NCRA 15m Radio telescop
Wrapper Design Document for the C & M System

Version 0.99 Jul 12, 2011

INDEX

1. CMS APPLICATION

1.1 CMS Application behavioural Points – Scope :

(i) CMS

1. Either from CMS-UI, BATCH or response to the STATE Send XML Commands to the WRAPPER.

2. Read CMD- Acknowledgement from Wrapper.

3. Read Intermediate and Final Response from the wrapper for the send Command.

4. Read Asynchronous Events from Sub-system (via wrapper).

5. Read Monitoring and System-State Parameters so that inputs can be given to Alarm, State-machine etc.

6. Enabling Periodic command From wrapper to Device (like DOMON <INTERVAL>, TRACK <INPUTPARA>, DOPLOT). Note : These commands will be like any other command except There Ids will have number greater than > for e.g. 200. But Ids and Name in SubSystem Other than servo have same name. How to handle this situation?).

7. All communication between CMS & Wrapper will be in predefined XML Format.

8. Log all activities in Database.

(ii) WRAPPER :

1. Receive XML Command from the Wrapper. Parse the XML command and send acknowledgement or no-acknowledgement for the command to CMS.

2. Once acknowledgement send to the CMS for the received command, store the command in a queue.

3. Read command from the queue and send it to device.

4. Receive the response from the device for the send command and receive the asynchronous event from the device (if any).

5. Send periodic commands (if enabled by CMS) to the device and receive the responses. e.g. DOMON, TRACK etc.

6. Convert response or asynchronous event from devices to XML format

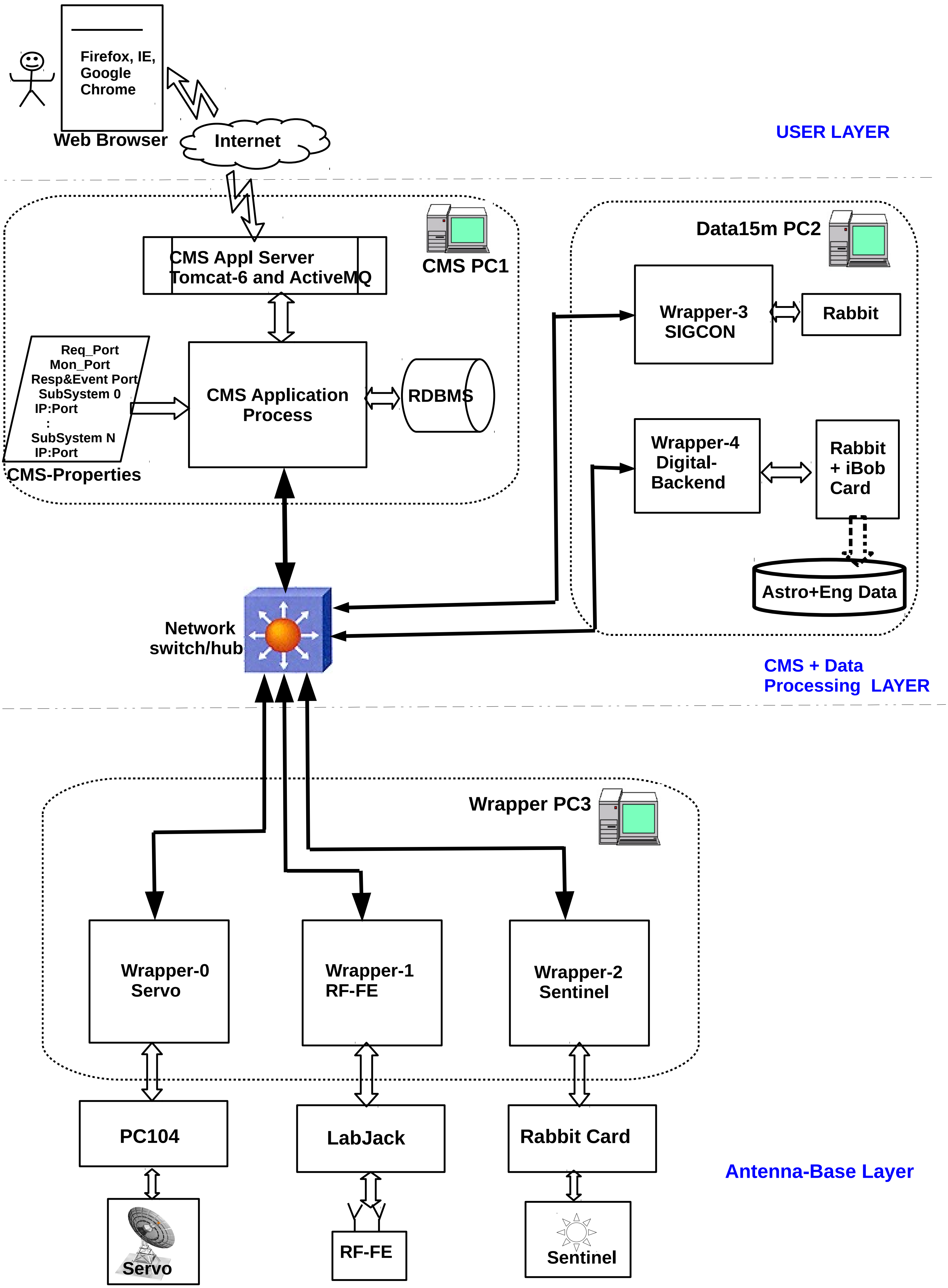
7. Send XML Responses to the CMS (monitoring information on monitoring-server and command,asynchronous responses to response-server).

8. Log XML command, it's response, monitoring information and asynchronous events from device/subsystem.

(iii) DEVICE :

1. Receive command from the wrapper.
2. Execute the command.
3. Send response for the received command to the wrapper
4. Send Asynchronouse events to the wrapper.
5. In case of critical/imergency situation in safety point of view, device process takes the safety major actions and send the alarm information to the wrapper.

1.3 CMS-Wrappers-Devices Functional Block-diagram



1.4 Predefined XML CMS-Properties and Assumptions :

1.4.1 `*cms.properties'* file – CMS application and Wrapper variables

```
# Response and Event Port will be the Same – CMS Server_1
  eventPort=5000
  responsePort=5000

# Monitoring Parameter Port - CMS_Server_2
  MonitoringParamPort=19999

# Default Command timeout in millisec
  defaultCommandTimeout=60000

# Default Response timeout in millisec
  responseTimeout=60000
```

(??) 1. Does it mean that total time since command-dispatcher issues the command till getting response Is total command plus response timeout i.e. 60000 + 60000 millisec?

