Verity INSTRUMENTS, INC.

Remote Communication Message Specification

Revision R

April 9, 2019~~January 30, 2008~~

Table of Contents

Implementation of some Commands is Limited

1 Overview

The Verity remote communications specification is partitioned in terms of protocol and messages where the protocol is the format of the information being communicated and the messages are the content of the information being communicated. This document defines the standard message set for the SD1024 family of instruments. Verity supplies a common C header file, “VTYRemoteMessages.h”, with definitions for message command and data packets. It is recommended that the header file be incorporated into customer software to minimize errors in definitions.

Beginning with revision “J” of this document, all messages containing references to the Fixed Length “`tVTYString`” element have been modified to optionally describe support of the `tVTYDynString` element. The `tVTYDynString` allows the implementor of the protocol to select use of either Fixed Length or Dynamic length string content, up to a maximum of 128 characters. This is especially beneficial in slow baud rate Serial implementations where message length can have an effect on transmission time and processing reactions

Warning: It is NOT possible to alternate between fixed and dynamic length string messages. Once a protocol format selection is made ALL messages containing strings must be formatted with the same string definition.
Please reference the VTY_CONNECT message and Section 5.4 for information on activating the use of Dynamic strings

Important: When using strings that are dynamic length, structures in the VTYRemoteMessages.h file, the structures that contain a tVTYString object cannot be used for casting the message data pointer to a compatible structure pointer. Structures that contain strings should be seen as a description of the repeating sets of fields in the command or reply. Each command will detail if a command can or cannot use the structure directly

2 References

2.1 Documents

Ref #	Doc # & Revision (or Date)	Title
1.	Rev 13+	VTYRemoteMessages.h “C/C++” Source Header
2.	Rev 11+	VirtualPortSocket.h “C/C++” Source Header

3 Definitions

Tool Controller	For the purposes of this document, the tool controller is the control computer on the customer tool that communicates with the Verity endpoint instrument (Server)
Verity Controller	The Verity controller is a PC or laptop running a Verity application that can control the instrument. The control computer on the customer tool that communicates with the Verity endpoint instrument. (Server to Tool / Client to Instrument)
Verity Instrument	The Verity instrument is the instrument that collects and processes the real-time data to report events such as endpoint. (Server)
Client	System responsible for initiating communication – Designated as Logical Port 1
Server	System responsible for responding to a communication request – Designated as Logical Port 2
Logical Port	A single digit value used to identify the originator of a message (Client or Server) The referenced “C” header has the following constants defined for this value:

	`VTY_LP_TOURI` 1 (Client) `VTY_LP_FROMURI` 2 (Server)
String Formats	All strings used in the messages specified in this document have one of 2 formats – Either Fixed length: 128 BYTE array followed by two BYTEs where: the first appended BYTE is the string type (zero denotes ASCII) and the 2nd BYTE is the length in BYTEs of the string. Reference the `tVTYString` structure type definition in reference document #1 -OR- – Either Dynamic length: formatted as three BYTEs representing 1) an <ESC> character – to denote dynamic 2) the string type (zero denotes ASCII) and 3) the length in BYTEs of the string. This is then followed by 0 (zero) or UP TO 127 BYTEs representing the string. The last byte in the string is a NULL character and is NOT included in the length. The `tVTYDynString` structure type definition in reference document #1 can be used to hold strings and facilitate the reading and writing the string from the stream of bytes in the data contents of messages.
Byte	8-bit value – Defined in the 'C' Header as `#define BYTE unsigned char"`
WORD	16-bit value – Defined in the 'C' Header as `#define WORD unsigned short"`
DWORD	32-bit value. – Defined in the 'C' Header as `#define DWORD unsigned long"`
URI	Universal Remote Interface

4 Verity Communication Protocol Basics

4.1 Layering

This document describes only the communication messaging interface between the tool controller and a Verity Controller (or Verity Instrument) from the software layer only. The protocol described is intended to be at a level higher than that implemented by the hardware. As such, this protocol is independent of the hardware method used for message transmission/reception (reference Figure 1).

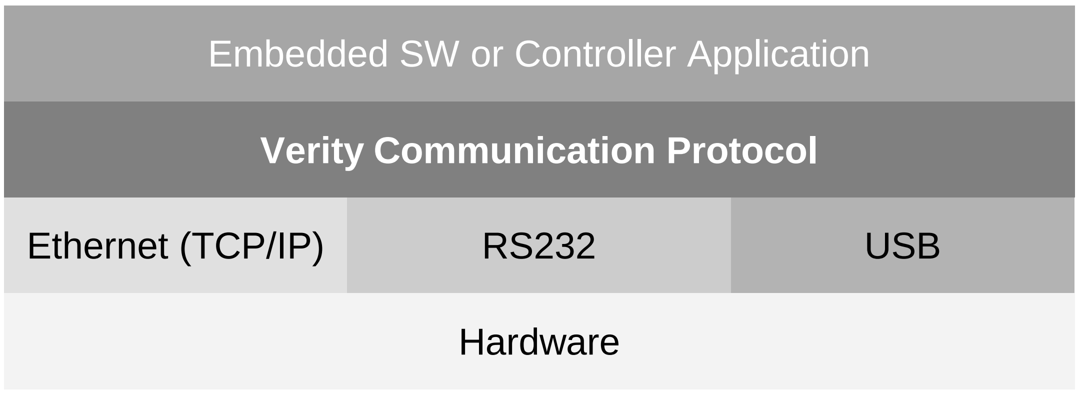

Embedded SW or Controller Application
Verity Communication Protocol
Ethernet (TCP/IP)
RS232
USB
Hardware

Figure 1 - Protocol Levels

Implementation of some Commands is Limited

4.2 Packet Format

All communication packets are built using the format detailed in Figure 1:

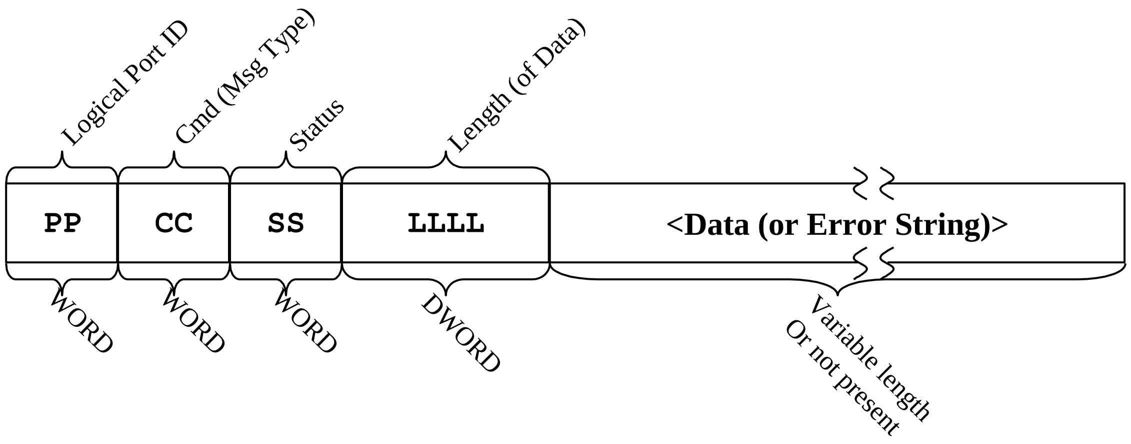


Figure 1 - Packet Structure

Where

Field:	**Contains**		**Has Length of:**
PP	The ***Logical Port ID*** as defined by the constants: `VTY_LP_TOURI` – Message Originated by the Client (Tool Controller) sent "TO" Verity (Universal Remote Interface [URI]) `VTY_LP_FROMURI`– Message Originated by the Server sent "FROM" Verity		16 bits (2bytes)
CC	The Message Type or ***Command ID*** – Each of these are detailed in Section 4.4		16 bits (2bytes)
SS	Message ***STATUS*** containing message specific information. For command message types it is used for holding the handshake reply. For event message types it is used for holding event specific information.		16 bits (2bytes)
LLLL	Data field ***LENGTH*** defining the number of BYTES in the data field		32 bits (4bytes)
<Data>	***Data*** values specific to the message type being transmitted		Variable*
	Specifically for Fixed Length String Data:		
	sA128	***128 Bytes of ASCII (Exclusive of <ESC>)***	Fixed 128 Bytes
	sT	***String Type*** Currently always 0 for ASCII	8 bits (1 Byte)
	sL	***String Length*** For this implementation of the protocol always 128 (hex 80)	8 bits (1 Byte)
	Specifically for Variable Length String Data:		
	sE	***ASCII <ESC> character*** Hex 0x1B or decimal 27	8 bits (1 Byte)
	sT	***String Type*** Currently always 0 for ASCII	8 bits (1 Byte)
	sL	***String Length*** For this implementation of the protocol any number from 1 to 127 inclusive	8 bits (1 Byte)
	sA	***0 to 127 Bytes of ASCII***	Variable sL
	s0	***Null String terminator*** (Not included in string length)	8 bits (1 Byte)
	For Data not containing strings please see messages or appendices:		

4.3 "Command" Message type Handshaking

In this communication scheme there is traditionally a single "Client" and, one or more, "Servers". In keeping with this concept the Tool Controller is considered the "Client" (being responsible for initiating communication) and the Verity Controller (or Verity Instrument) is considered the "Server" (responsible for responding to communication). Note- once the Client is defined it remains the Client for all current, and subsequent communication sessions.
Thus, Handshaking, for the purposes of this document, is considered to be the sequence of message transmission and response between a Client and any single Server (ex: the Tool Controller and Verity Controller). For each of the Command messages defined there is always a bi-directional handshake between the Client and Server whenever a message contains a "Command" (vs. an event). The Handshake sequence, in basic terms, is shown in Figure 2. For a formal example please reference appendix "A"
It is important to mention that a response to a client request will only be sent following completion of message processing by the server. In the case of message failures as much successful processing is performed as possible until the error / failure is detected.

