PERSISTENT

CMS Wrapper Communication

Version: 1.2 [Draft]

Date: 06-Jan-2011

INDEX

Revision History

Version	Summary of Change	Written By	Approver	Date
1.0	Released HLD	PSL Team		09-Sept-2010
1.1	CMS-Wrapper Communication	PSL Team		06-Jan-2011
1.2	Draft after review comments incorporation	PSL Team		11-Jan-2011

1 Introduction

1.1 Purpose

This document specifies the CMS wrapper communication.

1.2 Scope

The CMS and wrapper communication purpose is as mentioned below:

1. Sending Command from CMS to Wrapper
2. Receiving acknowledgement from Wrapper
3. Receiving response from Wrapper
4. Receiving asynchronous event from wrapper
5. Sending periodic keep alive/monitor command from CMS to Wrapper

Server socket requirement would be as below:

1. Command Server Socket:
   Server Socket for accepting commands (residing at wrapper)
   The number of Command Server socket would depend on the number of wrappers i.e. **each wrapper would run a Server Socket to accept commands**.
2. Event Server Socket:
   Server Socket listening for incoming asynchronous events and responses
   The CMS will have only one Event Server Socket.

1.3 Glossary & Acronyms

IUCAA – Inter-University Centre for Astronomy and Astrophysics
NCRA –National Centre for Radio Astrophysics
IGO – IUCAA Girawali Observatory
GMRT - Giant Meter Wavelength Radio Telescope
URL – Uniform Resource Locator
I/O – Input/output
IDE – Integrated Development Environment
JDE – Java Development Environment
HTTP – Hypertext Transfer Protocol
XML- Extensible Markup Language
GUI - Graphical User Interface
CMS - Control and Monitoring System
RDBMS – Relational Database Management System
Java EE – Java Enterprise Edition

1.4 References

Please refer the following documents available:

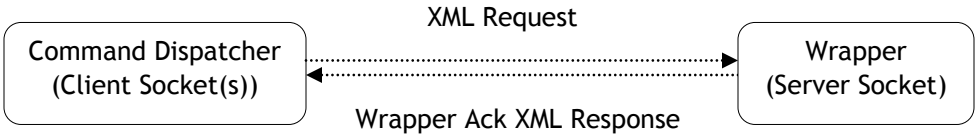
- Telescope_CMS_Requirement_Specifications_Document_V2[1].0.pdf
- Command_structure_ver1.3.pdf
- Servo_Telemetry_Commn_Protocol.pdf
- Technical Discussion meeting with IUCAA team
- Technical Discussion meeting with NCRA team

2 Overview

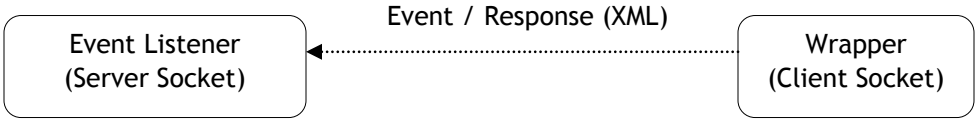
2.1.1 Communication Layer

This layer actually transports the commands to the wrapper as well as receives the response/event from the wrapper. All the communication between CMS and wrapper is in terms of structured well defined xml documents. The format of this xml document is being worked out and would be shared once it is finalized. This layer is made up of following components –

- **Command Dispatcher** – This component is responsible for delivering commands to appropriate wrapper for subsystem. The Command Dispatcher communicates with the wrapper layer using socket connection.



- **Event Listener** – This component receives various acknowledgements, responses and events from wrapper and hand it over to messaging layer so that business layer can process the response.



All wrappers will send the Events/Response to the same Server Socket.