2. Where to mention the Monitoring/State parameter reading time which is unique for the STATE-MACHINE? (domon <interval> is there, but can be define in a common variable).

```
# Sequence file – need to store seq_id of the command? // purpose need to understand.
   seq.file=/usr/ncra/seq.dat

# Wrapper information to configure sub-systems
subSystemConfig=ncra-subsystemconfig.xml
```

ncra-subsystemconfig.xml : There will be a one wrapper-server running per sub-system. Subsystems', wrapper-server IP & Port, along with file to be read to know valid commands for given the sub-system is mentioned as below. ***CMS Client*** can connect to these wrappers depends on for which sub-system command is issued (From user).

```
<?xml version="1.0" encoding="UTF-8"?>
<subsystems>
  <subsystem>
        <name>servo</name>
        <connectionurl>127.0.0.1:9000</connectionurl>
        <commandfile>servo_commands.xml</commandfile>
        <version>1</version>
  </subsystem>
  <subsystem>
        <name>sentinal</name>
        <connectionurl>127.0.0.1:8000</connectionurl>
        <commandfile>sentinal_commands.xml</commandfile>
        <version>1</version>
  </subsystem>
  <subsystem>
        <name>frontend</name>
        <connectionurl>127.0.0.1:8571</connectionurl>
        <commandfile>frontend_commands.xml</commandfile>
        <version>1</version>
  </subsystem>
  <subsystem>
        <name>backend</name>
        <connectionurl>192.168.3.6:9571</connectionurl>
        <commandfile>backend_commands.xml</commandfile>
        <version>1</version>
  </subsystem>
  <subsystem>
        <name>sigcon</name>
        <connectionurl>192.168.4.45:7570</connectionurl>
        <commandfile>sigcon_commands.xml</commandfile>
        <version>1</version>
  </subsystem>
</subsystems>
```

1.4.2 <SubSystem>_commands.xml :

The pre-defined XML Configuration file per sub-system per wrapper contains :

(a) supportedcommands : All supported commands to the <subsystem>. Command name at user level, translation to subsystem commands, validation for the command arguments with data types can be defined for the command.
(b) supportedresponses : Response to the commands with response validation parameters.
(c) supportedalarms : List of possible alarms that can be generated by sub-system.
(d) monitoringparams : sub-system monitoring and status parameters can be defined with tag-name & value.
(e) supportedbatches : batch-command to execute specific batch-file (containing list of valid sub-system Commands) can be defined here. (Note – batch-file can be interpreted up-to two level down).

Note : wrapper must send response, monitoring, and alarm to the CMS in XML format which is predfined In <subsyste>_command.xml file per sub-system.

(??. This contain not only command but response,monitoring, alarm etc. File name can be change to <subsystem>.xml only. -> it is possible in ncra-sub-system.xml properties but whether CMS will affect?)

e.g. servo_commands.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<supportedconfig>
        <supportedcommands>
                <command> // simple command
                        <name>coldstart</name>
                        <id>40</id>
                        <syntax></syntax>
                        <sample></sample>
                </command>

                <command>   // Position command
                        <name>position</name>      // user level command name
                        <id>42</id>                // device specific command, can it have string?
                        <syntax>ax,ang1,ang2</syntax>    // syntax
                        <sample>B,123:30:20,123:50:10</sample>   // sample-which appears in UI after selecting command
                        <params>                   // required parameters – i.e. command validation
                                <param required="true">
                                        <paramname>ax</paramname>
                                        <type>string</type>
                                        <validation>
                                                <values>
                                                        <value>A</value>
                                                        <value>E</value>
                                                        <value>B</value>
                                                </values>
                                        </validation>
                                </param>
                                <param>
                                        <paramname>ang1</paramname>
                                        <type>angle</type>
                                        <validation>
                                                <angle>
                                                        <degree>
                                                                <min>0</min>
                                                                <max>360</max>
                                                        </degree>
                                                </angle>
                                        </validation>
                                </param>
                                <param>
                                        <paramname>ang2</paramname>
                                        <type>angle</type>
                                        <validation>
                                                <angle>
                                                        <degree>
                                                                <min>30</min>
                                                                <max>300</max>
                                                        </degree>
                                                </angle>
                                        </validation>
                                </param>
                        </params>
```

```
<dependency_validations> // argument
dependency validation
            <params>
            <param required="true">
                    <paramname>ax</paramname>
                        <type>string</type>
                          <validation>
                             <values>
                                 <value>A</value>
                            </values>
                          </validation>
            </param>
            <param required="true">
                    <paramname>ang1</paramname>
                       <type>angle</type>
                         <validation>
                            <angle>
                            <degree>
                            <min>0</min>
                            <max>360</max>
                            </degree>
                           </angle>
                        </validation>
            </param>
            </params>
            <params>
            <param required="true">
                    <paramname>ax</paramname>
                        <type>string</type>
                          <validation>
                            <values>
                                <value>E</value>
                            </values>
                          </validation>
            </param>
                    <param required="true">
                    <paramname>ang2</paramname>
                       <type>angle</type>
                         <validation>
                            <angle>
                            <degree>
                            <min>17</min>
                            <max>90</max>
                            </degree>
                           </angle>
                        </validation>
            </param>
            </params>
            <params>
                   <param required="true">
                   <paramname>ax</paramname>
                        <type>string</type>
                            <validation>
                                <values>
                                    <value>B</value>
                                </values>
                           </validation>
                    </param>
                    <param required="true">
                    <paramname>ang1</paramname>
                       <type>angle</type>
                         <validation>
                            <angle>
                            <degree>
                            <min>0</min>
                            <max>360</max>
                            </degree>
                           </angle>
                        </validation>
                   </param>
                    <param required="true">
                    <paramname>ang2</paramname>
                       <type>angle</type>
                          <validation>
                                <angle>
                                <degree>
                                <min>17</min>
                                <max>90</max>
                                </degree>
                               </angle>
                          </validation>
       </dependency_validations>
</command>   // position command format end here
<command> ..</command>  // further commands.
</supportedcommands>

<supportedresponses>
  <response>
     <name>readangles</name>
     <id>30</id>
     <params>
        <param required="true">
        <paramname>timeofday</paramname>
         <type>datetime</type>
             <validation>
               <datetime>
                 <format>hh:mm:ss</format>
                </datetime>
              </validation>
          </param>
          <param required="true">
          <paramname>az_cp</paramname>
           <type>angle</type>
                <validation>
                   <angle>
                   <degree>
                       <min>0</min>
                       <max>360</max>
                    </degree>
                    </angle>
                </validation>
            </param>
            <param required="true">
             <paramname>az_tp</paramname>
               <type>angle</type>
                   <validation>
                       <angle>
                       <degree>
                           <min>30</min>
                           <max>300</max>
                       </degree>
                       </angle>
                   </validation>
             </param>
             <param required="true">
              <paramname>az_pp</paramname>
                 <type>angle</type>
                    <validation>
                        <angle>
                        <degree>
                             <min>30</min>
                             <max>300</max>
                        </degree>
                         </angle>
                     </validation>
               </param>
               <param required="true">  // further parameters can
be defined as mentioned above.

  </response>
</supportedresponses>
```

Note – In response parameters, alarm and event not mentioned within a supportedresponses block but seprately defined. However in format of request-response as a response from Wrapper, alarm and event can be part of response format. This need to check/confirm.

