *a*, int *b*) `[inline]`

string OPCService::OPCController::getPLCVariable (string *varName*) `[inline, private]`

string OPCService::OPCController::setPLCVariable (string *varName*, bool *value*) `[inline, private]`

string OPCService::OPCController::getStartDemo () `[inline]`

This method return if the demo scenario has been started.

Returns:

false = demo scenario is not started, true = it is started

Implements **OPCService::OPCControllerI** (*p.47*).

string OPCService::OPCController::getStatusClamps () `[inline]`

This method return the status of clamps in teh welding station.

Returns:

false = clamps are open, true = clamps are closed

Implements **OPCService::OPCControllerI** (*p.47*).

string OPCService::OPCController::getStatusRobot () `[inline]`

This method return the status of robots in teh welding station.

Returns:

false = robots are at rest pos, true = robots are welding

Implements **OPCService::OPCControllerI** (*p.47*).

string OPCService::OPCController::getSKIDStartPosition () `[inline]`

This method return if SKID is at the start position.

Returns:

false = SKID is NOT at the start position, true = SKID is at the start position

Implements **OPCService::OPCControllerI** (*p.48*).

string OPCService::OPCController::getSKIDWeldingPosition () `[inline]`

This method return if SKID is at the welding station.

Returns:

false = SKID is NOT at the welding station, true = SKID is at the welding station

Implements **OPCService::OPCControllerI** (*p.48*).

string OPCService::OPCController::getActuatorSKID () `[inline]`

This method return if SKID is at the welding station.

Returns:

false = SKID is NOT at the welding station, true = SKID is at the welding station

Implements **OPCService::OPCControllerI** (*p.48*).

string OPCService::OPCController::getActuatorClamps () `[inline]`

This method return the status of the Clamps motor.

Returns:

false = Clamp motors register that the clamps are open, true =clamps motor registers that the clamps are closed

Implements **OPCService::OPCControllerI** (*p.48*).

string OPCService::OPCController::getActuatorRobot () `[inline]`

This method return if Robot is at the welding station.

Returns:

false = Robot is NOT at the welding station, true = Robot is at the welding station

Implements **OPCService::OPCControllerI** (*p.48*).

string OPCService::OPCController::getStationRest () `[inline]`

This method return if the station is at the rest position.

Returns:

false = station is NOT at the rest position, true = Station is at the rest position

Implements **OPCService::OPCControllerI** (*p.48*).

string OPCService::OPCController::getStatusBodz () `[inline]`

Returns:

false = SKID is NOT at the welding station, true = SKID is at the welding station

Implements **OPCService::OPCControllerI** (*p.49*).

string OPCService::OPCController::getAlarm1 () `[inline]`

This method return if Alarm station is turned On.

Returns:

false = Alarm off, true = Alarm on

Implements **OPCService::OPCControllerI** (*p.49*).

string OPCService::OPCController::getAlarm2 () `[inline]`

This method return if Alarm station is turned On.

Returns:

false = Alarm off, true = Alarm on

Implements **OPCService::OPCControllerI** (*p.49*).

string OPCService::OPCController::getAlarm3 () `[inline]`

This method return if Alarm station is turned On.

Returns:

false = Alarm off, true = Alarm on

Implements **OPCService::OPCControllerI** (*p.49*).

string OPCService::OPCController::getAlarm4 () `[inline]`

This method return if Alarm station is turned On.

