ORSAY PHYSICS
TESCAN ORSAY HOLDING

Communication protocol description

MODIFICATIONS FOLLOW-UP				
Version N°	Modifications	Author	Approbator	Date
V1.0	Initial version	SM		08/01/12
V2.5	Exit client command/ Disconnection from server	SM		06/25/14
V2.6	Add InitOne for one parameter	SM		11/26/14
The authenticity of the document is guaranted by visa on original.				

For further informations, please contact Stéphane MICLO : stephane.miclo@orsayphysics.com

TABLE OF CONTENTS

1 INTRODUCTION 2

2 PRINCIPLE 2

2.1 Communication ports 2

2.2 XML messages 3

2.3 Other types of messages 3

3 HOW TO ESTABLISH A CONNECTION WITH THE SERVER 4

4 CONTENTS OF MESSAGES 5

4.1 Encoding 5

4.2 Notication from server 5
4.2.1 Description of the system 5
4.2.2 Parameter updates for every object 6
4.2.3 Parameter names and values for a specific object 6

4.3 Notifications from the client 7
4.3.1 Update a parameter value 7
4.3.2 Start an action on a device 8
4.3.3 Initialization/Description message 8
4.3.4 Initialization/Description message for one parameter 8
4.3.5 Conditions management (Backup) 8

5 SPECIAL MESSAGES 10

5.1 Exit Server from the client 10

5.2 Triggering a scenario before extinction 10

5.3 Error Messages 11
5.3.1 Communication error message 11

6 CONCLUSION 11

ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

1 Introduction

The purpose of this document is to present the server/client communication protocol. In this software, client and server are completely independent, they can be compiled separately and they do not need to share any library. To communicate with each other, they just exchange information on a TCP connection using the XML format (converted into a byte array encoded in UTF8). The main advantage of this format is that anybody can develop its own client for our server assuming that communication protocol is explained. This paper exists to explain how to develop a client able to interact with our server.

2 Principle

2.1 Communication ports

Every communication port used by the server is defined in the following file "Server/conf/connection/connection.xml" and can be modified there assuming that a complete restart of the server is necessary to make modifications effective. The configuration file shows three different communication ports with different purposes:

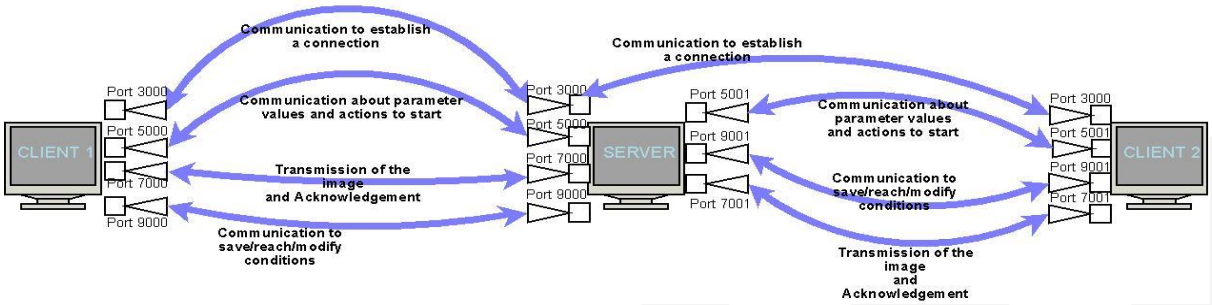
Establish a connection (ConnectionThread): The server is listening on this port to detect every client wanting to connect (default value : 3000)

Handle parameters and methods (MessageReaderThread): When the client is accepted and the communication is established, the client can communicate with the server. The server is listening on the defined port (default value: 5000). Each client connected on the server will not communicate on the same port. In the default configuration, the first client will discuss on the port 5000, the second one on the port 5001, etc…

Image: This is the specific communication canal for the transmission of the image delivered by the MISS. Considering that the MISS is sending a lot of information, it was decided to assign it a specific port of communication, so the communication will not be altered. The image will go through the port 7000 (default value) for the first client connected, then it will be 7001, 7002, etc…