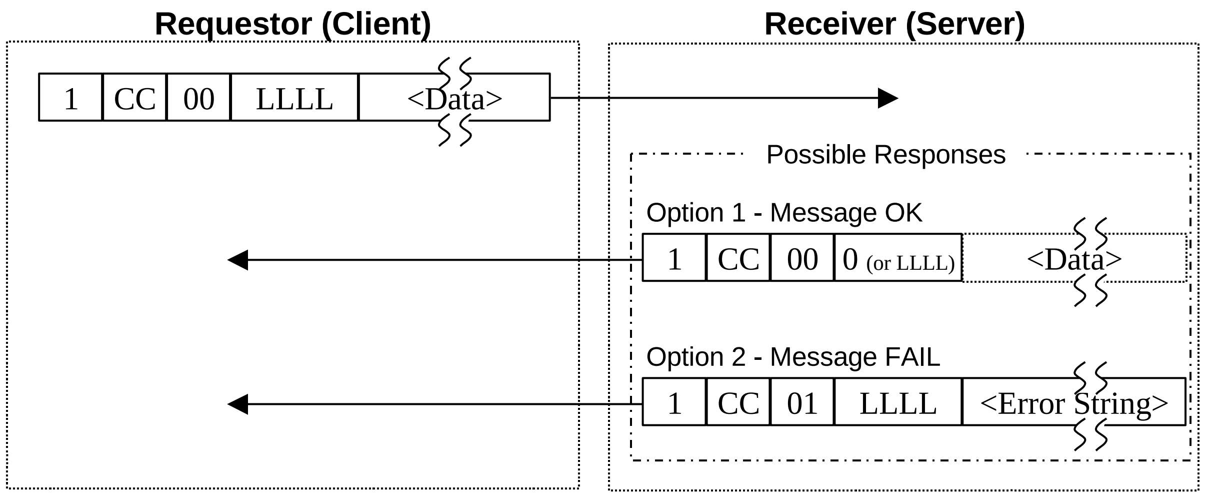


Figure 2 – "Command" Type Messages Handshaking

Note that the purpose of the Handshake is to confirm to the sender that the message was received. The sender can than determine the following based on the value of the Status Field:

Status	"C" Header Constant	Meaning
OK	`VTY_OK` (value 0)	Message was correctly Formatted and Processing was completed without error
FAIL	`VTY_FAIL` (value 1)	Message was incorrectly Formatted OR an error occurred during processing

4.4 Message Failure Responses

As stated previously, message processing continues until either a failure is detected, or a successful response is returned. In the diagram of Figure 2, the failure message returns an Error String. In the Verity communication scheme errors are reported as a descriptive text string of up to 128 characters.

5 Client originated "COMMAND" Messages

5.1 Message Definition lay-out

All COMMAND messages are defined using the following field format.

ID Number	Message Title	CMD ID as defined in Ref Doc #1
Message Description		
Product Implementation:	☐SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL<Data>
Where PP; CC; SS; and LLLL are as defined previously and <Data> is defined when needed	

"PP" field	"CC" field	"SS" field	"LLLL" field
VTY_LP_TOURI	<Cmd ID>	<Status>	<Length>
<Data>			

Reply Format - UNSUCCESSFUL	

"PP" field	"CC" field	"SS" field	"LLLL" field
VTY_LP_TOURI	<Cmd ID>	VTY_FAIL	<Length>
<Data>			

5.2 Generic Replies to Messages

All successful Message transmissions result in a reply to the sender (Client) with the following syntax unless otherwise specified in the command definition.

Reply Format - SUCCESSFUL	PPCCSSLLLL

"PP" field	"CC" field	"SS" field	"LLLL" field
VTY_LP_TOURI	<Cmd ID>	VTY_OK	0
<Optional> Data, as dependant on command type			

All UNsuccessful Message transmissions result in a reply to the sender (Client) with the following syntax unless otherwise specified in the command definition.

Reply Format - UNSUCCESSFUL	PPCCSSLLLL<Data>

"PP" field	"CC" field	"SS" field	"LLLL" field
VTY_LP_TOURI	<Cmd ID>	VTY_FAIL	4 to 131
A string structure defined by `tVTYString` –OR- tVTYDynString containing a description of the error per section 5.3			

NOTE if a reply is expected and is not returned within 6 seconds, it should be assumed that the device is not operational. VTY_FAIL may actually never be returned. On a failure, a timeout condition will occur typically due to the device or controller being unable to receive a message because of some transport issue such as the wrong baud rate, bad IP address, or unplugged cable.

5.3 Generic Definition of Data Fields for Messages Containing Strings

As stated previously, only one of the following string definitions (either All Fixed or All Variable) would be adopted for any given implementation of the protocol .

FIXED LENGTH STRING MESSAGES	PPCCSSLLLL<Data>			
"PP" field	**"CC" field**	**"SS" field**	**"LLLL" field**	
`<msg specific>`	`<msg specific>`	`<msg specific>`	`130`	
"<Data>" field				
sA128 (128Bytes of ASCII)			sT	sL

Use of Fixed length Strings will require the protocol logic will always resulting the data field length being equal to 130 bytes

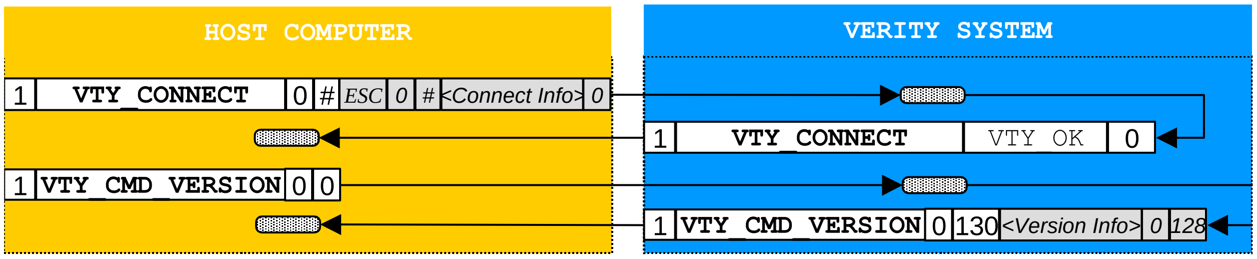
DYNAMIC (VARIABLE) LENGTH STRING MESSAGES	PPCCSSLLLL<Data>					
"PP" field			**"CC" field**	**"SS" field**	**"LLLL" field**	
`<msg specific>`			`<msg specific>`	`<msg specific>`	`4 to 131`	
"<Data>" field						
sE	sT	sL	sA (1 to 127 Bytes of ASCII)		s0 (zero)	

Use of Variable length Strings will require the protocol logic to calculate both the length of the complete data field as (String Length + 4) but the max length will always be 131
Remember the String Length(sL) DOES NOT include the NULL (0) character Byte, However the length of the Data Field does include this byte. Note that string length CAN be 0 thus, sL is 0 and there are no characters contained in the "sA" field

5.4 *Activation of Dynamic String Content for Messages Containing Strings*

The VTY_CONNECT message should always be the first message sent to the Verity system since this performs the necessary connection function for Ethernet (TCP/IP) sockets, and also provides a method of triggering use of either Fixed or Dynamic Strings. .
Use of the VTY_CMD_VERSION command reflects the success (or failure) of the String type selection in the response to the command.
The following diagrams reflect what would happen with both Prior Implementations of the protocol, and requests for Fixed or Dynamic Strings

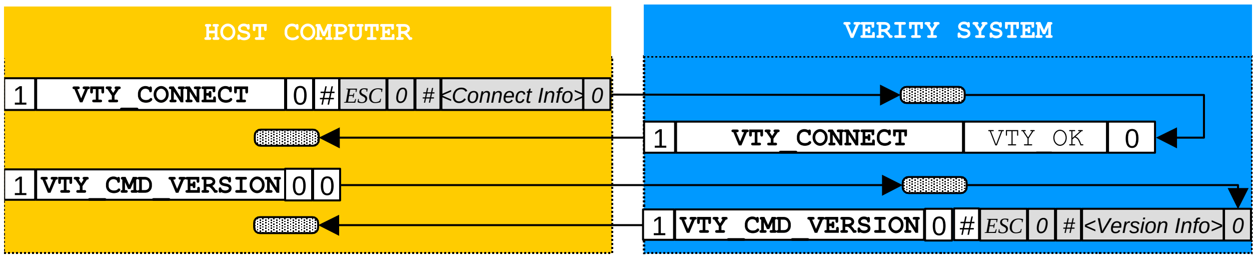
Example of Communication with Verity System
Prior to implementation of Dynamic Strings (Early Version of URI – RS232 only)**


HOST COMPUTER
VERITY SYSTEM
1 VTY_CONNECT 0 # ESC 0 # <Connect Info> 0
1 VTY_CONNECT VTY OK 0
1 VTY_CMD_VERSION 0 0
1 VTY_CMD_VERSION 0 130 <Version Info> 0 128

VTY_CMD_VERSION returns a Fixed String although Dynamic String used in VTY_CONNECT

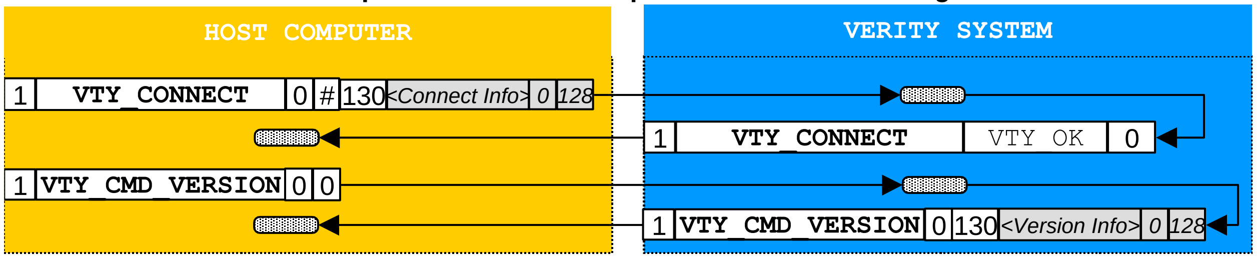
****Important:** Support for RS232 and use of the VTY_CONNECT message as a means to determine if Dynamic Strings are supported applies to URI versions prior to those supporting revision "J" of this document. However, use of Ethernet in the same manner, will only be supported in versions of URI updated with the content described subsequent to revision "J". This will begin with version 2.40 of URI.

