



Jaco API programming guide

R5.0.2

Version 1.0.0

June 5 2013

Warning

The information contained in this document is the property of Kinova. Except as specifically authorized in writing by Kinova, the holder of this document shall keep the information contained here confidential and shall protect same in whole or in part from disclosure and dissemination to third parties and use same for evaluation, operation and maintenance purposes only. This document is not an engagement of Kinova to develop, implement or design the product or techniques described here.

Table of content

1	General information.....	4
1.1	Purpose of this guide.....	4
2	Tutorial	5
2.1	Getting started.....	5
2.1.1	Files structure.....	5
2.2	Libraries	5
2.3	Hello world	6
2.4	Client configurations.....	8
2.5	Protection zones	11
2.6	Control mapping.....	12
2.6.1	Data layer 1: The control mapping charts	12
2.6.2	Data layer 2: The control mapping.....	13
2.6.3	Data layer 3: The control mode map	14
2.6.4	Data layer 4: The joystick event	15
2.6.5	Data layer 5: The functionality	17
2.7	Virtual Joystick	18
2.7.1	Class CJoystickValue.....	18
2.8	Control modes.....	19
2.8.1	Angular control type	19
2.8.2	Cartesian control type	20
2.9	Trajectories	20
2.10	Diagnostic and tool function	20
3	API	21
3.1	Overview	21
3.2	Managers	21
3.3	Configuration managers	22
3.3.1	General configuration	23
3.3.2	Client configuration	26
3.3.3	Protection zones	28
3.3.4	Control mapping.....	30
3.3.5	Profile.....	33
3.4	Diagnostic manager	34
3.4.1	Diagnostic data	34
3.4.2	CPosition.....	34
3.4.3	Tool.....	42
3.5	Control manager	43

3.6	Other functionalities.....	56
4	API Remote.....	57
4.1	Overview	57
4.2	Server GUI	57
4.2.1	Windows	57
4.2.2	Ubuntu	58
4.3	Example	59
4.3.1	Windows	59
4.3.2	Ubuntu	60
5	Troubleshooting.....	61
5.1	Activity log.....	61
5.2	Jacosoft.....	61
5.3	Contact us.....	61
6	F.A.Q.....	62

1 General information

1.1 Purpose of this guide

This guide is to help developer when they develop application that interact with the robotic arm Jaco via its API and it assumes that the API has been installed and works correctly. If you need explanation about the installation, please refer to the installation guide.

All the examples are written in `c#` but Ubuntu developers that use other languages will find their way very easily.

2 Tutorial

2.1 Getting started

2.1.1 Files structure

When you have installed Jacosoft, it has created a folder structure that, as a developer, you will use during the development of your application. Let's have a better look of what it really contains.

The first thing to know is that the root folder may change depending on the system you are on. On windows system it will be at: [Your application data folder] / Kinova. On Ubuntu system, it will be at HOME/Kinova/Application Data/.

Folder's name	Description
Kinova_Root	Kinova root folder
Kinova_Root \ Product	This is where all Kinova products are installed. It is located at: Kinova_Root\
Product \ Licenses	This is where all licence related files are kept.
Product \ Jacosoft	This is the root folder of Jacosoft.
Jacosoft \ API	This is the root folder of all Kinova API.
API \ Data	This is a folder where all serialized data are stored.
Data \ Config	This is a folder where you can store serialized configurations.
Data \ Logs	This is the root folder of all log files.
Logs \ LogErrors	This is where all errors log files are stored.
Logs \ Activities	This is where all activities log files are stored.
Data \ Positions	This is a folder where you can store serialized position.
Data \ Profiles	This is the root folder of all stored profiles.
Data \ Trajectories	This is a folder where you can store trajectories.

2.2 Libraries

Jaco's API is a set of libraries that you can access with your application. It is separated into several modules. Here is the complete list of all available libraries that a developer can access to interact with the robotic arm Jaco:

Library	Description
Kinova.API.Jaco.dll	
Kinova.DLL.Data.dll	

Kinova.DLL.SafeGate.dll
Kinova.DLL.Tools.dll
Kinova.DLL.USBManager.dll
Kinova.DLL.TestData.dll
Kinova.DLL.ReportBuilder
Kinova.DLL.CommData.dll
Kinova.DLL.TcpConnector.dll

You also have to copy the External_DLL folder and all of its content at the same place of your application.

2.3 Hello world

Now is the time to write our first Jaco application. You can follow the next section with the visual studio 2010 project named Hello World. It is located in the example folder of the installation package.

Open visual studio 2010 and create a C# console application with a main function and name it HelloWorld. Via the solution explorer, rename the file Program.cs file to HelloWorld.cs.

The file will look like this,

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace HelloWorld
{
    class HelloWorld
    {
        static void Main(string[] args)
        {
        }
    }
}
```

From there, the next thing to do is to declare and initialize an instance of an object CJacoArm. This object will be our entrance point to the API. The only parameter you need to provide to the constructor is an encrypted password.

Next, modify the main function your class like this,

```
static void Main(string[] args)
{
```

```

string MyValidPassword = "MyValidPassword";

try
{
    //An Object that represents the robotic arm Jaco.
    CJacoArm Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    //Is Jaco ready to communicate?
    if (Jaco.JacoIsReady())
    {
        System.Console.WriteLine("\n\nHello World ! ! !");
    }
    else
    {
        System.Console.WriteLine("Jaco is not ready to communicate.");
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("Exception during execution of the example. Verify " +
        "your API installation, verify if your Jaco is " +
        "connected and verify that " +
        "you have a valid password.");
}

System.Console.WriteLine("End of the example...");
System.Console.ReadKey();
}

```

Build your project and wait for some errors to show. It is quite normal to have errors here since we did not import the DLL from the API to communicate with Jaco. Add all API'S DLL as a reference to your project. Note that in the example project me put a copy of all needed libraries in a folder called bin/DLL inside the project folder. Make sure that a copy of the folder External_DLL is at the same location as your application. Most of the time, it is at bin\Release or bin\Debug. Once all of that is completed, add those lines at the beginning of the file.

```

using Kinova.API.Jaco;
using Kinova.DLL.SafeGate;

```

Rebuild your project. If you did everything correctly, the built should be completed without any errors.

Let's analyze the code a little bit,

```

string MyValidPassword = "MyValidPassword";

```

The previous line is a string that holds the password that has been given to you by Kinova. You have to replace the "MyValidPassword" by the real password.

```

CJacoArm Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

```

Here, we are declaring an instance of Jaco and we use the Crypto class to encrypt the password. Crypto is class from the DLL Kinova.DLL.SafeGate. So, we get an instance of Crypto (which is a singleton class) and we pass the result (CCypherMessage) to the constructor. It is done in one line but we could have done the same thing by doing:

```
CCypherMessage encryptedPassword = Crypto.GetInstance().Encrypt(MyValidPassword);

//An Object that represents the robotic arm Jaco.
CJacoArm Jaco = new CJacoArm(encryptedPassword);
```

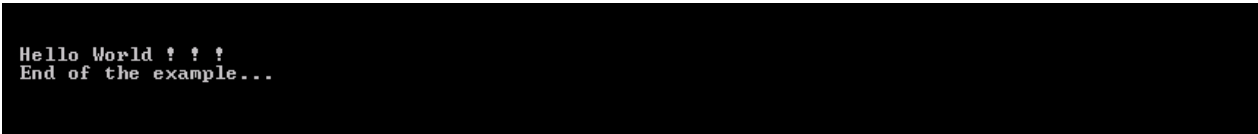
```
//Is Jaco ready to communicate?
if (Jaco.JacoIsReady())
{
    System.Console.WriteLine("\n\nHello World ! ! !");
}
else
{
    System.Console.WriteLine("Jaco is not ready to communicate.");
}
```

In the previous code snippet, use the method JacoIsReady() to verify if the communication with the robotic arm is good. If it is, we show our “HelloWorld !!!” on the console.

```
catch (Exception ex)
{
    System.Console.WriteLine("Exception during execution of the example. Verify " +
        "your API installation, verify if your Jaco is " +
        "connected and verify that " +
        "you have a valid password.");
}
```

The example catches all type of exception and display a message when a problem is hit.

Execute the application and it should look like:



```
Hello World ! ! !
End of the example...
```

Congratulation, you have just created your first Jaco application!

2.4 Client configurations

Now that we are able to communicate with Jaco, let's try something a little more concrete that interact with the data. The next example is about getting the basic configuration (client configuration) of the robot, modify it and send it back to Jaco. You can follow the next section with the visual studio 2010 project named ClientConfiguration. It is located in the example folder of the installation package.

First, create a project like you did in section [HelloWorld](#), make all the needed references to the API and rename it ClientConfiguration. Once it is done, make your main class look like this.

```
static void Main(string[] args)
{
    string MyValidPassword = "MyValidPassword";

    try
    {
        //An Object that represents the robotic arm Jaco.
        CJacoArm Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

        CClientConfigurations config = new CClientConfigurations();

        //Is Jaco ready to communicate?
        if (Jaco.JacoIsReady())
        {
            config = Jaco.ConfigurationsManager.GetClientConfigurations();
        }
        else
        {
            System.Console.WriteLine("Jaco is not ready to communicate.");
        }
    }
    catch (Exception ex)
    {
        System.Console.WriteLine("Exception during execution of the example. Verify " +
            "your API installation, verify if your Jaco is " +
            "connected and verify that " +
            "you have a valid password.");
    }

    System.Console.WriteLine("End of the example...");
    System.Console.ReadKey();
}
```

Let's dive in the code.

```
CClientConfigurations config = new CClientConfigurations();
```

This line instantiates a CClientConfigurations object which is a member of the namespace Kinova.DLL.Data.Jaco.Config. It contains all data specific to a Jaco's client.

```
config = Jaco.ConfigurationsManager.GetClientConfigurations();
```

This is the interesting line that you want to know about. It does a lot of work but it is quite simple to use. First, you access the configuration manager through Jaco and then you get the information about the client configurations.

For the next step, we will display the information, modify it, send it to Jaco and verify our modification. To do so, modify your main function to make it like this.

```

if (Jaco.JacoIsReady())
{
    //Get the data from Jaco
    config = Jaco.ConfigurationsManager.GetClientConfigurations();

    //Display the Data
    System.Console.WriteLine("                Name : " + config.ClientName);
    System.Console.WriteLine("Max linear speed : " + config.MaxLinearSpeed);
    System.Console.WriteLine("    Organization : " + config.Organization);

    config.ClientName = "James Bond";
    config.MaxLinearSpeed = 0.1f;
    config.Organization = "MI6";

    //We send the new data to Jaco
    Jaco.ConfigurationsManager.SetClientConfigurations(config);

    //Get the new data from Jaco
    config = Jaco.ConfigurationsManager.GetClientConfigurations();

    //Display the new Data
    System.Console.WriteLine("                Name : " + config.ClientName);
    System.Console.WriteLine("Max linear speed : " + config.MaxLinearSpeed);
    System.Console.WriteLine("    Organization : " + config.Organization);
}

```

```
System.Console.WriteLine("                Name : " + config.ClientName);
```

We display the client's name. You can use it to name your Jaco.

```
System.Console.WriteLine("Max linear speed : " + config.MaxLinearSpeed);
```

We display the max linear speed which corresponds to the speed when you control Jaco in cartesian mode.

```
System.Console.WriteLine("    Organization : " + config.Organization);
```

We display the client's organization.

```

config.ClientName = "James Bond";
config.MaxLinearSpeed = 0.1f;
config.Organization = "MI6";

```

Here, we modify the information.

Build your project and execute it. It should look like this:

```
      Name : DefaultClientName
Max linear speed : 0.15
      Organization : DefaultOrganizat
```