Handle Conditions (ConditionThread): When a message reader thread is created, a condition thread is created as well. Its purpose is to deal with the backup of a state of the whole system. We can save the actual values of every parameters or reach a previously saved state of the system. The default communication port for this type of exchange is 9000 for the first client, and then the communication will increase its value (9001 for the second client connected, 9002 for the third one…)

Details of the communication ports used between clients and the server



ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

2.2 XML messages

Once the communication is established between client(s) and server, notifications are exchanged through XML Format (converted into byte array encoded in UTF8). The server automatically sends two types of message (the notifications of the updated parameters and the notifications of exceptions). The client must be ready to catch these notifications but can also send several kinds of messages (asking for the update of a parameter, starting an action on a device, asking for the list of devices/parameters, asking for the list of registered backups…). The syntax of all these messages will be explained later in this paper.
Every XML message sent by the server starts with the same entitled: "*<?xml version=\"1.0\"?>*". If two messages arrive in the reception buffer at the same time, you will be able to separate them from each other by splitting the received string with this value.

2.3 Other types of messages

Two types of message are not in the "xml" format:
The transmission of the image from the server to the client: A byte array is sent by the server and the client needs to rebuild the image from it
The establishment of the connection between client and server is made through a specific protocol
The syntax of these two specific message types will be detailed later.

ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

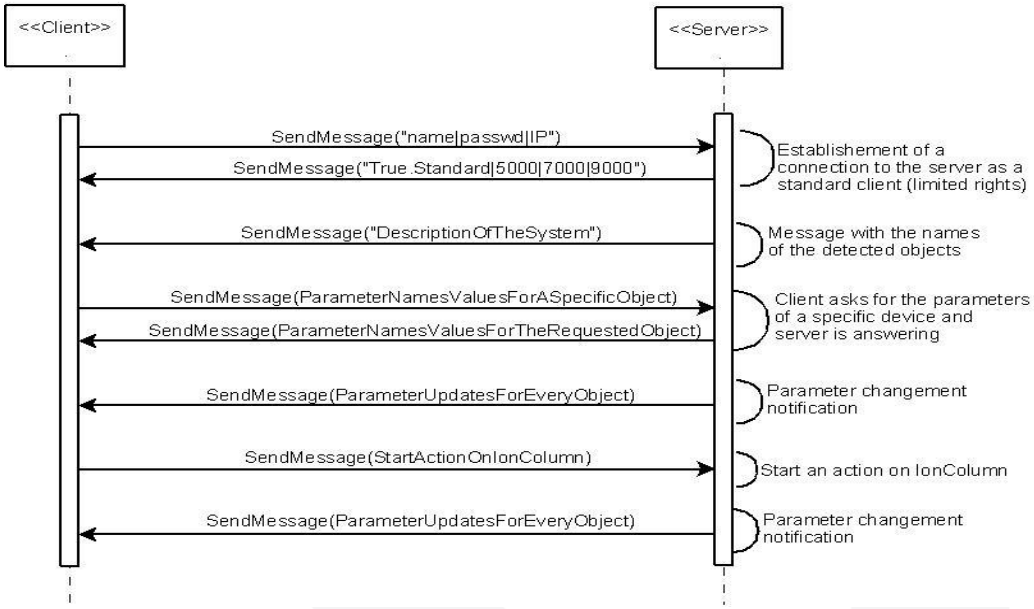
3 How to establish a connection with the server

After it has been started, the server is waiting for an entrance communication on the port 3000 (default value that can be changed on "Server/conf/connection/connection.xml"). To establish a connection, the client should contact the server through this port and send the following message "username|password|clientIpAddress" converted in byte[]. Obviously, you should replace username, password and clientIpAddress by their corresponding values. Receiving this message, the server checks that the user name and password are correct and sends back its answer:

- Connection accepted: The answer is formatted as follows: "True.userType|MessageReaderPort|MissPort|ConditionPort|Mode". The user type can be "Service", "Standard" or "Admin", "MessageReaderPort" is the communication port for the client sending information to the server and reciprocally, "MissPort" is the port the client must listen to receive the pictures as byte arrays, "ConditionPort" is the communication port used to save/reach a specific system state. The "Mode" can be "FIB" or "FEB" depending on the devices known by the Server.
- Connection refused : The answer is simply "False"

On the scheme displayed below, you will find a possible communication between a client and a server. As explained before, the client should first identify on the server and then some communication channels are open.