2.1.2 Socket Communication

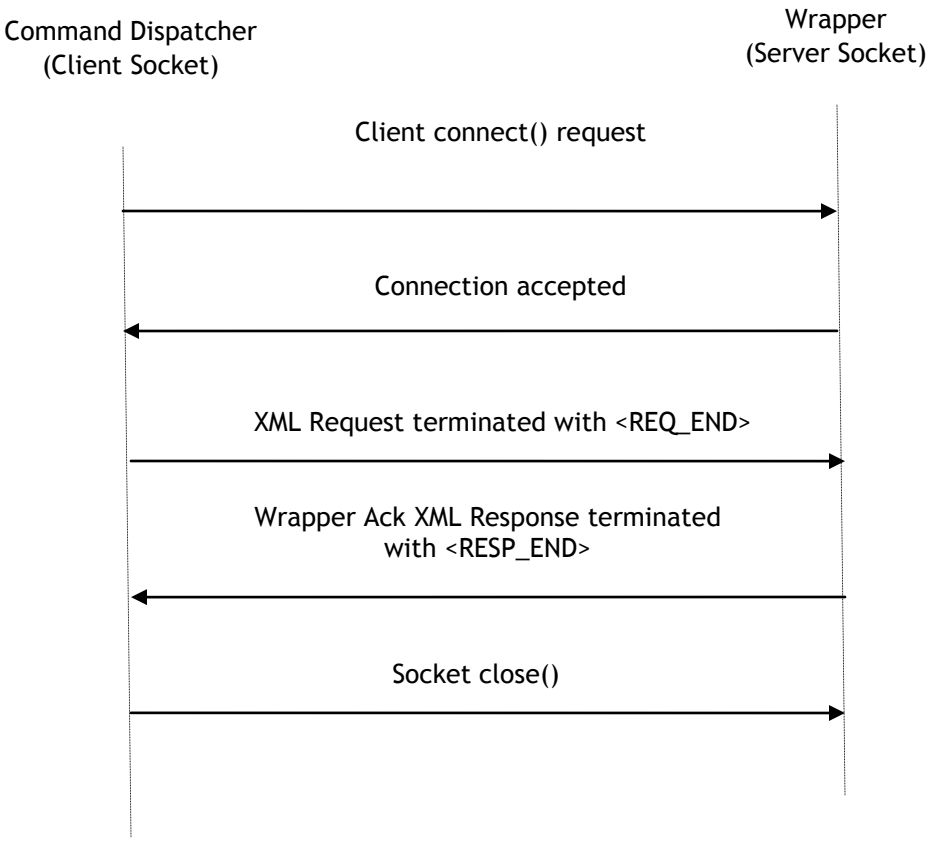
The Socket server needs to support multiple clients so needs to follow the Multithreading approach i.e. separate thread for each client connection.

Pros of Multithreaded Socket Server:

1. Multiple clients can get connected to server.
2. If one of the clients goes in IO Blocking mode, still the server is reachable through other clients.
3. This approach can support CMS scalability.

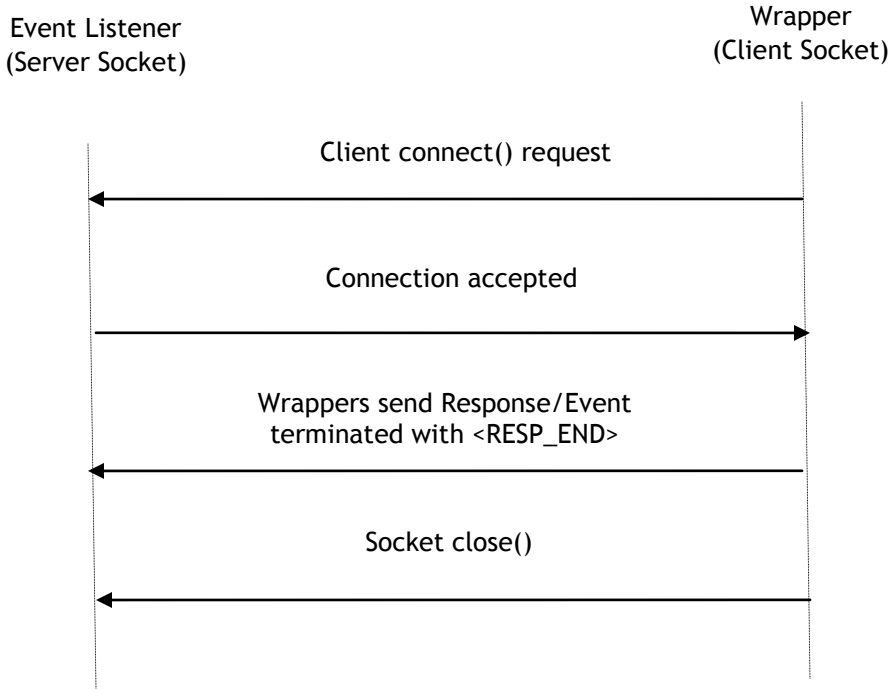
2.1.3 CMS Command Dispatcher

The command Dispatcher is responsible for sending the commands over the socket. For this it will implement a client socket connection on CMS end and a Server Socket at Wrapper end. The handshaking and data flow would be as specified in below figure:



2.1.4 CMS Event Listener

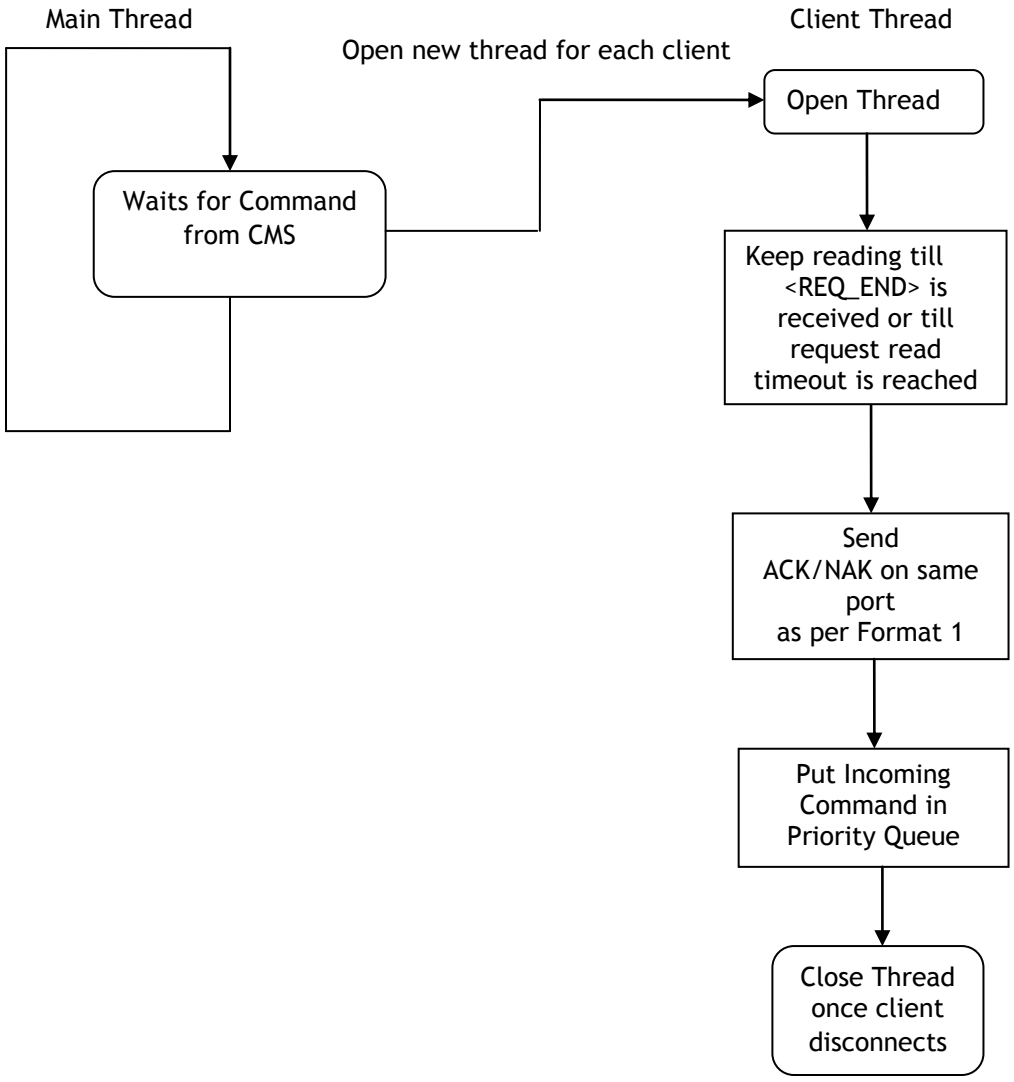
The EventListner is responsible for various acknowledgements, responses and events from wrapper. For this the CMS will implement a Server Socket and the wrapper will open a client socket. The handshaking and data flow would be as specified in below figure:



2.1.5 Wrapper Command Acceptor

The wrapper Command Acceptor accepts the command from the CMS and sends wrapper acknowledgement on the same connection.
Each Wrapper would have to implement a Server socket to accepts commands

Command Acceptor Module (Server socket for accepting commands)



Format 1:

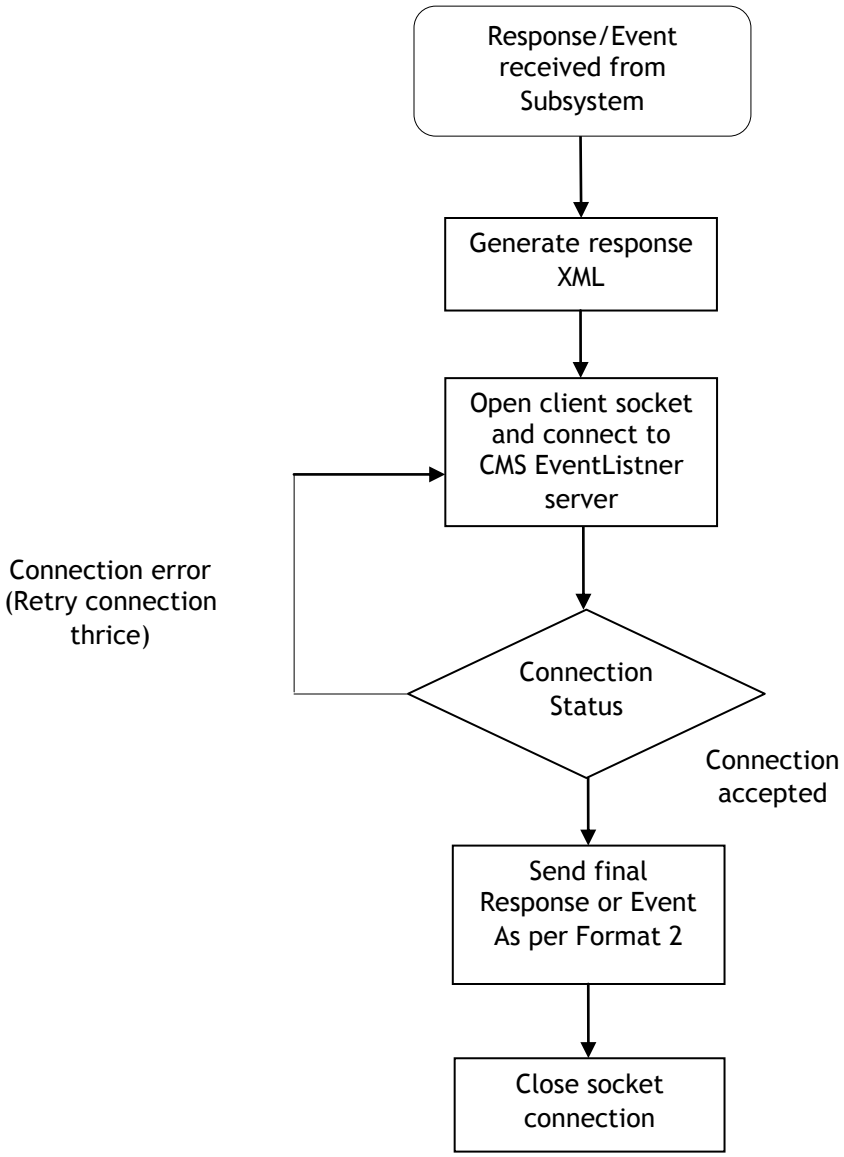
Wrapper-ACK – to the same port on which request arrived

```
<responses>
        <response>
                <seq>1</seq>
                <id>44</id>
                <name> MOVE</name>
                <systemid>servo</systemid>
                <version>1.0</version>
                <timestamp></timestamp>
                <code>10</code>
                <event>10</event>
                <message>wrapper ack</message>
                <data></data>
        </response>
</responses>
```

*note: Data indicated in the **purple color** is command dependent and will be available from the request sent by CMS. For more details on the response format refer to Request and Response format section of this document.

2.1.6 Wrapper Event/Response sender

This component sends the event or asynchronous response (intermediate and final) to CMS. For this the wrapper needs to open a client socket connection and connects to the CMS EventListner server and writes the response. Once response is written the client can disconnect from the server.



Format 2:

Final Response-

```
<responses>
        <response>
                <seq>1</seq>
                <id>44</id>
                <name>MOVE</name>
                <systemid>servo</systemid>
                <version>1.0</version>
                <timestamp></timestamp>
                <code>10</code>
                <event>12</event>
                <message>move successful</message>
                <data></data>
        </response>
</responses>
```

*note: Data indicated in the **purple color** is command dependent and will be available from the request sent by CMS. For more details on the response format refer to Request and Response format section of this document.

3 Request and Response Format

Following are request and response format for communication between CMS and Wrapper. This is structured is still being fine tuned so there may be few changes in final version.

3.1.1 Request Format

Following is the request format of the xml that would be sent to a wrapper by CMS.

```
<commands>
    <command>
            <seq>1</seq>
            <id>41</id>
            <name>position</name>
            <systemid>subsystem1</systemid>
            <version>1.0</version>
            <timestamp>2011-01-11T19:27:23.110+05:30</timestamp>
            <priority>1</priority>
            <data>
                    <param>
                            <name>subsystemid</name>
                            <value>1</value>
                    </param>
                    <param>
                            <name>ax</name>
                            <value>A</value>
                    </param>
                    <param>
                            <name>ang1</name>
                            <value>10</value>
                    </param>
            </data>
    </command>
</commands >
```

Following is brief description about the various fields in this format –

id – This represents the command id for CMS system, for various subsystems it would be actual opcode corresponding to the command fired. CMS will send this as part of Command request xml.

name - This is display name of the actual command fired by the user. CMS will send this as part of Command request xml.

seq - This will contain a unique number indicating the count of commands fired. CMS will populate this value and send it as part of Command request.

systemid - This is used to distinguish between various subsystems attached to CMS via wrapper. CMS will populate this as part of Command request.