Returns:

false = Alarm off, true = Alarm on

Implements **OPCService::OPCControllerI** (*p.49*).

string OPCService::OPCController::setStartDemo (bool *value*) `[inline]`

this method is used to start / stop the demo scenario

Parameters:

value false = stop scenario, true = start scenario

Returns:

Implements **OPCService::OPCControllerI** (*p.49*).

string OPCService::OPCController::setStatusClamps (bool *value*) `[inline]`

this method is used to register & unregister the clamps when they lock / unlock the SKID

Parameters:

value false = clamps open, true = clamps closed

Returns:

Implements **OPCService::OPCControllerI** (*p.50*).

string OPCService::OPCController::setStatusRobot (bool *value*) `[inline]`

this method is used to register & unregister the Robots when they start / finish the welding process

Parameters:

value false = robots at rest, true = robots are working

Implements **OPCService::OPCControllerI** (*p.50*).

string OPCService::OPCController::setSKIDStartPosition (bool *value*) `[inline]`

this method is used to register & unregister the SKID when it arrives / leaves to the start position

Parameters:

value true = SKID arrives at the start position, false = SKID leaves the start position/param>

Returns:

Implements **OPCService::OPCControllerI** (*p.50*).

string OPCService::OPCController::setSKIDWeldingPosition (bool *value*) `[inline]`

this method is used to register & unregister the SKID when it arrives / leaves to the welding station

Parameters:

value true = SKID arrives at the welding station, false = SKID leaves the station

Returns:

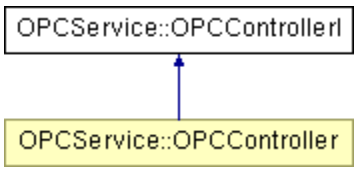
Implements **OPCService::OPCControllerI** (*p.50*).

Member Data Documentation

OPCDAClient.OPCDAClient OPCService::OPCController::opcClient `[private]`

4.17 OPCService::OPCControllerI Interface Reference

Inheritance diagram for OPCService::OPCControllerI:



Public Member Functions

- string **getStartDemo** ()
  This method return if the demo scenario has been started.
- string **getStatusClamps** ()
  This method return the status of clamps in teh welding station.
- string **getStatusRobot** ()
  This method return the status of robots in teh welding station.
- string **getSKIDStartPosition** ()
  This method return if SKID is at the start position.
- string **getSKIDWeldingPosition** ()
  This method return if SKID is at the welding station.
- string **getActuatorSKID** ()
  This method return if SKID is at the welding station.
- string **getActuatorClamps** ()
  This method return the status of the Clamps motor.
- string **getActuatorRobot** ()
  This method return if Robot is at the welding station.
- string **getStationRest** ()
  This method return if the station is at the rest position.

- string **getStatusBodz** ()
- string **getAlarm1** ()
  This method return if Alarm station is turned On.
- string **getAlarm2** ()
  This method return if Alarm station is turned On.
- string **getAlarm3** ()
  This method return if Alarm station is turned On.
- string **getAlarm4** ()
  This method return if Alarm station is turned On.
- string **setStartDemo** (bool value)
  this method is used to start / stop the demo scenario
- string **setStatusClamps** (bool value)
  this method is used to register & unregister the clamps when they lock / unlock the SKID
- string **setStatusRobot** (bool value)
  this method is used to register & unregister the Robots when they start / finish the welding process
- string **setSKIDStartPosition** (bool value)
  this method is used to register & unregister the SKID when it arrives / leaves to the start position
- string **setSKIDWeldingPosition** (bool value)
  this method is used to register & unregister the SKID when it arrives / leaves to the welding station

---

Member Function Documentation

string OPCService::OPCControllerI::getStartDemo ()

This method return if the demo scenario has been started.

Returns:

false = demo scenario is not started, true = it is started

Implemented in **OPCService::OPCController** (*p.42*).

string OPCService::OPCControllerI::getStatusClamps ()

This method return the status of clamps in teh welding station.

Returns:

false = clamps are open, true = clamps are closed

Implemented in **OPCService::OPCController** (*p.42*).

string OPCService::OPCControllerI::getStatusRobot ()

This method return the status of robots in teh welding station.

Returns:

false = robots are at rest pos, true = robots are welding

Implemented in **OPCService::OPCController** (*p.43*).

string OPCService::OPCControllerI::getSKIDStartPosition ()

This method return if SKID is at the start position.