```
<supportedalarms>
        <alarm>
              <name>motor_current_high</name>
              <id>1</id>
              <level>1</level>
              <message>motor current high</message>
        </alarm>
        <alarm>
             <name>temperature_high</name>
             <id>2</id>
             <level>1</level>
             <message>temperature is high</message>
        </alarm>
        <alarm>
             <name>unknown</name>
             <id>0</id>
             <level>1</level>
             <message>unknown error</message>
        </alarm>
</supportedalarms>

 <monitoringparams>
            <param>
                         <name>timeofday</name>
                         <value>10:12:20</value>
            </param>
            <param>
                         <name>az_cp</name>
                         <value>123:50:10</value>
            </param>
            <param>
                         <name>az_tp</name>
                         <value>10:50:10</value>
            </param>
            <param>
                         <name>az_pp</name>
                         <value>13:50:10</value>
            </param>
            <param>
                         <name>el_cp</name>
                         <value>123:50:10</value>
            </param>
            <param>
                         <name>el_tp</name>
                         <value>123:50:10</value>
            </param>
            <param>
                         <name>el_pp</name>
                         <value>123:50:10</value>
            </param>
            <param>
                         <name>az_motor1_current</name>
                         <value>35.12</value>
            </param>
            <param>
                         <name>az_motor2_current</name>
                         <value>34.56</value>
            </param>
            <param>
                         <name>az_tacho1</name>
                         <value>434</value>
            </param>
   // Further Monitoring parameters can be defined
  // as mentioned above.
            <param>...</param>
            <param>
                        <name>CWFL</name>
                         <value>0</value>
            </param>
</monitoringparams>
```

```
<supportedbatches>
       <batchcommand>
             <name>batch1(init-position)</name>
             <id>412</id>
                          <syntax>z</syntax>
                          <sample>2</sample>
                          <params>
                                  <param required="true">
                                  <paramname>z</paramname>
                                       <type>integer</type>
                                  </param>
                          </params>
                          <filepath>/usr/ncra/lib/batch1.txt</filepath>
        </batchcommand>
         <batchcommand>
                <name>batch2(init-track)</name>
                <id>1233</id>
                          <syntax></syntax>
                          <sample></sample>
                          <filepath>/usr/ncra/lib/batch2.txt</filepath>
          </batchcommand>
          <batchcommand>
               <name>batch3</name>
               <id>1233</id>
                          <syntax></syntax>
                          <sample></sample>
                          <filepath>/usr/ncra/lib/batch3.txt</filepath>
           </batchcommand>
             <batchcommand>
                   <name>batch4</name>
                   <id>1233</id>
                          <syntax></syntax>
                          <sample></sample>
                          <filepath>/usr/ncra/lib/batch4.txt</filepath>
             </batchcommand>
   </supportedbatches>

</supportedconfig>   // servo_commands.xml End here.
```

Note : batch_n.txt can contain any valid predefined command which is mentioned in <subsystem>_commands.xml.

(??)

(i) How to send send raw-monitoring data (for e.g. MCM channel data) and display or archive it?

(ii) How to define STATUS of sub-system in <monitoringparams>

(iii) What is significance of <id> in a batch mode?

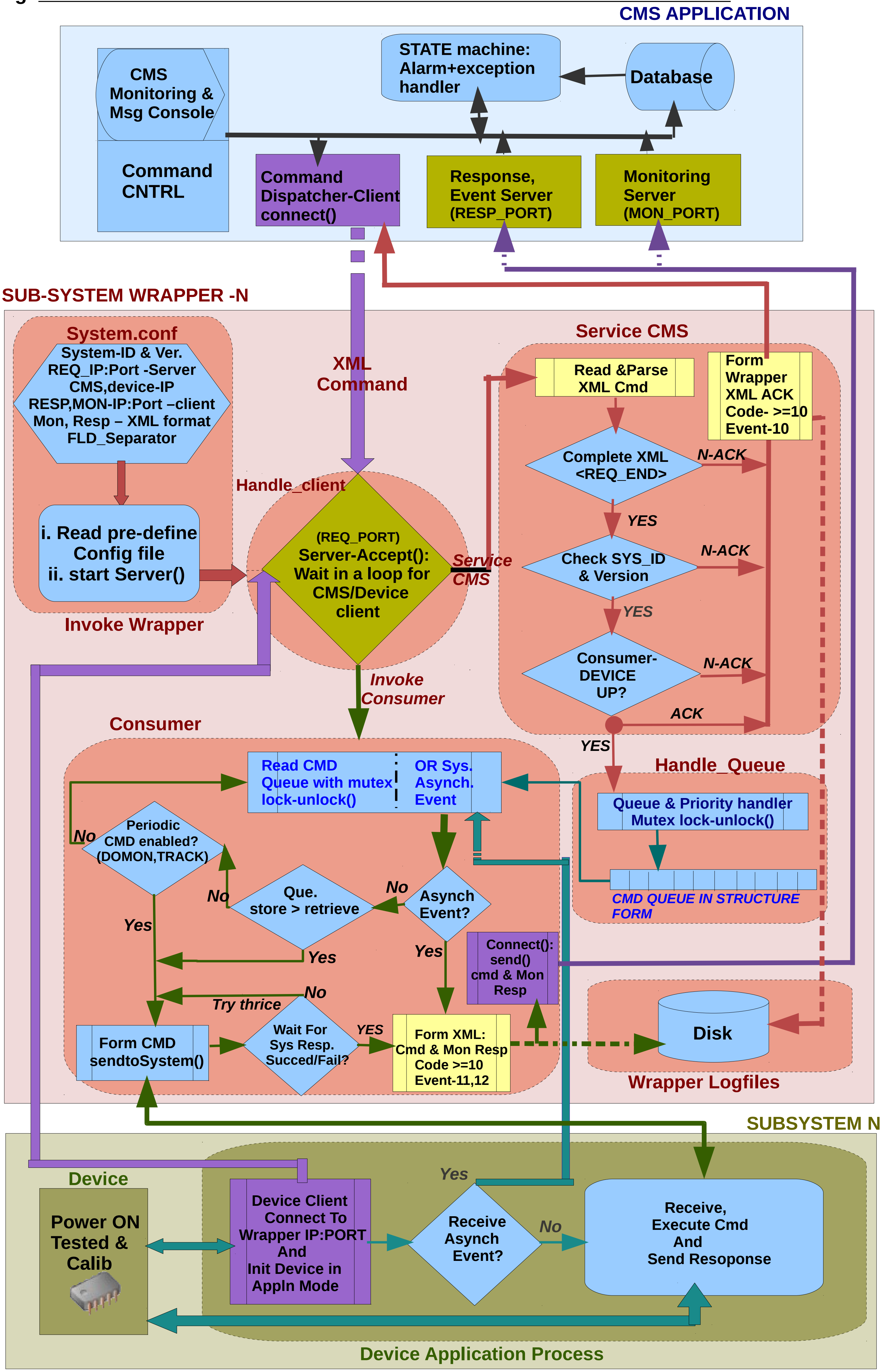
1.4.3 <subsystem>_engineering.xml

Using *<subsystem>_engineering.xml* file CMS creates web-based Engineering GUI for particular <subsystem>. There are main four section **Status param, Monitoring param, Basic commands and Detailed Commands** for particular <subsystem>. This file is loaded during the CMS initialization. Hence any changes needed in UI require to modify the <subsystem>_engineering.xml and re-initialize the CMS.

Note : All the UI parameters for Status, Monitoring and Commands should be predefined first in the ***<subsystem>_commands.xml*** (see section 1.4.2). If the particular status/monitoring parameter or command not defined in ***<subsystem>_commands.xml*** and available in the ***<subsystem_engineering>.xml*** it will have no effect and cause errors like monitoring/status parameter will not update or command not found message.

(see the Engineering UI Configuration document by the PSL in the archive of CMS release of Phase2.2. Further detail structure of <subsystem>_engineering.xml is avoided here as it is subset of <subsystem>_commands.xml).

2.1 Fig. CMS SUB-SYSTEM-DEVICE DETAILS : FUNCTIONAL BLOCK DIAGRAM


CMS APPLICATION
CMS Monitoring & Msg Console
Command CNTRL
STATE machine: Alarm+exception handler
Database
Command Dispatcher-Client connect()
Response, Event Server (RESP_PORT)
Monitoring Server (MON_PORT)
SUB-SYSTEM WRAPPER -N
System.conf
System-ID & Ver.
REQ_IP:Port -Server
CMS,device-IP
RESP,MON-IP:Port –client
Mon, Resp – XML format
FLD_Separator
i. Read pre-define Config file
ii. start Server()
Invoke Wrapper
XML Command
Handle_client
(REQ_PORT)
Server-Accept(): Wait in a loop for CMS/Device client
Service CMS
Service CMS
Read &Parse XML Cmd
Form Wrapper XML ACK Code- >=10 Event-10
Complete XML <REQ_END>
N-ACK
YES
Check SYS_ID & Version
N-ACK
YES
Consumer-DEVICE UP?
N-ACK
ACK
YES
Invoke Consumer
Consumer
Read CMD Queue with mutex lock-unlock()
OR Sys. Asynch. Event
Handle_Queue
Queue & Priority handler Mutex lock-unlock()
CMD QUEUE IN STRUCTURE FORM
No
Periodic CMD enabled? (DOMON,TRACK)
No
Que. store > retrieve
No
Asynch Event?
Yes
Yes
Yes
Connect(): send() cmd & Mon Resp
Try thrice
No
Form CMD sendtoSystem()
Wait For Sys Resp. Succed/Fail?
YES
Form XML: Cmd & Mon Resp Code >=10 Event-11,12
Disk
Wrapper Logfiles
SUBSYSTEM N
Device
Power ON Tested & Calib
Device Client Connect To Wrapper IP:PORT And Init Device in Appln Mode
Yes
Receive Asynch Event?
No
Receive, Execute Cmd And Send Resoponse
Device Application Process

2.2 Process Components overview :

2.2.1 CMS Application process

1. **Client_1** (Command dispatcher): Depending on SUBSYSTEM IP:PORT -> Connect to Subsystem wrapper. Send XML command and Receive ACK from the Wrapper for the send XML Command.

2. **Server_1** – To read Intermediate and Final Response for the XML command send by Client_1 or receive Asynchronous events from sub-system (via wrapper).

3. **Server_2** – To read Monitoring parameters of the system (via wrapper).

4. Web-based UI or batch-script interface to issue commands in XML format.

5. State-Machine – commands to be issued on *state-changes* or on *critical alarm* conditions.

6. Database For retreiving CMS Status and logging CMS activities.

2.2.2 Wrapper Process :

1. **Invoke Server :** This component reads the <subsystem>_server.conf (latter file name need to change wrapper_<subsystem>.conf) file. e.g. ***servo_server.conf*** file is given as below:

```
   # SubSystem ID
   export SUBSYSTEM=0

  # On this port, CMS will connect to SubSystem's wrapper.
    export REQ_PORT=9000

  # IP of CMS Machine
    export MON_IP=192.168.3.203
    export RESP_IP=192.168.3.203
  # Wrapper send Monitoring, event and command response  information on these ports
    export MON_PORT=9898
    export RESP_PORT=5000          //Event will also be send on RESP_PORT

 # IP of SubSystem (Servo) Machine
    export DEVICE_IP=192.168.3.6

 # Monitoring Interval between Wrapper and Device
    export MON_INT=3

 # Monintoring XML parameter file per Subsystem
    export MON_XML=ServoMon.xml

 # Format for Command Response XML
    export RESP_XML=response.xml

 # Log Path to save Command, Response and Monitoring XML log
    export LOGPATH=/home/pslncra/log

  # CMD Field Separator for the device Commands.
    export CMD_FLD=","

  # Mon or Response Seprator from the device response
     export RESP_FLD=","
```

Invoke Wrapper invoke following processes/functionalities (*invokeWrapper.c, wrapperCom.c*):

- Start server() socket call on **REQ_PORT.**
- Using ***handle_clients()*** : Wait in a continuous loop for the **client() connection** either from the Device or from the CMS.
- If connection received from the CMS (not a Device_IP), then sequentially execute methods defined in **service CMS block**
- If connection received from the device, then sequentially execute methods as mentioned in **Consumer block.**

2. service CMS (*servCMS.c*) : Receive XML command from the CMS-client over request-port (REQ_PORT) and send Wrapper acknowledgement/No-acknowledgement for the command to CMS-client over the same connection (REQ_PORT).

- parse XML command using XMLIB routines.
- Confirm whether complete command received or not by checking <REQ_END> tag in the XML command.
- Check whether command is for the <Sub-System> and <version>
- Check whether Wrapper to device communication is established.
- If the above three checks are OK then save Parsed XML structure in a queue and send Wrapper acknowledgement about Command receiving successfully, otherwise send wrapper No-acknowledgment for the CMS command.

The acknowledgement to the CMS command by wrapper is in XML format, the XML tags' values are :
EVENT=10 // Wrapper Acknowledgement.
CODE=10 // if sending ACK successful.
OR >10 // NACK - command not received successfully.

3. Handle Queue *(handle_Que.c)* : Once XML command decoded successfully by XMLIB parser and ACK send to the CMS over request-port. handle_queue component store the command in global structure format as mention Below.

- Queue is either circular or sliding
- writting the queue-element increment **queue.store** variable**,** and reading the queue-element increment **queue.retrieve** vaiable
- priority command stored at the head of the queue.

Note – handle queue algorithm will be reviesed later.

- **Global Queue structure :**

