



Overview of YNC / YRSC

Ver 1.0

YAMAHA CORPORATION
AV Receiver Group
Products Development Department
AV Products Division

1	Preface	4
2	What's YNC/YRSC?	4
3	Basic Concept of YNC / YRSC Protocol	5
3.1	Function Tree.....	5
3.1.1	Expression of PUT & GET commands.....	6
3.1.2	Expression of Event Notification command	6
3.2	PUT Command.....	6
3.3	GET Command.....	8
3.4	Event Notification Command	9
4	Overview of YNC Protocol	10
4.1	Communication Format.....	10
4.2	Device Detection.....	10
4.3	Command Format	10
4.4	Communication Protocol	12
5	Overview of YRSC Protocol	15
5.1	Communication Format.....	15
5.2	Command Format	15
5.3	Communication Protocol	17
6	Remarks	20

1 Preface

This document describes an overview of YNC / YRSC Protocol of Yamaha AV products.

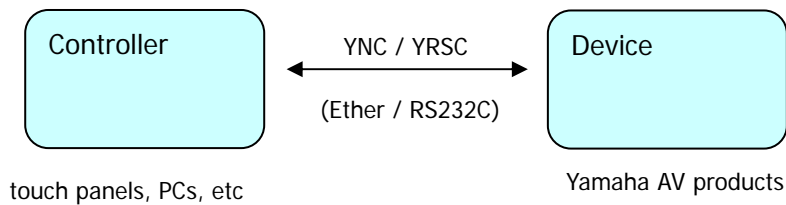
2 What's YNC/YRSC?

- YNC and YRSC are the communication protocols for device controllers such as touch panels or PCs to control Yamaha AV products. They are abbreviations for;

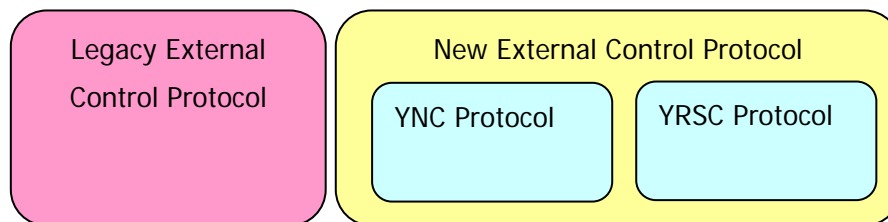
Yamaha Network Command (YNC)

Yamaha RS-232C Command (YRSC)

and they offer Ethernet or RS-232C communications respectively.



- Previous generations use a different Yamaha RS-232C protocol, but it wasn't adopted for a next generation protocol in an era of Ethernet network control. YNC/YRSC will be replacing it. (Note: You should be able to know which model uses which protocol by referring to its control protocol specification.)



- YNC/YRSC could be implemented both or either one of them in Yamaha AV products, that depends on model by model.

In the following chapters, the common characteristics of YNC/YRSC protocol will be initially provided as a basic concept. Then differences between YNC and YRSC are to be specified with actual command examples.