Returns:

false = SKID is NOT at the start position, true = SKID is at the start position

Implemented in **OPCService::OPCController** (*p.43*).

string OPCService::OPCControllerI::getSKIDWeldingPosition ()

This method return if SKID is at the welding station.

Returns:

false = SKID is NOT at the welding station, true = SKID is at the welding station

Implemented in **OPCService::OPCController** (*p.43*).

string OPCService::OPCControllerI::getActuatorSKID ()

This method return if SKID is at the welding station.

Returns:

false = SKID is NOT at the welding station, true = SKID is at the welding station

Implemented in **OPCService::OPCController** (*p.43*).

string OPCService::OPCControllerI::getActuatorClamps ()

This method return the status of the Clamps motor.

Returns:

false = Clamp motors register that the clamps are open, true =clamps motor registers that the clamps are closed

Implemented in **OPCService::OPCController** (*p.43*).

string OPCService::OPCControllerI::getActuatorRobot ()

This method return if Robot is at the welding station.

Returns:

false = Robot is NOT at the welding station, true = Robot is at the welding station

Implemented in **OPCService::OPCController** (*p.43*).

string OPCService::OPCControllerI::getStationRest ()

This method return if the station is at the rest position.

Returns:

false = station is NOT at the rest position, true = Station is at the rest position

Implemented in **OPCService::OPCController** (*p.44*).

string OPCService::OPCControllerI::getStatusBodz ()

Returns:

false = SKID is NOT at the welding station, true = SKID is at the welding station

Implemented in **OPCService::OPCController** (*p.44*).

string OPCService::OPCControllerI::getAlarm1 ()

This method return if Alarm station is turned On.

Returns:

false = Alarm off, true = Alarm on

Implemented in **OPCService::OPCController** (*p.44*).

string OPCService::OPCControllerI::getAlarm2 ()

This method return if Alarm station is turned On.

Returns:

false = Alarm off, true = Alarm on

Implemented in **OPCService::OPCController** (*p.44*).

string OPCService::OPCControllerI::getAlarm3 ()

This method return if Alarm station is turned On.

Returns:

false = Alarm off, true = Alarm on

Implemented in **OPCService::OPCController** (*p.44*).

string OPCService::OPCControllerI::getAlarm4 ()

This method return if Alarm station is turned On.

Returns:

false = Alarm off, true = Alarm on

Implemented in **OPCService::OPCController** (*p.44*).

string OPCService::OPCControllerI::setStartDemo (bool *value*)

this method is used to start / stop the demo scenario

Parameters:

value false = stop scenario, true = start scenario

Returns:

Implemented in **OPCService::OPCController** (*p.45*).

string OPCService::OPCControllerI::setStatusClamps (bool *value*)

this method is used to register & unregister the clamps when they lock / unlock the SKID

Parameters:

value false = clamps open, true = clamps closed

Returns:

Implemented in **OPCService::OPCController** (*p.45*).

string OPCService::OPCControllerI::setStatusRobot (bool *value*)

this method is used to register & unregister the Robots when they start / finish the welding process

Parameters:

value false = robots at rest, true = robots are working

Implemented in **OPCService::OPCController** (*p.45*).

string OPCService::OPCControllerI::setSKIDStartPosition (bool *value*)

this method is used to register & unregister the SKID when it arrives / leaves to the start position

Parameters:

value true = SKID arrives at the start position, false = SKID leaves the start position/param>

Returns:

Implemented in **OPCService::OPCController** (*p.45*).

string OPCService::OPCControllerI::setSKIDWeldingPosition (bool *value*)

this method is used to register & unregister the SKID when it arrives / leaves to the welding station

Parameters:

value true = SKID arrives at the welding station, false = SKID leaves the station

Returns:

4.18 OPCDAClient::OPCValue Class Reference

Substitution of OpcRcw.Da.OPCITEMSTATE.

