

**Contents**

[**Reference Material** 4](#_Toc58258114)

[**1.** **Introduction** 4](#_Toc58258115)

[**2.** **Physical Interface** 4](#_Toc58258116)

[**3.** **VerifyNow Network Adapter Configuration** 4](#_Toc58258117)

[**3.1.** **Background** 5](#_Toc58258118)

[**3.1.1.** **VerifyNow Interface** 5](#_Toc58258119)

[**3.2.** **ARP and Telnet** 7](#_Toc58258120)

[**4.** **Serial Communications Software Design Specification** 9](#_Toc58258121)

[**4.1.** **Overview** 10](#_Toc58258122)

[**4.2.** **Highlights of VerifyNow Serial Communication** 10](#_Toc58258123)

[**4.3.** **Enabling Serial Communication** 11](#_Toc58258124)

[**4.4.** **Physical Specifications** 11](#_Toc58258125)

[**4.5.** **Communications Interface** 12](#_Toc58258126)

[**4.6.** **The Use of Port Numbers** 12](#_Toc58258127)

[**5.** **Control Port (*NET\_CONTROL\_PORT*)** 13](#_Toc58258128)

[**5.1.** **Open Message** 13](#_Toc58258129)

[**5.1.1.** **Ack (Acknowledgement)** 14](#_Toc58258130)

[**6.** **Transmitting “Live” Test Results** 15](#_Toc58258131)

[**6.1.** **Status** 15](#_Toc58258132)

[**6.2.** **Result Port (*NET\_RESULT\_PORT*)** 15](#_Toc58258133)

[**6.3.** **Command Port (*NET\_COMMAND\_PORT*)** 15](#_Toc58258134)

[**1.** **Data Element Attributes** 16](#_Toc58258135)

[**7.1.** **Transmitting Past Results and Other Information** 19](#_Toc58258136)

[**7.2.** **Handling the *status request* Message** 21](#_Toc58258137)

[**7.3.** **Processing Input *command* Messages** 21](#_Toc58258138)

[**7.4.** **Challenges** 24](#_Toc58258139)

[**APPENDIX** 25](#_Toc58258140)

[**Data Format** 25](#_Toc58258141)

[**Sequence Diagrams** 30](#_Toc58258142)

[**Basic send + ack** 31](#_Toc58258143)

[**ack’ing Multiple Messages with Single ack** 32](#_Toc58258144)

[**Processing *command* Messages** 33](#_Toc58258145)

[**Alarm and Error records only** 34](#_Toc58258146)

[**Synapse Link Installation** 39](#_Toc58258147)

[**GEMweb Plus 500 Connectivity Guide – VerifyNow** 47](#_Toc58258148)

[**VerifyNow Installation Procedure** 47](#_Toc58258149)

[**GWP Configuration** 48](#_Toc58258150)

[**Custom Mapping** 50](#_Toc58258151)

[**Sending Quality Control data to GWP** 51](#_Toc58258152)

[**Host Connection Information** 51](#_Toc58258153)

[**Instrument Mapping** 52](#_Toc58258154)

[**Custom Mapping** 52](#_Toc58258155)

# **Reference Material**

|  |  |
| --- | --- |
| **Reference Documents** | **Revision** |
| Accumetrics VerifyNow LIS Interfacing Guide | PN: 14364;   REVISION: A |
| VerifyNow Synapse Link Installation Guide | Draft |
| VerifyNow System (RoHS) Serial Communications Software Design Specification | VFN-SDS-18-0006 |
| GEMweb Plus 500 Connectivity Guide – VerifyNow with External Barcode Scanner | Rev.01 |

# **Introduction**

The VerifyNow System provides a communications interface allowing operator access to result data for patient assays, electronic quality control tests, and wet quality control assays. This access requires utilizing the VerifyNow Network Adapter.



# **Physical Interface**

The Network Adapter is an accessory included with the VerifyNow instrument. The Network Adapter connects to the serial port on the back of the VerifyNow instrument and allows one-way data transmission (from the VerifyNow instrument to a host system) via a hard-wired Ethernet connection.