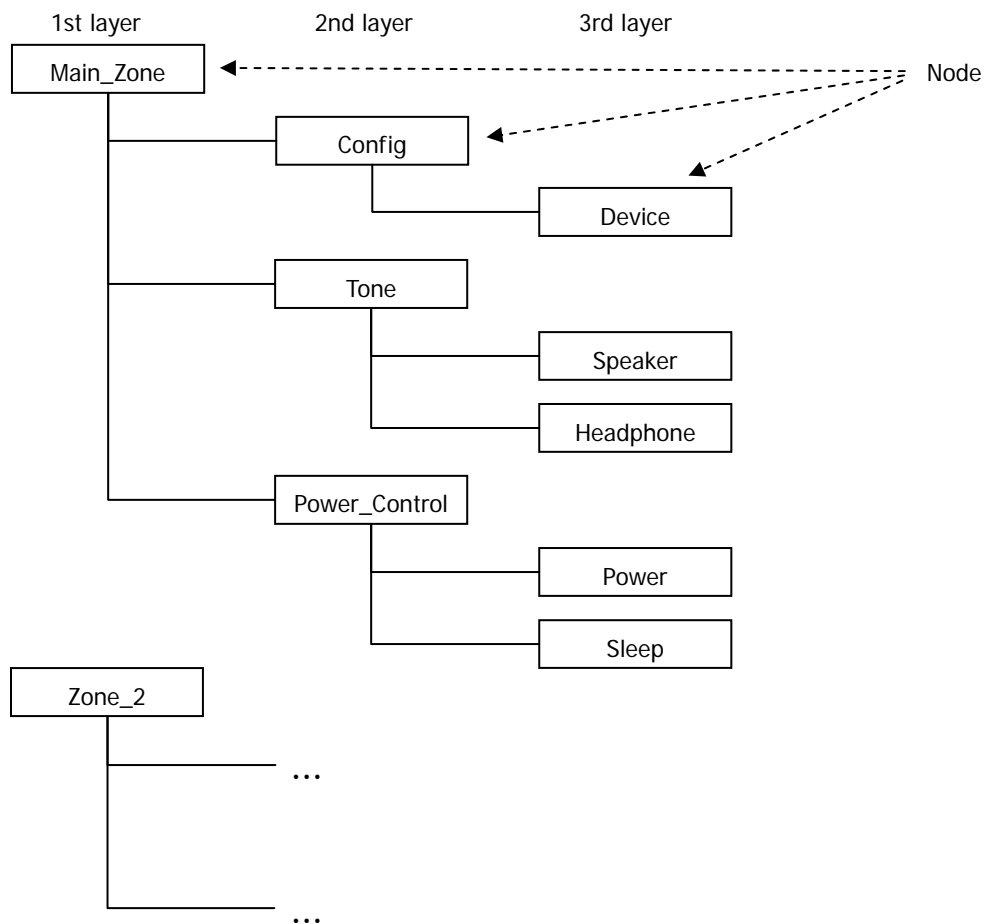
3 Basic Concept of YNC / YRSC Protocol

In this chapter, the basic concept of YNC / YRSC Protocol are explained.

3.1 Function Tree

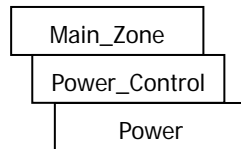
The command structure of YNC/YRSC is reflected in the “function tree”, which is represented by “nodes” of the Device’s function in a structured (hierarchical) tree form. The following diagram shows an example of the function tree.

Example of Function Tree



3.1.1 Expression of PUT & GET commands

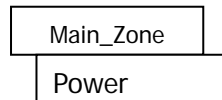
To control the Device using YNC/YRSC is just to change (PUT) or read (GET) a certain status of end nodes in the function tree. So the essential command structure in the case of controlling the end node of "Power" (found in the diagram of the previous page) can be represented as shown below;



In this chapter, the command structure of PUT/GET commands to a node will be represented abstractly like shown above (not in the actual form when used).

3.1.2 Expression of Event Notification command

Events arise when status (as will be described in detail later) changes in the Device and the corresponding Event Notification commands are sent to the Controller simultaneously (note: generally the Controller will try to read changed status (or send GET commands) when it receives Event Notification commands from the Device). In addition, events are separately managed in a zone-by-zone unit. For example, when the status of "Power" in "Main Zone" is changed, its Event Notification command structure can be abstractly represented as shown below in compliance with the same rule mentioned in the previous section, which will be applicable through this chapter.