Public Member Functions

- **OPCValue** ()
- **OPCValue** ()

Properties

- int **id** `[get, set]`
  unique client item id, OpcRcw.Da.OPCITEMDEF.hClient, OPCVariableDescription.id
- object **value** `[get, set]`
  value, a scalar variable, OpcRcw.Da.OPCITEMDEF.vDataValue
- DateTime **datetime** `[get, set]`
  time stamp, OpcRcw.Da.OPCITEMDEF.ftTimeStamp
- **OPCValueQuality quality** `[get, set]`
  *value quality, rough resume of OpcRcw.Da.OPCITEMDEF.wQuality, of enum **OPC.OPC_QUALITY_STATUS***

Detailed Description

Substitution of OpcRcw.Da.OPCITEMSTATE.

Constructor & Destructor Documentation

OPCDAClient::OPCValue::OPCValue () `[inline]`

OPCDAClient::OPCValue::OPCValue () `[inline]`

Property Documentation

int OPCDAClient::OPCValue::id `[get, set]`

unique client item id, OpcRcw.Da.OPCITEMDEF.hClient, OPCVariableDescription.id

object OPCDAClient::OPCValue::value `[get, set]`

value, a scalar variable, OpcRcw.Da.OPCITEMDEF.vDataValue

DateTime OPCDAClient::OPCValue::datetime `[get, set]`

time stamp, OpcRcw.Da.OPCITEMDEF.ftTimeStamp

OPCValueQuality OPCDAClient::OPCValue::quality `[get, set]`

value quality, rough resume of OpcRcw.Da.OPCITEMDEF.wQuality, of enum **OPC.OPC_QUALITY_STATUS**

4.19 TestConsole::Program Class Reference

Static Private Member Functions

- static bool **verifyRegistry** ()
  This function verifies if the necessary rigistry setting is made. If the program is lanched in Studio Debug mode, the module name is "TestConsole.vshost.exe" and verification fails. If verification fails, AuthenticationLevel will not be read by OLE mechanism and OPC communication will not work with remote hosts.
- static void **Main** (string[] args)
- static bool **verifyRegistry** ()
  This function verifies if the necessary rigistry setting is made. If the program is lanched in Studio Debug mode, the module name is "TestConsole.vshost.exe" and verification fails. If verification fails, AuthenticationLevel will not be read by OLE mechanism and OPC communication will not work with remote hosts.
- static void **Main** (string[] args)

Private Attributes

- const string **mySelf** = "{33EA2EAF-0A02-4A3E-B61D-BC757C555BF1}"

Member Function Documentation

static bool TestConsole::Program::verifyRegistry () `[inline, static, private]`

This function verifies if the necessary rigistry setting is made. If the program is lanched in Studio Debug mode, the module name is "TestConsole.vshost.exe" and verification fails. If verification fails, AuthenticationLevel will not be read by OLE mechanism and OPC communication will not work with remote hosts.

Returns:

static void TestConsole::Program::Main (string[] *args*) `[inline, static, private]`

static bool TestConsole::Program::verifyRegistry () `[inline, static, private]`

This function verifies if the necessary rigistry setting is made. If the program is lanched in Studio Debug mode, the module name is "TestConsole.vshost.exe" and verification fails. If verification fails, AuthenticationLevel will not be read by OLE mechanism and OPC communication will not work with remote hosts.

Returns:

static void TestConsole::Program::Main (string[] *args*) `[inline, static, private]`

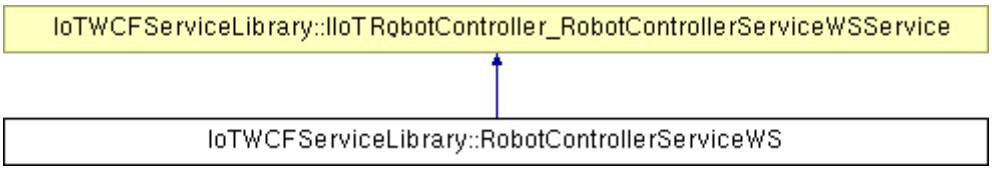
Member Data Documentation

const string TestConsole::Program::mySelf = "{33EA2EAF-0A02-4A3E-B61D-BC757C555BF1}" [private]

4.20 LinkSmart Robot Proxy IoTWCFServiceLibrary::RobotControllerServiceWS Class Reference

Exposed methods from the robot controller to LinkSmart.