```
Struct {
   Long seq // Sequence Number of Command

   Int SubsystemId
   Float version

   char ID;  // Command ID – Device specific.
   char CommandName[512]; // command name at user level

   char TimeStamp[256];
   int timeout ;                // (??) CMS to wrapper or wrapper to device timeout period?
   // --------- Above information re-used by XMLIB to form XML ACK/NACK/RESPONSE ---//

   int NoofArg;
   Char CommandArg[256][512];
   Char CMD_FLD // Field separater for device-specific command
   Char RESP_FLD // Field separator for resp-specific command
 } SUBSYSTEM_CMD
```

4. Consumer *(ConsumeCMS.c)* : This component in a continous loop, either read asynchronous event From device or read command from the queue (queue.retrieve < queue.store). The asynchronous event and command-response is send to the CMS in XML format.

- Read asynchronous event or Alarm; form XML packet (using XMLIB) and send to the CMS on response port (i.e. RESP_PORT).
- Read command from a queue - if queue.retrieve counter is less than the queue.store counter. Form Device specific command and send it to device
- If periodic command enabled (like DOMON, TRACK) form the DEVICE specific Command and send to device.
- Read the response from the device for the send command (if note succeed try CMD execution 3 times).
- Form XML packet (using retrieved queue-command information) for Command-response.
- send the XML packet to MONITORING (MON_PORT) or RESPONSE/EVENT Port (RESP_PORT).
- Consumer uses non-persistent connection (connect() call) to MON_PORT or RESP_PORT to send the information.

Note – (i) Routine/Block to handle STATE_CHANGE depending upon either receiving Monitoring information (e.g. For Servo system 'Target – actual encoder position' error) or any asynchronous event information not mentioned here. However this need to incorporate at later stage.
(ii) However, actions/commands need to execute as a response to STATE_CHANGE (in safety point of view) is handled by DEVICE & CMS processes which is not considered/expected to be done by the WRAPPER.

5. Wrapper Logfile : This component is responsible to store the wrapper transaction in a routine logfiles.

- XML format acknowled/Not-acknowldged commands.
- XML format - Asynchronous event or alarm
- XML format - Monitoring inromation

This module need to develop and currently not available, Monitoring/Asynchronous/Alarm responses can be stored in a separate Logfile as Device monitoring information. Command and it's response can be stored in a separate logfile using SEQ number.

6. **XML libxml based component *(xmlparse.c)***:

- This component is mainly parse the XML format based command and store information in a global Command structure format so that it can be store as element for queue.
- Wrapper acknowledgement, command responses, events, alarm and monitoring information received from the devices converted into XML format so that it can send to the CMS.

7. Wrapper Communication to Device and CMS (*wrapperCom.c*) :

- Routines to init server socket, accepts and process communications from the clients i.e. CMS and Device.
- Routines to connect the CMS server so that response from the devices can be send.

2.2.3 Device Application Process :

- Device specific application process can either run on PC104,Rabbit or Labjack kind of Micro-controller or on a separate PC to interface the Device/hardware i.e. Subsystem

- Once DEVICE+ Interface h/w (mirco-controller or PC) Powered ON, it will do establish basic Initialization chain (i.e. establish connection with the h/w) and does the basic calibration.

- Upon successfull connection establishment and calibration of the Device h/w, it will connect to the WRAPPER process through socket communication calls using the predefined WRAPPER- IP and PORT. i.e. **Device application process will be a CLIENT and it will have persistent connection to the Wrapper server.**

- As a part of the CMS initialization process it will receive CMS commands mentioned in init-batch script. For e.g. Set-time, set-default settings, DOMON (read monitoring parameters), etc.

- Device application process is responsible for
  i. Receive asynchronous events or alarm – take action in safety or perfomance point of view.
  ii. Keep monitoring data ready so that it can send immediately upon receiving periodic monitoring command.
  Iii. Receive commands from the wrapper, execute it and send responses to the received commands.

- After power failure it will repeat the above actions.

3. Global Request- Response, ALARM, and Asynchronous EVENT : XML Formats and Assumptions

As a title suggests, this section discuss about the generic XML formats and assumptions in a command, responses, Monitoring and asynchrnous events transaction between CMS and the wrappers.

3.1 Generic Request Structure :

Assumptions :

- Wrapper will run the one server per <SUBSYSTEM> (or device) using REQ_PORT.
- The CMS command-dispatcher client() will connect to the Wrapper-server to issue XML-Request.
- The CMS command-dispatcher client() and Wrapper-Server communication is non-persistent.
- One Request (command) XML packet may contain multiple requests/commands.
- XML Command send by the CMS is completed by the tage <REQ_END>.
- The Wrapper send acknolwedgement (ACK or NACK) to the CMS on same REQ_PORT.
- This structured is still being fine tuned so there may be "Few changes" a in Final version.

Generic Request Structure

```
    <commands>   // Multiple command can be in a one packet.

      <command>
              <seq>1</seq>
               <id>41</id>                   // Applicable to the Servo while formating command to the device
              <name>position</name> // Applicable to the Sub-system while formating cmd to dev.
              <systemid>subsystem-name</systemid>  // Subsystem ID.
              <verison>1.0</version>                          // Version of device and wrapper server.
              <timestamp>2011-01-11T19:27:23.110+5:30</timestamp>
              <priority> 1 </priority>   // (??)  Is CMS checking Priority in ACTIVEMQ How?
                                        //  Wrapper will handle the priority by putting Command
                                        //  in Circular or shifting Queue.
               <timeout> </timeout>  // (??) Variable need to define for the CMS-Wrapper; how about
                                       // Wrapper-Device. Actually CMS-Wrapper one can have fixed time? But
                                       // suppose command is in queue then?
//--------------------- Above information can be reused while sending the response to the commands ------//
               <data>
                  <param>
                     <name>subsystemid</name>  // This subsystemid where single wrapper address
                     <value>1</value>                // multiple sub-system if any at device level.
                  </param>
                  <param>
                     <name>ax</name>
                     <value>A</value>
                  <param>
                  <param>
                     <name>ang1</name>
                     <value>10</value>
                  <param>

               </data>
          <command>

        </commands><REQ_END>
```

id , name : id and name can be same for the sub-systems other than Servo system.
Seq : is Unique for each command.
systemid : This is used to distinguish between various sub-systems attached to CMS via wrapper. CMS will populate this as a part of Command Request.
timestamp : Timestamp at which the request was created.
priority : 0 Low priority; 1- 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 commands may need additional input apart from opcode (i.e. ID) 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 to form the command.

3.2 Generic Response Structure:

Assumptions :

- Generic response structure used to send the wrapper acknowledgement, response, asynchrnous events, alarms, and monitoring information in XML format to the CMS.
- Wrapper Acknowledgement as a immediate response send to the CMS Client by Wrapper Server (using same connection of REQ_PORT).
- Wrapper client send Intermediate and Final response (which include alarm, asynchronous events etc.) to the RESP_PORT of the CMS server.
- Monitoring information about device parameters (i.e. Live status) and STATE_CHANGES variables is send as a part of monitoring information in XML format on the MON_PORT.
- All the responses from the wrapper is end by the tag <RESP_END>
- The tage value for <***event***> **</event>** is **10 - wrapper acknowledgement**
  11 - Intermediate device from the device.
  12 - Final response from the device.
  >12 – Asynchronous Event handling.
- The tage value for **<code></code>**> is **10 – Successful**
  >10 - Failed.
- Multiple responses can be combined in the one packet of XML response.

Sample Response format :

```
  <responses>  // multiple responses can be combined in one XML-response packet.
          <response>
            <seq>1</seq>
            <id>41</id>
            <name>position</name>
            <systemid>System-name</systemid>
            <version>1</version>
            <timestamp>2011-01-11T19:27:23.110+5:30</timestamp>  // time-stamp when XML Req. issued.
            <priority>1<priority>
           // --------- Already available XML parsed data from the received CMD by the wrapper server  --//
              <code>10</code>
              <event>12</event>
              <message> position Successfully<message> // (??) This message need to be dynamically
                                                      //          formed depending upon the code, event and
                                                     //          Command-name values.
              // Data as a part of the response to the command, this information is optional if there is no data
              // requirement for the command.
              <data>
                   <param>
                      <name>sub-sub-system-id</name> // (??) If the device is having sub systems.
                      <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>0 </level>   // Level - 1 information, 2- Warning, 3- Critical.
               </alarm>
        <response>