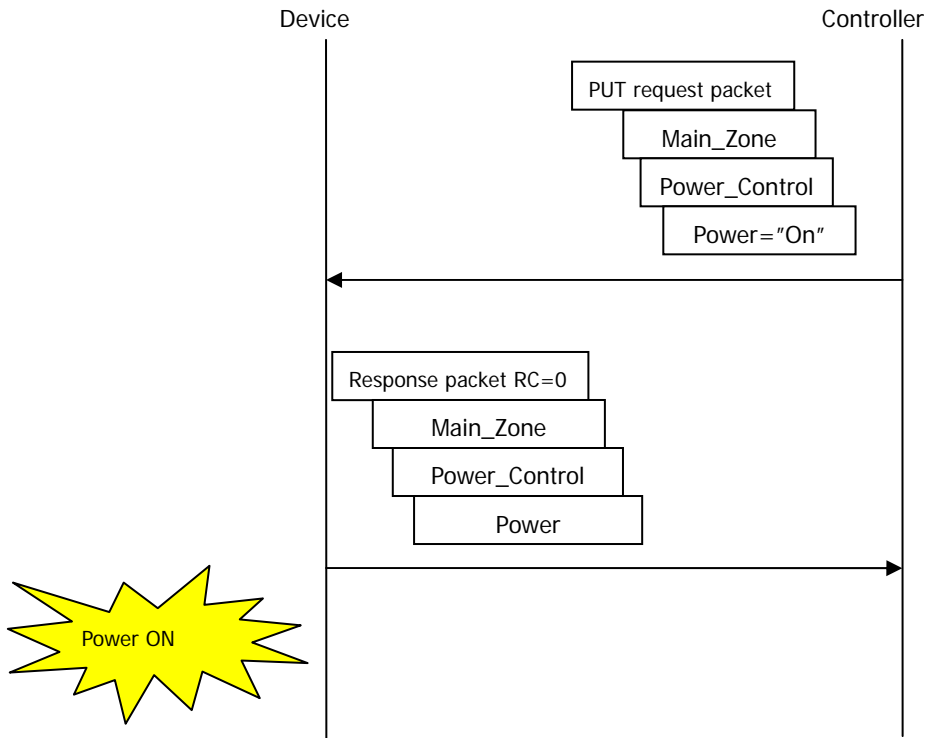


3.2 PUT Command

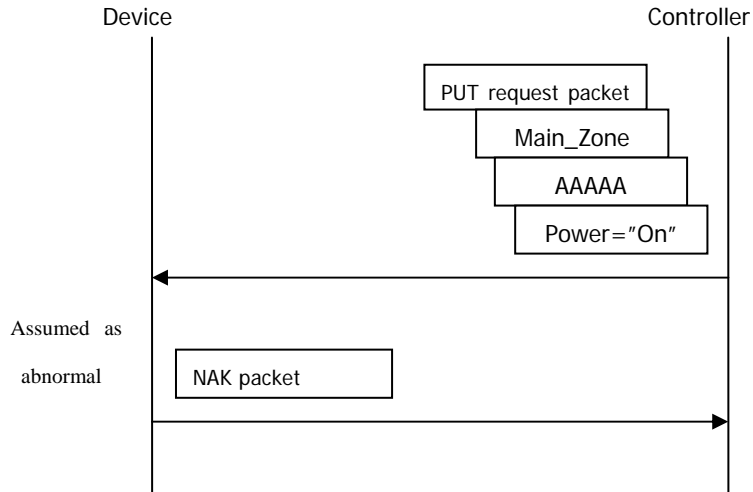
Put Command is the process to set the status of the end node for a Device from the Controller. The Device that received a request packet sends a response packet to the Controller after setting the status.

If the request packet is not well-formed, then a negative acknowledgment packet (NAK) is returned. The response packet includes RC (Return code) value that is embedded in the response packet as the processed result for the request packet.

Example of PUT command normal processing



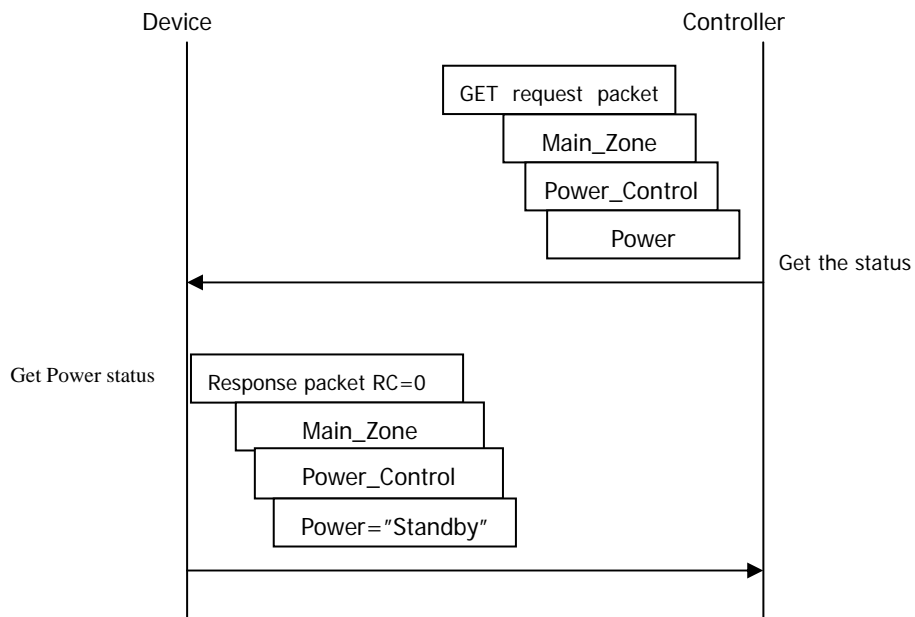
Example of PUT command abnormal processing