# **VerifyNow Network Adapter Configuration**

Configure the VerifyNow Network Adapter to interact with the user’s network.

| **Reference** | **Document** |
| --- | --- |
| Enabling the telnet client in Windows 10 | <https://social.technet.microsoft.com/wiki/contents/articles/38433.windows-10-enabling-telnet-client.aspx> |
| Lantronix Direct User Guide | <http://www.lantronix.com/wp-content/uploads/pdf/xDirect_UG.pdf> |
| Lantronix Device Installer | <https://www.lantronix.com/products/deviceinstaller/> |

# **Background**

# **VerifyNow Interface**

The VerifyNow Network Adapter is a custom configured version of an off-the-shelf network adapter. The settings to interface with the VerifyNow System have been hard coded so that they cannot be changed by the end user. The captive cable coming from the Network Adapter has been customized to work with the VerifyNow System. The off-the-shelf network adapter will not work without a custom adapter cable.

The off-the-shelf version by default is configured to obtain a dynamic IP address. Once a dynamic IP address is obtained, a third-party piece of software (or DOS-command level interface) is needed to find the IP address it was assigned. To prevent this extra level of complication, a static IP address has been assigned to our custom configuration. The IP address was chosen to be one on a globally recognizable local subnet so that it should be able to be plugged directly into most PC’s Ethernet port and be able to communicate.

In case the preassigned address is not accessible from any available PCs, instructions are included to reassign the address using commonly available network protocols that should not require an additional installation. In case these are also not available, a link to the third-party application is included but not supported by Instrumentation Laboratory.

The VerifyNow Network Adapter is also custom configured with Instrumentation Laboratory branding so that it is easily recognizable that it is specifically for the VerifyNow System.

**Pre-requisites**VerifyNow Network Adapters are pre-configured to have a static IP address of 192.168.0.100. The PC used to configure the Network Adapter must recognize 192.168.0.255 as a local subnet.

**Power**: The VerifyNow Network Adapter must be powered to configure it. The Network Adapter is powered directly from the VerifyNow System but can also be powered through a standard Mini-USB cable either connected to a PC USB port or a USB AC adapter.

**Network Connection**: Additionally, to configure the VerifyNow Network Adapter, it must be connected to a PC or to an Ethernet network using a standard Ethernet cable.

**Procedure**

**IP Address**: A static IP Address will need to be obtained from the IT department that can be permanently associated with the instrument.

**Web Interface**: After powering the Network Adapter and connecting it to the PC, use any web browser to browse to <http://192.168.0.100:80>.

A webpage like Figure 1 will be displayed.

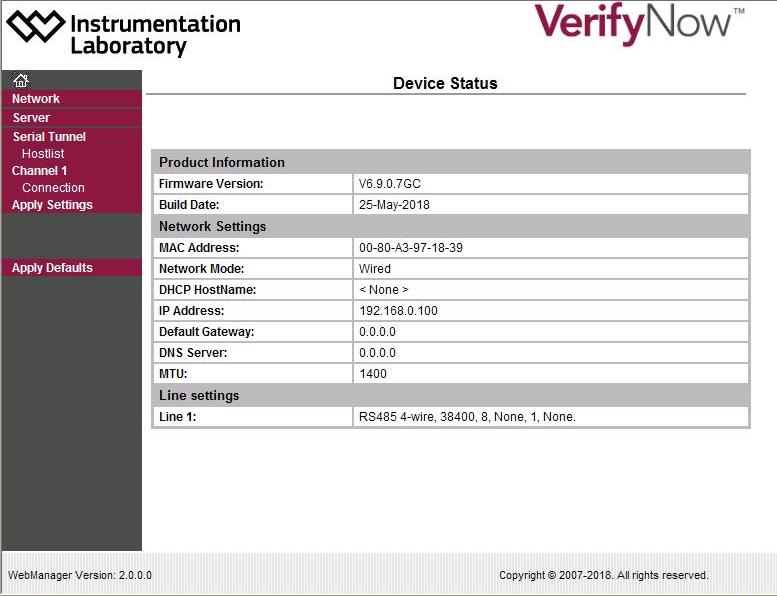


Figure 1

Click on the “Network” option as indicated. On the screen shown in Figure 2, the IP address and Subnet Mask can be configured. Once these are set, click the “OK” button, then the “Apply Settings” option on the left.