```
Sending the information to Jaco...
```

```
      Name : James Bond
Max linear speed : 0.1
      Organization : MI6
End of the example...
```

2.5 Protection zones

There is a built in protection zone system inside Jaco that can be accessed with the API. A protection zone is specific portion of space that possesses a speed limitation. The first thing to know is that a protection zone's shape can only be a rectangular prism of type `CJacoStructures.ShapeType.PrismSquareBase_Z`.

.

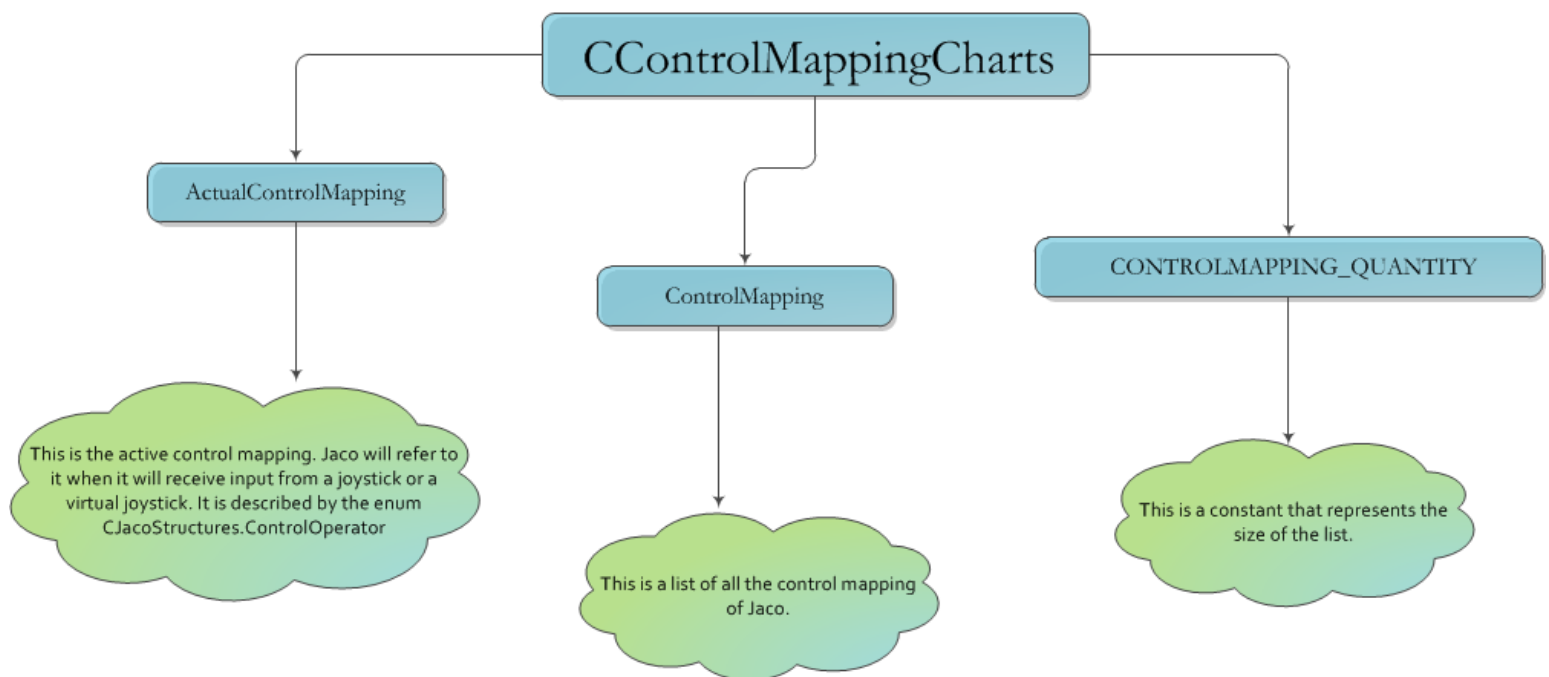
There is 2 way of building a protection zone. The first one is to do it manually by instantiating a `CZone` object and fill its member with values. The other way is to use the static method `Czone.CreateXYBaseZone()` which takes, as parameters, 4 points to represents the base of the prism, the height of the prism, an angular speed limit and linear speed limit.

2.6 Control mapping

The Jaco's control mapping system is way to associate a specific functionality to a virtual third party control device. The data structure behind the system can seems, at first, a little complicated but once you are familiar with it, this become very useful if you want to implement your own control device. To begin, let's take a look at the different data's layer.

2.6.1 Data layer 1: The control mapping charts

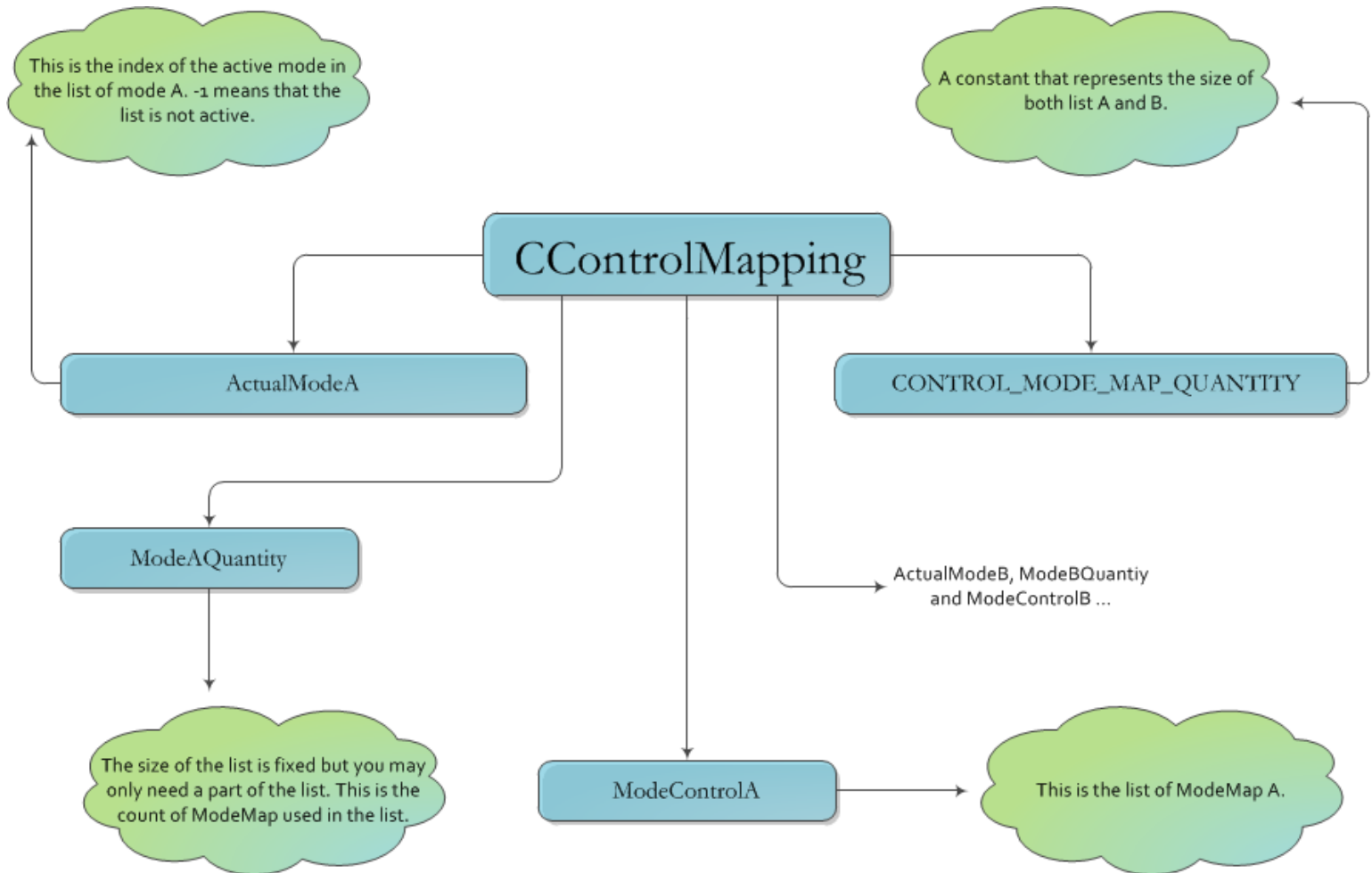
This is the root of the mapping system. It is a chart that contains a list of control mapping each control mapping from the list is associated with a specific device. As an example, by default, there is a control mapping associated with the 3 axis Kinova joystick. Every time you use the joystick, Jaco look into the control mapping charts, find the control mapping associated with the joystick and execute the functionality mapped to the input (like pressing the button 1) you are sending. At any time this chart can be modified but keep in mind that if you change the mapping of the kinova joystick it will affect the behaviour of the robot next time someone will use it. Ultimately, at any time, you can execute the [\[Restore default factory\]](#) functionality to get back the default control mapping charts.



2.6.2 Data layer 2: The control mapping

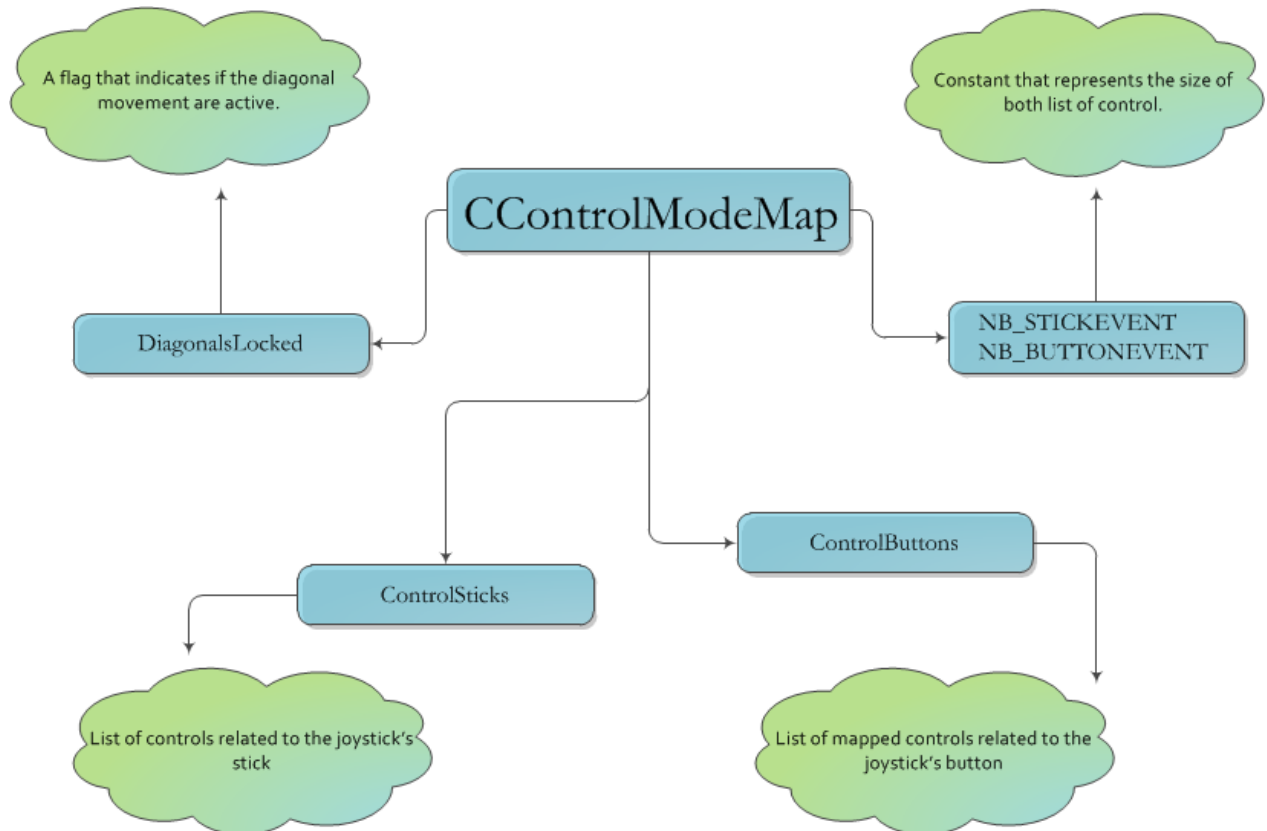
A control mapping is a configured group of functionalities associated to a specific control device. As an example, by default, Jaco has a control mapping for a 3 axis joystick, a 2 axis joystick and another one for the API.

A control mapping has 2 list of control mode map that can be iterated by your application.



2.6.3 Data layer 3: The control mode map

A control mode map represents a specific control configuration of a joystick. Usually, a control mapping has several mode map that can be used. As an example, if you are using the kinova 3 axis joystick with its default mapping, if you press the right button on the top of the stick, you iterate between to mode map: one that control translation and another that move the wrist. Also, this is where you specify if you allow the robot to perform diagonal movement.

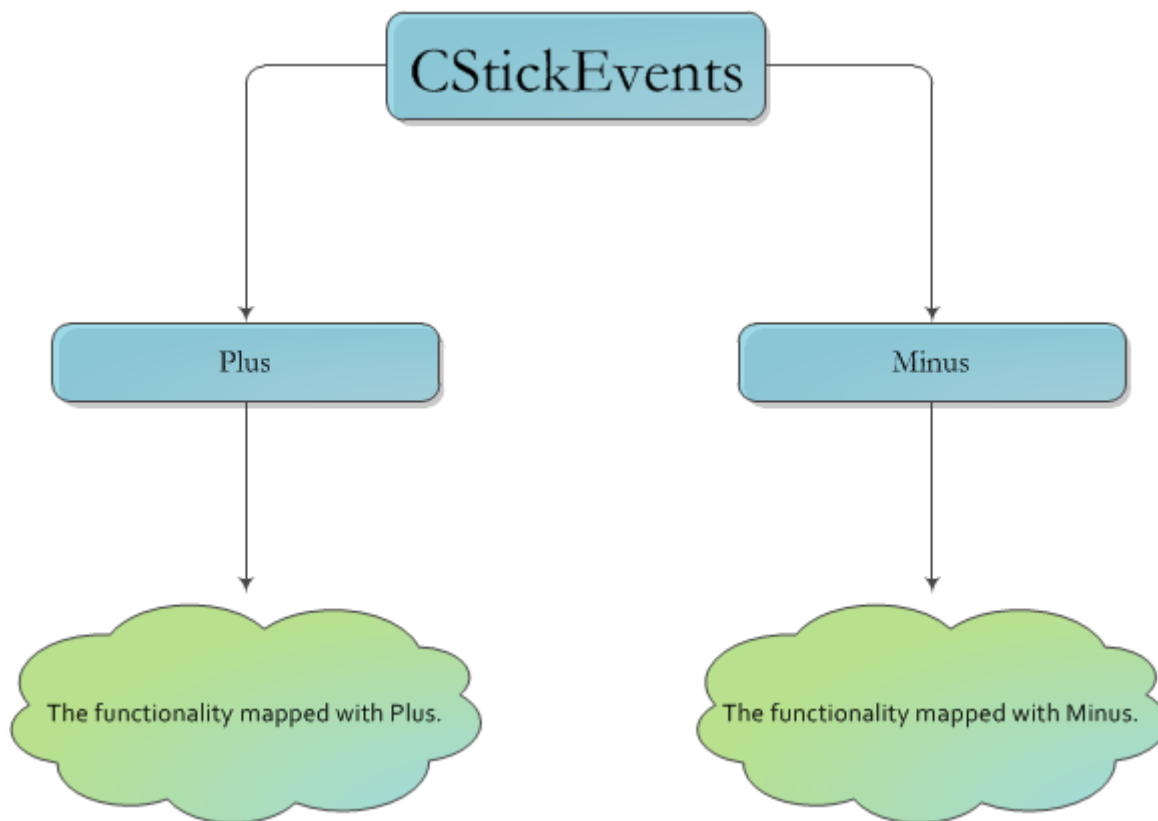


2.6.4 Data layer 4: The joystick event

This data layer represents data from a joystick. It is separated into 2 sub layer: data from the stick and the data from a set of buttons.

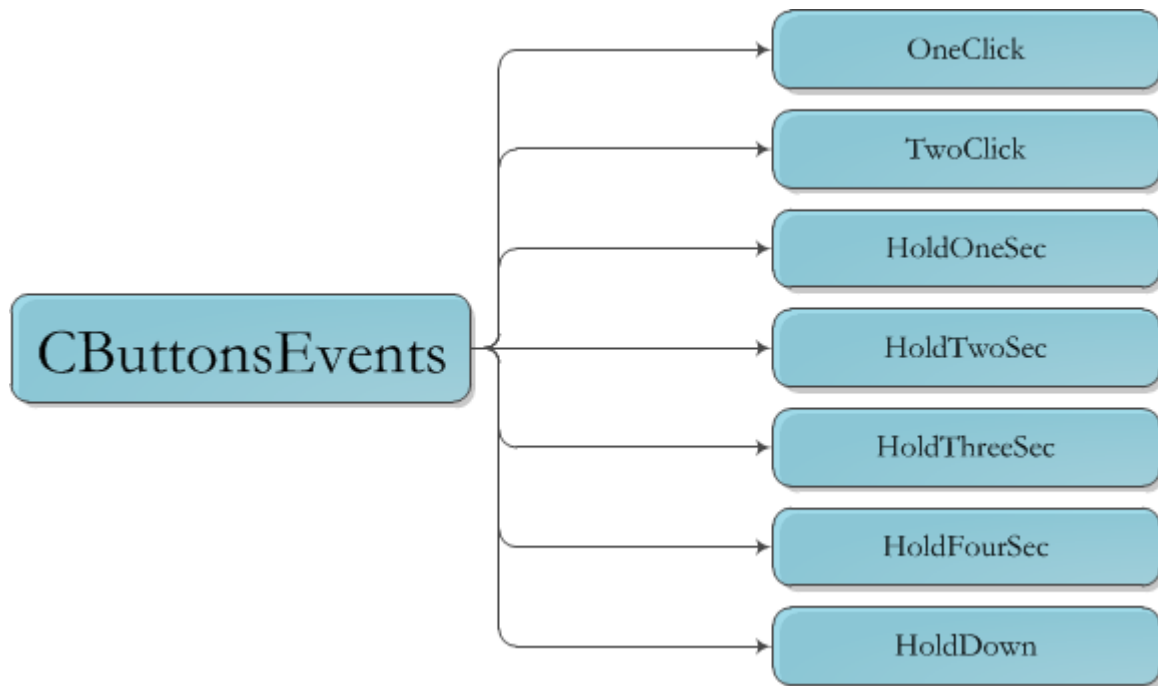
2.6.4.1 Stick Events

A stick event represents the action of moving a joystick's stick along a specific axis. As an example, if your joystick can move along three axis: forward/back, left/right and rotation of the stick, you will need 3 joystick event to cover all functionality of your joystick. When you map a stick event with a stick movement, you have to take into consideration 2 things: "what action to perform when I go positive along the axis and what action to perform when I go negative". Plus and Minus event are purely abstract, it is up to you to decide the part of the movement that is represented by the Plus and which one is represented by the Minus. As an example, say you have to map a left/right stick movement, you could map Plus with the movement to the right and the Minus with the movement to the left.



2.6.4.2 Button Events

A button event represents a single event from a button. It covers the regular event that you can be triggered by a button: a single click, a double-click, holding the button down and holding it for a specific time. The main thing you have to keep in mind here is that you can only map one functionality of type “holding” for the same CButtonsEvents object.