Possible communication between a client and a server



ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

4 Contents of messages

4.1 Encoding

Information sent through the communication ports is encoded as a byte array. There are two types of encoding:
The message can be an XML document converted in byte array encoded in UTF8.
The message is an image or a part of an image depending on the scan speed you chose: the first 13 items contains information about the (part) image and the rest of the data is the value of each pixel of the image. Here is the proper description:

- Convert byte in message[0] to "boolean" ; if its value is true with this message you received the last part of the image
- Convert bytes in message[1] and message[2] to "int" to get the width of the complete image
- Convert bytes in message[3] and message[4] to "int" to get the height of the complete image
- Convert bytes in message[5] and message[6] to "int" to get the x-coordinate of the first pixel
- Convert bytes in message[7] and message[8] to "int" to get the y-coordinate of the first pixel
- Convert bytes in message[9] and message[10] to "int" to get the width of the part image received
- Convert bytes in message[11] and message[12] to "int" to get the height of the part image received
- Convert bytes in message[13] and message[14] to "int" to get the value of the top left pixel
- Convert bytes in message[15] and message[16] to "int" to get the value of the second pixel of the first line
- …and then every pixel of the (part) image is detailed (line by line). The pixel value corresponds to an image encoding as a 16bpp grayscale image.

When the client is ready to deal with a new image, it must send an acknowledgment to the server. Sending a byte set to 1 would be perfectly fine.

4.2 Notication from server

4.2.1 Description of the system

Assuming that the configuration of the FIB is not fixed, the software can handle a not fully equipped FIB. When the user is identified on the server, the server notifies the client with the name of the devices it has detected. The message is formatted in this xml style:

```
<?xml version=\"1.0\"?>
<Description>
   <Object>Miss</Object>
   <Object>IonColumn(MVA)</Object>
   <Object>Scanner</Object>
</Description>
```


ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

4.2.2 Parameter updates for every object

Periodically, the server checks the values on every device and sends to the client all the parameter values that have changed since the last check. The message is formatted in this xml style:

```
<?xml version="1.0"?>
<Update>
  <Object>
    <Name>IonColumn(MVA)</Name>
    <Param name=\"MVAProbe_Y_Actual\" type=\"string\">-25346.283</Param>
  </Object>
  <Object>
  <Name>IonColumn(MVA)</Name>
  <Param name=\"CondensorVoltage_Actual\" type=\"string\">0</Param>
  </Object>
  <Object>
  <Name>IonColumn(MVA)</Name>
  <Param name=\"CondensorVoltage_Wobblerstate\" type=\"int32\">0</Param>
  </Object>
  <Object>
    <Name>Gis_0001</Name>
    <Param name=\"Line1Valve_Actual\" type=\"string\">True</Param>
  </Object>
  ...
</Update>
```

What is important about this message is that it is automatically generated by the server as soon as a client is connected and it concerns every device of the FIB system.

4.2.3 Parameter names and values for a specific object

This message can be sent by the server only if the client asks it to do so. As a response to the description message (ref III.3.c) of the client, the server is describing every parameter of a specific device.
The message is formatted this way:

```
<?xml version="1.0"?>
<Update>
  <Object>
    <Name>ObjectName</Name>
    <Param name=\"Parameter1Name\" type=\"string\">0</Param>
    <Param name=\" Parameter2Name \" type=\"int32\">0</Param>
    <Param name=\" Parameter3Name \" type=\"double\">0</Param>
    <Param name=\" Parameter4Name \" type=\"string\">0</Param>
    <Param name=\" Parameter5Name \" type=\"int32\">0</Param>
    <Param name=\" Parameter6Name \" type=\"double\">0 </Param>
    ...
  </Object>
</Update>
```


ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

4.3 Notifications from the client

4.3.1 Update a parameter value

Here is the explanation to follow when you want to change the value of a parameter. The update of a parameter should be done on two steps. First you access to the target value of the parameter you want to reach and then you can call the function "Update" on the parameter and it will try to reach its "Target" value.
In order to understand clearly, a parameter is an object programmatically speaking composed by a set of attributes and one function named "Update". The attributes are the following ones:

- *Actual* : The actual value of the parameter
- *Target* : The target value of the parameter
- *Current* : Usually the leak current of the power supply
- *Maxvalue* : Usually the maximum voltage of the power supply
- *Minvalue* : Usually the minimum voltage of the power supply
- *Enabled* : Boolean saying whether the parameter is enabled
- *Haswobbler* : Boolean saying whether the parameter can wobble
- *Wobblerstate* : Status of the wobbler. 1 if it is activated, 0 otherwise
- *Wobblerstrength* : Value of the wobbler
- *Status* : Integer representing the state of the parameter
- *Name* : Name of the parameter
- *Text* : Textual description of the parameter
- *Unit* : Unit of the parameter
- *StepFine* : The value of the step in fine mode
- *StepCoarse* : The value of the step in coarse mode

The definition of these attributes is perfectly right when it concerns high voltages parameter and has been adapted when it concerns other kind of parameters. You will found a complete list of parameters in the annex part of this document.

Messages sent to update a specific parameter are formatted in this xml style:

- Set the Target value of the parameter:

```
<?xml version="1.0"?>
<Setter>
  <ObjectConcerned>ObjectName</ObjectConcerned>
  <Parameters>
    <Parameter>
      <Name>ParameterName_Target</Name>
      <Value>2</Value>
    </Parameter>
  </Parameters>
</Setter>
```

- Try to reach the target value previously set

```
<?xml version="1.0"?>
<Command>
<ObjectConcerned>ObjectName</ObjectConcerned>
<Name>ParameterName_Update</Name>
</Command>
```


ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

4.3.2 Start an action on a device

To start an action on a specific device, you must give two kinds of information: the name of the device and the name of the command. In the annex part, you can find every name of commands and a brief description of what it is supposed to do. The message sent to start a command looks like this one:

```
<?xml version="1.0"?>
<Command>
   <ObjectConcerned>IonColumn(MVA)</ObjectConcerned>
   <Name>Initialization</Name>
</Command>
```

This command will start the process of initialization on the object named "IonColumn(MVA)"

4.3.3 Initialization/Description message

This type of message can be used when you want to get every parameter value of a specific device. It might be useful when you launch a graphical user interface for the first time so you can fill every field. The message is formatted in this xml style:

```
<?xml version="1.0"?>
<Init>
   <ObjectConcerned>ObjectName</ObjectConcerned>
</Init>
```

With this specific message, the server will give you back every parameter name and value of the device named "ObjectName" on the server. The returned message is the one called "Parameter names and values for a specific object" in this document.

4.3.4 Initialization/Description message for one parameter

This type of message can be used when you want to get one parameter value of a specific device. It might be useful when you launch a graphical user interface for the first time so you can fill one field. The message is formatted in this xml style:

```
<?xml version="1.0"?>
<InitOne>
   <ObjectConcerned>ObjectName</ObjectConcerned>
   <ParameterName>ParameterName</ParameterName>
</InitOne>
```

With this specific message, the server will give you back one parameter name and value of the device named "ObjectName" and the parameter named "ParameterName" on the server.

4.3.5 Conditions management (Backup)

On a specific port, defined in connections.xml file (starting at 9000 by default), you can communicate with the server about registering/reaching backups.

- To get the list of backups

```
<?xml version="1.0"?>
  <WorkingCondition>
  <ID>Mode (FIB or FEB</ID>
  <Name>GetListOfWC</Name>
</WorkingCondition>
```


ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

And the server will answer with the following message containing the name of every condition stored and their main parameter values:

```
<?xml version="1.0" ?>
  <WorkingConditions>
  <WorkingCondition>
    <Name>Condition1</Name>
    <Energy>17394</Energy>
    <ApertureSize>N/A</ApertureSize>
    <ApertureNumber>N/A</ApertureNumber>
    <Condensor>12857</Condensor>
    <BeamCurrent>N/A</BeamCurrent>
  </WorkingCondition>
  <WorkingCondition>
    <Name>Condition2</Name>
    <Energy>25006</Energy>
    <ApertureSize>251</ApertureSize>
    <ApertureNumber>1</ApertureNumber>
    <Condensor>13540</Condensor>
    <BeamCurrent>N/A</BeamCurrent>
  </WorkingCondition>
</WorkingConditions>
```