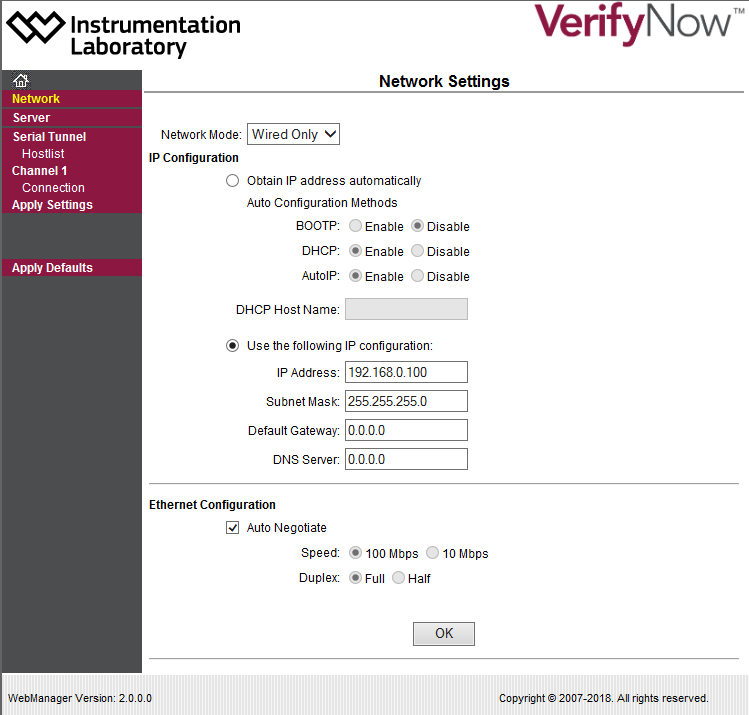


Figure 2

After applying the settings, the Network Adapter will reboot and apply the new IP address. Since the IP address is changing, the webpage will no longer work until you update your browser to connect to the new IP address.

# **ARP and Telnet**

If the IP address of the Network Adapter is unknown, it can be set using arp and telnet. Please read all instructions before attempting to configure as there are some time sensitive steps.

To complete this procedure, you need a PC that has ARP and Telnet and are both accessible from the current account. It may be necessary to run the DOS command prompt as an administrator to gain access to ARP. Telnet is not always installed by default on Windows PCs so it would either need to be downloaded or a third-party tool could be used, such as PuTTY or HyperTerminal.

Identify the hardware address of the Network Adapter. In the image below, the address is 00-80-A3-97-18-39 (the dashes must be added).



Figure 3

Open a DOS command prompt, and assign a temporary address (in this example, 10.62.64.200) by typing the following: **arp –s 10.62.64.200 00-80-A3-97-18-39**

Open a telnet connection to port 1. If telnet is installed in Windows, type the following into the DOS command prompt. The connection will fail quickly, but this triggers the Network Adapter to change its IP address: **telnet 10.62.64.200 1**

Within five seconds, open a telnet connection to port 9999. This will connect you to the text-based Setup Mode: **telnet 10.62.64.200 9999**

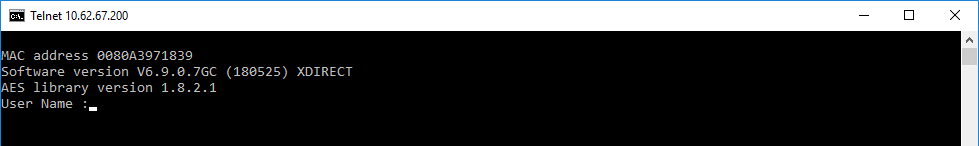


Figure 4

Login with the User Name “admin” and the password “Werfen01!” then press ENTER (note that you do not need to press ENTER after typing the user name; this will actually cause you to enter a blank password which will fail and you will need to start the process over).

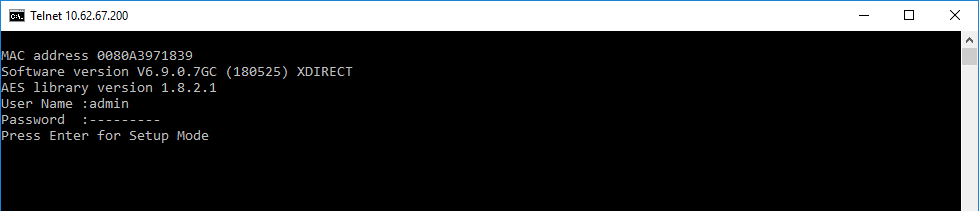


Figure 5

Type the number “0” (without quotes) then press ENTER to change the Server set up.