2.6.5 Data layer 5: The functionality

This data layer represents all functionality that can be mapped to a joystick or any other kind of custom control. It is described by the enum `CJacoStructures.ControlFunctionnalityValues`.

Functionality	Description
NoFunctionality	This is a default value.
Disable_EnableJoystick	This turn [on / off] the kinova joystick.
Retract_ReadyToUse	Switch between NORMAL, READY and RETRACT position.
Change_TwoAxis_ThreeAxis	Switch between 2 axis joystick mapping and 3 axis joystick mapping.
Change_DrinkingMode	Switch between normal and drinking mode.
ChangeMode_Left	Iterate control mode map from the list A.
ChangeMode_Right	Iterate control mode map from the list B.
DecreaseSpeed	Divide the linear(cartesian) speed by 2.
IncreaseSpeed	Multiply the linear(cartesian) speed by 2.
Goto_Position1	Move Jaco to the recorded position 1.
Goto_Position2	Move Jaco to the recorded position 2.
Goto_Position3	Move Jaco to the recorded position 3.
Goto_Position4	Move Jaco to the recorded position 4.
Goto_Position5	Move Jaco to the recorded position 5.
RecordPosition1	Store the current position into RecordPosition1.
RecordPosition2	Store the current position into RecordPosition2.
RecordPosition3	Store the current position into RecordPosition3.
RecordPosition4	Store the current position into RecordPosition4.
RecordPosition5	Store the current position into RecordPosition5.
X_Plus	Move Jaco along the X +.
X_Minus	Move Jaco along the X -.
Y_Plus	Move Jaco along the Y +.
Y_Minus	Move Jaco along the Y -.
Z_Plus	Move Jaco along the Z +.
Z_Minus	Move Jaco along the Z -.
Xtheta_Plus	Rotate Jaco around the X axis counter clockwise.
Xtheta_Minus	Rotate Jaco around the X axis clockwise.

Ytheta_Plus	Rotate Jaco around the Y axis clockwise.
Ytheta_Minus	Rotate Jaco around the Y axis counter clockwise.
Ztheta_Plus	Rotate Jaco around the Z axis clockwise.
Ztheta_Minus	Rotate Jaco around the Z axis counter clockwise.
OpenHandTwoFingers	Open 2 fingers of the hand. You can configure which fingers via the client configuration.
CloseHandTwoFingers	Open 2 fingers of the hand. You can configure which fingers via the client configuration.
OpenHandThreeFingers	Open all fingers of the hand.
CloseHandThreeFingers	Close all fingers of the hand.

2.7 Virtual Joystick

This feature let you emulate command that you can send with a Kinova joystick. The relating data is contained in the class `Kinova.DLL.Data.Jaco.CJoystickValue`. You send those values to Jaco with the method `CJacoArm.ControlManager.SendJoystickFunctionnality()`. The behaviour of Jaco will depend on the active mapping. For more details, please see section [[Send joystick functionality](#)].

2.7.1 Class CJoystickValue

This represents the value of a joystick. It contains an array of 26 (`CJoystickValue.BUTTON_VALUE_SIZE`) possible button values and 6 stick values. A button value is 0(RELEASED) or 1(PRESSED). A stick value can be between -1 to 1. Note that all stick values between -0.2 and 0.2 will be interpret as a 0.

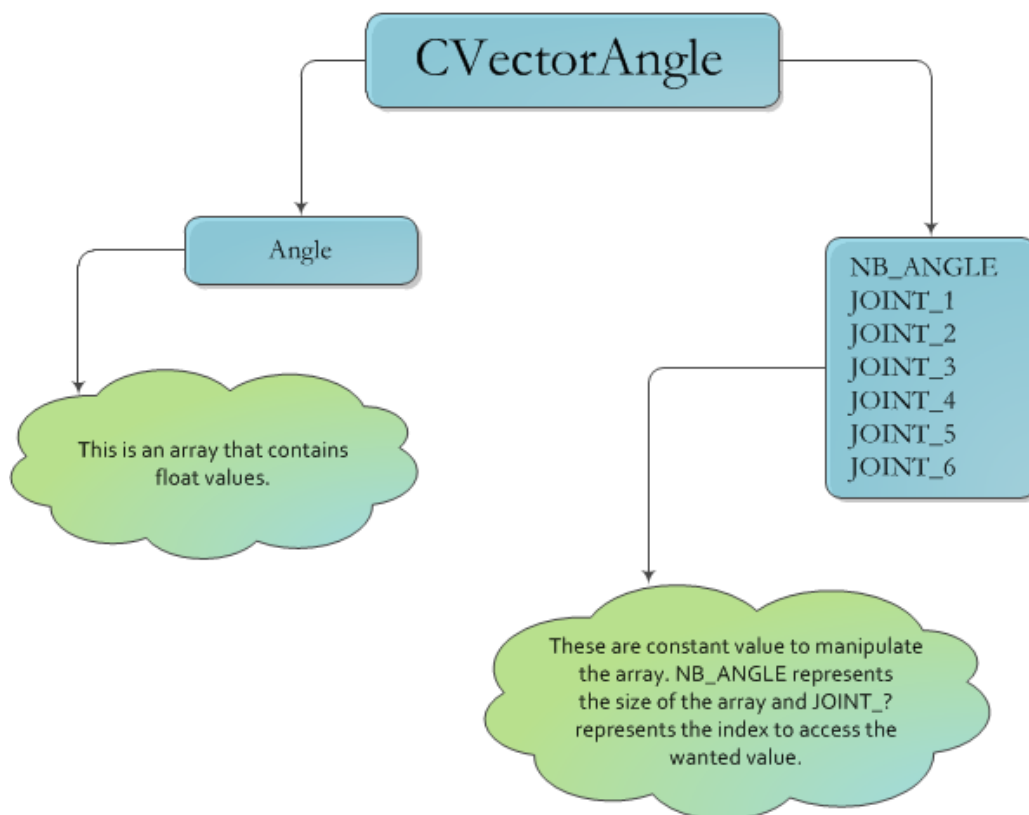
2.8 Control modes

The robotic arm Jaco has 2 types of control: the cartesian control type and the angular control type. The API provides functionalities to switch from one type to the other via [\[SetCartesianControl\]](#) and [\[SetAngularControl\]](#).

2.8.1 Angular control type

This type of control is pretty simple. It consists of moving the robotic arm joint by joint by specifying angles to each of them. The most important thing you need to know about this type of control is that there is no advance security and it is quite possible for the arm to hurt itself. Just know what you do when you control Jaco in angular mode. There is some limitation on the angle, the joint 1, 4, 5 and 6 have a range of $[-10\ 000^{\circ}, 10\ 000^{\circ}]$, joint has a range of $[42^{\circ}, 318^{\circ}]$ and joint 3 has a range of $[17^{\circ}, 343^{\circ}]$.

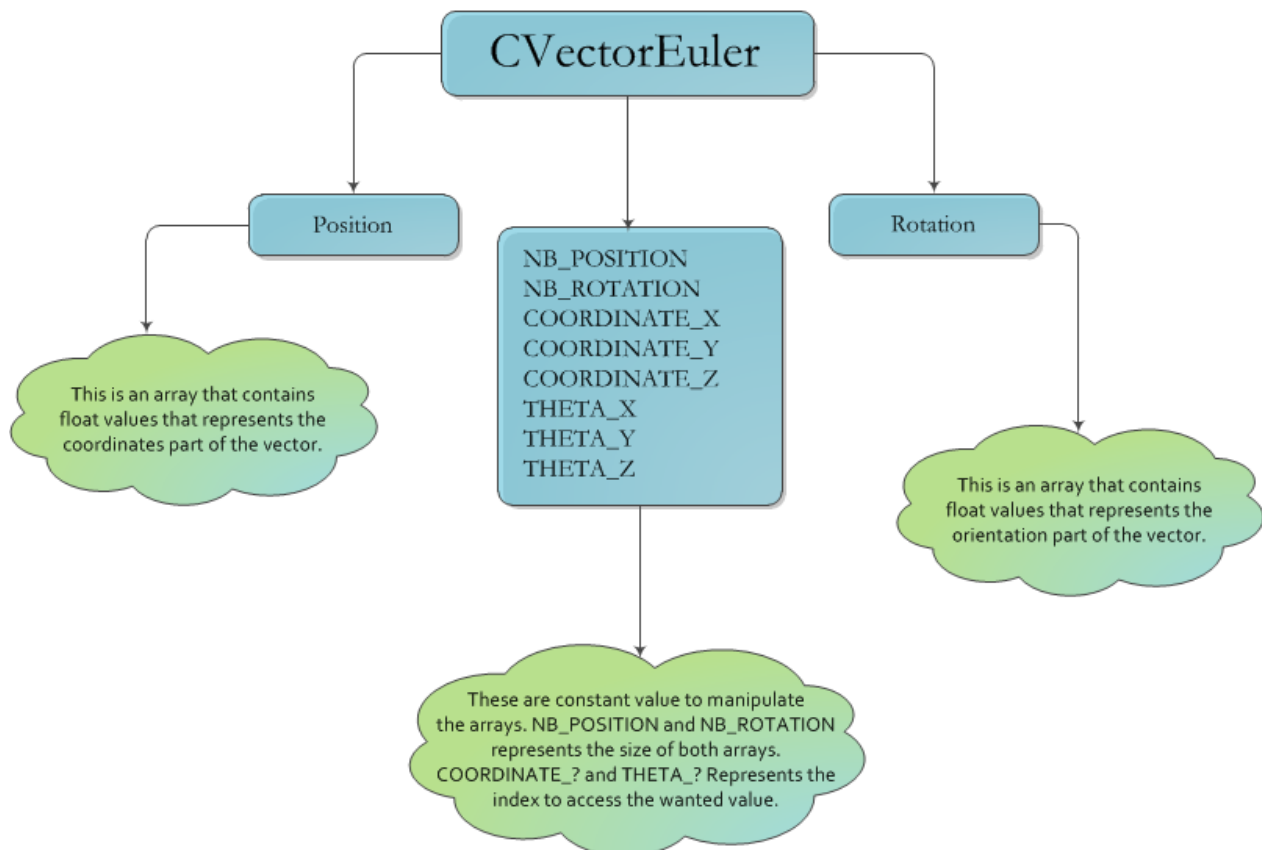
Most of time, when you deal with value of the joints in Jaco, the data structure CVectorAngle is involved. This main purpose of this class is to hold information about the joint of Jaco. For example, if you want to get the current value of all joint in Jaco, you can use the functionality [\[GetJointsPosition\]](#) and it will returns a CVectorAngle object that contains 6 values; each representing an angle value in degree.



2.8.2 Cartesian control type