 </responses>
```

NOTE -

(1) seq, id, name, systemid, version, timestamp, priority - Allocated by the CMS; Wrapper should use this information while forming the Response Packet. CMS Expects Wrapper to send back "SAME Value" for :
i. As part of each Command and every ACK
ii. Intermediate or FINAL Response for corresponding Command.

(2) The response structure is fairly generic to accommodate various types of responses from sub-system. It may be noted that "IN SOME CASES SOME OF THE FIELDS CAN NOT BE POPULATED BY THEWRAPPER" . For e.g.

(a) Fields like id,name,seq are not relevant when wrapper hardware send failure alarm.
(b) (??) Asynchronous events send by the Devices i.e. <event> " value greater than 12" </event>
(c) (??) If a command was executed successfully it cannot have ALARM information on it.!!!
(d) Responses for some data may not have data associated with it.

ONLY IN SUCH CASES Wrapper can exclude these fields while sending information back to CMS.

(3) (??) Since the response structure is generic – can the Alarm or asynchronous event can be formed as part of The Monitoring data to MON_PORT. This will help while debugging the MON Data?

(4) While sending responses, device response time also need to incorporate in the response structure so that User can know at what time the command execution completed at the device level.

3.2.2 Code and Event Assumptions for the Response :

- Code and Event fields are populated by the WRAPPER while sending either acknowledgement or responses the received command by wrapper (which includes alarm, asynchronous events and monitoring information as well).
- **<message> ..</message> field can be formed dynamically using the "seq, id, name, subsystem-name, code and event values".**
- <data>..</data> field for the command is optional, if required data field can be populated by the wrapper.

#	id or name	CODE	Event	Message
1	Cmd-Name	10	10	<Cmd-Name>: Wrapper <subsystem> ACK successful
2		11	10	<CMD-Name>: Wrapper <subsystem> Time Out.
3		12	10	<CMD-Name> : Wrapper XML Command incomplete.
4		13	10	<CMD-Name> : Wrapper invalid subsystem or version.
5		<x>	10	<CMD-Name> : Wrapper XML Any other unknown error.
6		10	11	<CMD-Name> : <subsystem> ACK the command
7		>11	11	<CMD-NAME>: <subsystem> give the error <CODE>
8		10	12	<CMD-NAME> : <subsystem> Final Response succeed
9		>11	12	<CMD-NAME>: <subsystem Final Response Fail <code>

Note (??):
The format of communication between wrapper and device need to be decide about how to differentiate Between the ALARM, Asynchronous events and command response.
- Command response can be differentiated as it is the part of transaction i.e. Command send and awaiting For the response.
- on the other hand if sub-system or device need to send some information about <alarm></alarm> or asynchronous event, wrapper must have some flag protocol to differentiate between alarm and Asynchronous events.

3.2.3 Alarm – XML format :

In some cases like hardware failure or critical safety situation occures, device (subsystem) can generate alarming condition. The alarm from devices can send by the WRAPPER to CMS using the **<alarm>...</alarm>** tag.

- alarms are part of the response structure as mentioned in the previous section
- alarms can be raised by the CMS when monitoring parameter goes out of range or certain flags are up in the monitoring information (e.g. STATUS variable, or digital variable in the SERVO subsystem).

The Alarm associated with a monitoring param is specified in the <supportedalarms> section of the <subsystem>_ commands xml. The syntax to be followed for alarm name is alarm_<monitoring param name>. For e.g. the name for wind_vel1 parameter the alarm name would be alarm_wind_vel1

(??) Checked servo_commands.xml "