Example of Successful Request to Use DYNAMIC Strings


HOST COMPUTER
VERITY SYSTEM
1 VTY_CONNECT 0 # ESC 0 # <Connect Info> 0
1 VTY_CONNECT VTY OK 0
1 VTY_CMD_VERSION 0 0
1 VTY_CMD_VERSION 0 # ESC 0 # <Version Info> 0

VTY_CMD_VERSION returns a Dynamic String

Example of Successful Request to Use FIXED Strings


HOST COMPUTER
VERITY SYSTEM
1 VTY_CONNECT 0 # 130 <Connect Info> 0 128
1 VTY_CONNECT VTY OK 0
1 VTY_CMD_VERSION 0 0
1 VTY_CMD_VERSION 0 130 <Version Info> 0 128

VTY_CMD_VERSION returns a Fixed String

Implementation of some Commands is Limited

5.5 Message Listing

-101	Connect	VTY_CONNECT

The command is essential to most communication protocols to identify the application or device desired and informs verity software what version of the command set is being used. If an older version, backward compatibility will be supported.
NOTE: This command has a negative value. This is very important for communications that use an ACK/NAK that are handled by the Client. RS232 is one example of where negative commands will not respond with an ACK or NAK, the tool should wait for the reply alone. Socket communication however performs these verifications internally so a command with a negative value will be handled exactly the same as any other command.

: (Spec Rev "L"addition) In the a-typical situation that could exist when a host loses the connection (re-boot…etc) and tries to re-establish a connection with the SpectraView system, the SpectraView system (URI) will recognize this command as equivalent to the VTY_RECONNECT command (described below) and discard any existing connection and create a new connection with the host ONLY if the URI "TOOLS" menu option indicates this should occur. The default setting of this indicator is "False" or "Off" thus causing the VTY_CONNECT command to reject (and return an error to) the VTY_CONNECT command if the requested connection already exists

The re-connect functionality described is provided to support existing installed-bases of the SpectraView system without needing associated Tool Host interface changes

Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CONNECT	Command set version Until a change is required 0 will be the version.	[String Data Length] – (Reference section 5.3)

DATA Field:
The name of the device or application the controller is seeking a connection should be sent in a string structure defined by either `tVTYString` or tVTYDynString (reference the definitions and section 5.3)
NOTE – As indicated in section 5.4 if this command is formatted with a FIXED string all subsequent commands will receive and reply with FIXED strings
If this command is sent with a Dynamic (or variable length) string, use the VTY_CMD_VERSION command next to determine if the request for use of Dynamic strings was successful

Reply Format - SUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CONNECT	VTY_OK	tVTY_SystemInfo

This structure gives details about URI and system capabilities. Advanced event reporting levels and related commands are only available if this structure is returned, and if EventReportingLevels is >=1.

Reply – UNSUCCESSFUL	A connection failure was detected

-102	Reconnect	VTY_RECONNECT

The command is the preferred method for re-establishing lost communication. If a connection needs to be re established, Use of this command will result in discard any existing connection and create a new connection with the host.

Implementation of some Commands is Limited

Prior to this re-assignment for the connection VTY_CMD_RESET and VTY_CONNECT will be sent to the connected SpectraView instance so that the spectrograph system is returned to a known state, and is in that state when the new connection is established. This command will first check for an existing connection. If an existing connection DOES NOT exist (can not send Reset command first), this command will fail and return an error to the sender. The sender MUST then use the VTY_CONNECT command to establish an initial connection **NOTE**: This command also has a negative value. Negative commands will not respond with an ACK or NAK, so the tool should wait for the reply alone. .
Product Implementation: ☑ SpectraView; ☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
`VTY_LP_TOURI`	`VTY_RECONNECT`	0.	[String Data Length] – (Reference section 5.3)
DATA Field: The name of the device or application the controller expecting to be re-connected to the host connection should be sent in a string structure defined by either `tVTYString` or tVTYDynString (reference the definitions and section 5.3) **Important:** It is NOT possible to use this command for defining /determining Fixed or Dynamic (or variable length) strings, use the VTY_CONNECT command only for this. This command will accept a Fixed –or- Dynamic string but return a string of the type defined by the original VTY_CONNECT type			

Reply Format - SUCCESSFUL		PPCCSSLLLL<Data>	
`VTY_LP_TOURI`	`VTY_CONNECT`	`VTY_OK`	`tVTY_SystemInfo`
This structure gives details about URI and system capabilities. Advanced event reporting levels and related commands are only available if this structure is returned, and if EventReportingLevels is >=1.			

Reply – UNSUCCESSFUL	A prior connection did not exist

99	Disconnect	**VTY_DISCONNECT**
The disconnect command is important to politely inform Verity communications components of the intent to disconnect eminently. Resources are freed, connection states are maintained, and some communication methods close, or prepare to close, their connection to the controller.		
Product Implementation: ☑ SpectraView; ☐Spectrograph;		

Transmission Format –		PPCCSSLLLL	
`VTY_LP_TOURI`	`VTY_DISCONNECT`	0	0
N/A			

Reply - UNSUCCESSFUL	Disconnection failure no disconnect occurred

100	Reset	**VTY_CMD_RESET**
This command informs the Verity controller(instrument) that it should perform a reset. This will put the Verity controller(instrument) in a state as if it were just powered up. Previously communicated settings are erased, SpectraView data collection / monitoring is stopped, and any current error is reset including the operator notification		

Implementation of some Commands is Limited

window being cleared.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Status Field - One of the following sub-command codes…
VTY_RSC_NORMAL: This resets only the endpoint system ***VTY_RSC_RESETALL***: This resets the endpoint system AND the device

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_RESET	0	0
N/A			

Reply –UNSUCCESSFUL	Reset failure message description

101	Test Communications	VTY_CMD_TEST

This is much like a "ping" to test the communications chain needed to test the Verity system's availability. This is a useful command if the tool wishes to keep in contact with the Verity system. A reply with a status of VTY_OK means the required elements for automated processing are in place and there is not a current system error/fault being displayed in the Operator Notification Window. A reply with the status of VTY_FAIL means that communication with at least one component is currently available but either a secondary component is missing, the system is in an error condition, or an error/fault has not been acknowledged in the Operator Notification Window. The error string returned the failure response contains details about the system issue or the string from the Operator Notification Window.

If this command does not reply within TBD seconds (Time Specified by Interrogating Host, but should be a minimum of 3 sec), it should be assumed that the device is not operational. On a failure, a timeout condition will occur because the device or controller was unable to receive the message because of some transport issue such as the wrong baud rate, bad IP address, or unplugged cable.

NOTE1: Use VTY_CMD_PRESENT to test the system availability AND the readiness to accept a start command. (See VTY_CMD_PRESENT)
NOTE2: To Limit the size of the Communication Log File This command WILL NOT BE LOGGED

Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_TEST	0	0
N/A			

Reply - UNSUCCESSFUL	Message format error

102	Instrument Available	VTY_CMD_PRESENT

This command queries the Verity system and instrument to determine that they are not only available, but that no error/fault is being displayed in the Operator Notification Window and the system is ready to begin processing.

If this command does not reply within TBD seconds(Time Specified by Interrogating Host, but should be a minimum of 1 sec), it should be assumed that the device is not operational. On a failure, a timeout condition will occur because the device or controller was unable to receive the message because of some transport issue such as the

wrong baud rate, bad IP address, or unplugged cable. NOTE1: This **should not** be used for a “ping” type message because when the system is successfully running an error is returned that states the system is already processing. NOTE2: Reception of this command also causes **clearing of all Wafer Information Fields** NOTE3: To Limit the size of the Communication Log File This command WILL NOT BE LOGGED
Product Implementation: ☑ SpectraView; ☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_PRESENT	0	0
N/A			

Reply – UNSUCCESSFUL	Error/fault description

103	Get Embedded SW Versions	VTY_CMD_VERSION
This command queries the Verity instrument for the current versions of embedded software		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_VERSION	0	0
N/A			

Reply Format - SUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_VERSION	VTY_OK	#versions x [String Data Length] – (Reference section 5.3)
One or more string structures defined by `tVTYString-OR- tVTYDynString` each containing the Version #(s) for the associated embedded processors and SpectraView			

Reply - UNSUCCESSFUL	Version numbers could not be returned

104	Get List of Configurations	VTY_CMD_CFG_LIST
This command requests a list of configuration names stored the Verity controller or Instrument. The reply to this message is a list of stored names along with each configuration’s corresponding time / date stamp and its size.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_CFG_LIST	0	0
N/A			

Reply Format – SUCCESSFUL The reply is an array of configuration elements.		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_CFG_LIST	VTY_OK	Size of tVTYConfigurationDesc x #of cfg list entries

Implementation of some Commands is Limited

A series of configuration ["File"]Names, each detailed by a stream that contains the members of the tVTYConfigurationDesc structure. The structure should only be used to note the order and placement of each member variable in the stream. The elements of the structure noted as tVTYString objects will be either tVTYString or tVTYDynString based on the string usage implemented for the protocol. (reference the definitions section) Each configuration entry contains a configuration name and time/date stamp dynamic strings followed by a DWORD that returns the BYTE size of the configuration file.