This type of control is more intuitive than the angular control and it use the onboard kinematics system of Jaco. Basically, you specify a coordinate with its orientation and Jaco goes there if it can.

Most of the time, when you deal with cartesian position, the data structure CVectorEuler is involved. The main purpose of this class is to hold information about the cartesian position of Jaco.



2.9 Trajectories

This feature let the developer send a set of point, that we call trajectory. For more information please see the section [\[Send a trajectory\]](#).

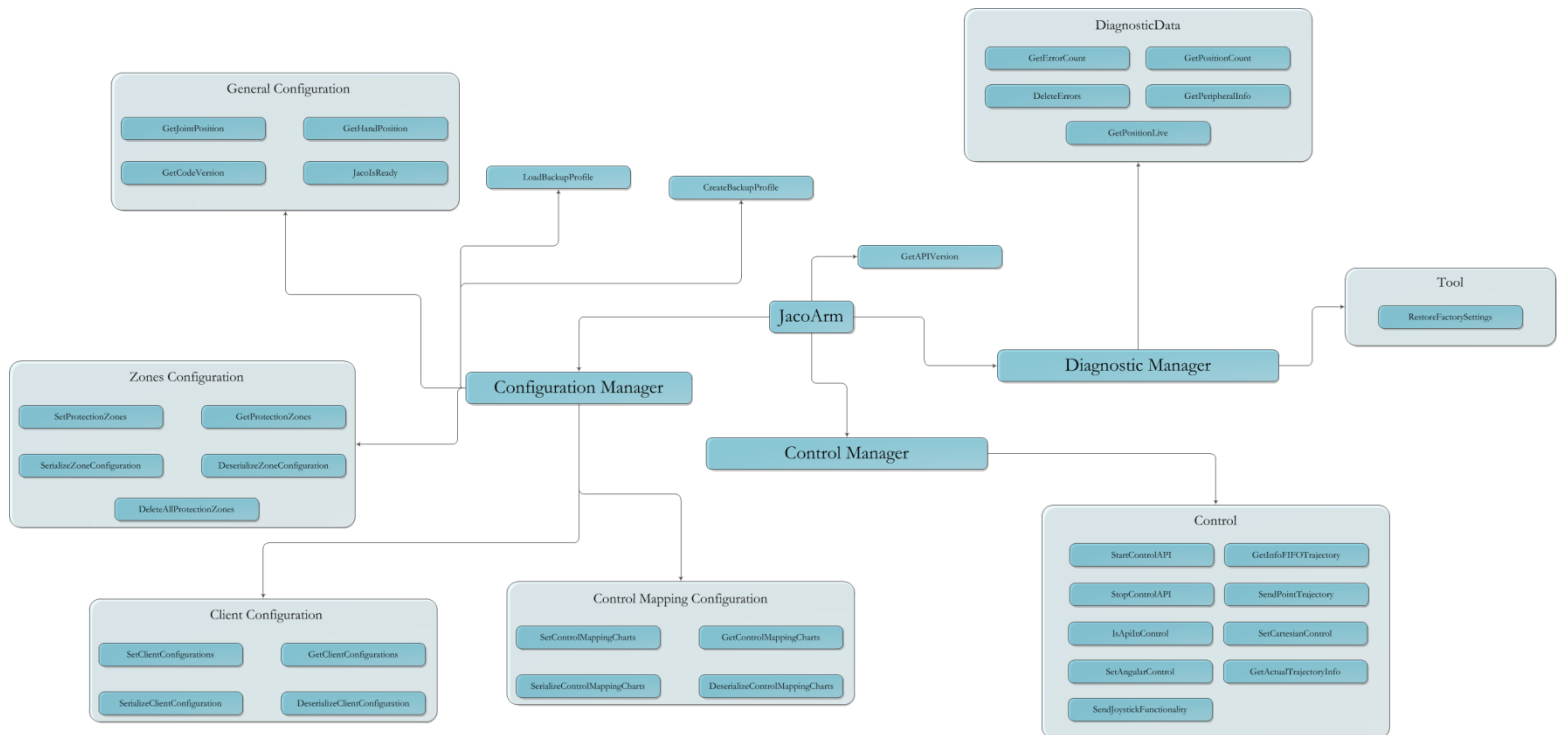
2.10 Diagnostic and tool function

See section [\[Diagnostic manager\]](#) and [\[Tool\]](#).

3 API

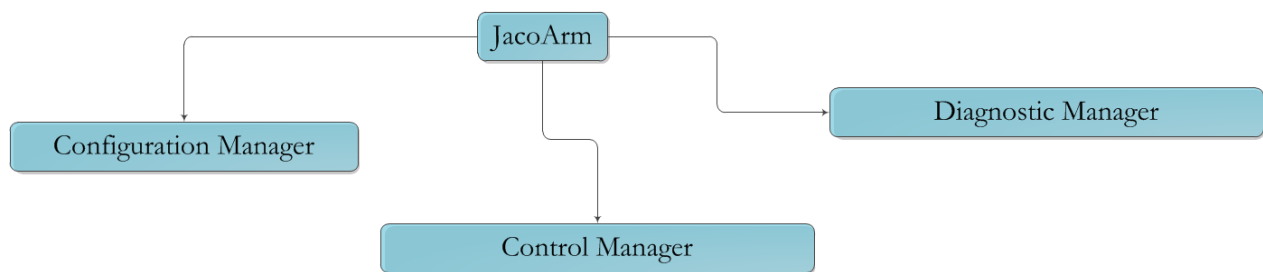
3.1 Overview

Jaco's API (Application Programming Interface) is a set of .NET DLL (Dynamic Linked Library) that let the developer interact with the robotic arm Jaco. As the may be other communication protocol implemented in the future, currently, the link between the terminal and Jaco is a USB 2 cable. Note that you can only communicate with one robot at a time.



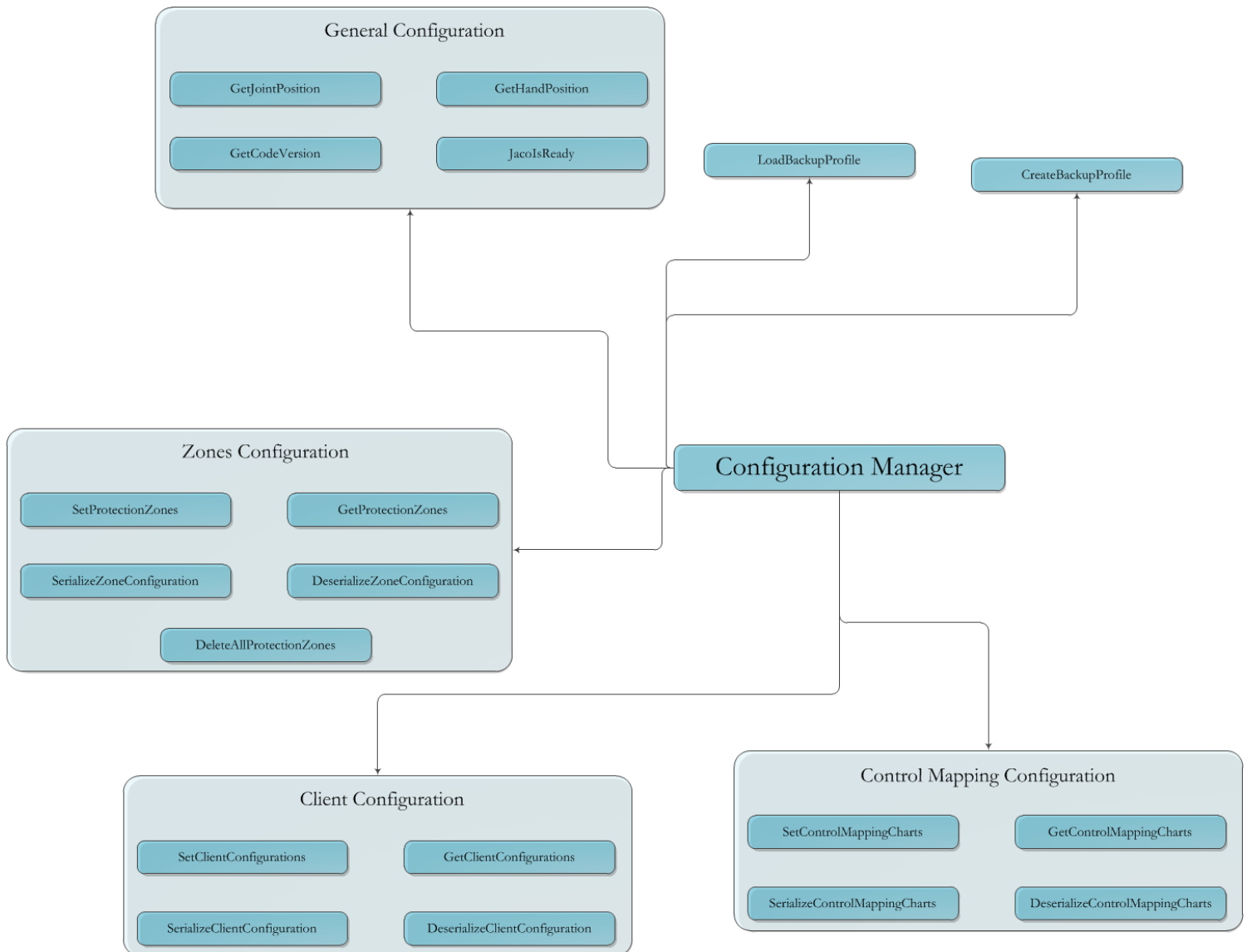
3.2 Managers

The API is divided in 3 groups of functionalities that can be access through object called managers. A manager is an entity that acts as a façade to a set of methods. Each manager can be accessed (so you don't have to create or declare them) via an initialized C.JacoArm object. Currently, 3 managers are available: the configuration manager, the control manager and the diagnostic manager.



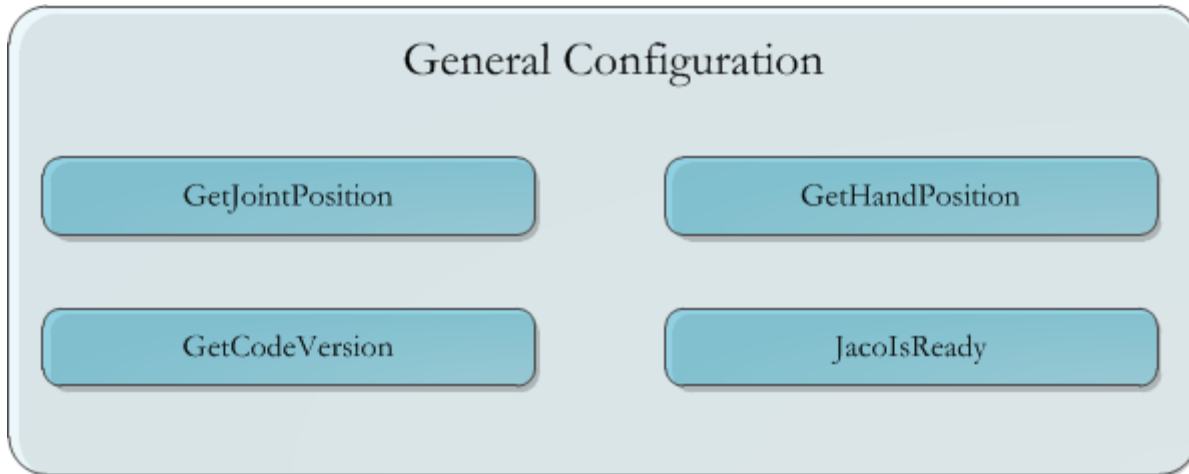
3.3 Configuration managers

Jaco's configuration is divided into 4 groups of functionalities. The first group is composed of data that are client specific related such as the client's name, the speed of the robot, its laterality and many more. The second group is composed of data that represents the protection zones of the robot. The third set of functionalities regroups a data structure known as the control mapping and the last group is composed of general method related directly to the API or the robot.



3.3.1 General configuration

Basically, the general configuration is all methods that can't be placed in a specific group. They are method related to the robot itself or in some case to the API directly.



3.3.1.1 Get joints position

3.3.1.1.1 Description

This functionality provides the values of all Jaco's actuators angles. The data is regrouped and returned in an object called CVectorAngle which basically contains a float array of 6 values. The size of the array and the possible values are defined by constants inside the CVectorAngle class.

3.3.1.1.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm (Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        CVectorAngle JointPosition = m_Jaco.ConfigurationsManager.GetJointPositions();

        //Display the value of the joint #2 at index 1
        System.Console.WriteLine("Angle value of Joint #2 = " +
            JointPosition.Angle[CVectorAngle.JOINT_2]);
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.1.2 Get hand's position

3.3.1.2.1 Description

This functionality provides the Cartesian position of the robot's hand. The data is regrouped and returned in an object called CVectorEuler which basically contains a float array of 3 values that represent the space coordinates and a float array of 3 values that represent the orientation of the hand. The size of the array and the possible values are defined by constants inside the CVectorEuler class.

3.3.1.2.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        //Use ConfigurationManager as much as we want
        CVectorEuler HandPosition = m_Jaco.ConfigurationsManager.GetHandPosition();

        float XPosition = HandPosition.Position[CVectorEuler.COORDINATE_X];
        float YPosition = HandPosition.Position[CVectorEuler.COORDINATE_Y];
        float ZPosition = HandPosition.Position[CVectorEuler.COORDINATE_Z];

        float ThetaX = HandPosition.Rotation[CVectorEuler.THETA_X];
        float ThetaY = HandPosition.Rotation[CVectorEuler.THETA_Y];
        float ThetaZ = HandPosition.Rotation[CVectorEuler.THETA_Z];

        System.Console.WriteLine("Position X = " + XPosition);
        System.Console.WriteLine("Position Y = " + YPosition);
        System.Console.WriteLine("Position Z = " + ZPosition);

        System.Console.WriteLine("Theta X = " + ThetaX);
        System.Console.WriteLine("Theta Y = " + ThetaY);
        System.Console.WriteLine("Theta Z = " + ThetaZ);
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}
```