```
 <supportedalarms>
        <alarm>
                <name>motor_current_high</name>
                <id>1</id>
                <level>1</level>
                <message>motor current is high</message>
        </alarm>
        <alarm>
                <name>temperature_high</name>
                <id>2</id>
                <level>1</level>
                <message>temperature is high</message>
        </alarm>
        <alarm>
                <name>unknown</name>
                <id>0</id>
                <level>1</level>
                <message>unknown error</message>
        </alarm>
       </supportedalarms>
```

" : There is no sharing monitoring-parameter name; also in AlarmRules.drl = there is no variable like "alarm_wind_vel1?".

3.3 Monitoring response structure :

- Once monitoring is enabled by the CMS “DOMON <INTERVAL>” command, wrapper will send the periodic command for getting the information from the wrapper.

- The device is expected to send monitoring information values :
  (i) In a sequence as defined in the **MON_XML=**<subsystem>_mon.xml OR (??)
  (ii) Tag_name and Tag_value separated by the response field (**RESP_FLD**). *// see section 2.2.2*
  // wrapper process

- Wrapper will encapsulate the Device Monitoring parameter values in the XML format and send to the CMS server on the separate port (MON_PORT).

- In case of the wrapper-Device connection disabled/not connected/broken- Device Monitoring data have zero values (reset) and the same will be communicated using <CODE>11</CODE> <EVENT>10</EVENT> (code=11- Device time out) as a wrapper Acknowledgement. (This is the case where CMS and Wrapper is UP but device is down).

- If monitoring parameters at specific interval not received by the CMS for particular system for which DOMON enabled, CMS will put that <sub-system> in the suspend state

.

- The monitoring parameter will be validated against the validation specified in the <supportedresponse> XML section of the <subsystem>_commands.xml file at the CMS.

- If DOMON <-1 or 0 > command send, wrapper will disable the periodic DOMON command to the Device.

Note (??) :

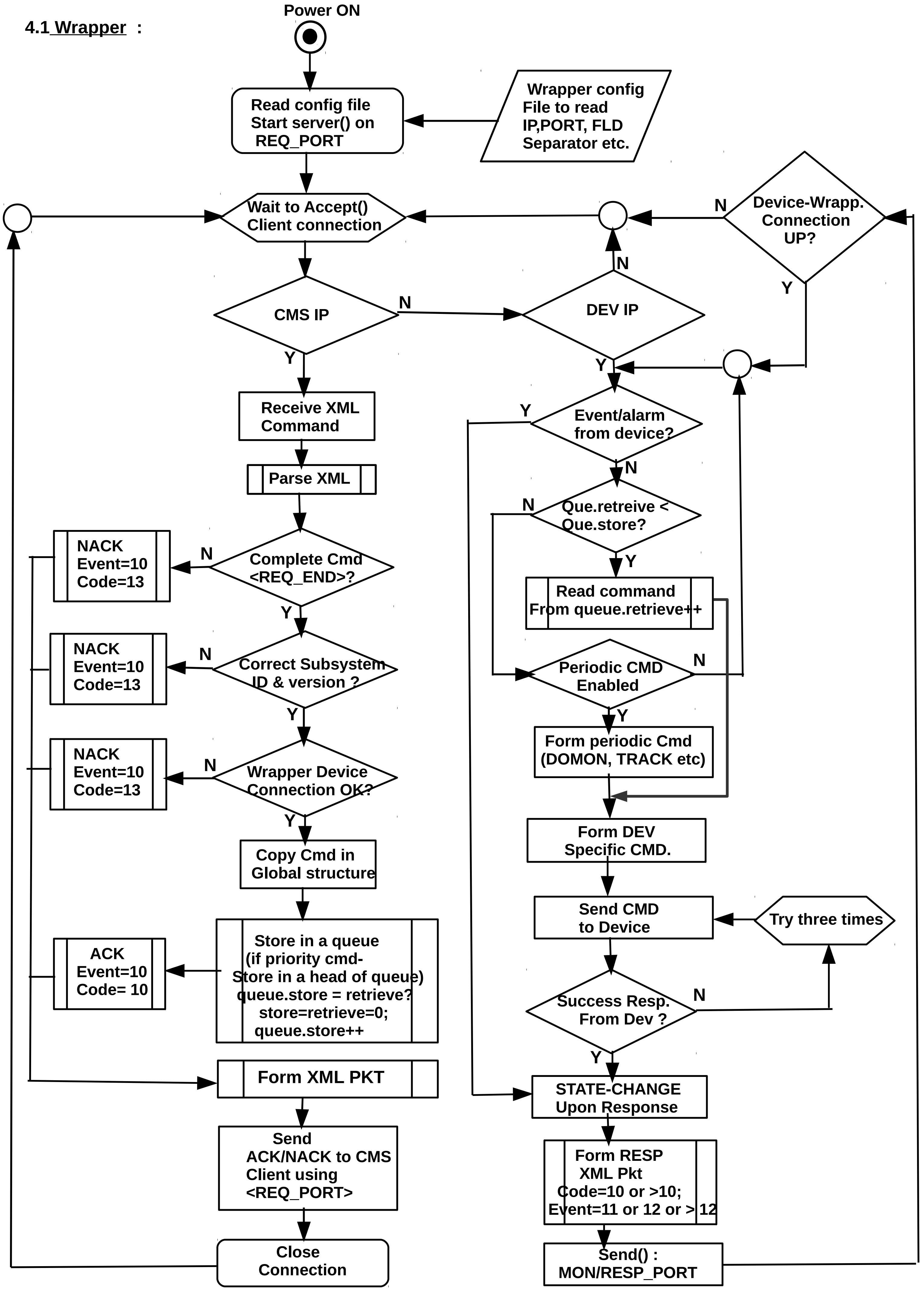
Whether command is periodic or normal will be decided by using some tag, since values for the tags <id> and <name> are same for the sub-system (other than servo subsystem), we either need to define separate tag or existing *<priority>x</priority*> tag will be used where priority with some specific value will say about - whether command is a *priority command, normal command or periodic command*. As there is not only a DOMON command but TRACK or DOPLOT kind of commands are also needed.

Response Monitoring strcuture :

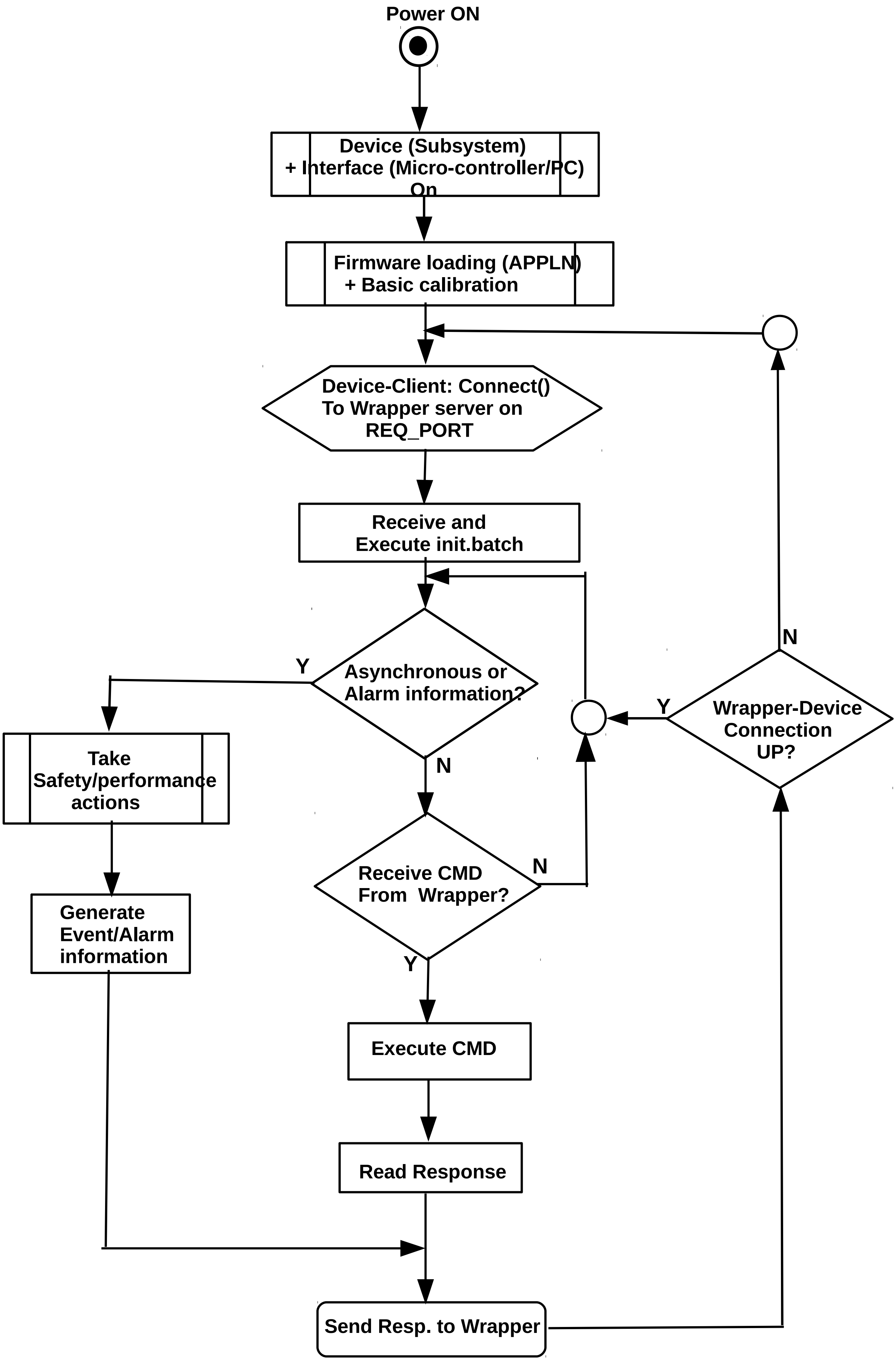
```
<responses>
<response>
    <seq>1</seq>
   <id>41</id>
   <name>Monitoring_Data</name>
   <systemid>servo</systemid>
   <version>1.0</version>
   <timestamp></timestamp>
   <code>10</code>
   <event>12</event>                // Event 12 means information from sub-system/device.
   <message>System Mon</message>
   <data>
  <param> <name>AMPOL</name> <value>0</value> </param>
  <param> <name>POSG</name> <value>1</value> </param>
  <param> <name>az_motor2_current</name> <value>42.56</value> </param>
  <param> <name>az_motor1_current</name> <value>43.12</value> </param>
  <param> <name>CWRP</name> <value>0</value> </param>
  <param><name>az_tacho1</name><value>442</value></param>
  <param><name>az_tacho2</name><value>31</value></param>
  <param><name>SSCOK</name><value>0</value></param>
  <param><name>STPOS</name><value>0</value></param>
  <param><name>ENCKOK</name><value>0</value></param>
  <param><name>CWPL</name><value>0</value></param>
  <param><name>SLWG</name><value>0</value></param>
  <param><name>el_pp</name><value>131:50:10.0</value></param>
  <param><name>LOC</name><value>0</value></param>
  <param><name>CWFL</name><value>0</value></param>
  <param><name>TRKG</name><value>0</value></param>
  <param><name>wind_vel2</name><value>72</value></param>
  <param><name>az_cp</name><value>131:50:10.0</value></param>
  <param><name>timeofday</name><value>02:32:22</value></param>
  <param><name>wind_vel1</name><value>72</value></param>
  <param><name>el_tacho1</name><value>986</value></param>
  <param><name>el_cp</name><value>131:50:10.0</value></param>
  <param><name>REM</name><value>0</value></param>
  <param><name>BRK1</name><value>0</value></param>
  <param><name>el_tp</name><value>131:50:10.0</value></param>
  <param><name>BRK2</name><value>0</value></param>
  <param><name>ACKOK</name><value>0</value></param>
  <param><name>el_motor1_current</name><value>42</value></param>
  <param><name>RUN</name><value>0</value></param>
  <param><name>el_motor2_current</name><value>20.5</value></param>
  <param><name>WND50</name><value>0</value></param>          // for e.g. Digital Flag
  <param><name>WND85</name><value>0</value></param>
  <param><name>CCPL</name><value>0</value></param>
  <param><name>MAN</name><value>0</value></param>
  <param><name>STRLSD</name><value>0</value></param>
  <param><name>CCFL</name><value>0</value></param>
  <param><name>STWD</name><value>0</value></param>
  <param><name>az_tp</name><value>18:50:10.0</value></param>
  <param><name>az_pp</name><value>21:50:10.0</value></param>
  </data>

