VON NETWORK USER MANUAL

Manual Version 0.1

Written by: Syvil

SEL COMPUTER & NETWORKING

I. Table of contents

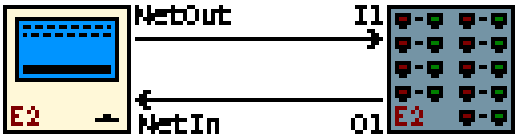
Table of Contents

II. VNet Basics

The VON network (VNet) operates on the sending and receiving of VON encoded tables. The content of the encoded table consists of named values, including but not limited to numbers, text, vectors, angles, arrays, tables, and any other type supported by VON encoding. For the remainder of this document, the VON strings will be referred to as “packets”, and the tables encoded within them as the “contents” of the packet. As well, a matching pair of inbound and outbound connections will be referred to as a “channel”.

Devices on the network are expected to have a string input and string output for sending and receiving packets. As a standard for devices with a single network channel, the input and output connections should be named “NetIn” and “NetOut” respectively. In the case of devices designed to handle multiple channels, the input and output connections may be abbreviated to “I#” and “O#” respectively, where # is replaced with an identifying number for the channel.

When connecting two devices, the network input of the first device should be connected to the network output of the second device, and the network input of the second device to the network output of the first device. If either of the devices has multiple channels, the input and output used should have the same identifier.



When sending a packet, the NetOut string should be set to the packet for 10ms, then reset to an empty string. To receive a packet, the receiving device should be set up so that a change to the NetIn triggers the code. When NetIn is changed and is not an empty string (“”), then there is a packet. Resetting NetOut allows for a repeat of the same packet to trigger the receiving device.

If a device receives a packet on the same tick that it sent a packet, the code may be executed again before certain flags have been reset. For example, an execution is triggered by a timer, a packet is sent, and a response packet received. The execution triggered by the response would have both the flag for being triggered by an input AND the flag for being triggered by a timer true. A possible solution to this is to check that the inputClk() returns 0 when checking if a timer triggered the execution.

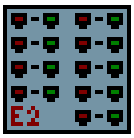
In certain device names or code, you may notice the term JNet used. This is a result of the fact that JSON was the original encoding choice. Due to the number of unsupported data types, JSON was abandoned and VON used to replace it. Any remaining instance of the term JNet is merely a holdover from before the switch.

It should be noted that Expression 2 does not use VON properly, expecting a second “types” table to identify the data type of each value stored in the VON table. While this has no effect in E2-to-E2 connections, starfall to E2 communication will require additional steps in the decoding process. Due to the substantially higher efficiency, the starfall device should be expected to handle this conversion.

III. Packet Contents

The following sections will discuss the data values expected by various VNet capable devices and what the purpose of each value is.

A. VON Data Router



Despite its name, the VON Data Router behaves like a network switch, not a router.

Data Value Name	Data Type	Example
ipr	Number	1245
Ips	Number	2
command	String	“identify_self”
[Additional Data]	Any	

Value Usage

‘ipr’: The router removes the first (left most) digit from the ‘ipr’ value and uses that to select the channel the packet will be sent on. A value of 1-9 will send the packet out on that channel. A value of 0 means the packet is for the router itself. It will process the contents and respond accordingly. If no ‘ipr’ is found, the router will send a NACK packet to the sender. The value of the NACK is a string containing “rout_bad_ip”.

‘ips’: The router adds the number of the channel that packet arrived on to the start of the ‘ips’ value. The purpose of the ‘ips’ value is for a device to return a response or otherwise identify the sender. This edit is not performed if the packet was for the router itself. An empty/missing ‘ips’ value is automatically added.

command: If the router itself is the destination (see ipr) then this command determines how it responds. The following commands are supported by the VON Data Router:

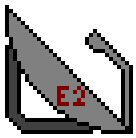
1. “net_identify”: Returns the device type and entity id to the sender. The packet is formatted as follows (assuming example values are received.)

Data Value Name	Data Type	Value
ipr	Number	2
ips	Number	0
command	String	“net_identity”
string	String	“jnet_router”
number	Number	entity():id()

Any other command will cause the router to send a NACK packet to the sender. The value of the NACK is a string containing “invalid command:” followed by the contents of the ‘command’ value. If the router is not the destination, ‘command’ falls under [Additional Data]

[Additional Data]: This encompasses any additional data values that are included in the packet. These are simply passed along to the outbound packet if the destionation is an outbound channel (see ipr)

B. JNet Uplink



For this page, we will assume the signal exceeds minimum strength.

Data Value Name	Data Type	Example
ips	Number	2
rec	number	45
command	String	"identify_self"
angle	Vector	45, 50, 0
string	String	"vxcvr"
number	Number	42
[Additional Data]	Any	

Value Usage

'ips': If the packet was for the uplink itself (see rec), the 'ips' value is used as the 'ipr' value of any response packets generated. Otherwise 'ips' falls under [Additional Data].

'rec': If the 'rec' value is not zero, and the signal exceeds minimum strength, the uplink will transmit the packet as a global range data signal. The uplink will use "rec_" prepended to the 'rec' value for the name of the data signal. If the 'rec' value is 0, then the packet is for the uplink itself, and it will then process the contents and respond accordingly. If 'rec' is missing from the packet, it will default to 0. By default, the signal group is "vxcvr" but that can be changed via network commands (see command: set_id).

'command': If the packet was for the uplink itself (see rec), then this value determines how the uplink reacts. The following commands are supported by the JNet Uplink:

1. "net_identify": Uplink returns the device type and entity id to the sender. The packet is formatted as follows (assuming example values are received.)

Data Value Name	Data Type	Value
ipr	Number	2
command	String	"net_identity"
string	String	"jnet_transciever"
number	Number	entity():id()

2. "get_ang": Uplink returns the current dish angle as a vector to the sender.
3. "get_id": Uplink returns the current receiver ID as a number to the sender. By default, the receiver ID is equal to the entity id of the uplink.
4. "get_group": Uplink returns the target signal group as a string to the sender. By default, the target signal group is "vxcvr".
5. "get_strength": Uplink returns the current signal strength as a number to the sender.

6. “get_threshold”: Uplink returns the minimum signal strength for transmission as a number to the sender.
7. “get_speed”: Uplink returns the current max turning rate in degrees per 100ms as a number to the sender.
8. “set_ang”: Uplink returns the current dish angle as a vector to the sender. The target angle for the dish is then changed to the ‘angle’ value of the packet and the dish begins rotating to that new angle.
9. “set_id”: Uplink returns the current receiver ID as a number to the sender. The receiver ID is then changed to the ‘number’ value of the packet. By default, the receiver ID is equal to the entity id of the uplink.
10. “set_group”: Uplink returns the target signal group as a string to the sender. The target signal group is then changed to the ‘string’ value of the packet. By default, the target signal group is “vxcvr”.

The Uplink will *not* send a NACK signal if an invalid command is provided. If the uplink is not the target of the packet, then ‘command’, ‘angle’, ‘string’, and ‘number’ all fall under [Additional Data].

[Additional Data]: This encompasses any additional data values that are included in the packet. These are simply passed along to the transmitted packet if the receiver is not the target of the packet (see ipr).

Additional note: When the uplink signal is above minimum strength, and a data signal is received on the current target signal group, with the name “rec_” followed by the uplink’s receiver ID, the string data of the signal is sent to NetOut. The Uplink does not keep track of the transmitting dish; therefore, any return transmission packets must be handled by the end device.