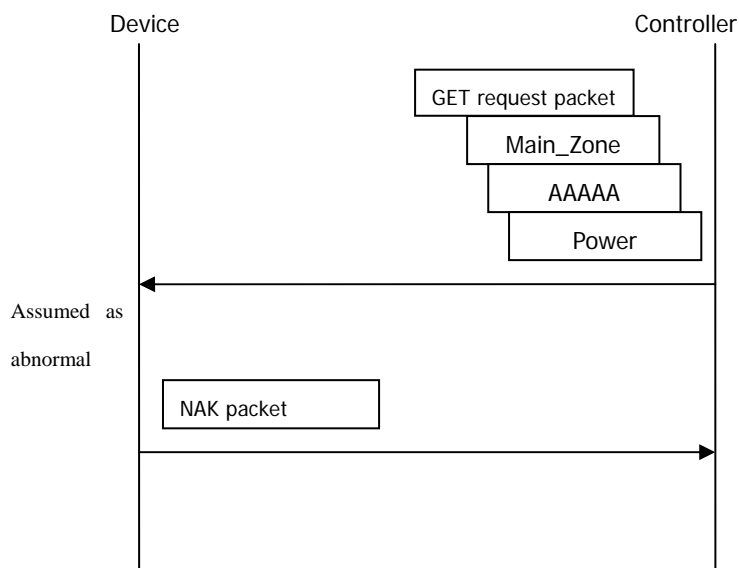
3.3 GET Command

Get Command is the process to get the status of the end node of the Device from the Controller. If the Device receives a request packet it sends either a response packet or negative acknowledgment (NAK) packet as PUT Command.

Example of GET command normal processing



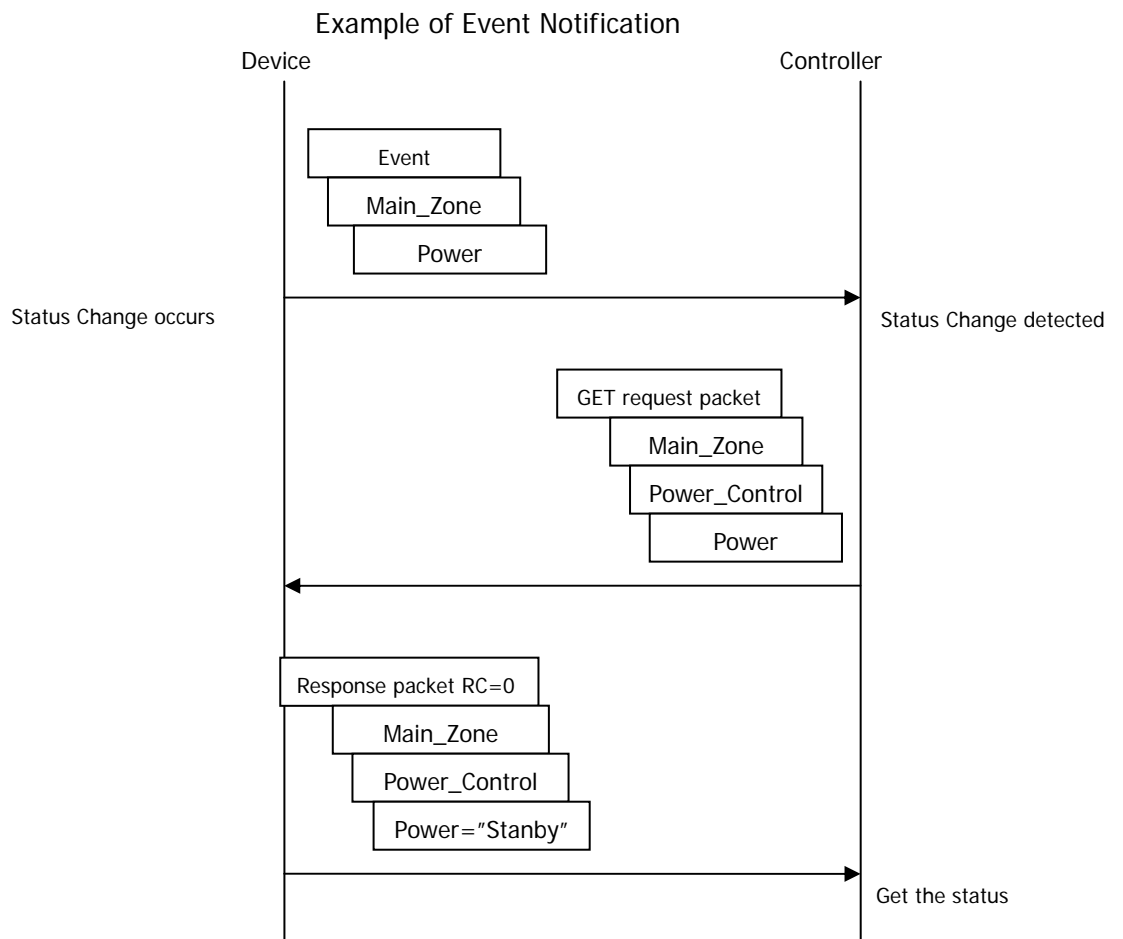
Example of GET command abnormal processing