</response>
  </responses>
<RESP_END>
```

4. FLOW-CHART

4.1 Wrapper :


Power ON
Read config file Start server() on REQ_PORT
Wrapper config File to read IP,PORT, FLD Separator etc.
Wait to Accept() Client connection
N
Device-Wrapp. Connection UP?
Y
CMS IP
N
DEV IP
N
Y
Y
Receive XML Command
Parse XML
Y
Event/alarm from device?
N
N
Que.retreive < Que.store?
Y
Read command From queue.retrieve++
N
Complete Cmd <REQ_END>?
NACK Event=10 Code=13
Y
N
Correct Subsystem ID & version ?
NACK Event=10 Code=13
Periodic CMD Enabled
N
Y
Y
Form periodic Cmd (DOMON, TRACK etc)
N
Wrapper Device Connection OK?
NACK Event=10 Code=13
Y
Copy Cmd in Global structure
Form DEV Specific CMD.
Send CMD to Device
Try three times
Store in a queue (if priority cmd- Store in a head of queue) queue.store = retrieve? store=retrieve=0; queue.store++
ACK Event=10 Code= 10
Success Resp. From Dev ?
N
Y
Form XML PKT
STATE-CHANGE Upon Response
Send ACK/NACK to CMS Client using <REQ_PORT>
Form RESP XML Pkt Code=10 or >10; Event=11 or 12 or > 12
Close Connection
Send() : MON/RESP_PORT

4.2 Device (Subsystem)



5. TEST Cases

TEST Case CMS	WRAPPER	DEVICE (SUBSYSTEM)
1. UP (i) Read the previous state (ii) Connect to Subsystems' Wrappers on given IP:REQ_PORT (iii) Initialize the subsystems (iv) As a UI, Batch or state resp. Send XML CMD, wait for ACK or NACK from the wrapper. (v) Read the responses for the CMD Send and Asynchronous event/ Alarm on RESP_PORT. (v) Read Monitoring/alarm/state Information periodically on MON_PORT	UP (i) Read the wrapper config file (ii) start server and wait to accept Connection on REQ_PORT (iii) If Connection from device - Start consumer process. (iv) If the connection from the CMS, parse XML command, send It to device and read response. (v) Send Responses (event, cmd Response, alarms on RESP_PORT). (vi) Send periodic Subsystem Monitoring on MON_PORT.	UP (i) Do the basic calibration and Initialization. (ii) Try to Connect to the Wrapper on given IP:REQ_PORT. IF Succed goto iii ELSE goto ii. (iii) Execute the init.batch script (iv) Read the alarm or events from System if any and send it to the Wrapper (v) Receive CMD from the wrapper, Execute it and send the response To the wrapper. (vi) If connection broken goto ii. Else goto iv.
2. DOWN	UP (i) Wait to accept connection from the CMS on REQ_PORT. (ii) If Periodic command enabled (i.e. DOMON/TRACK) send to device (iii) Read monitoring/STATE or alarm from system. (iv) Try to send XML-PKT to MON_PORT (v) If packet fails, increment Failure counter. (vi) if failure counter do not exceed certain threshould go to i. (vii) Shutdown the system i.e. Park antenna	UP Same as in Test-case 1.
3. UP Same normal mode as mentioned in Test-case 1. except : (i) If the CMS could not read the Wrapper information from a particular sub-system, put that Subsystem in SUSPEND mode. (ii) If use-case require that CMS is also in SUSPEND Mode due to a particular subsystem is down, put the CMS in SUSPEND Mode (iii) Keep polling the i.e. Trying To establish connection. (iv) If connection establish, execute last-restoring stage. (v) Read the MON para from sub-system. (vi) Wait so that Manual intervation need to take out CMS From the SUSPEND to NORMAL Mode. (needed?)	DOWN	UP Same as in Test-case 1. Except – Client Keep on polling so that WRAPPER-Device connection Is established once the wrapper is up. Untill that time the Device is In stage (ii) or (iv) of Test-case 1. i.e. Asynchronous event can be Read, if require safety action can Be take irrespctive of whether wrapper is communicating or not.
4. UP Same as test case-3 Except (iii) of test case-3 i.e. The CMS don't have to poll the wrapper as connection already established.	UP (i) If wrapper could not establish Communication to the DEVICE. Put wrapper-Device communication Variable false. (ii) Form the XML message with Event=10, Code=13 with message <SUBSYSTEM> is giving Timeout	DOWN

5. TEST Cases

TEST Case	CMS	WRAPPER	DEVICE (SUBSYSTEM)
5.	UP Same mode as mentioned in Test-case 3.	DOWN	DOWN
6.	DOWN	UP Same as Test case 1 up to item ii.	DOWN
7.	DOWN	DOWN	UP Same as Test case 1.

END