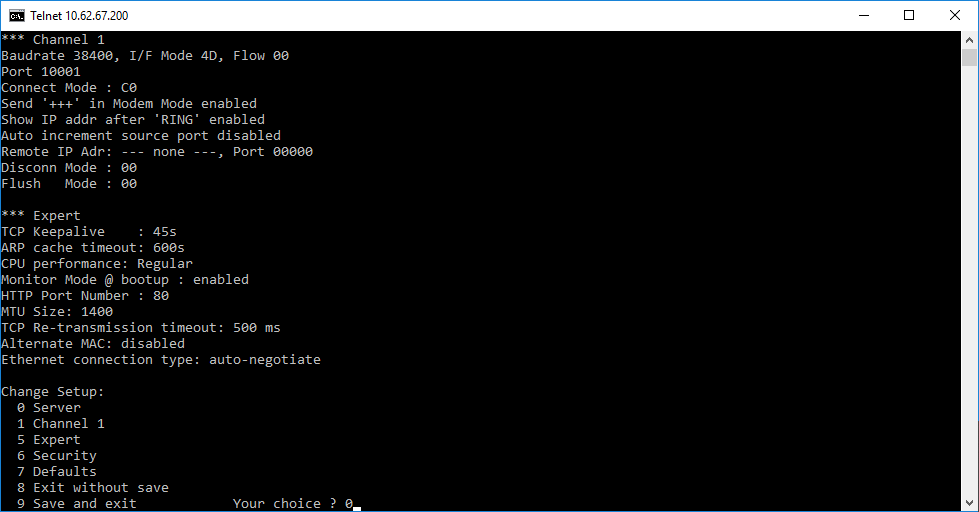


Figure 6

Enter the desired IP address then press ENTER.

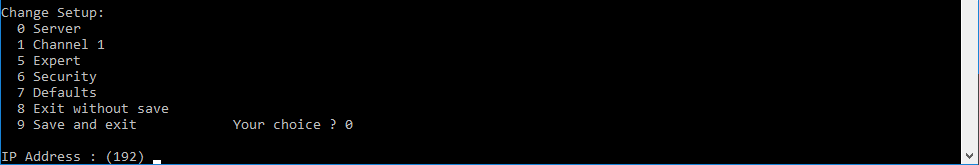


Figure 7

Do not change any of the subsequent settings unless required. Press ENTER four times to retain the current settings and exit back out to the menu.

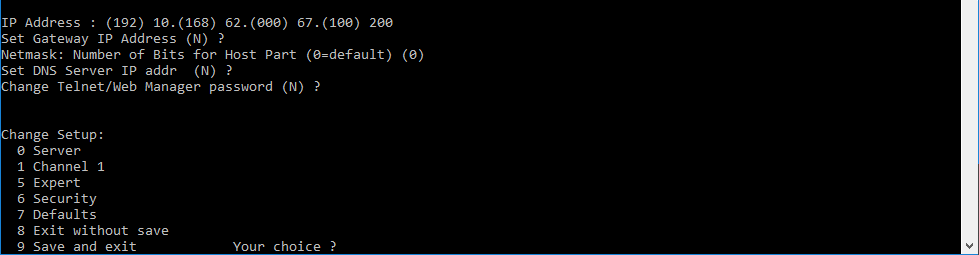


Figure 8

Type the number “9” (without quotes) and press ENTER to save the settings.

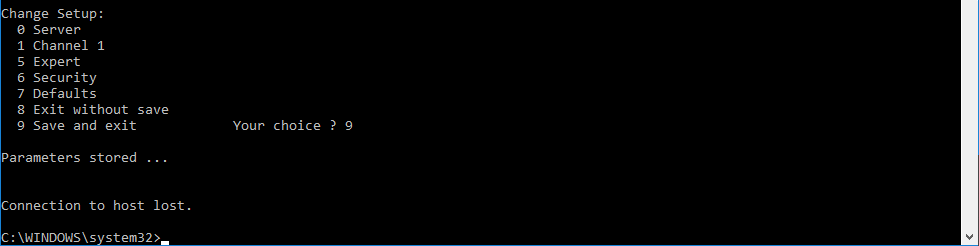


Figure 9

The Lantronix Device Installer is an easier way to reconfigure the IP address, but this piece of third-party software is not supported by IL. A link to download the software is included as well as a link to the User Guide is in the table at the beginning of this section.

*Note:* If looking to connect more than one VerifyNow instrument, the IP addresses of the two (or more) instruments need to be reconfigured. If two instruments are both using the default IP address, there will be a problem in talking to them, since both are using the same IP address.