Reply - UNSUCCESSFUL	Configuration list could not be returned

105	Get Embedded Configuration	VTY_CMD_GET_CFG
This command returns the details of the specified configuration stored in flash memory in the instrument		
Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_GET_CFG	0	String Data Length – (Reference section 5.3)
The name of the device or application the controller is seeking a connection should be sent in a string structure defined by either tVTYString or tVTYDynString (reference the definitions and section 5.3) NOTE a VTY_CMD_CFG_LIST request should have been performed first to determine the name of configuration needing to be returned			

Reply Format – SUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_GET_CFG	VTY_OK	Size of tVTYConfigurationDesc + Cfg Data Streem
A single configuration descriptor stream similar including the configuration name, date and time stamp, size of the configuration representing the elements of a tVTYConfigurationDesc structure followed by a stream of configuration parameters Details related string content within the structure apply as defined in VTY_CMD_CFG_LIST			

Reply - UNSUCCESSFUL	Configuration could not be returned

106	Send Configuration	VTY_CMD_SET_CFG
This command stores a runtime configuration in flash memory of the instrument.		
Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_SET_CFG	0	Size of tVTYConfigurationDesc + Cfg Data Streem
A single configuration descriptor stream similar including the configuration name, date and time stamp, size of the configuration representing the elements of a tVTYConfigurationDesc structure followed by a stream of configuration parameters			

Details related string content within the structure apply as defined in VTY_CMD_CFG_LIST

Reply - UNSUCCESSFUL	Configuration could not be stored

107	Delete Single Configuration	VTY_CMD_DEL_CFG
This command instructs the instrument to delete the specified run configuration.		
Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_DEL_CFG	0	Size of tVTYConfigurationDesc
A single configuration descriptor stream similar including the configuration name, date and time stamp, size of the configuration representing the elements of a tVTYConfigurationDesc Details related string content within the structure apply as defined in VTY_CMD_CFG_LIST NOTE a `VTY_CMD_CFG_LIST` request should have been performed first to determine the name of configuration needing to be deleted			

Reply - UNSUCCESSFUL	Configuration could not be deleted

108	Delete All Embedded Configurations	VTY_CMD_DEL_ALLCFG
This command instructs the instrument to delete all run configurations. For safety, this command is currently ignored by all controllers that are not “Smart Detectors” using flash memory.		
Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_DEL_ALLCFG	0	0
N/A			

Reply - UNSUCCESSFUL	Configuration could not be deleted

109	Configure URI	VTY_CMD_SET_URI
This command transmits a Universal Remote Interface configuration for storage in the instrument		
Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_SET_URI	0	Size of
Binary URI configuration			

Reply - UNSUCCESSFUL	URI configuration could not be stored

110	Get URI Configuration	VTY_CMD_GET_URI
This command returns a Universal Remote Interface configuration from the instrument		

Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_GET_URI	0	0
Binary URI configuration			

Reply Format – SUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_GET_URI	VTY_OK	Size of
Binary URI configuration			

Reply - UNSUCCESSFUL	URI configuration could not be returned

111	Tool Is Host	VTY_CMD_TOOLISHOST
This command tells the instrument that the tool controller is the primary interface. Only the tool controller or the Verity controller (if connected) can be the host. They cannot both be the host at the same time.		
Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data – Optional>	
VTY_LP_TOURI	VTY_CMD_TOOLISHOST	<See Below>	<See Below>

Status Field Bit Assignments

BIT Position	Constant	Description
0x0000	VTY_ITYPE_NONE	indicates that no data is to be passed to the tool controller
0x0001	VTY_ITYPE_RAWSPECTRUM	indicates that raw spectra selected in the configuration information is to be passed to the tool controller
0x0002	VTY_ITYPE_SPECTRALEQU	Spectral Equations flagged to be sent during execution are desired by the controller
0x0004	VTY_ITYPE_REGIONEQU	Region Equations flagged to be sent during execution are desired by the controller.
0x0008	VTY_ITYPE_TRENDEQU	Trend Equations flagged to be sent during execution are desired by the controller.
0x0010	VTY_ITYPE_ADV_TREND	Advanced Equation trend outputs, flagged to be sent during execution, are desired by the controller
0x0020	VTY_ITYPE_ADV_SPECTRA	Advanced Equation spectral outputs, flagged to be sent during execution, are desired by the controller
0x003f	VTY_ITYPE_ALL	indicates that spectral and trend data selected in the configuration information is to be passed to the tool controller.

Data Field
This data block is optional and currently ignored in all systems. However functionality detailed here will eventually be supported and even enhanced. Defaults will always be used if this information is not supplied. The specification given here is preliminary and will most likely be altered before it is supported
For a Status field with the VTY_ITYPE_RAWSPECTRUM bit set the structure of the data field is as follows:

WORD 1	Data type desired	This may or may not be honored based on many different factors. If 0, then default will be used at least for this type of information.

WORD 2	Data speed	This entry will specify how timely the data must be. Sending the data in near real-time may slow the system and inversely, sending larger data events less often can reduce communication overhead
WORD 3	Various bit flags	Bit one will allow the system to send only the current spectra instead of all spectra collected and/or calculated since the last data event message was sent. This can free the tool from sending data that may only be for informational or visual inspection. If the data is required to be reprocessed or to be used for detailed analysis then this flag should not be used. NOTE: All trend information types are unaffected.

Reply - UNSUCCESSFUL	the requested data could not be supplied or the tool controller was not accepted as host

112	Verity Controller Is Host	VTY_CMD_TOOLNOTHOST
This command tells the instrument that the Verity controller is the primary interface. This is the power up condition. Only the tool controller or the Verity controller (if connected) can be the host. They cannot both be the host at the same time.		
Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL		
VTY_LP_TOURI	VTY_CMD_TOOLNOTHOST	0	0

Reply - UNSUCCESSFUL	Verity controller was not accepted as host

113	Store Wafer Information	VTY_CMD_WAFERINFO
This command sends wafer information to the Verity instrument(controller) for the next wafer to be run, writing over the existing wafer information if it exists.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_WAFERINFO	<See Below>	<See Below>

Status Field - The number of wafer array elements **OR** one of the following sub-command codes…

VTY_WISC_NEWWAFER: Current information is cleared, replaced with the new information, and VTY_CMD_WAFERCOMPLETE is performed as well. The result is the next step will be in a new directory even if path and filename substitution are the same as previous.
NOTE: This is the default when status is used as a number of structures

VTY_WISC_UPDATE: Information that already exists is updated, and any new information is appended to the end of the existing information. Only fields that are to be added or updated need to be sent. The information will be considered as part of the same wafer and will be saved with processing of the next step in the recipe.

VTY_WISC_APPEND: All current information is kept unchanged. Passed information is appended to existing wafer information without looking for duplicates.

NOTE: If duplicates exist or field type codes are used more than once, all text substitution for path and filenames will use the first entry that contains the correct field type bit.

Data Field
The data consists of an array of wafer information descriptors. Each description mimics the tVTYWaferDesc structure by having elements for a string containing the field label, a string containing the field information, and a DWORD containing the field type. Strings may be either Fixed Length or Dynamic. If Dynamic, the structure cannot be used directly. (reference the definitions and section 5.3)
The field types are as follows:

BIT Position	Constant	Description
0x00000001	VTY_WAFER_TOOLID	Indicates the id of the current tool,
0x00000002	VTY_WAFER_WORKFLOW	Indicates the name of the workflow
0x00000004	VTY_WAFER_RECIPE	Indicates the name of the recipe
0x00000008	VTY_WAFER_WAFERID	Indicates the name of the current wafer
0x00000010	VTY_WAFER_LOT_NAME	Indicates the current wafer lot name
0x00000020	VTY_WAFER_CASSETTE	Indicates the name of the current cassette
0x00000040	VTY_WAFER_SLOT	Indicates the name of the current slot
0x00000080	VTY_WAFER_OTHER_INFO	This or a 0 can be used for any non-specific information that is to be saved in the data file for informational purposes.
0x00000100	VTY_WAFER_STEP_NUM	Indicates the step number of the recipe
0x00000200	VTY_WAFER_CUSTOM1	Customizable information that can be used in path and file names or any place else that wafer information can be used to customize the endpoint system.

0x00000400	VTY_WAFER_CUSTOM2	Used in same way as VTY_WAFER_CUSTOM1
0x00000800	VTY_WAFER_CUSTOM3	Used in same way as VTY_WAFER_CUSTOM1
0x00001000	VTY_WAFER_CUSTOM4	Used in same way as VTY_WAFER_CUSTOM1
0x00002000	VTY_WAFER_CUSTOM5	Used in same way as VTY_WAFER_CUSTOM1
0x00004000	VTY_WAFER_DATE	Date to be associated with data file
0x00008000	VTY_WAFER_TIME	Time to be associated with data file

VTY_WAFER_SYNCBIT can be combined with VTY_WAFER_DATE or VTY_WAFER_TIME to synchronize the endpoint computer's date and/or time to the passed date and time. The date must be formatted as either YYYY/MM/DD or MM/DD/YYYY. The acceptable time formats are HH:MM:SS, H:MM:SS, H:MM:SSam, and HH:MMam. If used for file or directories. The ':' and '/' symbols are replaced with '-' automatically.

