Class mARC.Connector.Connector

Version 1.0

Date 24/10/2014

Table Of Content

[Connecting to a mARC server 1](#_Toc401915481)

[Executing a command step by step 2](#_Toc401915482)

[Executing an encapsulated command 3](#_Toc401915483)

[1 line command execution 3](#_Toc401915484)

[2 Script execution 4](#_Toc401915485)

# 0 OverView

It includes

* a client socket
* an associated object mARCResult to handle results sent from the server

# Connecting to a mARC server

The function connect() allows to connect via TCP to a mARC server identified with a port and a IP address:

import mARC.Connector.\*;

Connector connector = new Connector();

connector.**Lock**();// low level locking for multithreading

connector.setIp("127.0.0.1");

connector.setPort("1254");

connector.connect() ;

connector.UnLock(); //unlocking

One can also use the other constructor

Connector connector = new Connector("MyName", "127.0.0.1","1254");

connector.Lock();

connector.setIp("127.0.0.1");

connector.setPort("1254");

connector.UnLock();

If connection was successful, the method getIsConnected returns true, false otherwise

# Executing a command step by step

provided a connector has been instantiated from one of the above constructor and connected as above, a command may be computed from the general scheme described below:

connector. directExecute = true; // the connector is in ‘command-line’ mode

connector.**OpenScript** (); // clear internal buffers and allows new script

connector.**push** ("Session.Connect");

connector.**AddFunction** (); // builds the command to send to the server

marc->ExecuteScript (); // send the command to the server and get results

the command ‘**Session.Connect() ;** ‘ is computed and sent to the server via TCP.

The method **getToSend()** returns the current command/script. The Method **getReceived()** contains the raw string returned by the server.

If a syntax error or any command error has been detected server-side the method getError() return true, on successful execution, it returns true.

The method getExecutionErrorMsg() returns the error message sent by the server after execution of the command/script sent.

The associated object mARC.Connector.mARCResult contains the encapsulated data returned by the server. At low-level, the server returns a (nxm) matrix of data. This matrix columns may be accessed through the following method:

**getDataByName(String name, int idx)**

The first argument refers to the name of the variable returned by a given command from the server, the second argument refers to the line number (base 0) of the current script. The default value is -1 indicating we need the result of last line of the script.

You may also retrieve the data from a matrix-line point of view with:

**getDataByLine(int row, int idx)**

idx refers to the line number (zero base) of the script line

row refers to the line index (zero base) inside the line referenced by idx.

**Consider this example:**

connector.**OpenScript** ();

connector.**push** ("Table.Get");

connector.**push** ("Instances");

connector.**addFunction** ();

connector.ExecuteScript ();

String[] tbls = connector.getDataByName("tables", -1);

the method GetReceived() returns :

**3 1 2 1 8 0 tables <6 Eudata/> <8 Eudetail/>;**

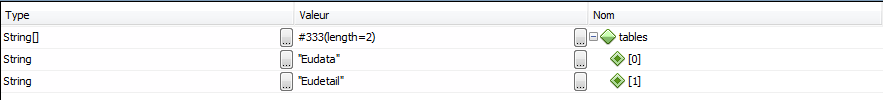
The API reference tells us that the matrix is 1x2 i.e. one line and two columns. The variable **tables** contains “Eudata” and “Eudetail”.

The line

String[] tbls = connector.getDataByName("tables", -1);

allows to retrieve the data contained in the server result.

This line returns an arrays of strings in tbls which contains:



# Executing an encapsulated command

All API functions are encapsulated in the connector class itself.

## 1 line command execution

You can execute a script command by command by using

connector. directExecute = true;

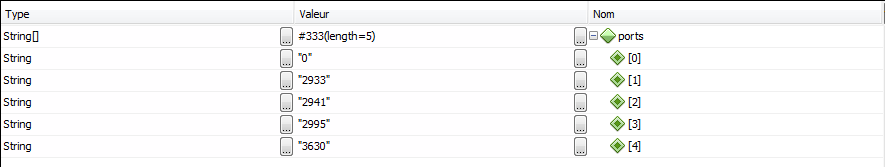
Consider the script, it is executed line by line from the server-side:

connector.directExecute = true;

connector.SERVER\_GetConnected("1", "-1");

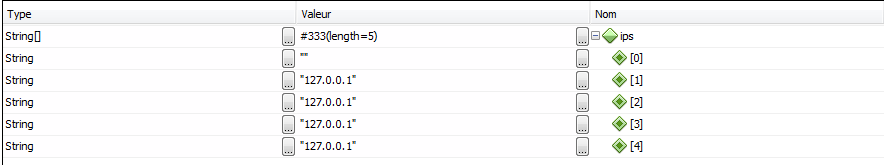
String[] ports = connector.getDataByName("Port", -1);

Ports is string array:



String[] ips = connector.getDataByName("IP", -1);

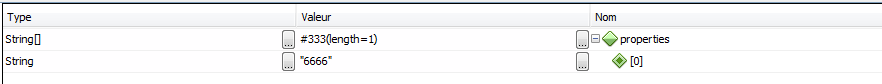
ips also :



connector.SERVER\_GetProperties("port");

String[] properties = connector.getDataByName("prop\_value", -1);

Properties contains:

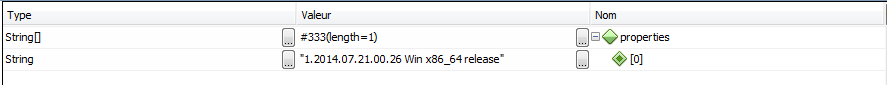


Which is the listening port of the server.

connector.SERVER\_GetProperties("build");

properties = connector.getDataByName("prop\_value", -1);

properties now contains:



Which is the server IP address.