Version – subsystem version

timestamp – The timestamp at which the request was created.

priority - This is used to set the priority. 0 - being low priority and 1 being high priority.
Note: Since priority is added as a part of request, a separate Server Socket would not be required on wrapper end to cater to High Priority commands.

data - Some of the commands may need additional input apart from opcode and systemid, all such additional information will be sent as name-value pair under DATA field. Wrapper can parse DATA fields to retrieve corresponding information.

3.1.2 Response Format

Following is the sample response format of the xml that would be sent by a wrapper to CMS.

```
<responses>
        <response>
                <seq>1</seq>
                <id>41</id>
                <name>position</name>
                <systemid>subsystem1</systemid>
                <version>1.0</version>
                <timestamp>2011-01-11T19:27:23.110+05:30</timestamp>
                <priority>1</priority>
                <code>10</code>
                <event>12</event>
                <message>Positioned successfully</message>
                <data>
                        <param>
                                <name>subsystemid</name>
                                <value>1</value>
                        </param>
                        <param>
                                <name>ax</name>
                                <value>A</value>
                        </param>
                        <param>
                                <name>ang1</name>
                                <value>10</value>
                        </param>
                </data>
```

```
                <alarm>
                        <alarmid></alarmid>
                        <description></description>
                        <level></level>
                </alarm>
        </response>
</responses>
```

Following is the descriptive information about response format that CMS expects from wrapper.

seq – This will contain a unique number indicating the count of commands fired. CMS will populate this value and send it as part of Command request, CMS expects wrapper to send back same value as part of each and every ack, intermediate or final response for corresponding command. This field is optional for events as they are generated asynchronously.

id – This represents the command id for CMS system, for various subsystems it would be actual opcode corresponding to the command fired. CMS will send this as part of Command request xml and expects wrapper to send it back for each and every ack, intermediate or final response for corresponding command.

name – This is display name of the actual command fired by the user. CMS will send this as part of Command request xml and expects wrapper to send it back for each and every ack, intermediate or final response for corresponding command.

systemid – This is used to distinguish between various subsystems attached to CMS via wrapper. CMS will populate this as part of Command request, CMS expects wrapper to send same value back as part of each and every ack, intermediate or final response for corresponding command.

version – subsystem version

timestamp – The timestamp at which the request was created. Optional for events

priority – This provides the priority of response. High priority responses will be consumed first.

code – This field is populated by wrapper while sending response back to CMS for a given command. Following are the various values and their interpretation by CMS. CMS expects wrapper to adhere to this convention while sending back ack, intermediate, final response.

VALUE	Meaning
10	SUCCESS
>10	FAILURE

event – This field is populated by wrapper while sending response back to CMS for a given command or any asynchronous event. Following are the various values and their interpretation by CMS. CMS expects wrapper to adhere to this convention while sending back ack, intermediate, final response.

VALUE	Meaning
10	Wrapper Ack
11	Subsystem Ack

12	Final response
>12	Any asynchronous event or intermediate response

message – This field can be used to communication information such as command success, failure or any other message to the end user. In case of general failure like invalid command syntax wrapper will need to indicate the error message. This field can be used for the same.

data – This field will be populated by wrapper while sending intermediate, final response or asynchronous response to CMS. This field can contain any number of name-value pairs. Optional if no data is to be sent.

alarm – In some cases like hardware failure, subsystem will generate alarm and send it to CMS via wrapper. In such cases wrapper can make use of ALARM tag, which contains following fields-

alarmid – alarm identifier
description – Contains the brief description about the ALARM, which will be shown to the user in ALARM window.
level – Wrapper can assign the priority for such ALARM and share the information with CMS using this field.
Level can be 1 - Information, 2 - Warning, 3 - Critical

Note that the response structure is fairly generic to accommodate various types of responses from subsystem. It may be noted that in some cases some of the fields cannot be populated by wrapper for e.g.

a) Fields like id, name, seq are not relevant when wrapper sends hardware failure alarms
b) If a command was executed successfully it cannot have ALARM information on it
c) Response for some of commands may not have any data associated with it so <data> field cannot be populated

Only in such cases wrapper can exclude these fields while sending information back to CMS.