Reply – UNSUCCESSFUL	Command was not accepted

114	Start Step	VTY_CMD_START
This command informs the Verity instrument(controller) that the tool controller is starting a step. Note that this command only needs to be sent when the tool controller step being started requires associated Verity data collection and processing.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_START	0	String Data Length – (Reference section 5.3)
The name of the configuration the controller is requesting be used for data collection should be sent in a string structure defined by either `tVTYString` or tVTYDynString (reference the definitions and section 5.3) NOTE a `VTY_CMD_CFG_LIST` request could have been performed first to determine the valid names of configurations available to be used for processing			

Reply - UNSUCCESSFUL	indicates that an error occurred and the step was not started.

115	<Reserved>	<Cmd 115>
N/A		

116	Stop Step	VTY_CMD_STOP
This command informs the Verity instrument(controller) that the currently running or paused step should be stopped. This will cause data collection and processing to be stopped.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_STOP	0	0
N/A			

Reply – UNSUCCESSFUL	indicates that the instrument was unable to stop the step NOTE if a failure is received, the tool

	controller should send a RESET command prior to any subsequent commands

117	Pause Step	**VTY_CMD_PAUSE**
This message commands the Verity instrument to pause the currently running step. This will cause data collection and processing to be paused		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL		
VTY_LP_TOURI	VTY_CMD_PAUSE	0	0
N/A			

Reply – UNSUCCESSFUL	indicates that the instrument was unable to pause the step

118	Continue Step	**VTY_CMD_CONTINUE**
This message commands the Verity instrument to continue the paused step. This will cause data collection and processing to resume		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL		
VTY_LP_TOURI	VTY_CMD_CONTINUE	0	0
N/A			

Reply - UNSUCCESSFUL	indicates that the instrument was unable to continue the step.

119	Step Complete- Wafer End	**VTY_CMD_COMPLETE**
This command informs the Verity instrument that the currently running wafer is complete. Use of this command ensures that a new data file directory is created subsequent to the next STOP command		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL		
VTY_LP_TOURI	VTY_CMD_START	0	0
N/A			

Reply - UNSUCCESSFUL	indicates that the instrument did not accept end of wafer

120	<Reserved>	**<Cmd 120>**
N/A		

121	<Reserved>	**<Cmd 121>**
N/A		

Implementation of some Commands is Limited

122	Verify Reliability of a List of Configurations	VTY_CMD_CFG_VERIFY
This command requests a verification of the existence and integrity of a list of configuration files about to be used		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_VERIFY	Number of configurations	String Data Length – (Reference section 5.3) * number to check
One or more multiple Fixed / Dynamic length strings. Each containing configuration name to be verified (reference the definitions and section 5.3)			

Reply Format - UNSUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_VERIFY	VTY_FAIL	#issues x [tVTYValidateReply structure] – (Reference section 5.3)
One or more tVTYValidateReply structures. Each containing an issue found with a configuration and the type of issue. The issue types are either a validation problem or the configuration not being found.			

123	Validate a Configuration	VTY_CMD_CFG_VALIDATE
This command validates a configuration to verify it is valid for use		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_VALIDATE	0	String Data Length – (Reference section 5.3)
The name of the configuration to validate in a tVTYString or tVTYDynString			

Reply Format – UNSUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_VALIDATE	VTY_FAIL	#messages x [tVTYValidateReply structure] – (Reference section 5.3)
One or more tVTYValidateReply structures are returned if there is an issue with the configuration executing in the sytem. Each structure contains a line of text from validation and the issue type (Text, Warning, or Error). The validation only is considered a failure if one or more errors is found.			

124	Remote Shutdown of PC	VTY_CMD_SHUTDOWN
This command informs the Verity instrument(controller) to perform a software shut-down of all applications and of windows. Before commanding windows to shutdown, the communication interface software will first request the device to stop data collection, and close any open data file. A time-out of 2 seconds will be allowed for this action. Also, this command only causes the commanded instance to close before shutdown. If other instances are active they should be commanded to Stop (VTY_CMD_STOP) prior to this command being sent NOTE: Although sending of this command is directed to a specific instance of SpectraView, all instances, and any other applications will be closed as a result of the low-level windows command forcing a shutdown. IMPORTANT: Verity recommends all customer protocols, and maintenance activities adopt and include communication of this shutdown command prior to powering-off the Verity instrument(controller)		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL		
VTY_LP_TOURI	VTY_CMD_SHUTDOWN	0	0
N/A			

Reply – NONE	Since the Communication Interface will be terminated there can be no reply of success

125	Set Variable Values	VTY_CMD_SET_VAR
This command sends new values for one or more variables in the system. Although the command can be sent at any time, the command should be sent while the system is idle in most cases. Changing the value of variables during a step can cause reprocessing issues and the timing would be almost guaranteed to be inconsistent from step to step.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL<Data>		
VTY_LP_TOURI	VTY_CMD_SET_VAR	0	<See Below>

Data Field The data consists of an array of variable description elements. Each element consists of a string containing the variable name and a float containing the new value for the variable. Strings are defined by string structure defined by `tVTYString` (reference the definitions section). The structure in the header file is called tVTYVariable.
Data Field The data consists of an array of variable description streams. Each stream holds the data for the elements of a tVTYVariable structure. If the protocol is implemented using dynamic string content the structure cannot be used to directly reference the stream of bytes. (reference the definitions and section 5.3) The elements consist of a string (Fixed or variable length) containing the variable name and a float containing the new value for the variable.

Reply – UNSUCCESSFUL	Command was not accepted

Implementation of some Commands is Limited

126	Get Variable Values	VTY_CMD_GET_VAR
This command requests the current value for one or more variables in the system. The command can be sent at any time.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL<Data>		
VTY_LP_TOURI	VTY_CMD_GET_VAR	0	<See Below>

Data Field The retrieve information about specific variables, pass one or more Fixed / Dynamic length strings (reference the definitions and section 5.3) containing the names of the variables. If variable name strings are not sent, then all variables currently being tracked in the system are returned.

Reply Format – SUCCESSFUL	PPCCSSLLLL<Data>		
VTY_LP_TOURI	VTY_CMD_GET_VAR	VTY_OK	Size of 0 or more tVTYVariable structures (adjusted for String lengths)

All variables requested, that were found to be actively in use by the system, are returned with their current value. Variables requested that were not being used will not have a return record. Each variable returned will be represented by a stream for elements of a tVTYVariable structure using either Fixed / Dynamic strings. (reference the definitions and section5.3)

Reply – UNSUCCESSFUL	Command was not accepted

Implementation of some Commands is Limited

127	Open a Data File	VTY_CMD_OPEN_DATAFILE
This command requests the system to open a data file in preparation of reprocessing or details request (VTY_CMD_GET_PROCESSDETAILS). The command will return an error if the file is not found or the system is not idle.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_OPEN_DATAFILE	0	0

Data Field The file is found based on wafer information. The wafer information that was used to create the file will be needed to locate the correct file. The data consists of an array of wafer information descriptors. Each description mimics the tVTYWaferDesc structure by having elements for a string containing the field label, a string containing the field information, and a DWORD containing the field type. Strings may be either Fixed Length or Dynamic. If Dynamic, the structure cannot be used directly. (reference the definitions and section 5.3)

Reply – UNSUCCESSFUL	indicates that the instrument was unable to open the data file

128	Get Processing Details	VTY_CMD_GETPROCESSDETAILS
Get details about currently processing data or open data file. An error is returned if there isn't a data file for details to be returned.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_GETPROCESSDETAILS	0	0

Reply Format – SUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_GETPROCESSDETAILS	VTY_OK	tVTYProcessInfo structure
The `tVTYProcessInfo` structure details the name of the data file, data interval, number of raw samples, system state, and the response time adjustment.			

129	Reprocess the open Data File	VTY_CMD_START_REPROCESS
This command requests the system to reprocess the open data file. The open file can be the one from the last step, or open opened using VTY_CMD_OPEN_DATAFILE. The command will return an error if there was not a data file open, the configuration was not found, or the system is not idle.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL<Data>		
`VTY_LP_TOURI`	VTY_CMD_START_REPROCESS	`0`	String Data Length – (Reference section 5.3)
The name of the configuration the controller is requesting be used for reprocessing should be sent in a string structure defined by either `tVTYString` or tVTYDynString (reference the definitions and section 5.3) NOTE a `VTY_CMD_CFG_LIST` request could have been performed first to determine the valid names of configurations available to be used for processing			

Reply - UNSUCCESSFUL	indicates that an error occurred and the reprocessing was not started.

130	<Reserved>	<Cmd 130>
N/A		

131	Set Event/Error Reporting Level	VTY_CMD_SETEVENTREPORTING
This command can be used to change the event reporting level of the Verity instrument(controller).		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL<Data>		
`VTY_LP_TOURI`	VTY_CMD_SETEVENTREPORTING	`<See Below>`	0

Status Field - The error and event level being requested…

VTY_EVENTLEVEL_NORMAL (Default): User events and system errors are reported when they occur only during execution of the system

VTY_EVENTLEVEL_LEVEL_ADV1: Events and errors are reported when they occur, are ACKed, and resolved during execution and when idle

NOTE: The reporting level always starts at the default whenever the system is loaded.

