# Message Format

# ver 0.8

**Message Packet Structure**

- NodeBus내 메세지는 Packet단위로 구성되며 Packet은 Header와 Data로 조합된다. Data은 옵션이며 Data가 있을 경우 Header내에 그 길이가 기록되어야 한다. Header는 XML 형식이며 반드시 start code와 XML의 길이를 명시한다. Data는 바이트 배열로 취급한다.

**Packet = <header><data: optional>**

**<header>**

**start string(11 bytes) + XML length(4 bytes) + <Header XML>**

**<data>**

**bytes array**

**start string = AsTaLaViStA**

**example) 67421 bytes log event, XML header: 108 bytes**

**AsTaLaViStA** **108** <event type=”LOG” format=”JSON” length=”**67421**”/><error type=”0”/><tvc>description:system crashed</tvc></event>kernel oops:..a.siasdf.,.,slfjalksf address=0xjfdiaffdklladfjlasdkjflkafalsdjflkasdlfalkjd...

**example) 72233 bytes image data, XML header: 94 bytes**

**AsTaLaViStA** **94** <event type=”4”><img type=”jpeg” length=”**72233**”/><tvc>detail:captured image file</tvc></event>jflaksjdflkasdlkfflasdfalsd...

1. Event

|  |  |
| --- | --- |
| Event Type | Description |
| REGOK | Register OK |
| REGFAIL | Register Failure |
| LOG | New Log Data |
| ERROR | Error |
| IMG | Image Data |
| TC | Testcase result(End of Testcase) |
| CMD | Command result |
| INFO | System information |

|  |  |
| --- | --- |
| Log Format | Description |
| TXT | Simple Text |
| JSON | JSON format |
| XML | XML format |
| BIN | Binary format |

|  |  |
| --- | --- |
| Error ID | Description |
| SYS | System Error |
| CMD | Command Error |

Syntax:

<event type=”string” | format=”string” length=”string” | id=”string” | res=”string”>

[<command></command>]

</event>

Examples:

<event type=”REGOK”/> //(From server to client) Client Registered to Server Success.

<event type=”LOG” format=”JSON” length=”29113”/> // Log JSON Message

<event type=”LOG” format=”TXT” length=”31582”/> // Log TXT Message

<event type=”ERROR” id=”SYS”/> // System error

<event type=”IMG” format=”JPG” length=”22423”/> // Image data

<event type=”TC” id=”1234” res=”1”/> // TestCase 1234 completed OK

<event type=”CMD” res=”0”> //Command failure

<command type=”RPC”> // ATM RPC

<arg>KADP\_FID\_2</arg>

<arg>ATLIB\_KADP\_SYS\_ReadReg</arg>

<arg>0xc0035009 0x10</arg>

</command>

</event>

<event type=”INFO” format=”JSON” length=”1422”/> //System information JSON

<event type=”INFO” format=”TXT” length=”1891”/> //System information TXT

2. Command

|  |  |
| --- | --- |
| Command Type | Description |
| STAY | Stay |
| INFO | Get system information |
| LOG | Get log data |
| IMG | Capture Image |
| STOP | Stop AutoTest |
| PAUSE | Pause AutoTest |
| WAIT | Wait |
| RESUME | Resume AutoTest |
| EPK | Flashing EPK |
| LUA | Run lua script |
| FILE | Get target file |
| KEY | Send key strings to target |
| RPC | Call ATLib Function |
| ARPC | Call ATLib Function Asynchronously |
| TC | Run remote TC file |
| SH | Run shell script |
| CFG | Config ATMgr |
|  |  |

Examples)

<command type=”STAY”/> // Just stay.

<command type=”INFO”/> // Get System Information.

<command type=”LOG”/> // Get Log data

<command type=”IMG”/> // Capture Image

<command type=”STOP”/> // Stop AutoTest

<command type=”PAUSE”/> // Pause AutoTest

<command type=”RESUME”/> // Resume AutoTest

<command type=”EPK”> // Flash EPK

<arg>tftp://10.10.10.10/epk/test.epk</arg>

</command>

<command type=”INFO”> // Retrieve target simple info

<arg>soc</arg> // get soc name of target

</command>

<command type=”RPC”> // Call ATLib function

<arg>KADP\_FID\_2</arg> // Function ID

<arg>ATLIB\_KADP\_SYS\_ReadReg</arg> // Function Name

<arg>0xc0035009 0x10</arg> // Function parameter

</command>

<command type=”ARPC”> // Call function of ATLib asynchronously

<arg> Remote Log URL </arg> // upload the log of func to this URL

<arg>KADP\_FID\_2</arg> // Function ID

<arg>ATLIB\_KADP\_SYS\_ReadReg</arg> // Function Name

<arg>0xc0035009 0x10</arg> // Function parameter

</command>

<command type=”TC”> // download TC file and execute it

<arg>tftp://10.10.10.10/tc/testcase1.tc</arg> // tc file

<arg> tftp://10.10.10.10/tc/log/tc1.log </arg> // upload the tc log

</command>

<command type=”SH”> // Download shell script file and then execute it in the target

<arg>tftp://10.10.10.10/sh/system\_info.sh</arg> //shell file

</command>

<command type=”CFG”> // Set ATMgr. configuration

<arg>AM</arg> // config key(AM : Autotest Mode)

<arg>1</arg> // 1: enable ,0:disable

</command>

3. Testcase

<testcase id=”5123”> // Testcase 5123.

<repeat>10000</repeat> // Run 10000 times.

<commands>

<command type=”RPC”>

<arg>KADP\_FID\_2</arg>

<arg>ATLIB\_KADP\_SYS\_ReadReg</arg>

<arg>0xc0035009 0x10</arg>

</command>

<command type=”WAIT” time=”100”/> // wait 100ms

<command type=”RPC”>

<arg>KADP\_FID\_2</arg>

<arg>ATLIB\_KADP\_SYS\_ReadReg</arg>

<arg>0xc0035009 0x10</arg>

</command>

<command type=”WAIT” time=”200”/> // wait 200ms

</commands>

</testcase>