3.4 Event Notification Command

When one of the four types of Device status of Power, Input, Volume, or Play_Info are changed, the device instantly sends an Event Notification command to the controller, notifying the function name and Zone name to which it belongs. However, the Event Notification does *not* give any detail of how exactly the status was changed while telling it was *just changed*. Therefore the controller may have to send necessary GET Command(s) in order to get content of changed status. In addition, the Event Notification command is a one-way process from the Device to the Controller. So the controller shall not send any response packets to the Device as a reaction to the Event Notification commands.

Also, it is recommended that the Controller poll the four types of Device status; Power, Input, Volume, and Play_Info at regular intervals by GET commands since it is *NOT* guaranteed for Event Notification commands to be surely delivered to the Controller due to the communication traffic.



Note: In this example, Main_Zone - Power_Control - Power command is sent to get the changed contents of status "Power", but this is just an example. Main_Zone - Basic_Status - Power_Control - Power command can also be an alternative for getting the same status

4 Overview of YNC Protocol

In this chapter, the concrete concept of YNC Protocol is explained.

4.1 Communication Format

- TCP/IP

PUT Command, GET Command : HTTP (TCP)

Event Notification Command, Device Detection : UPnP AV/DLNA

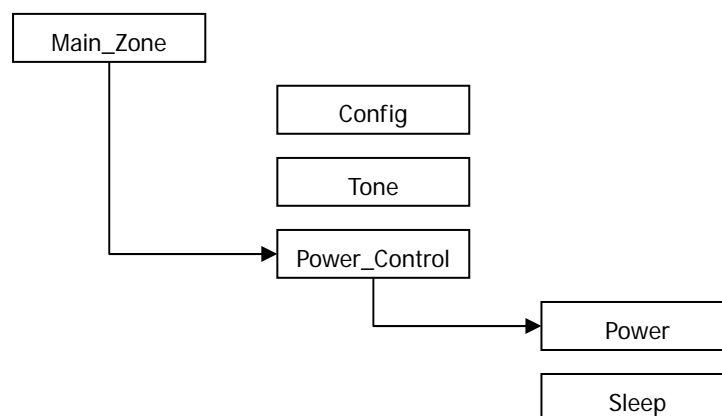
4.2 Device Detection

- The Device complies with UPnP AV/DLNA of Media Renderer and the Controller detects the Device using M-SEARCH method.

The Controller can also detect the Device with the multicast NOTIFY packet periodically sent by the Device. (See `***_ETHERNET_IF_Spec_e.xls` for more details. (the Device name is assigned to `***`))

4.3 Command Format

- PUT command & GET command are communicated using the post method of http. The body of POST method shall be documented in XML form which starts from the top node of the Device function tree to the end node.



The XML form is illustrated as follows when "Power" is set to "On" in this function tree.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<YAMAHA_AV cmd="PUT">
  <Main_Zone>
    <Power_Control>
      <Power>On</Power>
    </Power_Control>
  </Main_Zone>
</YAMAHA_AV>
```

These commands are generated by the tool of ***_EXT_CTRL_Function_Tree.xls
(the Device name is assigned to ***).

They can be also generated from the dump list of each model.

- The Event Notification command uses NOTIFY method of UPnP AV/DLNA.
The body of NOTIFY method is documented in XML form to be traced back to find out which Zone function has been changed.
The XML form is illustrated as follows when "Power" is set from "On" to "Standby" in Zone 2.

```
<?xml version="1.0" encoding="utf-8"?>
<YAMAHA_AV cmd="EVENT">
  <Zone_2>
    <Property>Power</Property>
  </Zone_2>
</YAMAHA_AV>
```

4.4 Communication Protocol

4.4.1 PUT Command

- If there is no error in response commands, the status code 200 is returned.
- If there is any problem in the respond packet such as the command is not in XML well-formed, then the status code 400 is returned.
- RC value is documented as an attribute of YAMAHA_AV tag upon its return.
- It is the command to set up the Device setting; therefore, the next request packet shall not be sent until a response packet or NAK packet is returned.

Request Sample (Controller=>Device):

```
POST /YamahaRemoteControl/ctrl HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-length: 114
Host: 192.168.2.74
```