Reply – UNSUCCESSFUL	Command was not accepted

Implementation of some Commands is Limited

132	Acknowledge/Clear User Events	**VTY_CMD_ACK_USEREVENT**
This command requests one or more user events to be cleared from the error/event tracking		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_ACK_USEREVENT	<See Below>	Zero or more tVTYEventData structures
The status field can be used to clear all events, or a group of a certain severity level. The status code of the command should be set to a bit mask (detailed below) or set to zero if specific events are to be acknowledged. NOTE: VTY_CMD_RESET can be used to clear all user events and system issues (that are not status issues). If the status of VTY_ACKCMDBIT_STRINGS(0) (VTY_ACKCMDBIT_SENDACKS is allowed) is sent, `tVTYEventData` structures should be passed in the data. Each containing the details of the event sent originally in a VTY_EVENT_USEREVENT message. The “Text” string and the “DateTime” strings are used to find the matching event to be acknowledged. If the DateTime string is empty, then all user events with the same text string will be acknowledged.			

Status Field - Bit mask of what severity levels should be acknowledged
VTY_ACKCMDBIT_STRINGS: Specific event structures are being sent. (Can only be combined with VTY_ACKCMDBIT_SENDACKS) **VTY_ACKCMDBIT_NOTIFY**: All notification level user events should be cleared **VTY_ACKCMDBIT_WARNING**: All warning level user events should be cleared **VTY_ACKCMDBIT_ERROR**: All error level user events should be cleared **VTY_ACKCMDBIT_FAULT**: All fault level user events should be cleared **VTY_ACKCMDBIT_ALLLEVELS**: All user events, of all levels should be cleared **VTY_ACKCMDBIT_SENDACKS**: Send VTY_EVENT_USEREVENT_ACK events for every event cleared. NOTE: This could cause latency for slow protocols like RS232.

Reply Format – UNSUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_ACK_USEREVENT	VTY_FAIL	indicates that at least one event was not found or could not be cleared

Implementation of some Commands is Limited

133	Acknowledge/Clear System Issues	VTY_CMD_ACK_ERROR
This command requests one or more system issues/errors to be cleared from the error/event tracking. "Status" type errors cannot be cleared with this request since they will only be resolved automatically. The best example is when the spectrograph is turned off. The VTY_EVENT_ERROR will detail the loss of connection to the device and flag the error with the VTY_ERRORSBIT_STATUS flag set to differentiate that error from an error that can be acknowledged and cleared by the user or tool.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_ACK_ERROR	<See Below>	Zero or more tVTYEventData structures
The status field can be used to clear all error, or a group of a certain severity level. The status code of the command should be set to a bit mask (detailed below) or set to zero if specific errors are to be acknowledged. NOTE: VTY_CMD_RESET can be used to clear all user events and system issues (that are not status issues). If the status of VTY_ACKCMDBIT_STRINGS(0) (VTY_ACKCMDBIT_SENDACKS is allowed) is sent, `tVTYEventData` structures should be passed in the data. Each containing the details of the error sent originally in a VTY_EVENT_ERROR message. The "Text" string and the "DateTime" strings are used to find the matching error to be acknowledged. If the DateTime string is empty, then all error events with the same text string will be acknowledged.			

Status Field - Bit mask of what severity levels should be acknowledged

VTY_ACKCMDBIT_STRINGS (0): Specific event structures are being sent. (Can only be combined with VTY_ACKCMDBIT_SENDACKS)

VTY_ACKCMDBIT_NOTIFY: All notification level user events should be cleared

VTY_ACKCMDBIT_WARNING: All warning level user events should be cleared

VTY_ACKCMDBIT_ERROR: All error level user events should be cleared

VTY_ACKCMDBIT_FAULT: All fault level user events should be cleared

VTY_ACKCMDBIT_ALLLEVELS: All user events, of all levels should be cleared

VTY_ACKCMDBIT_SENDACKS: Send VTY_EVENT_ERROR_ACK events for every event cleared. NOTE: This could cause latency for slow protocols like RS232.

Reply Format – UNSUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_ACK_ERROR	VTY_FAIL	indicates that at least one issue was not found or could not be cleared

134	Get Current System Issues	VTY_CMD_GETSTATUSERRORS
This command requests a list of the current system issues/errors being tracked. All system issues/errors, that match the mask in the status field, will be returned in a series of tVTYEventData structures.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_ACK_ERROR	<See Below>	0

Status Field - Bit mask of what severity levels should be returned
VTY_GETSTATUSBIT_NOTIFY: Return all notification (VTY_EVENTCODE_NOTIFY) level issues
VTY_GETSTATUSBIT_WARNING: Return all warning (VTY_EVENTCODE_WARNING) level issues
VTY_GETSTATUSBIT_ERROR: Return all error (VTY_EVENTCODE_ERROR) level issues
VTY_GETSTATUSBIT_FAULT: Return all fault (VTY_EVENTCODE_FAULT) level issues
VTY_ACKCMDBIT_ALLLEVELS (or 0): Return all issues…of all levels

Reply Format - SUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_ACK_ERROR	VTY_OK	Zero or more tVTYEventData structures
Zero or more structures defined by `tVTYEventData,` each describing the issue that occurred. If no events returned, then there are none being tracked in the system.			

Implementation of some Commands is Limited

135	Get Current User Events	VTY_CMD_GETSTATUSUSEREVENTS
This command requests a list of the user events being tracked. All user events, that match the mask in the status field, will be returned in a series of tVTYEventData structures.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_TOURI	VTY_CMD_ACK_ERROR	<See Below>	0

Status Field - Bit mask of what severity levels should be returned
VTY_GETSTATUSBIT_NOTIFY: Return all notification (VTY_EVENTCODE_NOTIFY) level events **VTY_GETSTATUSBIT_WARNING**: Return all warning (VTY_EVENTCODE_WARNING) level events **VTY_GETSTATUSBIT_ERROR**: Return all error (VTY_EVENTCODE_ERROR) level events **VTY_GETSTATUSBIT_FAULT**: Return all fault (VTY_EVENTCODE_FAULT) level events **VTY_ACKCMDBIT_ALLLEVELS (or 0)**: Return all issues…of all events

Reply Format - SUCCESSFUL		PPCCSSLLLL<Data>	
VTY_LP_TOURI	VTY_CMD_GETSTATUSUSEREVENTS	VTY_OK	Zero or more tVTYEventData structures
Zero or more structures defined by `tVTYEventData`, each describing the user event that occurred. If no events returned, then there are none being tracked in the system.			

6 Server originated "EVENT" Messages

The following section defines the system event messages. The event messages are used to communicate information that is transmitted not in reply to a control computer command. These messages are used to communicate status and data information from the instrument to the control computer.

6.1 Generic Replies to Messages

All Event messages are sent without a reply.

NO Reply – SUCCESSFUL or UNSUCCESSFUL	

The Server assumes that if the Hardware delivered the message the message was accepted and processed correctly

6.2 Message Listing

200	Endpoint Detected	VTY_EVENT_ENDPOINT
This event informs the Client that endpoint has been detected.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_FROMURI	VTY_EVENT_ENDPOINT	0	<See Below>
A tVTYEventData structure with adjustment as neede for Fixed / Dynamic Length Strings that contains the user event text as entered in the configuration's sequence item, the severity code, and relative time since processing began			

201	Instrument operating in Local mode	VTY_EVENT_LOCAL
This event informs the host that the Verity instrument (controller) has been switched to the local operating mode by the operator. NOTE – Default mode upon initial startup is Local		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_FROMURI	VTY_EVENT_LOCAL	0	0
N/A			

202	Instrument operating in Remote mode	VTY_EVENT_REMOTE
This event informs the host that the Verity instrument (controller) has been switched to the remote operating mode by the operator. NOTE – Mode on subsequent startup will be Remote if previously in remote		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_FROMURI	VTY_EVENT_REMOTE	0	0
N/A			

203	Instrument is Powered	VTY_EVENT_RUNNING
This event informs the host that the Verity instrument (controller) has received a start command and is collecting data. Running always follows VTY_EVENT_NOTREADY mainly for support of legacy command sets, VTY_EVENT_READY is analogous to NOTRUNNING.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_FROMURI	VTY_EVENT_RUNNING	0	0
N/A			

204	Instrument Stopped/Ready	VTY_EVENT_READY
This event informs the host that the Verity instrument (controller) has been stopped, has completed processing and is ready to start the next data collection step.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_FROMURI	VTY_EVENT_READY	0	0
N/A			

205	Instrument Running	VTY_EVENT_NOTREADY
This event informs the host that the Verity instrument (controller) has received the start command, is collecting data and will be unavailable until a stop command is received.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL	
VTY_LP_FROMURI	VTY_EVENT_NOTREADY	0	0
N/A			

206	User Event Encountered	VTY_EVENT_USEREVENT
This event informs the host that the Verity instrument (controller) has recognized a "user defined" system event during the running of a process.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_FROMURI	VTY_EVENT_USEREVENT	0	<See Below>
A stream that represents the elements of a tVTYEventData structure but with adjustment for Fixed / dynamic length strings. (reference the definitions and section 5.3) The event holds the user event text as entered in the configuration's sequence item, the severity code, the relative event time, and a date/time stamp.			

207	<Reserved>	**<Event 207>**
Reserved for Future Use		

208	Data Identification Matrix	**VTY_EVENT_MATRIX**
This event outputs a list of strings and string identifiers. Each string / string identifier structure corresponds to data that will be output from the instrument. Each output, as selected in the configuration, sends an identifier to indicate the equation or other configuration element the data was generated by. This matrix is most importantly a map between the identifiers in the VTY_EVENT_DATABLOCK and the names of each item. NOTE: This event will only be sent once prior to the first VTY_EVENT_DATABLOCK		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_FROMURI	VTY_EVENT_MATRIX	<See Below>	<See Below>

Matrix Size – indicates the number of Data Identifier / Data Name pairs.