Inheritance diagram for IoTWCFServiceLibrary::RobotControllerServiceWS:



Public Member Functions

- **RobotControllerServiceWS** (**RobotController. RobotController** theDevice)
- System.Int32 **GetAxis1Current** ()
  Gest the axis 1 current for the robot.
- System.Int32 **GetAxis1Voltage** ()
  Gest the axis 1 voltage for the robot.
- System.Int32 **GetAxis2Current** ()
  Gest the axis 2 current for the robot.
- System.Int32 **GetAxis2Voltage** ()
  Gest the axis 2 voltage for the robot.
- System.Int32 **GetAxis3Current** ()
  Gest the axis 3 current for the robot.
- System.Int32 **GetAxis3Voltage** ()
  Gest the axis 3 voltage for the robot.
- System.Int32 **GetAxis4Current** ()
  Gest the axis 4 current for the robot.
- System.Int32 **GetAxis4Voltage** ()
  Gest the axis 4 voltage for the robot.
- System.Int32 **GetAxis5Current** ()
  Gest the axis 5 current for the robot.
- System.Int32 **GetAxis5Voltage** ()
  Gest the axis 5 voltage for the robot.
- System.Int32 **GetAxis6Current** ()
  Gest the axis 6 current for the robot.
- System.Int32 **GetAxis6Voltage** ()
  Gest the axis 6 voltage for the robot.

Private Attributes

- **RobotController. RobotController m_robotcontroller**

Detailed Description

Exposed methods from the robot controller to LinkSmart

Constructor & Destructor Documentation

IoTWCFServiceLibrary::RobotControllerServiceWS::RobotControllerServiceWS (RobotController. RobotController *theDevice*) `[inline]`

Member Function Documentation

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis1Current () `[inline]`

Gest the axis 1 current for the robot.

Returns:

Axis 1 current in Ampere

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis1Voltage () `[inline]`

Gest the axis 1 voltage for the robot.

Returns:

Axis 1 voltage in Volt

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis2Current () `[inline]`

Gest the axis 2 current for the robot.

Returns:

Axis 2 current in Ampere

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis2Voltage () `[inline]`

Gest the axis 2 voltage for the robot.

Returns:

Axis 2 voltage in Volt

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis3Current () `[inline]`

Gest the axis 3 current for the robot.

Returns:

Axis 3 current in Ampere

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis3Voltage () [inline]

Gest the axis 3 voltage for the robot.

Returns:

Axis 3 voltage in Volt

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis4Current () [inline]

Gest the axis 4 current for the robot.

Returns:

Axis 4 current in Ampere

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis4Voltage () [inline]

Gest the axis 4 voltage for the robot.

Returns:

Axis 4 voltage in Volt

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis5Current () [inline]

Gest the axis 5 current for the robot.

Returns:

Axis 5 current in Ampere

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis5Voltage () [inline]

Gest the axis 5 voltage for the robot.

Returns:

Axis 5 voltage in Volt

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis6Current () [inline]

Gest the axis 6 current for the robot.

Returns:

Axis 6 current in Ampere

System.Int32 IoTWCFServiceLibrary::RobotControllerServiceWS::GetAxis6Voltage () [inline]

Gest the axis 6 voltage for the robot.

Returns:

Axis 6 voltage in Volt

Member Data Documentation

RobotController. RobotController IoTWCFServiceLibrary::RobotControllerServiceWS::m_robotcontroller
[private]

References

(LINKSMART, 2012) http://www.hydramiddleware.eu/news.php, visited 2012-08-15.

(LINKSMART2,2012) http://sourceforge.net/projects/linksmart/, visited 2012-08-15.