```
<?xml version="1.0" encoding="utf-8"?>
<YAMAHA_AV cmd="PUT">
  <System>
    <Mem_Guard>Off</Mem_Guard>
  </System>
</YAMAHA_AV>
```

Response Sample(Device=>Controller):

```
HTTP/1.1 200 OK
Server: YAMAHA_RX-V3900/DSP-AX3900
Content-Type: text/xml; charset=utf-8
Content-length: 118
```

```
<?xml version="1.0" encoding="utf-8"?>
<YAMAHA_AV rsp="PUT" RC="0">
  <System>
    <Mem_Guard></Mem_Guard>
  </System>
</YAMAHA_AV>
```

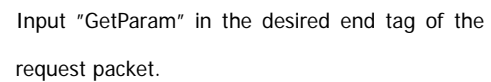
4.4.2 GET Command

- If there is no error in response commands, the status code 200 is returned.
- If there is any problem in the respond packet such as the command is not in XML well-formed, then the status code 400 is returned.
- RC value is documented as an attribute of YAMAHA_AV tag upon its return.

Request Sample (Controller=>Device):

```
POST /YamahaRemoteControl/ctrl HTTP/1.1
Host: 192.168.1.102
Content-Length: 178
Cache-Control: no-cache
```

```
<?xml version="1.0" encoding="utf-8"?>
<YAMAHA_AV cmd="GET">
<Main_Zone>
  <Power_Control>
    <Power>GetParam</Power>
  </Power_Control>
</Main_Zone>
</YAMAHA_AV>
```



Input "GetParam" in the desired end tag of the request packet.

Response Sample(Device=>Controller):

```
HTTP/1.1 200 OK
Server: YAMAHA_Network_Receiver/2.0 (RX-Z7)
Content-Type: text/xml; charset=utf-8
Content-Length: 111
```

```
<?xml version="1.0" encoding="utf-8"?>
<YAMAHA_AV rsp="GET" RC="0">
<Main_Zone>
  <Power_Control>
    <Power>On</Power>
  </Power_Control>
</Main_Zone>
</YAMAHA_AV>
```

4.4.3 Event Notification Command

- Multicast transmission from the Device

Event Notify Sample (Device=>Controller):

NOTIFY * HTTP/1.1

Host: 239.255.255.250:1900

NT: urn:yamaha-com:service:YamahaRemoteControl:2

NTS: yamaha:propchange

USN: uuid: <UUID>::urn:yamaha-com:service:YamahaRemoteControl:2

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<YAMAHA_AV cmd="EVENT">
```

```
  <Main_Zone>
```

```
    <Property>Power</Property>
```

```
  </Main_Zone>
```

```
</YAMAHA_AV>
```

5 Overview of YRSC Protocol

In this chapter, it explains the concrete concept of YRSC Protocol.

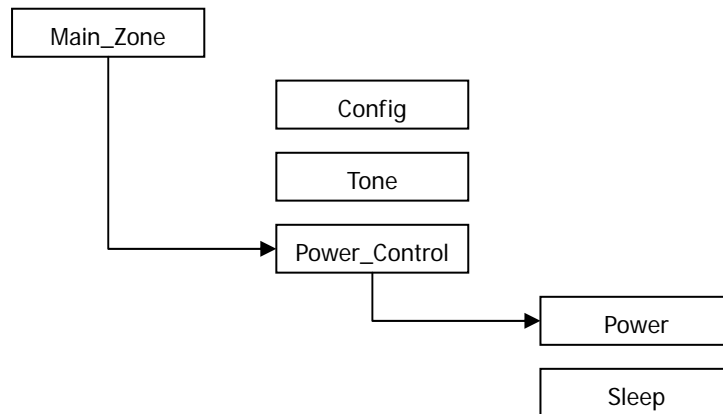
5.1 Communication Format

- RS-232C Compliance Full duplex Asynchronous communication

5.2 Command Format

- PUT command & GET command are communicated using the text format.

(see `***_RS232C_IF_Spec_e.xls` for the details of the format . (the Device name is assigned to `***`))



The command when "Power" is set to "On" in this function tree is illustrated as follows.

```
PUT,Main_Zone(Power_Control(Power=On))
```

The command when the value of "Power" is obtained in this function tree is illustrated as follows.

```
GET,Main_Zone(Power_Control(Power))
```

These commands are generated by the tool of `***_EXT_CTRL_Function_Tree.xls` (the Device name is assigned to `***`).

They can be also generated from the dump list of each model.