Data Field Variable length data consisting of one or more tVTYEventInfo structures		
Data Item	Size	Description
1	tVTYString –OR- tVTYDynString	**Data Name** – associated with the corresponding data identifier.
2	WORD	**Data Item ID** - Unique value corresponding to the following Data Name
3	WORD	**Item Type Code** – a code indicating the type of data (spectral, trend, etc.)
4	WORD	**Data Interval** – the millisecond interval between data items that will be received

209	Data Output	**VTY_EVENT_DATABLOCK**
This event outputs the latest values for all data marked for export in the configuration AND have been included in the type mask given in the VTY_CFG_TOOLISHOST.		
Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_FROMURI	VTY_EVENT_DATABLOCK	<See Below>	<See Below>

Status Field - the number of data description structures found at the start of the data block

Data Field
The data block consists of two sections. First the data item descriptor structures that identify each piece of data and give an offset to the actual data in the second section of the data buffer. The data after the structures is a dynamic buffer containing the actual data described by the description structures
Refer to tVTY DataItem and tVTYRawSpectrumHdr in reference document #1 for details

Section 1 - Data Item descriptor [One or more] consist of the following:

Data Item	Size	Description
1	WORD	**Data Item ID** identifying the spectra being transferred as defined in the VTY_DATA_ID_MATRIX event
2	WORD	**Item Type Code** using the same codes used by VTY_CFG_TOOLISHOST command to describe what type of data is represented
3	DWORD	**Offset** from the beginning of the data block where the actual data described here can be found.
4	BYTE	**Data Type** describing the data for each nm of data in the spectrum
5	WORD	**Number** of spectra found at the offset
6	float	**First Item Time** - Relative time of the first data point in the set of data
7+	UNION	ONE of the following Data "Sub" Structures Raw Spectra (18 Bytes) Spectral Equation (18 Bytes) Region Equation (2 Bytes) Trend Equation (2 Bytes) Advanced Equation Trend: (2 Bytes) Advanced Equation Spectrum (18 Bytes)

Section 2 – Dynamic Buffer [One for each descriptor, Pointed to by the "Offset" entry of the descriptor]:

*	16 Bytes	Raw Spectrum Header (16 Bytes)
--	Variable Len	Data values associated with each Data Item Sub Structure

* - Raw Spectrum Header information can occur at any point following Section 1, but will only occur immediately before Raw Spectra

Reference Appendix "B" for a Sample Data Block Diagram

210	<Reserved>	<Event 210>
Reserved for Future Use		

211	Instrument Error	VTY_EVENT_ERROR
This event informs the Client that the Verity instrument has detected an error.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –	PPCCSSLLLL<Data>		
VTY_LP_FROMURI	VTY_EVENT_ERROR	0	<See Below>
A stream that represents the elements of a tVTYEventData structure but with adjustment for Fixed / dynamic length strings. (reference the definitions and section 5.3) The event holds the error text as entered in the configuration's sequence item, the severity code, the relative event time, and a date/time stamp.			

212	Instrument is Powered	**VTY_EVENT_POWERUP**
This event informs the host that the Verity instrument powered-up and can communicate.		
Product Implementation:	☐ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
`VTY_LP_FROMURI`	`VTY_EVENT_POWERUP`	0	[String Data Length] – (Reference section 5.3)
The data contains a string as defined by `tVTYString or tVTYDynString` that contains the name of the device.			

213	<Reserved>	**<Event 213>**
Reserved for Future Use		

214	<Reserved>	**<Event 214>**
Reserved for Future Use		

215	<Reserved>	**<Event 215>**
Reserved for Future Use		

216	<Reserved>	**<Event 216>**
Reserved for Future Use		

217	System Issue Acknowledgment	**VTY_EVENT_ERROR_ACK**
This event informs the host that an error has been acknowledged or resolved. If the issue is flagged as a “status” issue then the issue has simply been resolved. Otherwise this is being sent due to a user acknowledging the error or is a requested ACK as part of a VTY_CMD_ACK_ERROR command. NOTE: Event reporting level must be set to >=1 using the VTY_CMD_SETEVENTREPORTING command, or request acknowledgement with VTY_CMD_ACK_ERROR, to receive these.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

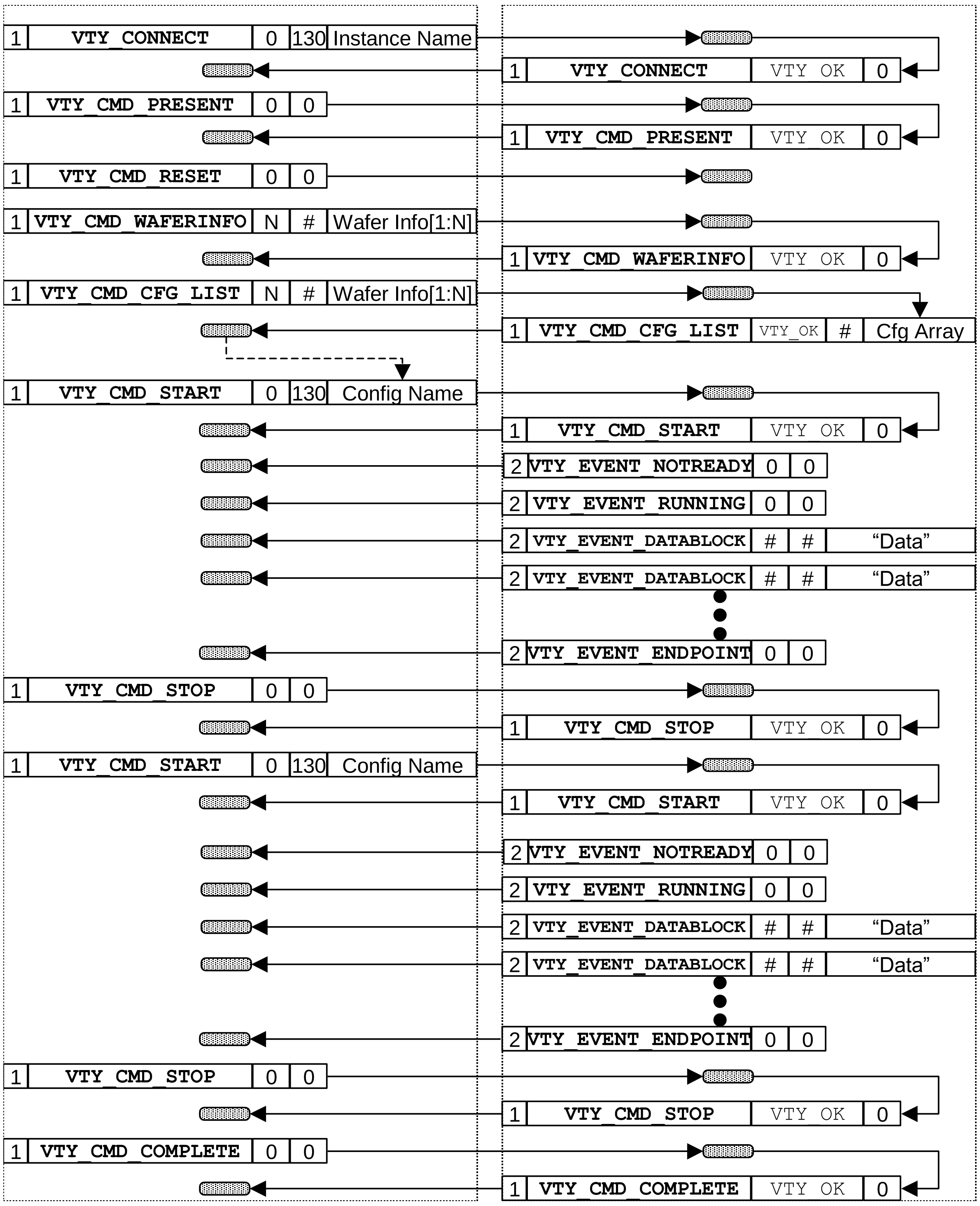
Transmission Format –		PPCCSSLLLL<Data>	
`VTY_LP_FROMURI`	`VTY_EVENT_ERROR_ACK`	0	`<See Below>`
A stream that represents the elements of a tVTYEventData structure but with adjustment for Fixed / dynamic length strings. (reference the definitions and section 5.3) The event holds the details of the issue as it was originally sent as part of the VTY_EVENT_ERROR event.			

218	User Event Acknowledgment	VTY_EVENT_USEREVENT_ACK
This event informs the host that a user event has been acknowledged. This is being sent due to a user acknowledging the error or is a requested ACK as part of a VTY_CMD_ACK_USEREVENT command. NOTE: Event reporting level must be set to >=1 using the VTY_CMD_SETEVENTREPORTING command, or request acknowledgement with VTY_CMD_ACK_USEREVENT, to receive these.		
Product Implementation:	☑ SpectraView;	☐Spectrograph;

Transmission Format –		PPCCSSLLLL<Data>	
VTY_LP_FROMURI	VTY_EVENT_ERROR_ACK	0	<See Below>
A stream that represents the elements of a tVTYEventData structure but with adjustment for Fixed / dynamic length strings. (reference the definitions and section 5.3) The event holds the details of the user event as it was originally sent as part of the VTY_EVENT_USEREVENT event message.			