3.3.1.3 Get the code version

3.3.1.3.1 Description

This functionality provides the code version of all firmware that are inside Jaco. The possible values are defined by constants (VERSION_XXXX) inside the class CJacoArm.

3.3.1.3.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        int[] CodeVersion;
        String DSPVersion = "";

        CodeVersion = m_Jaco.ConfigurationsManager.GetCodeVersion();

        //We convert the hexadecimal number into a decimal number.
        DSPVersion = CodeVersion[CJacoArm.VERSION_DSP].ToString("x1");

        //Output of this is: DSP Code version: 040105 (if version 4.01.05)
        System.Console.WriteLine("DSP Code version : " + DSPVersion);
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.1.4 Ask if Jaco is ready to communicate

3.3.1.4.1 Description

This functionality indicates if Jaco is ready to communicate. When the method returns FALSE, check if your cable is unplugged or the power is off.

3.3.1.4.2 C# Example

```
try
{
    CCypherMessage cypherPass;

    //MyValidPassword is a string containing the password.
    cypherPass = Crypto.GetInstance().Encrypt(MyValidPassword);
    CJacoArm m_Jaco = new CJacoArm(cypherPass);

    if (m_Jaco.JacoIsReady())
    {
        System.Console.WriteLine("Jaco is ready.");
    }
    else
```

```

    {
        System.Console.WriteLine("Jaco is not ready.");
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.3.2 Client configuration

3.3.2.1 Set the client configuration

3.3.2.1.1 Description

This functionality send to Jaco the client's configuration specified in the input parameters. Note that all values of the client's configuration must be within acceptable range. Those ranges are specified in the class Kinova.DLL.Data.Jaco.CThreshold.

3.3.2.1.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        CClientConfigurations cfg = new CClientConfigurations();

        // ...
        // Initialize the configuration
        // ...

        m_Jaco.ConfigurationsManager.SetClientConfigurations(cfg);
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.3.2.2 Get the client configuration

3.3.2.2.1 Description

This functionality gets the actual client's configuration from Jaco.

3.3.2.2.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

```

```

    if (m_Jaco.JacoIsReady())
    {
        CClientConfigurations cfg = new CClientConfigurations();

        cfg = m_Jaco.ConfigurationsManager.GetClientConfigurations();
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.3.2.3 Serialize the client configuration

3.3.2.3.1 Description

This functionality provides a way to serialize on disk a CClientConfigurations object in a binary file. If you don't specify a name for your serialization, the object will be serialized at PathCatalog.BASE_JACOAPI_CLIENTCONFIG_PATH. If you specify a name, the object will be serialized at PathCatalog.BASE_JACOAPI_PROFILES_PATH/YourName. In both case, the name of the file will be ClientConfig.bin.

3.3.2.3.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ConfigurationsManager.SerializeClientConfiguration("MyClientProfile");
        //Your client profile has been saved in a binary file at location :
        //PathCatalog.BASE_API_PROFILES_PATH.
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.3.2.4 Deserialize the client configuration

3.3.2.4.1 Description

This functionality provides a way to deserialize a CClientConfigurations object from a binary file on disk. If your serialization had a specific name, you have to specify it when you deserialize.

3.3.2.4.2 C# Example

```

try

```

```

{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ConfigurationsManager.DeserializeClientConfiguration();
        //Your client profile has been load from a binary file at the location :
        //PathCatalog.BASE_API_CLIENTCONFIG_PATH.
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.3.3 Protection zones

3.3.3.1 Set the protection zone list

3.3.3.1.1 Description

This functionality sends and set the protection zones to Jaco. The method takes a CZoneList object as parameters.

3.3.3.1.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        CZoneList ZoneList = new CZoneList();
        // ...
        // Initialize the list
        // ...
        try
        {
            m_Jaco.ConfigurationsManager.SetProtectionZones(ZoneList);
            System.Console.WriteLine("SUCCESS");
        }
        catch (Exception ex)
        {
            System.Console.WriteLine("ERROR");
        }
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.3.3.2 Get the protection zone list

3.3.3.2.1 Description

This functionality gets the protection zones from Jaco. The method returns a CZoneList object that contains a list of protection zone.

3.3.3.2.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        CZoneList ZoneList = new CZoneList();

        ZoneList = m_Jaco.ConfigurationsManager.GetProtectionZones();
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.3.3 Delete all protection zones

3.3.3.3.1 Description

This functionality erases all active protection zones in Jaco.

3.3.3.3.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ConfigurationsManager.DeleteAllProtectionsZones();
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.3.4 Serialize the protection zone list

3.3.3.4.1 Description

This functionality provides a way to serialize on disk a CZoneList object in a binary file. If you don't specify a name for your serialization, the object will be serialized at PathCatalog.BASE_JACOAPI_PROTECTIONZONE_PATH. If you specify a name, the object

will be serialized at PathCatalog.BASE_JACOAPI_PROFILES_PATH/YourName. In both case, the name of the file will be ProtectionZones.bin.

3.3.3.4.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {

        m_Jaco.ConfigurationsManager.SerializeZoneConfiguration("MyZoneProfile");
        //Your zone profile has been saved in a binary file at location :
        //PathCatalog.BASE_API_PROFILES_PATH.
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.3.5 Deserialize the protection zone list

3.3.3.5.1 Description

This functionality provides a way to deserialize a CZoneList object from a binary file on disk. If your serialization had a specific name, you have to specify it when you deserialize the structure.

3.3.3.5.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ConfigurationsManager.DeserializeZoneConfiguration("MyZoneProfile");
        //Your zone profile has been load from a binary file at location :
        //PathCatalog.BASE_API_PROFILES_PATH.
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.4 Control mapping

3.3.4.1 Set the control mapping

3.3.4.1.1 Description

This functionality sends and set the control mapping of Jaco. The method takes a CControlMappingCharts object as parameters.

3.3.4.1.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        CControlMappingCharts MappingCharts = new CControlMappingCharts();

        // ...
        //Initialize the mapping
        // ...

        m_Jaco.ConfigurationsManager.SetControlMappingCharts(MappingCharts);

        //Your mapping charts is now operational inside JACO
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.4.2 Get the control mapping

3.3.4.2.1 Description

This functionality gets the protection zones from Jaco. The method returns a CControlMappingCharts object that contains a list of CControlMapping.

3.3.4.2.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        CControlMappingCharts MappingCharts = new CControlMappingCharts();

        m_Jaco.ConfigurationsManager.GetControlMappingCharts();
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.4.3 Serialize the control mapping

3.3.4.3.1 Description

This functionality provides a way to serialize on disk a CControlMappingCharts object in a binary file. If you don't specify a name for your serialization, the object will be serialized at PathCatalog.BASE_JACOAPI_MAPPING_PATH. If you specify a name, the object will be serialized at PathCatalog.BASE_JACOAPI_PROFILES_PATH/YourName. In both case, the name of the file will be MappingCharts.bin.

3.3.4.3.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));
    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ConfigurationsManager.SerializeControlMappingCharts();
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.4.4 Deserialize the control mapping

3.3.4.4.1 Description

This functionality provides a way to deserialize a CControlMappingCharts object from a binary file on disk. If your serialization had a specific name, you have to specify it when you deserialize.