- Text format is used for the Event Notification command.
(See `***_RS232C_IF_Spec_e.xls` for the details of the format. . (the Device name is assigned to `***`))

The XML form is illustrated as follows when "Power" is set from "On" to "Standby" in Zone 2.

```
EVENT,(Main_Zone(Power),Zone_2(Power))
```


5.3 Communication Protocol

5.3.1 Packet Format

- A header and footer are added to the command mentioned in section 5.2 and then send it as a packet.

The size of a PUT command or a GET command can be big; therefore, it shall be divided into several packets with one packet being max 256 bytes. (See ****_RS232C_IF_Spec_e.xls* for more details. (the Device name is assigned to *****))

5.3.2 Packet Sample

- PUT Request Packet
`0,1,28,PUT,System(Memory_Guard=Off),SumCRLF`
- GET Request Packet
`0,1,24,GET,System(Memory_Guard), SumCRLF`
- Event Notification Packet
`0,1,47,EVENT,(Main_Zone(Power,Input,Volume,Play_Info)), SumCRLF`
- GET Response Divided Packet 1/2
`1,2,256,GET,RC=0,NET_USB(List_Info(Menu_Layer=1,Menu_Name==,Current_List(Li
ne_1(Txt==,Container=False),Line_2(Txt==,Container=False),Line_3(Txt==,Contain
er=False),Line_4(Txt==,Container=False),Line_5(Txt==,Container=False),Line_6(Txt
==,Container=False),Line,216CRLF`
- GET Response Divided Packet 2/2
`0,2,103,_7(Txt==,Container=False),Line_8(Txt==,Container=False)),Cursor_Position
(Current_Line=1,Max_Line=0))),147CRLF`

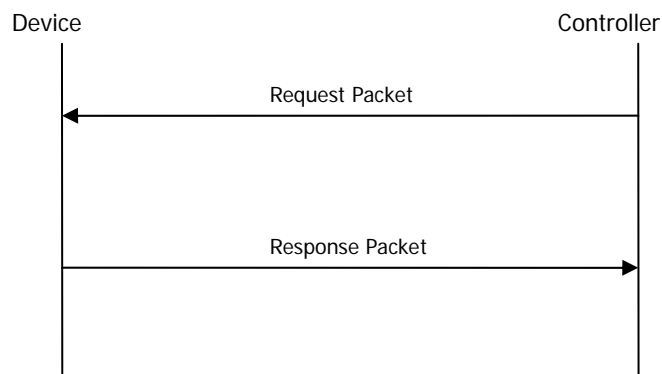
5.3.3 Divided Packet Sequence of PUT Command & GET Command

- The request packet from the Controller shall not send the next request packet until it completes receiving the response packet from the Device.
- If the size of one command exceeds 256 bytes, it shall be divided into several packets with one packet being under 256 bytes.
- Once the Device receives a divided "command" packet from the Controller, it sends back the NEXT packet to the Controller. The Controller shall not send the next divided command unless it receives the NEXT packet from the Device (If the Device couldn't get the next divided command from the Controller within 5 seconds after it sends the NEXT packet, the process will time out).

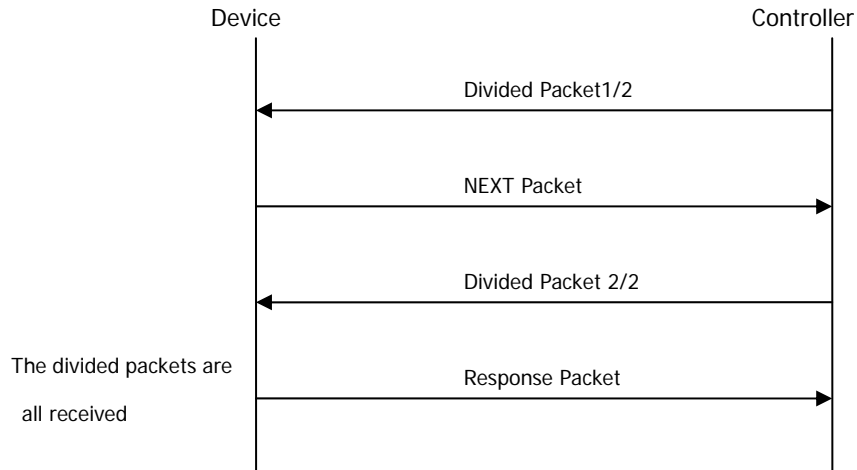
Similarly, once the Controller receives a divided "response" packet from the Device, it shall send back the NEXT packet to the Device in order to let the Device send the next one. The Device won't send the next divided response unless it gets the NEXT packet from the Controller (If the Device couldn't get the NEXT packet from the Controller within 5 seconds, the process will time out).