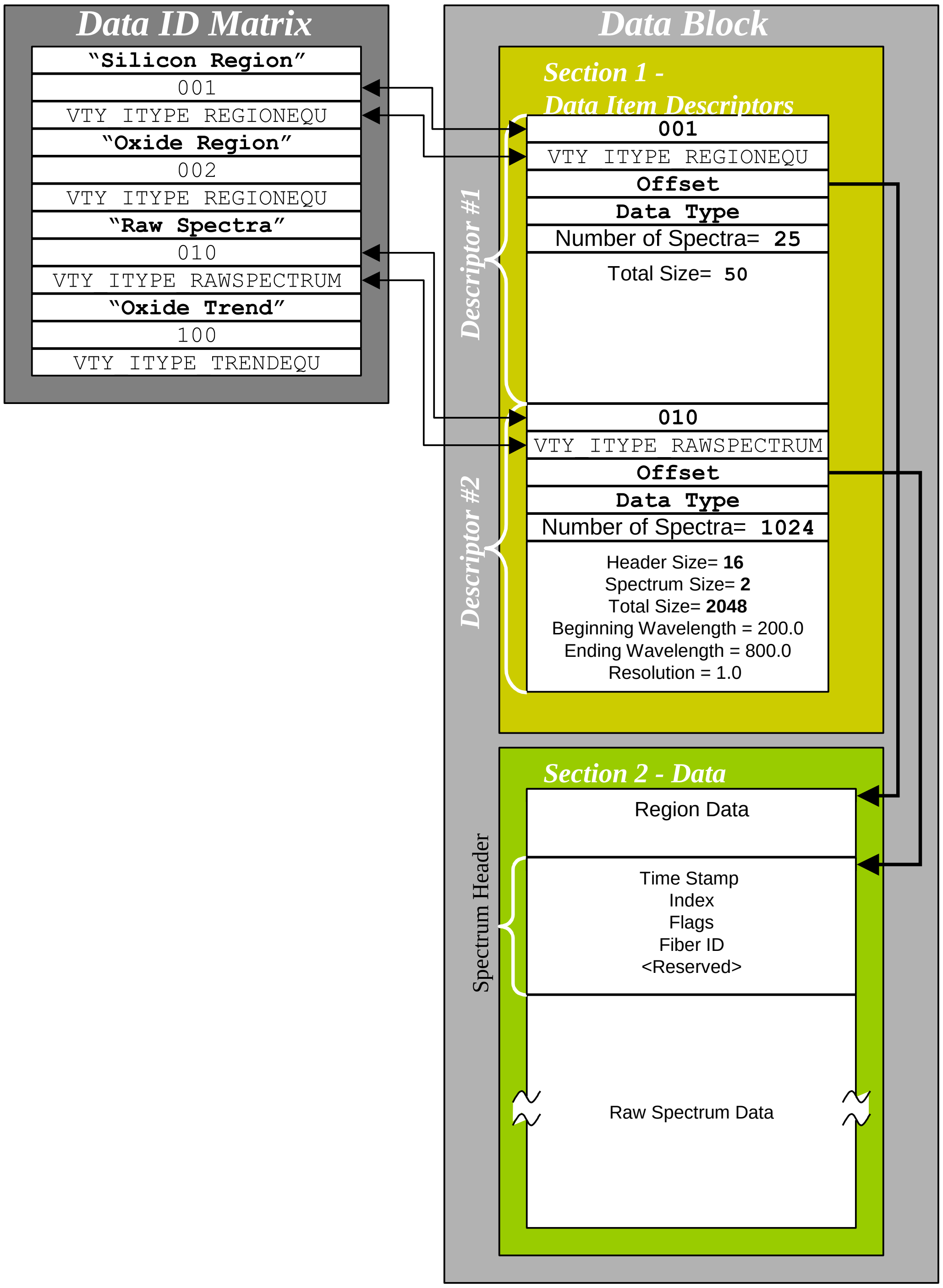
7 Appendicies

Appendix A. Example Command Sequence


1 VTY_CONNECT 0 130 Instance Name
1 VTY_CONNECT VTY_OK 0
1 VTY_CMD_PRESENT 0 0
1 VTY_CMD_PRESENT VTY_OK 0
1 VTY_CMD_RESET 0 0
1 VTY_CMD_WAFERINFO N # Wafer Info[1:N]
1 VTY_CMD_WAFERINFO VTY_OK 0
1 VTY_CMD_CFG_LIST N # Wafer Info[1:N]
1 VTY_CMD_CFG_LIST VTY_OK # Cfg Array
1 VTY_CMD_START 0 130 Config Name
1 VTY_CMD_START VTY_OK 0
2 VTY_EVENT_NOTREADY 0 0
2 VTY_EVENT_RUNNING 0 0
2 VTY_EVENT_DATABLOCK # # "Data"
2 VTY_EVENT_DATABLOCK # # "Data"
2 VTY_EVENT_ENDPOINT 0 0
1 VTY_CMD_STOP 0 0
1 VTY_CMD_STOP VTY_OK 0
1 VTY_CMD_START 0 130 Config Name
1 VTY_CMD_START VTY_OK 0
2 VTY_EVENT_NOTREADY 0 0
2 VTY_EVENT_RUNNING 0 0
2 VTY_EVENT_DATABLOCK # # "Data"
2 VTY_EVENT_DATABLOCK # # "Data"
2 VTY_EVENT_ENDPOINT 0 0
1 VTY_CMD_STOP 0 0
1 VTY_CMD_STOP VTY_OK 0
1 VTY_CMD_COMPLETE 0 0
1 VTY_CMD_COMPLETE VTY_OK 0

Implementation of some Commands is Limited

Appendix B. Example Data Block



Implementation of some Commands is Limited

Appendix C. Example of Fixed String Message

Example message for

VTY_CMD_START

The Configuration Name used for this example will be:

`ChamberTest1`

Using the Field codes (as defined in section 4.3) for message content defined, this message is formatted as:

```
PPCCSSLLLLsA128sTsL
```

Substituting mnemonic constants or strings the result is:

```
VTY_LP_TOURI | VTY_CMD_START | 0 | 130 | ChamberTest1\0 [115 NULLS] | VTY_ASCII |128
```

Converting to HEX for all Fields results in (Line wrap begins after null in Configuration Name) :

```
0100 | 7200 | 0000 | 82000000 | 4368616d626572546573743100
0000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000
| 00 | 80
```

Total Bytes transmitted: **140**

Appendix D. Example of Dynamic String Message

Example message for

VTY_CMD_VALIDATE

The Configuration Name used for this example will be:

`PolyEtchStep`

Using the Field codes (as defined in section 4.3) for message content defined, this message is formatted as:

```
PPCCSSLLLLsEsTsLsAs0
```

Substituting mnemonic constants or strings the result is:

```
VTY_LP_TOURI | VTY_CMD_CFG_VALIDATE | 0 | 16 | <ESC>| VTY_ASCII | 12 | PolyEtchStep |0
```

Converting to HEX for all Fields results in:

```
0100 | 7B00 | 0000 | 10000000 | 1B | 00 | 0C | 506F6C794574636853746570 | 00
```

Total Bytes transmitted: **26**

Implementation of some Commands is Limited

8 Revision History

Revision	Changes	Change Date
A	Document Created	
B	Reformatted	2001
C	Minor Changes and Additions	Oct 2001
D	- Added Revision History Section - Updated VTY_CMD_TEST and VTY_CMD_PRESENT to more accurately reflect usage and function. - Updated VTY_CMD_WAFERINFO to add “NEW”, “UPDATE” and “APPEND” options. Also added “Step No” and 5 “Custom” fields - Updated VTY_EVENT_ENDPOINT to contain data related to the conditions at EP	Apr, 12, 2001
E	- Accepted Tracked Changes from revision D - Added definition for Command #124 (SHUTDOWN) - Updated Page Footer for Limitation and Confidential	Apr 22, 2002
F	- Added descriptions for the VTY_CMD_WAFERINFO status codes	
G	- Further clarified the VTY_CMD_WAFERINFO status codes	Oct 24, 2002
H	- Added the sub-command codes for VTY_CMD_RESET	Nov 20, 2002
I	- Minor adjustments to inaccuracies in the document discovered while detailing the dynamic string version of the protocol	May 2, 2003
J	- Merged functionality of Dynamic string content so that one document could be referenced for either Fixed Length and Dynamic (variable) length strings	May 7, 2003
K	- Added section 5.4 and updated VTY_CONNECT to definedmethod for activating use of Dynamic strings - Added appendix “C” and “D”	May 15, 2003
L	- Added command #(-)minus102 VTY_RECONNECT and Updated VTY_CONNECT to allow either command to be used to re-establish a connection with the host, typically due to the host being reset / rebooted - Updated Appendix “A” to include VTY_CONNECT as the first message	Dec 2, 2003
M	- Added commands 127 128, and 129 for Verity use between URI and SpectraView in support of Externally initiated reprocessing, Such as could be performed by Verity test simulators to aid the automation of formal validation testing	March 1 2004
N	- Updated VTY-CONNECT and VTY RECONNECT command replies to return system specifics - Added Commands 131 thru 135 and Events 217 and 218 in support of Advanced Status Messaging	October 18, 2005
O	- Updated data definition for VTY_CMD_ACKUSEREVENT and VTY_CMD_ACK_ERROR	Jan 12, 2006
P	- Clarified Conditions for VTY_RECONNECT	Feb 6, 2006
Q	- Modified Data Definition for VTY_CMD_CFG_VERIFY and VTY_CMD_CFG_VERIFY	Feb 13, 2006

Revision	Changes	Change Date
R	- Updated Document Version History	October 4, 2006

Implementation of some Commands is Limited