3.3.4.4.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ConfigurationsManager.DeserializeControlMappingCharts("MyMapping");
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}
```


3.3.5 Profile

3.3.5.1 Load a profile

3.3.5.1.1 Description

This functionality provides a way to load configuration profile on disk. Basically, it works the same way as all other deserialization of the API but you are forced to specify a name.

3.3.5.1.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ConfigurationsManager.LoadProfileBackup("MyJACOProfile");
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.3.5.2 Save a profile

3.3.5.2.1 Description

This functionality provides a way to save a configuration profile on disk. You need to specify a name for your profile. All profiles are at PathCatalog.BASE_JACOAPI_PROFILES_PATH.

3.3.5.2.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

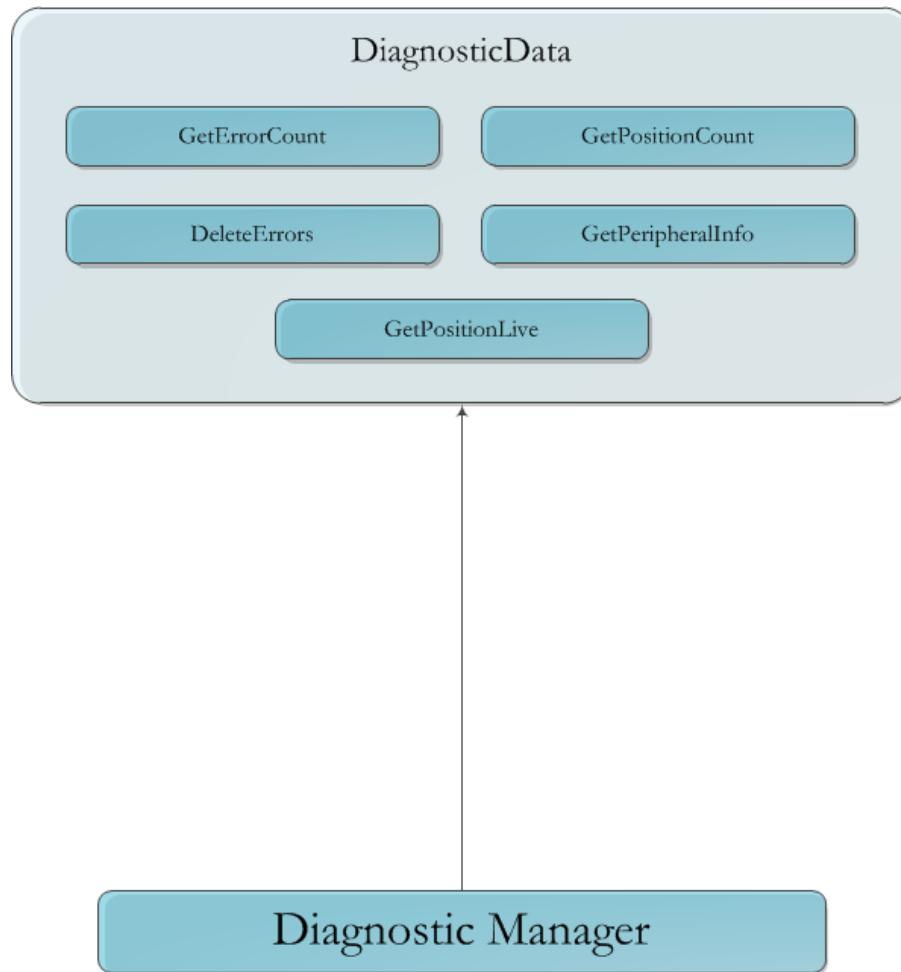
    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ConfigurationsManager.CreateProfileBackup("MyJACOProfile");
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.4 Diagnostic manager

The diagnostic manager's purpose is to get all kind of data from the robotic arm Jaco to help the developer to diagnose a problem. Of course, that data can be use in many other ways but keep in mind that if you want to modify something in the robot, you are at the wrong place.

3.4.1 Diagnostic data

Diagnostic data is basically all information about Jaco. Most of it is located in a class called CPosition.



3.4.2 CPosition

3.4.2.1 The age of the robot.

This is the total time since the assembly of the robot in millisecond. It is accessible via CPosition.TimeAbsolute.

3.4.2.2 The time since the last boot.

This is the total time since the last start up in millisecond. It is accessible via CPosition.TimeFromStartup.

3.4.2.3 The position's index in the circular list since start up.

This is the start up index of the Jaco's storage circular list. When Jaco is powered up, it automatically store information in a circular list within the main board. It is accessible via `CPosition.IndexStartup`.

3.4.2.4 Recording time of each position.

This represents the recording rate in second. For example, if it returns 0.5, it means that information is recorded every 0.5 second. It is accessible via `CPosition.TimeStampSaving`.

3.4.2.5 The voltage.

The actual voltage in volt. It is accessible via `CPosition.SupplyVoltage`.

3.4.2.6 The current consumed.

It is the actual current consumed of the arm in Amp. It is accessible via `CPosition.CurrentConsumed`.

3.4.2.7 The average power.

This is the average power consumed by the arm in watt. It is accessible via `CPosition.AveragePower`.

3.4.2.8 Value of the acceleration captor X

The value of returned by the acceleration captor X in G. It is accessible via `CPosition.AccelerationCaptorX`.

3.4.2.9 Value of the acceleration captor Y

The value of returned by the acceleration captor Y in G. It is accessible via `CPosition.AccelerationCaptorY`.

3.4.2.10 Value of the acceleration captor Z

The value of returned by the acceleration captor Z in G. It is accessible via `CPosition.AccelerationCaptorZ`.

3.4.2.11 The code's version.

This is the version of the code that is running in the main board. It is accessible via `CPosition.CodeVersion`.

3.4.2.12 The code's revision.

This is the revision number of the code that is running in the main board. It is accessible via `CPosition.CodeRevision`.

3.4.2.13 The control operator type.

This represents the entity that controls the robotic arm Jaco. The value returned is described by the enum `CJacoStructures.ControlOperator`. Most of the time, as a software developer, you should be preoccupied by those 3 values: `ExternalJoystick3Axis`, `ExternalJoystick2Axis` and `GUI`. `ExternalJoystick3Axis` means that the Kinova joystick (3-axis mode) is controlling the arm, `ExternalJoystick2Axis` means that the Kinova joystick (2-axis mode) is controlling the arm and

GUI means that the API is controlling the arm. Note that `CJacoStructures.ControlOperator` also represents the index of the control mapping charts.

3.4.2.14 The hand mode.

This is the mode in which the hand is being used. The value returned is described the enum `CJacoStructures.HandMode`. It is accessible via `CPosition.HandMode`.

3.4.2.15 Quantity of joints that connected

This is the connected joint count. It is accessible via `CPosition.ConnectedJointQuantity`.

3.4.2.16 The type of position

This is the type of the actual position. It is described by the enum `CJacoStructures.PositionType`. It is accessible via `CPosition.PositionType`.

3.4.2.17 Error count on main SPI

This is the error count on the main SPI interface. It is accessible via `CPosition.NbErrorsSpiPrincipal`.

3.4.2.18 Error count on external SPI

This is the error count on the external SPI interface. It is accessible via `CPosition.NbErrorsSpiExternal`.

3.4.2.19 Error count on main CAN

This is the error count on the main CAN interface. It is accessible via `CPosition.NbErrorsCanPrincipal`.

3.4.2.20 Error count on external CAN

This is the error count on the external CAN interface. It is accessible via `CPosition.NbErrorsCanExternal`.

3.4.2.21 Joystick activity flag

It is a flag that indicates if the joystick is connected or not. It is accessible via `CPosition.SystemStatus.JoystickActive`.

3.4.2.22 The retract status

This indicates the retract status. With this value, you can know if you are in a normal position, a retract position or in a ready position (HOME). It is described by the enum `CJacoStructures.RetractMode`. It is accessible via `CPosition.SystemStatus.RetractStatus`.

3.4.2.23 Drinking mode flag

This is a flag that indicates if Jaco is in drinking mode. It is accessible via `CPosition.SystemStatus.DrinkingMode`.

3.4.2.24 The arm laterality

That indicates if the Jaco is right handed or left handed. It is accessible via `CPosition.SystemStatus.ArmLaterality`.

3.4.2.25 Translation activity flag

This is a flag that indicates if Jaco is currently in cartesian translation mode. It is accessible via `CPosition.SystemStatus.TranslationActive`.

3.4.2.26 Rotation activity flag

This is a flag that indicates if Jaco is currently in a cartesian rotation mode. It is accessible via `CPosition.SystemStatus.RotationActive`.

3.4.2.27 Fingers activity flag

This is flag that indicates if Jaco is currently in finger mode. It is accessible via `CPosition.SystemStatus.FingersActive`.

3.4.2.28 Warning force overcharge (arm) flag

This is a flag that warns about force overcharge on the arm. It is accessible via `CPosition.SystemStatus.WarningOverchargeForce`.

3.4.2.29 Warning force overcharge (fingers) flag

This is a flag that warns about force overcharge on the fingers. It is accessible via `CPosition.SystemStatus.WarningOverchargeFingers`.

3.4.2.30 Warning low voltage warning flag

This is a flag that warns about low voltage in the arm. It is accessible via `CPosition.SystemStatus.WarningLowVoltage`.

3.4.2.31 Major error occurred flag

This is a flag that indicates if a major error occurred at this position.

3.4.2.32 The actual cartesian position

This is the actual cartesian position of the hand. The values (meter) are contained in an object called `CVectorEuler`. `CVectorEuler` contains the 3 cartesian positions (translation) X, Y and Z. It also contains the 3 Euler angle to establish the orientation (rotation) of the hand. The translation reference of the hand is the center top of Jaco's base (a little lower than the center of the first joint). The reference of the orientation is the angle of the first joint. Euler angle order is XYZ. It is accessible via `CPosition.UserPosition.Position`.

3.4.2.33 The actual angular position

This is the actual angular position of Jaco. The values (degrees) are in an object called `CVectorAngle` which contains an array of 6 angles (joint 1 to 6). It is accessible via `CPosition.UserPosition.AnglesJoints`.

3.4.2.34 Current on each joint

This is the value, in Amp, of the current sensor of each joint of Jaco. The value is in an object called `CVectorAngle` which contains an array of 6 angles (joint 1 to 6). It is accessible via `CPosition.UserCurrent.AnglesJoints`.

3.4.2.35 Force on each joint

This is the force value, in Newton, of each joint of Jaco. The value is in an object called CVectorAngle which contains an array of 6 angles (joint 1 to 6). It is accessible via CPosition.UserForce.AnglesJoints.

3.4.2.36 The actual limitation

These are the actual limitation in Jaco represented by an object called CZoneLimitation. It is accessible via CPosition.ActualLimitation.

3.4.2.37 Value of the joystick connected

This represents the value received by the Kinova joystick. It is accessible via CPosition.JoystickValue.

3.4.2.38 Temperature on each joint

This is the temperature, in °C, of each joint of Jaco. It is accessible via CPosition.ActuatorsTemperatures which is a float array of CPosition.ACTUATORS_TEMPERATURE_QUANTITY values.

3.4.2.39 Temperature on each finger

This is the temperature, in °C, of each joint of Jaco, It is accessible via CPosition.FingersTemperatures, which is a float array of CPosition.FINGERS_TEMPERATURE_QUANTITY values.

3.4.2.40 Comm error count on each actuator

This is the communication error count of each joint of Jaco. It is accessible via CPosition.ActuatorsErrorsCommunication which is a float array of CPosition.ACTUATORS_ERRORS_COMMUNICATION values.

3.4.2.41 Comm error count on each finger

This is the communication error count of each finger of Jaco. It is accessible via CPosition.FingersErrorsCommunication which is a float array of CPosition.FINGERS_ERRORS_COMMUNICATION values.

3.4.2.42 Total time of control

This is the total control time since the assembly of the robot in millisecond. It is accessible via CPosition.ControlTimeAbsolute. The control time encompass the joystick control time and the emulated joystick control time.

3.4.2.43 Time of control since start up

This is the total control time since the last start-up in millisecond. It is accessible via CPosition.ControlTimeStartup. The control time encompass the joystick control time and the emulated joystick control time.

3.4.2.44 Get the errors count

3.4.2.44.1 Description

Every time that a major error occurs in Jaco, it is logged inside a memory on the main board. This functionality returns the count of those errors.

3.4.2.44.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        ulong Count = m_Jaco.DiagnosticManager.DataManager.GetErrorLogCount();

        System.Console.WriteLine("Error count : " + Count);
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.4.2.45 Get the positions count

3.4.2.45.1 Description

When the robot is powered, there is process that store information over time. The information stored corresponds to the class CPosition. This functionality indicates the quantity of CPosition that has been stored.

3.4.2.45.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        int Count = m_Jaco.DiagnosticManager.DataManager.GetPositionLogCount();

        System.Console.WriteLine("CPosition count : " + Count);
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.4.2.46 Get a specific position from Jaco

3.4.2.46.1 Description

This functionality returns a specific position from Jaco. It is mostly used to get data from the past. If you want to get live data, you should use the method `GetPositionLiveFromJaco`. You can specify the index of the position where it is stored inside Jaco. If you use the value returned by the method `GetPositionLogCount` as an index, you will have the latest recorded position. A new `CPosition` object is recorded every 0.5 second.

The example will show how to get data of the last 25 seconds of Jaco. You will see that sometime, the first positions are not valid because of the synchronization between your method call and the recording system of Jaco. This is normal and you just have to ignore those few first positions.

3.4.2.46.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    List<CPosition> list = new List<CPosition>();

    int index = m_Jaco.DiagnosticManager.DataManager.GetPositionLogCount();

    for (int i = index; i > index - 50; i--)
    {
        list.Add(m_Jaco.DiagnosticManager.DataManager.GetPositionFromJaco(i));
    }

    for (int i = 0; i < list.Count; i++)
    {
        System.Console.WriteLine("time absolute #" + i + " = " + list[i].TimeAbsolute);
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.4.2.47 Get the peripherals information

3.4.2.47.1 Description

This functionality returns information about the peripheral that is connected to Jaco. Most of the time, it is the kinova joystick.

3.4.2.47.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
```



```

        CPeripheralInformation info =
            m_Jaco.DiagnosticManager.DataManager.GetPeripheralInformationFromJaco();

        System.Console.WriteLine("Device ID : " + info.DeviceID);
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.4.2.48 Get the actual position

3.4.2.48.1 Description

This functionality returns the actual position of the robot along with all information relative to it. That information is contained in the class CPosition. Basically, when you need information of all kind about Jaco, you call that method. If you are looking for the cartesian or the angular position of Jaco, it is possible via this method but you should use the [\[Get hand's position\]](#) or [\[Get joints positions\]](#) functionalities, it will be a lot faster.

3.4.2.48.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {

        //We get a list of position and we take the first element.
        CPosition Position =
            m_Jaco.DiagnosticManager.DataManager.GetPositionLogLiveFromJaco();

        System.Console.WriteLine("Average power consumed : " + Position.AveragePower);
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.4.2.49 Delete all errors

3.4.2.49.1 Description

This functionality deletes all major errors listed inside Jaco.

3.4.2.49.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())

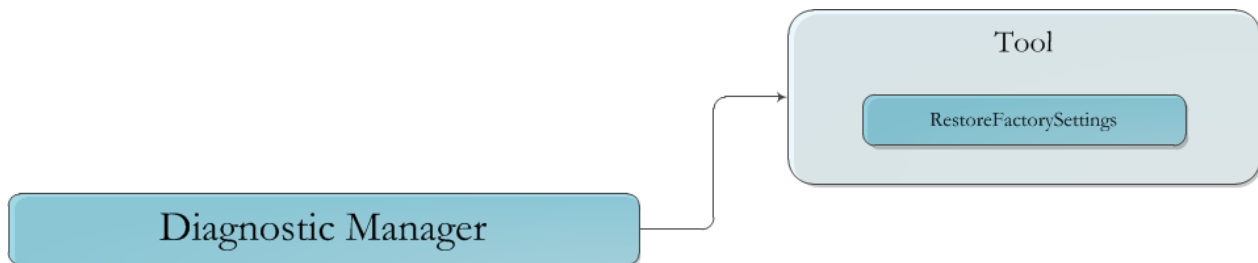
```

```

    {
        m_Jaco.DiagnosticManager.DataManager.DeleteErrorLog();
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.4.3 Tool



3.4.3.1 Restore the default factory settings

3.4.3.1.1 Description

This functionality restores Jaco to its factory default settings. You have to reboot the robot for the action to take its full effect. Before doing that, if you have special configuration profile, it is suggested that you [save a copy of the profile](#) because all configurations in Jaco will be overwritten.

3.4.3.1.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

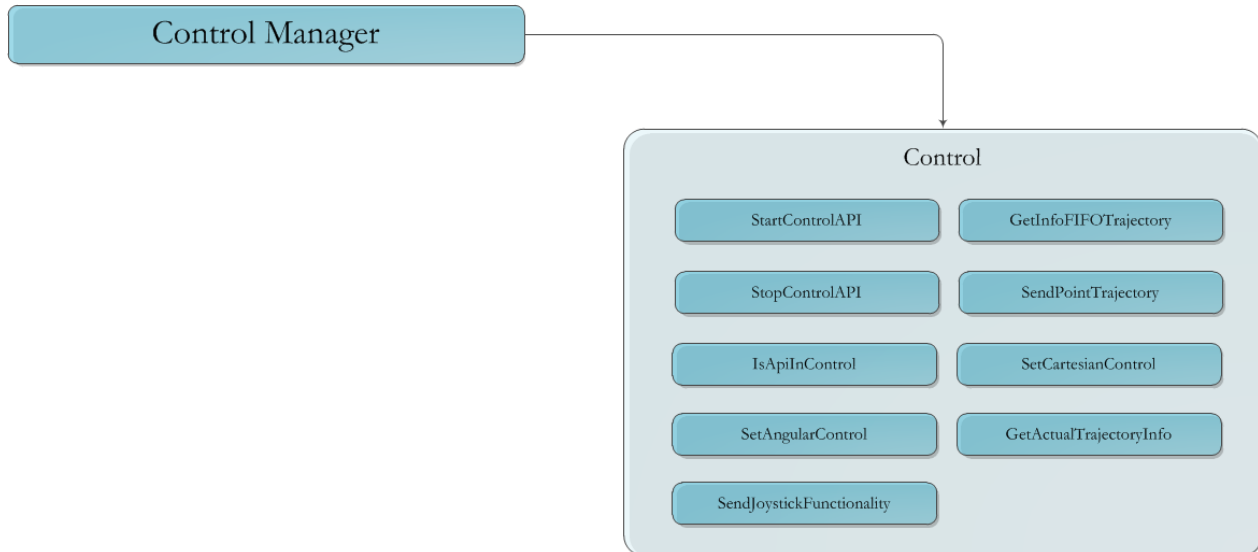
    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.DiagnosticManager.ToolManager.RestoreFactorySettings();

        //REBOOT your Jaco to complete the operation.
        //You now have a factory default JACO.
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.5 Control manager

This manager's purpose is to manage all method that physically moves Jaco. From there, you have access to the [trajectory features](#) and the [Kinova joystick emulation](#) as well. It is also the right place to switch the robotic arm between [angular](#) and [cartesian](#) control type.



3.5.1.1 Start to control Jaco with the API

3.5.1.1.1 Description

This functionality tells Jaco that from now on, it is the API that will control it. You have to use this method each time you start a procedure that require an API control. Note that if you move the kinova joystick while you are controlling via the API, the joystick will take back the control because it has a higher priority than the API. In this case, you need to re-execute the method to tell Jaco that the API is taking back the control.

3.5.1.1.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        //From now on, API can move the arm.
        m_Jaco.ControlManager.StartControlAPI();

        //The API will lose control if the StopControlAPI is called, if the USB connection
        //broke or if another interface with a higher priority move the arm.
        //For example, the 3 axis joystick directly connected to JACO have an higher
        //priority.
    }
}
catch (Exception)
{
}
```

```
System.Console.WriteLine("EXCEPTION");  
}
```

3.5.1.2 Stop to control Jaco with the API

3.5.1.2.1 Description

This functionality tells Jaco that from now on, the API is no longer controlling Jaco. See section [\[Start to control Jaco with the API\]](#) for information about the control priority.

3.5.1.2.2 C# Example

```
try  
{  
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));  
  
    if (m_Jaco.JacoIsReady())  
    {  
  
        //From now on, API can move the arm.  
        m_Jaco.ControlManager.StopControlAPI();  
  
    }  
}  
catch (Exception)  
{  
    System.Console.WriteLine("EXCEPTION");  
}
```

3.5.1.3 Get information about the trajectory FIFO

3.5.1.3.1 Description

This functionality returns information about the trajectory FIFO inside Jaco. Most of the time, this method is used to know if Jaco is still executing a trajectory. As an example, if `CInfoFIFOtrajectory.StillInFIFO` equals 0, that means that there is no trajectory at the moment in Jaco.

3.5.1.3.2 C# Example

```
try  
{  
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));  
  
    if (m_Jaco.JacoIsReady())  
    {  
        //Declaration of the data structure that will hold information about Jaco's  
        //trajectories FIFO.  
        CInfoFIFOtrajectory info;  
  
        //We get the information from the robotic arm Jaco.  
        info = m_Jaco.ControlManager.GetInfoFIFOtrajectory();  
  
        System.Console.WriteLine("Quantity of trajectories still waiting to be executed :  
" + info.StillInFIFO);  
        System.Console.WriteLine("FIFO max size : " + info.MaxSize);  
    }  
}
```

```

    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.5.1.4 Send a trajectory

3.5.1.4.1 Description

This functionality adds a trajectory to Jaco's trajectory FIFO. Basically, a trajectory is a list of point that you want Jaco to move over. There is 2 methods that send trajectories to Jaco, the first one, SendTrajectoryFunctionnality, send the complete trajectory to the robot and the other one, SendBasicTrajectory, send only physical position of each point. SendBasicTrajectory is faster if you don't need to apply restriction on the robot.

3.5.1.4.2 C# Example (angular trajectory)

```

try
{
    CCypherMessage cypherPass;

    //MyValidPassword is a string containing the password.
    cypherPass = Crypto.GetInstance().Encrypt(MyValidPassword);
    CJacoArm m_Jaco = new CJacoArm(cypherPass);

    CPointsTrajectory m_PointsTrajectory = new CPointsTrajectory();
    if (m_Jaco.JacoIsReady())
    {
        //From now on, API can move JACO.
        m_Jaco.ControlManager.StartControlAPI();

        CVectorAngle vector = new CVectorAngle();
        CTrajectoryInfo point1 = new CTrajectoryInfo();

        point1.UserPosition.AnglesJoints = vector;
        point1.UserPosition.PositionType = CJacoStructures.PositionType.AngularPosition;

        point1.UserPosition.HandMode = CJacoStructures.HandMode.PositionMode;