## 2 Script execution

The above script may also be executed at once on the server side:

connector.directExecute = false;

connector.openScript(null);

connector.SERVER\_GetConnected("1", "-1");

connector.SERVER\_GetProperties("port");

connector.SERVER\_GetProperties("build");

String toSend = connector.getToSend();

connector.executeScript(); // execute script server-side at once !!!

String Received = connector. getReceived ();

// retrieve results at once !!!

String[] ports = connector.getDataByName("Port", 0); // retrieve data from line #0

String[] ips = connector.getDataByName("IP", 0); // retrieve data from line #0

String[] propertiesPort = connector.getDataByName("prop\_value", 1); // retrieve data from line #1

String[] propertiesBuild = connector.getDataByName("prop\_value", 2); // retrieve data from line #2

String[] s = connector.getDataByLine(5, 0); // retrieve line 5 in line#0

the string toSend contains:



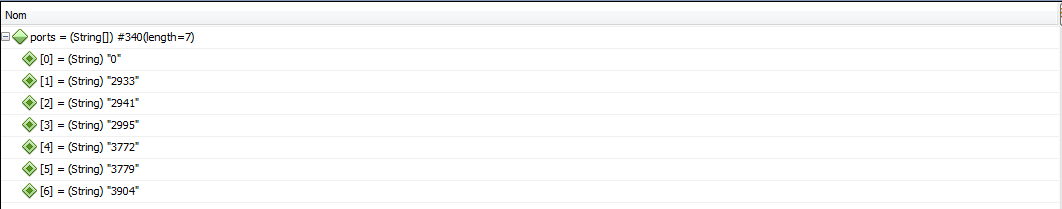
The string Received contains after execution:

Received = (String) "5 1 7 2 8 0 IP 8 0 Port <0 /> <1 0/> <9 127.0.0.1/> <4 2933/> <9 127.0.0.1/> <4 2941/> <9 127.0.0.1/> <4 2995/> <9 127.0.0.1/> <4 3772/> <9 127.0.0.1/> <4 3779/> <9 127.0.0.1/> <4 3904/> ; 1 4 8 0 prop\_name 8 0 prop\_value 8 0 prop\_type 8 0 prop\_access <4 port/> <4 6666/> <6 string/> <1 r/> ; 1 4 8 0 prop\_name 8 0 prop\_value 8 0 prop\_type 8 0 prop\_access <5 build/> <37 1.2014.07.21.00.26 Win x86\_64 release/> <6 string/> <1 r/> ; "

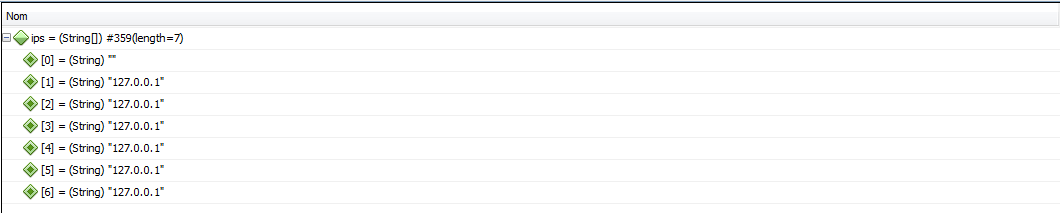
As you can see we received three lines corresponding to the three commands to be executed:

|  |  |
| --- | --- |
| Line #0 | 7 2 8 0 IP 8 0 Port <0 /> <1 0/> <9 127.0.0.1/> <4 2933/> <9 127.0.0.1/> <4 2941/> <9 127.0.0.1/> <4 2995/> <9 127.0.0.1/> <4 3772/> <9 127.0.0.1/> <4 3779/> <9 127.0.0.1/> <4 3904/> |
| Line #1 | 1 4 8 0 prop\_name 8 0 prop\_value 8 0 prop\_type 8 0 prop\_access <4 port/> <4 6666/> <6 string/> <1 r/> |
| Line #2 | 1 4 8 0 prop\_name 8 0 prop\_value 8 0 prop\_type 8 0 prop\_access <5 build/> <37 1.2014.07.21.00.26 Win x86\_64 release/> <6 string/> <1 r/> |

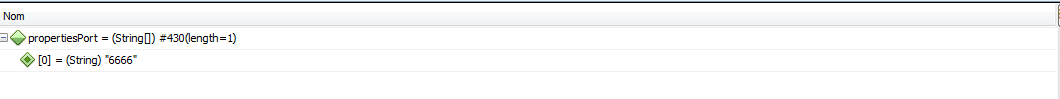
The array string ports contains:



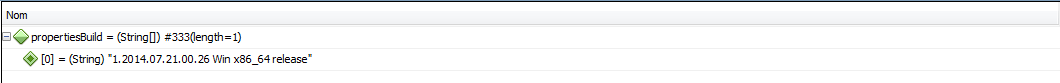
The string array ips contains:



The string array propertiesPort contains:



And the string array propertiesBuild contains:



The line #5 of line script #0 is retrieved by string array s:



All the data are the same as in the preceding section but they were retrieved at once.

# Methods references :

bool Lock (void);

bool UnLock (void);

void Connect (void); //connecting

bool isConnected (void);

bool isValid (void); // socket is valid

void OpenScript (void); // clears command buffers : ready for an new command

void Push (GPString s); //push a string on the command buffer

void AddFunction (void);

bool ExecuteScript (void);

# members references :

String ip; //IP address: defaut 127.0.0.1

String port; // Port default: 1254

String SessionId; // id of session