(See `***_RS232C_IF_Spec_e.xls` for NEXT packet format. (the Device name is assigned to `***`))

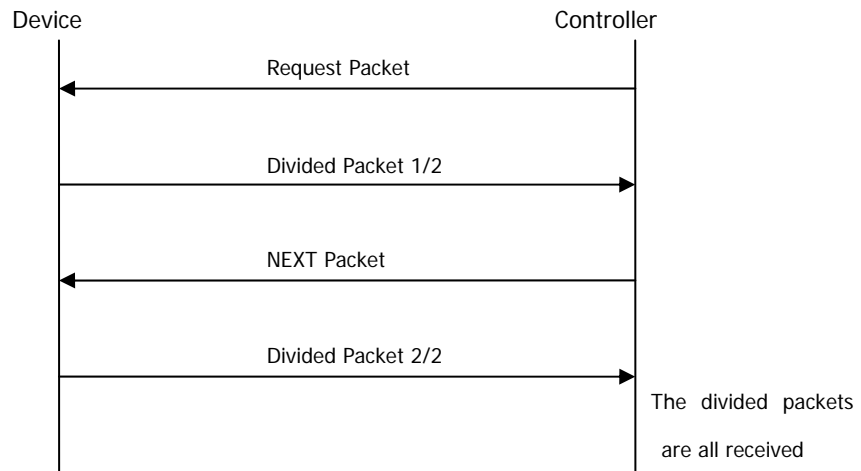
Both the request and response packets are not divided (under 256 Bytes)



The command from the controller (request packet) is divided

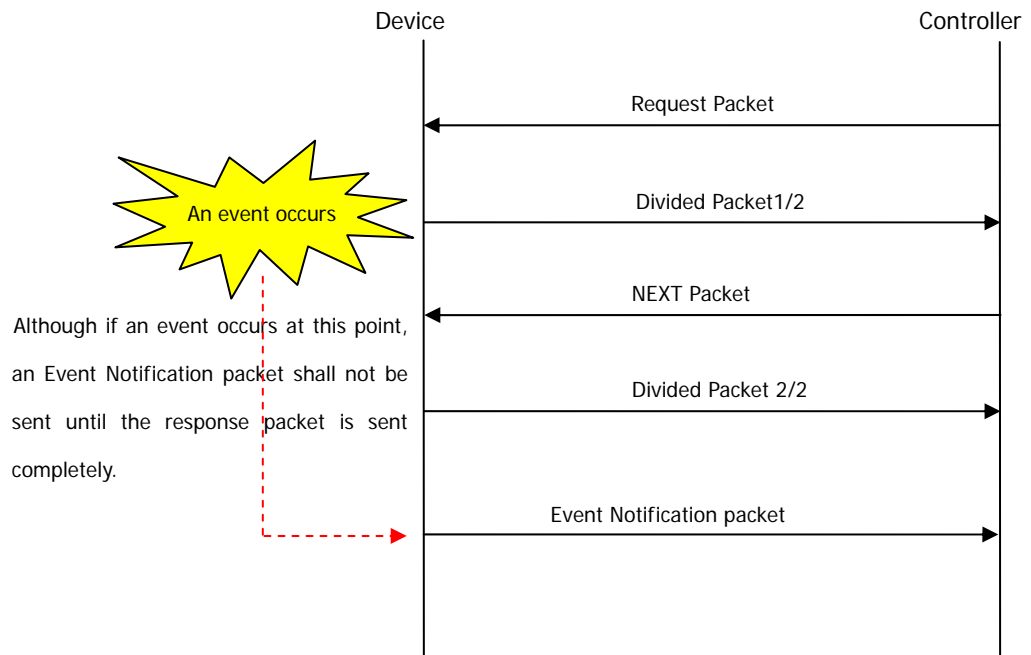


The command from the device (response packet) is divided



5.3.4 Sequence of the Event Notification command

- The Event Notification command is not divided into several packets as PUT command or GET command.
- When an event occurred in the Device while PUT command or GET command are being processed, the Device shall not send an Event Notification command until the response packet is sent completely.



6 Remarks

References

- `***_ETHERNET_IF_Spec_e.xls` . (the Device name is assigned to `***`)
 ...The detailed specification regarding YNC Protocol
- `***_RS232C_IF_Spec_e.xls` . (the Device name is assigned to `***`)
 ...The detailed specification regarding YRSC Protocol
- `***_EXT_CTRL_Function_Tree.xls` (the Device name is assigned to `***`).
 ...The tool to generate YNC or YRSC commands
- The entire dump list
 ...A list of YNC or YRSC commands of each model