        //This position is near the READY(HOME) position.
        point1.UserPosition.AnglesJoints.Angle[CVectorAngle.JOINT_1] = 273.4177f;
        point1.UserPosition.AnglesJoints.Angle[CVectorAngle.JOINT_2] = 185.4481f;
        point1.UserPosition.AnglesJoints.Angle[CVectorAngle.JOINT_3] = 114.9165f;
        point1.UserPosition.AnglesJoints.Angle[CVectorAngle.JOINT_4] = 251.9101f;
        point1.UserPosition.AnglesJoints.Angle[CVectorAngle.JOINT_5] = 80.5050f;
        point1.UserPosition.AnglesJoints.Angle[CVectorAngle.JOINT_6] = 32.8332f;

        point1.UserPosition.FingerPosition[0] = 10;
        point1.UserPosition.FingerPosition[1] = 30;
        point1.UserPosition.FingerPosition[2] = 40;

        m_PointsTrajectory.Add(point1);
    }
}

```

```

        //Here, we use the SendBasicTrajectory method but you could also
        //use the SendTrajectoryFunctionnality. SendBasicTrajectory
        //is faster because it does not send the limitation structure.
        m_Jaco.ControlManager.SendBasicTrajectory(m_PointsTrajectory);
    }
}
catch (Exception)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.5.1.4.3 C# Example (cartesian trajectory)

```

try
{
    CCypherMessage cypherPass;

    //MyValidPassword is a string containing the password.
    cypherPass = Crypto.GetInstance().Encrypt(MyValidPassword);
    CJacoArm m_Jaco = new CJacoArm(cypherPass);

    CPointsTrajectory m_PointsTrajectory = new CPointsTrajectory();
    if (m_Jaco.JacoIsReady())
    {
        //From now on, API can move JACO.
        m_Jaco.ControlManager.StartControlAPI();

        CTrajectoryInfo point1 = new CTrajectoryInfo();

        //This position is near the READY(HOME) position.
        point1.UserPosition.Position.Position[CVectorEuler.COORDINATE_X] = 0.20f;
        point1.UserPosition.Position.Position[CVectorEuler.COORDINATE_Y] = -0.41f;
        point1.UserPosition.Position.Position[CVectorEuler.COORDINATE_Z] = 0.43f;
        point1.UserPosition.Position.Rotation[CVectorEuler.THETA_X] = -1.5f;
        point1.UserPosition.Position.Rotation[CVectorEuler.THETA_Y] = 0.48f;
        point1.UserPosition.Position.Rotation[CVectorEuler.THETA_Z] = -3.17f;

        point1.UserPosition.FingerPosition[CUserPosition.FINGER_1] = 39.25f;
        point1.UserPosition.FingerPosition[CUserPosition.FINGER_2] = 39.25f;
        point1.UserPosition.FingerPosition[CUserPosition.FINGER_3] = 40.75f;

        point1.UserPosition.PositionType = CJacoStructures.PositionType.CartesianPosition;

        m_PointsTrajectory.Add(point1);

        //Here, we use the SendBasicTrajectory method but you could also
        //use the SendTrajectoryFunctionnality. SendBasicTrajectory
        //is faster because it does not send the limitation structure.
        m_Jaco.ControlManager.SendBasicTrajectory(m_PointsTrajectory);
    }
}

```

3.5.1.5 Send a joystick functionality

3.5.1.5.1 Description

This functionality sends a virtual joystick command which will be executed by Jaco according to the active mapping in effect.

3.5.1.5.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        //From now on, API can move JACO and we assume the active mapping is GUI.
        m_Jaco.ControlManager.StartControlAPI();

        //Default mapping for the API is that the third button(index = 2) control the
        //retract&ready movement.
        CJoystickValue Value = new CJoystickValue();
        Value.ButtonValue[2] = 1;

        //Send a PRESS event on the third button(index 2).
        m_Jaco.ControlManager.SendJoystickFunctionality(Value);

        //We wait 4 second to simulate someone holding the button.
        Thread.Sleep(4000);

        Value.ButtonValue[2] = 0;

        //Send a RELEASE event on the third button(index 2).
        m_Jaco.ControlManager.SendJoystickFunctionality(Value);

        m_Jaco.ControlManager.StopControlAPI();

        // ...
        //A PRESS and a RELEASE is considered like you have clicked the second button.
        //Based on the mapping that has been configured, JACO will perform
        //the functionality. For example, if the functionality Retract_ReadyToUse is set
        //on the second button, JACO will move according to that.
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.5.1.6 Ask if API is controlling Jaco

3.5.1.6.1 Description

This functionality asks Jaco if it is still the API that is controlling Jaco. If it returns false and you want to take back control of Jaco, use functionality [\[Start to control Jaco with API\]](#)

3.5.1.6.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        if(m_Jaco.ControlManager.IsApiInControl())
        {
            System.Console.WriteLine("API is in control.");
        }
        else
        {
            System.Console.WriteLine("API is not in control.");
        }
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.5.1.7 Set Jaco to Cartesian control

3.5.1.7.1 Description

This functionality switches the arm to cartesian control.

3.5.1.7.2 C# Example

```
try
{
    CCypherMessage cypherPass;

    //MyValidPassword is a string containing the password.
    cypherPass = Crypto.GetInstance().Encrypt(MyValidPassword);
    CJacoArm m_Jaco = new CJacoArm(cypherPass);

    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ControlManager.SetCartesianControl();

        //JACO is now in Angular control.
        System.Console.WriteLine("JACO is now in cartesian mode");
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```


3.5.1.8 Set Jaco to angular control

3.5.1.8.1 Description

This functionality switches the arm to angular control.

3.5.1.8.2 C# Example

```
try
{
    CCypherMessage cypherPass;

    //MyValidPassword is a string containing the password.
    cypherPass = Crypto.GetInstance().Encrypt(MyValidPassword);
    CJacoArm m_Jaco = new CJacoArm(cypherPass);

    if (m_Jaco.JacoIsReady())
    {
        m_Jaco.ControlManager.SetAngularControl();

        //JACO is now in Angular control.
        System.Console.WriteLine("JACO is now in angular mode");
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.5.1.9 Get information about the current trajectory

3.5.1.9.1 Description

This functionality returns information about the trajectory that Jaco is currently executing.

3.5.1.9.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {

        //Start a trajectory...

        //Give some time to the trajectory to be executed.
        Thread.Sleep(3000);

        //From now on, API can move the arm.
        CTrajectoryInfo info = m_Jaco.ControlManager.GetActualTrajectoryInfo();

        //Show information that we received
        System.Console.WriteLine("Are the limitation currently active? : " +
            info.LimitationActive);
    }
}
```

```

}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.5.1.10 Erase all trajectories in Jaco

3.5.1.10.1 Description

This functionality erase all trajectory from Jaco's trajectory FIFO.

3.5.1.10.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        //We create a trajectory
        CPointsTrajectory trajectory = new CPointsTrajectory();
        CTrajectoryInfo point1 = new CTrajectoryInfo();
        CTrajectoryInfo point2 = new CTrajectoryInfo();
        CTrajectoryInfo point3 = new CTrajectoryInfo();
        CTrajectoryInfo point4 = new CTrajectoryInfo();

        CVectorEuler ActualPosition = m_Jaco.ConfigurationsManager.GetHandPosition();

        point1.UserPosition.HandMode = CJacoStructures.HandMode.PositionMode;
        point1.UserPosition.Position = ActualPosition;
        point1.UserPosition.FingerPosition[0] = -200;
        point1.UserPosition.FingerPosition[1] = -200;
        point1.UserPosition.FingerPosition[2] = -200;

        point2.UserPosition.HandMode = CJacoStructures.HandMode.PositionMode;
        point2.UserPosition.Position = ActualPosition;
        point2.UserPosition.FingerPosition[0] = 200;
        point2.UserPosition.FingerPosition[1] = 200;
        point2.UserPosition.FingerPosition[2] = 200;

        point3.UserPosition.HandMode = CJacoStructures.HandMode.PositionMode;
        point3.UserPosition.Position = ActualPosition;
        point3.UserPosition.FingerPosition[0] = -200;
        point3.UserPosition.FingerPosition[1] = -200;
        point3.UserPosition.FingerPosition[2] = -200;

        point4.UserPosition.HandMode = CJacoStructures.HandMode.PositionMode;
        point4.UserPosition.Position = ActualPosition;
        point4.UserPosition.FingerPosition[0] = 200;
        point4.UserPosition.FingerPosition[1] = 200;
        point4.UserPosition.FingerPosition[2] = 200;

        trajectory.Add(point1);
        trajectory.Add(point2);
        trajectory.Add(point3);
    }
}

```

```

        trajectory.Add(point4);

        m_Jaco.ControlManager.StartControlAPI();

        //We send the trajectory.
        m_Jaco.ControlManager.SendTrajectoryFunctionnality(trajectory);

        //Anytime while the robot is executing the trajectory the user can press a button
        to stop the trajectory.
        System.Console.ReadKey();

        m_Jaco.ControlManager.EraseTrajectories();

        //We wait a bit...
        Thread.Sleep(4000);

        //The robot is ready to receive other trajectory so we resend another one.
        m_Jaco.ControlManager.SendTrajectoryFunctionnality(trajectory);
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.5.1.11 Get information about the cartesian position

3.5.1.11.1 Description

This functionality returns information specific to the cartesian position.

3.5.1.11.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        //Start a trajectory...

        //Give some time to the trajectory to be executed.
        Thread.Sleep(3000);

        //From now on, API can move the arm.
        CTrajectoryInfo info = m_Jaco.ControlManager.GetActualTrajectoryInfo();

        //Show information that we received
        System.Console.WriteLine("Are the limitation currently active? : " +
            info.LimitationActive);
    }
}
catch (Exception ex)
{
}

```

```
System.Console.WriteLine("EXCEPTION");  
}
```

3.5.1.12 Get information about the angular position

3.5.1.12.1 Description

This functionality returns information specific to the angular position.

3.5.1.12.2 C# Example

```
try  
{  
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));  
  
    if (m_Jaco.JacoIsReady())  
    {  
  
        //Declaration of the data structure that will hold information about the angular  
        position.  
        CAngularInfo info;  
  
        //We get the information from the robotic arm Jaco.  
        info = m_Jaco.ControlManager.GetPositioningAngularInfo();  
  
        System.Console.WriteLine("    Joint 1 : " + info.Joint1);  
        System.Console.WriteLine("    Joint 2 : " + info.Joint2);  
        System.Console.WriteLine("    Joint 3 : " + info.Joint3);  
        System.Console.WriteLine("    Joint 4 : " + info.Joint4);  
        System.Console.WriteLine("    Joint 5 : " + info.Joint5);  
        System.Console.WriteLine("    Joint 6 : " + info.Joint6);  
        System.Console.WriteLine("    Finger #1 : " + info.Finger1);  
        System.Console.WriteLine("    Finger #2 : " + info.Finger2);  
        System.Console.WriteLine("    Finger #3 : " + info.Finger3);  
  
    }  
}  
catch (Exception ex)  
{  
    System.Console.WriteLine("EXCEPTION");  
}
```

3.5.1.13 Get information about the cartesian command

3.5.1.13.1 Description

This functionality returns information specific to the cartesian command.

3.5.1.13.2 C# Example

```
try  
{  
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));  
  
    if (m_Jaco.JacoIsReady())  
    {
```

```

//Declaration of the data structure that will hold information about the
cartesian command.
CCartesianInfo info;

//We get the information from the robotic arm Jaco.
info = m_Jaco.ControlManager.GetCommandCartesianInfo();

System.Console.WriteLine("      Command X : " + info.X);
System.Console.WriteLine("      Command Y : " + info.Y);
System.Console.WriteLine("      Command Z : " + info.Z);
System.Console.WriteLine("  Command Theta X : " + info.ThetaX);
System.Console.WriteLine("  Command Theta Y : " + info.ThetaY);
System.Console.WriteLine("  Command Theta Z : " + info.ThetaZ);
System.Console.WriteLine(" Command Finger #1 : " + info.Finger1);
System.Console.WriteLine(" Command Finger #2 : " + info.Finger2);
System.Console.WriteLine(" Command Finger #3 : " + info.Finger3);

    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.5.1.14 Get information about the angular command

3.5.1.14.1 Description

This functionality returns information specific to the angular command.

3.5.1.14.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

        if (m_Jaco.JacoIsReady())
        {
            //Declaration of the data structure that will hold information about the
            cartesian command.
            CAngularInfo info;

            //We get the information from the robotic arm Jaco.
            info = m_Jaco.ControlManager.GetCommandAngularInfo();

            System.Console.WriteLine("  Command Joint 1 : " + info.Joint1);
            System.Console.WriteLine("  Command Joint 2 : " + info.Joint2);
            System.Console.WriteLine("  Command Joint 3 : " + info.Joint3);
            System.Console.WriteLine("  Command Joint 4 : " + info.Joint4);
            System.Console.WriteLine("  Command Joint 5 : " + info.Joint5);
            System.Console.WriteLine("  Command Joint 6 : " + info.Joint6);
            System.Console.WriteLine(" Command Finger #1 : " + info.Finger1);
            System.Console.WriteLine(" Command Finger #2 : " + info.Finger2);

```

```

        System.Console.WriteLine(" Command Finger #3 : " + info.Finger3);
    }
}
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.5.1.15 Get information about the angular force

3.5.1.15.1 Description

This functionality returns information specific to the angular force.

3.5.1.15.2 C# Example

```

try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

        //Declaration of the data structure that will hold information about the
        angular force.
        CAngularInfo info;

        //We get the information from the robotic arm Jaco.
        info = m_Jaco.ControlManager.GetForceAngularInfo();

        System.Console.WriteLine(" Force Joint 1 : " + info.Joint1);
        System.Console.WriteLine(" Force Joint 2 : " + info.Joint2);
        System.Console.WriteLine(" Force Joint 3 : " + info.Joint3);
        System.Console.WriteLine(" Force Joint 4 : " + info.Joint4);
        System.Console.WriteLine(" Force Joint 5 : " + info.Joint5);
        System.Console.WriteLine(" Force Joint 6 : " + info.Joint6);
        System.Console.WriteLine(" Force Finger #1 : " + info.Finger1);
        System.Console.WriteLine(" Force Finger #2 : " + info.Finger2);
        System.Console.WriteLine(" Force Finger #3 : " + info.Finger3);
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.5.1.16 Get information about the cartesian force

3.5.1.16.1 Description

This functionality returns information specific to the cartesian force.

3.5.1.16.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        //Declaration of the data structure that will hold information about the
        //cartesian force.
        CCartesianInfo info;

        //We get the information from the robotic arm Jaco.
        info = m_Jaco.ControlManager.GetForceCartesianInfo();

        System.Console.WriteLine("      Force X : " + info.X);
        System.Console.WriteLine("      Force Y : " + info.Y);
        System.Console.WriteLine("      Force Z : " + info.Z);
        System.Console.WriteLine(" Force Theta X : " + info.ThetaX);
        System.Console.WriteLine(" Force Theta Y : " + info.ThetaY);
        System.Console.WriteLine(" Force Theta Z : " + info.ThetaZ);
        System.Console.WriteLine(" Force Finger #1 : " + info.Finger1);
        System.Console.WriteLine(" Force Finger #2 : " + info.Finger2);
        System.Console.WriteLine(" Force Finger #3 : " + info.Finger3);

    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}
```

3.5.1.17 Get information about the angular current

3.5.1.17.1 Description

This functionality returns information specific to the angular current.