- Save the actual state of the system by sending

```
<?xml version="1.0" ?>
<WorkingCondition>
  <Name>StoreWC</Name>
  <ID>Mode (FIB or FEB)</ID>
  <Param>NameOfWorkingCondition</Param>
</WorkingCondition>
```

- Reach a previously saved state by sending

```
<?xml version="1.0" ?>
<WorkingCondition>
  <Name>ReachWC</Name>
  <ID>Mode (FIB or FEB)</ID>
  <Param>NameOfWorkingCondition</Param>
</WorkingCondition>
```

- Remove a saved condition

```
<?xml version="1.0" ?>
<WorkingCondition>
  <Name>DeleteWC</Name>
  <ID>Mode (FIB or FEB)</ID>
  <Param>NameOfWorkingCondition</Param>
</WorkingCondition>
```

- Remove all working condition

```
<?xml version="1.0" ?>
<WorkingCondition>
  <Name>  DeleteAllWC</Name>
  <ID>Mode (FIB or FEB)</ID>
</WorkingCondition>
```


ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

5 Special Messages

5.1 Exit Server from the client

In order to exit the server from the client you have to send this special message command

```
<?xml version="1.0" ?>
<Command>
<ObjectConcerned>Server</ObjectConcerned >
  <Name>Quit</Name>
  </Command>
```

In response the server will send this notification to all the clients connected:

```
<?xml version="1.0" ?>
<Error>
<Command>Disconnection</Command >
  </Error>
```

5.2 Triggering a scenario before extinction

Before sending the quit command to the server you have the possibility to trigger a scenario by sending this command

```
<?xml version="1.0" ?>
<Command>
<ObjectConcerned>Server</ObjectConcerned >
  <Name>Stop</Name>
  </Command>
```

The scenario is described in the node Stop in the file stopHTProcedure.xml located in the following path C:\OrsayPhysics\Configuration Files\conf\StopHTProcedure\

Ex :

```
<Stop>
  <Object name="MotherBoard">
   <Parameter name="Valve1_Target" type="string"
waitingFor="GunElectron:GunState:OFF">CLOSE</Parameter>
  </Object>
  <Object name="MotherBoard">
   <Parameter name="Valve2_Target" type="string"
waitingFor="GunIon:GunState:OFF">>CLOSE</Parameter>
  </Object>
        <Object name="GunElectron">
   <Parameter name="GunState_Target" type="string">OFF</Parameter>
  </Object>
      <Object name="GunIon">
   <Parameter name="GunState_Target" type="string">OFF</Parameter>
  </Object>
      <Object name ="Sed">
      <Parameter name="Collector_Target" type="string">0</Parameter>
      <Parameter name="Detector_Target" type="string">0</Parameter>
      <Parameter name="Level_Target" type="string">0</Parameter>
      <Parameter name="PMT_Target" type="string">0</Parameter>
      </Object>
 </Stop>
```

You can add a special attribute waitingFor in a parameter node, to notify that the action must wait for the completion of the parameter in the waitingFor attribute. The syntax is *waitingFor="ObjectName:Short Parameter Name:Value"*

ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033
www.orsayphysics.com

5.3 Error Messages

The Client can receive error messages sent by the server following this syntax :

```
<?xml version="1.0" ?>
<Error>
<ObjectName>[device name]</ ObjectName >
  <Message>[ message] </Message>
  </Error>
```

5.3.1 Communication error message

When the supervisor board does not respond well this message will be sent by the server

```
<?xml version="1.0" ?>
<Error>
<Command>Communication</Command >
  </Error>
```

The solution to solve this issue is to check if the usb plug is correctly plugged or if the power is on.

6 Conclusion

Following this guide, you should be able to develop your own client communicating with our server: authenticate yourself on the server, ask it to update parameters, ask it to launch actions, receive the notifications of the server (concerning parameters and image) and deal with backups of the system.

ZAC Saint-Charles - 95, Avenue des Monts Auréliens - 13710 Fuveau FRANCE
Tél. : +33 442 538 090 - Fax : +33 442 538 091
SA au capital de 846 000 euros - N°TVA intracommunautaire : FR 79 350 421 996 - Siret : 350 421 996 00033