3.5.1.17.2 C# Example

```
try
{
    CJacoArm m_Jaco = new CJacoArm(Crypto.GetInstance().Encrypt(MyValidPassword));

    if (m_Jaco.JacoIsReady())
    {
        //Declaration of the data structure that will hold information about the angular
        //current.
        CAngularInfo info;

        //We get the information from the robotic arm Jaco.
        info = m_Jaco.ControlManager.GetCurrentAngularInfo();

        System.Console.WriteLine(" Current Joint 1 : " + info.Joint1);
        System.Console.WriteLine(" Current Joint 2 : " + info.Joint2);
        System.Console.WriteLine(" Current Joint 3 : " + info.Joint3);
    }
}
```

```

        System.Console.WriteLine("    Current Joint 4 : " + info.Joint4);
        System.Console.WriteLine("    Current Joint 5 : " + info.Joint5);
        System.Console.WriteLine("    Current Joint 6 : " + info.Joint6);
        System.Console.WriteLine(" Current Finger #1 : " + info.Finger1);
        System.Console.WriteLine(" Current Finger #2 : " + info.Finger2);
        System.Console.WriteLine(" Current Finger #3 : " + info.Finger3);
    }
}
catch (Exception ex)
{
    System.Console.WriteLine("EXCEPTION");
}

```

3.6 Other functionalities

3.6.1.1 Get the API's version

3.6.1.1.1 Description

This functionality returns the version of the API.

3.6.1.1.2 C# Example

```

try
{
    CCypherMessage cypherPass;

    //MyValidPassword is a string containing the password.
    cypherPass = Crypto.GetInstance().Encrypt(MyValidPassword);
    CJacoArm m_Jaco = new CJacoArm(cypherPass);

    if (m_Jaco.JacoIsReady())
    {
        System.Console.WriteLine("API Version = " + m_Jaco.GetAPIVersion());
    }
}
catch (Exception ex)
{
    //Manage the Exception
}

```


4 API Remote

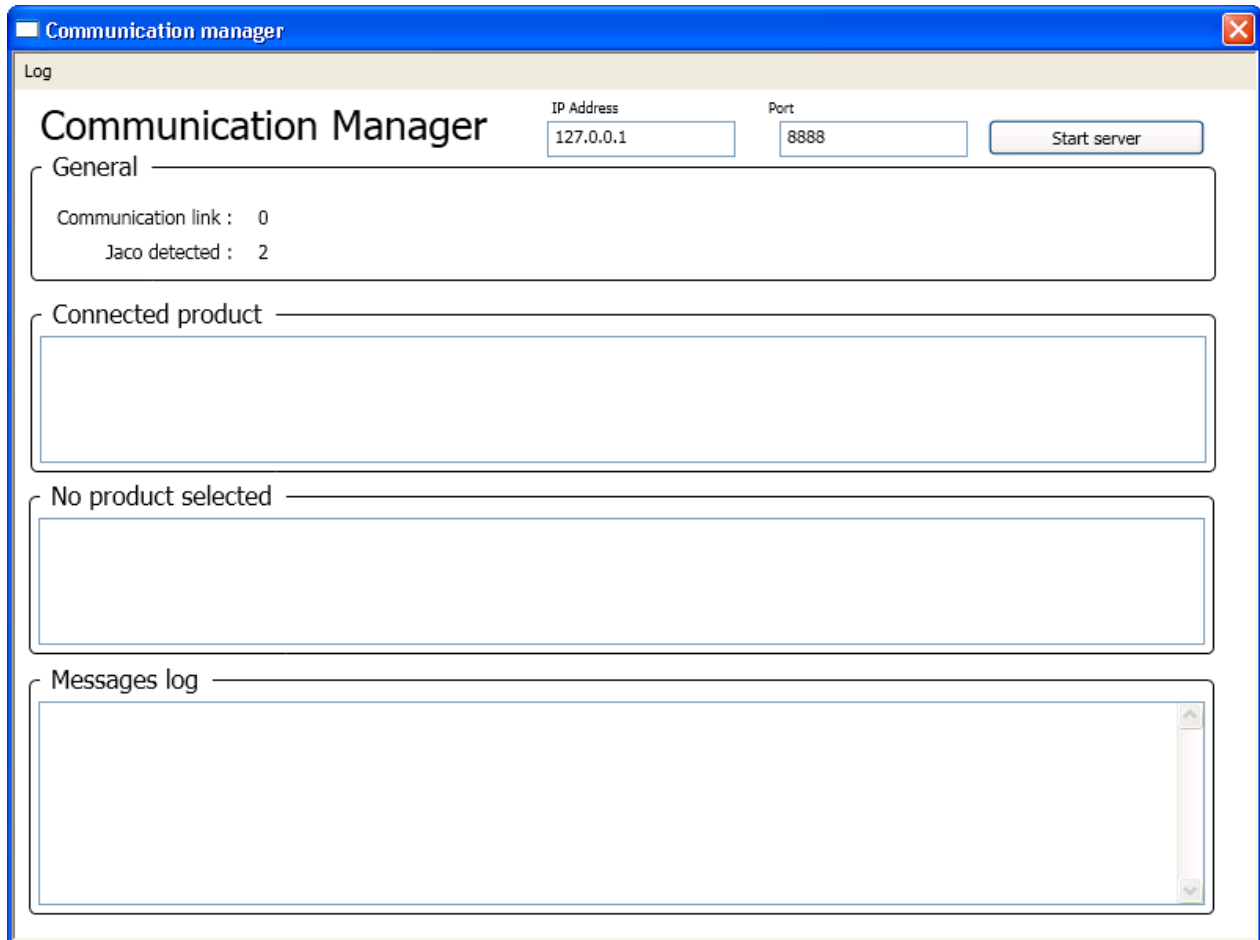
4.1 Overview

The remote API offers the option to communicate with many Jaco at the same time. It is called remote because all the commands pass through a communication server ([Kinova.GUI.CommunicationManager](#)). This version only support local server so you cannot communicate with a Jaco via a network but it will be possible in a future version.

Basically, what you can do in this version is to communicate locally with more than 1 Jaco at the same time. Note that if you want to use the remote feature of the API, you have to make sure that the server is up and running.

4.2 Server GUI

4.2.1 Windows



4.2.1.1 Start / Stop server

That button start and stop the server. If you have active connections with a product during the closure, it will be broken. You must start the server before using the remote API.

4.2.1.2 Log menu

The Log menu has only 2 items, the first one activate / deactivate the event logger on the GUI and the second is under construction so it has not been release in this version.

4.2.1.3 General tab

The general tab shows how many Jaco is connected to the server and how many connections have been made with the server.

4.2.1.4 Connected product tab

The Connected product tab shows all the products that are used by the API. If you select a product, information about it will show in the product tab.

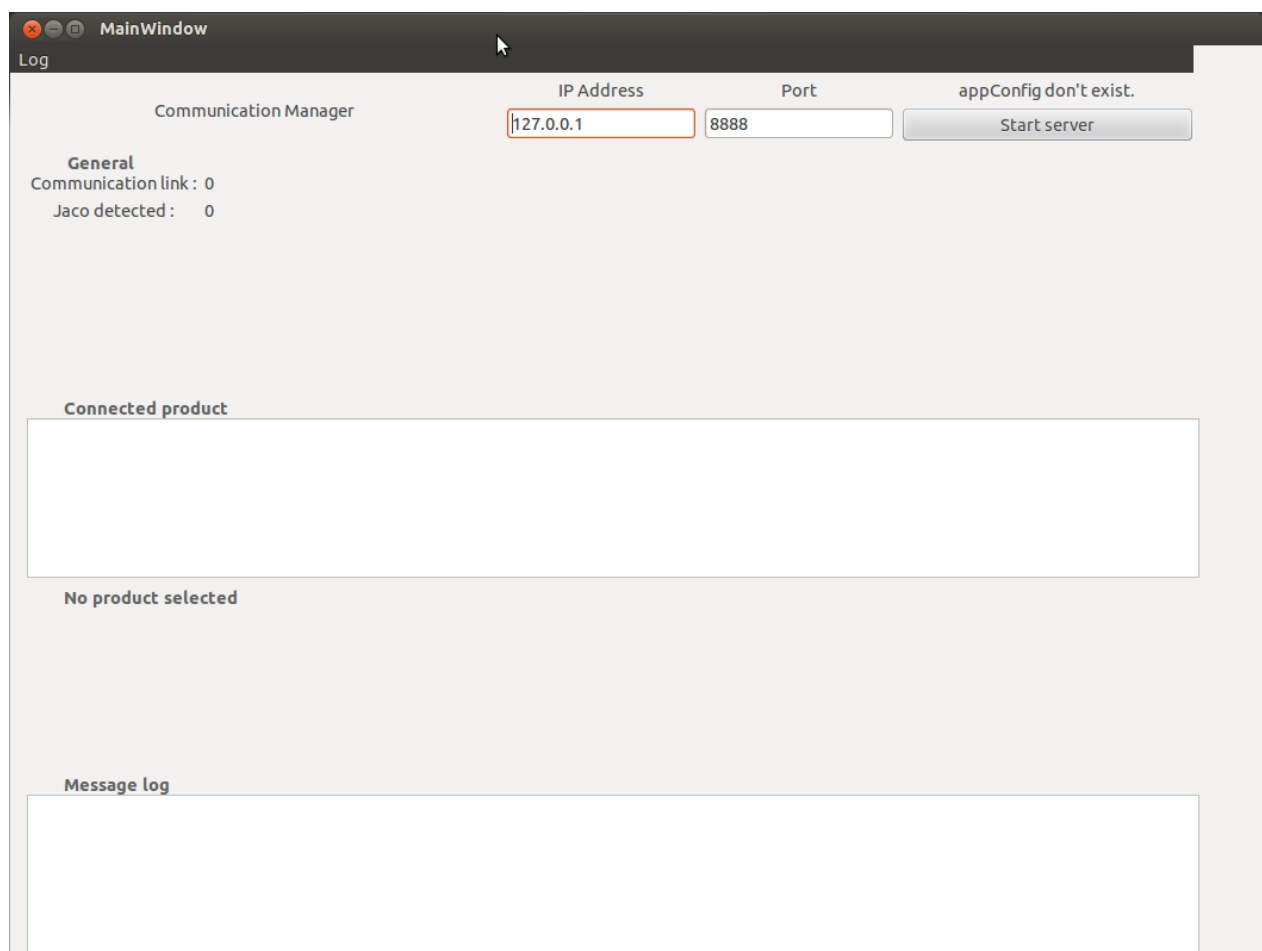
4.2.1.5 Product tab

This tab shows information about the selected product.

4.2.1.6 Message log tab

If the feature is on, it shows the event logged by the logger.

4.2.2 Ubuntu



4.3 Example

All of the examples assume that the communication server is up and running and it has been started.

4.3.1 Windows

4.3.1.1 Description

This example gets information from the server, connect to all available product and get the client configuration from all of them. Note some difference with the

4.3.1.2 Code

```
try
{
    //A Jaco connector the can be accessed remotely.
    CJacoArmRemote jaco = new CJacoArmRemote(Crypto.GetInstance().Encrypt(m_APIpass),
    IPAddress.Parse("127.0.0.1"), true);

    //A list to store all the product Jaco connected.
    List<CProductJaco> listProduct = new List<CProductJaco>();

    //We get the info from the server and store the returned command.
    CGetCommInfoCmd commInfo = jaco.GetCommunicationInfo();

    //A Initialization command that will be use for each product.
    CInitConnectionCmd cmd = new CInitConnectionCmd();

    uint AvailableConnectionCount = commInfo.ProductAvailableJaco;
    int NextID = commInfo.NextAvailableJacoID;

    AvailableConnectionCount = commInfo.ProductAvailableJaco;
    Console.WriteLine("We found " + AvailableConnectionCount + " Jaco product
connected.");

    for (int i = 0; i < AvailableConnectionCount; i++)
    {
        commInfo = jaco.GetCommunicationInfo();

        Console.WriteLine("Connecting to Jaco product ID " +
commInfo.NextAvailableJacoID);
        cmd = jaco.InitProductConnection(commInfo.NextAvailableJacoID);

        Console.WriteLine("Initializing the product");
        CProductJaco tempProduct = new CProductJaco();
        tempProduct.HandleRead = cmd.Product.HandleRead;
        tempProduct.HandleWrite = cmd.Product.HandleWrite;

        listProduct.Add(tempProduct);

        Thread.Sleep(500);
    }
}
```

```

//Now we have a list of all product connected that we can use.

CClientConfigurations tempClientConfig = new CClientConfigurations();
for (int i = 0; i < listProduct.Count; i++ )
{
    tempClientConfig =
jaco.ConfigurationsManager.GetClientConfigurations(listProduct[i]);
    Console.WriteLine("Nam of Jaco " + i + " is " + tempClientConfig.ClientName);
}

for (int i = 0; i < listProduct.Count; i++)
{
    jaco.CloseConnection(listProduct[i]);
}

System.Console.ReadLine();
}
catch (CTcpConnectionException errorTCP)
{
    System.Console.WriteLine("TCP error, make sure the server is up and running");
}
catch(Exception ex)
{
    System.Console.WriteLine("Error while executing");
}

System.Console.WriteLine("Example done...");
System.Console.Read();

```

4.3.2 Ubuntu

4.3.2.1 Description

4.3.2.2 Code

5 Troubleshooting

5.1 Activity log

Most of the functionalities are logged in a file located at:

[Windows]

(Application Data directory)\Kinova\Products\Jacosoft\API\Data\Logs\Activities

[Ubuntu]

Home/Kinova/Application Data/Products/Jacosoft/API/Data/Logs/Activities

You can use this file to have a better look of what happened before a bug.

5.2 Jacosoft

JacoSoft is an application (windows only) that provides a GUI to configure Jaco and some tools to visualize the data from Jaco. Sometime it is useful to use the Jacosoft while you are debugging your application.

5.3 Contact us

If you need help, want to write comments or want to submit a bug, please contact us at:

support@kinova.ca