Reconfigure the IP address for one VerifyNow instrument, using the web interface, then reconfigure the IP address for any additional VerifyNow instruments. There is a separate web interface for each instrument, so there will never be a Channel 2, for example.

# **Serial Communications Software Design Specification**

This section describes how the VerifyNow instrument uses its serial port to communicate with attached devices. Specific instrument features related to the use of the serial port are also presented.

The intended audience for this document includes engineering, technical and support personnel who have a need or interest to understand how the VerifyNow System (RoHS) communicates over its serial port.

This information applies to the VerifyNow instrument with software part number SW-100-00 and SW-100-01. It emphasizes the serial messaging interfaces employed by the instrument. Details about how the interfaces are designed and implemented in the instrument software are beyond the scope of this document.

# **Overview**

The VerifyNow instrument has two external communication ports:

1. A DB25 parallel port used to send test results and other information to a printer. It is also used by R&D and the factory to update the instrument software.
2. A DB9 serial port, used for two-way communication between the instrument and an attached serial device, such as the Network Adapter or a PC.

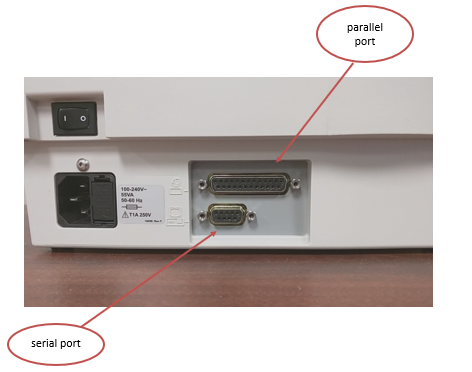


Figure 10

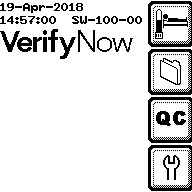
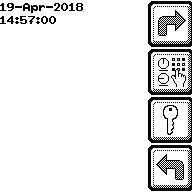
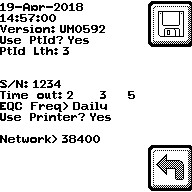
The communication ports on the VerifyNow

# **Highlights of VerifyNow Serial Communication**

* Physical interface is RS422, 8 data bits, no stop bits, 1 parity bit; maximum baud rate is 38400.
* Serial communication is enabled by default, with the baud rate set to the maximum, 38400.
* The instrument will send an open message every 15 seconds until it receives a valid response, at which point the communication session is established.
* Once communication is established, any messages transmitted by the instrument require an ACK by the attached device. If the instrument does not receive an ACK after 3 seconds, the message will be sent one more time. Likewise, the instrument will send an ACK for any message it receives.
* Assay results are always transmitted at the conclusion of a successful test.
* Earlier assay results, as well as errors, alarms, and statistics, may be transmitted on-demand via the instrument UI.
* The instrument responds to incoming status and command messages received over the serial port. Some commands cause the instrument to transmit information like assay results, other commands trigger diagnostic actions, like turning motors or valves on and off.

# **Enabling Serial Communication**

Serial communication is enabled by default, with the baud rate set to its maximum supported value of 38400. Below is the UI flow to view or modify the baud rate:

From the rightmost panel shown in **Figure 2**, the left and right arrow keys on the keypad may be used to modify the baud rate for the serial port. Setting the baud rate to NONE disables serial communication.

# **Physical Specifications**

|  |  |
| --- | --- |
| Electrical | RS-422, DB9 connector |
| Data bits | 8 |
| Parity | None |
| Stop bits | 1 |
| Baud Rate | 1200,2400,4800,9600,19200,38400, selectable via the UI |

# **Communications Interface**

Messages transmitted or received over the serial port have the following general format.

<STX><crc16><length><body><ETX>. Message format summarized in this table:

|  |  |  |
| --- | --- | --- |
| **Field** | **Format** | **Description** |
| <STX> | 8 bits unsigned | Start-of-text, contains ASCII code 0x02. Signals the beginning of the message. |
| <crc16> | 16 bits unsigned | Contains the CRC value of the <length> and <body> fields. |
| <length> | 16 bits unsigned | Contains the length of the <body> field. |
| <body> | Variable length, 8-bit unsigned | Contains the body of the message. |
| <ETX> | 8 bits unsigned | End-of-text, contains ASCII code 0x03. Signals the end of the message. |

The body of the message consists of the following fields. Body format summarized in this table:

|  |  |  |
| --- | --- | --- |
| **Field** | **Format** | **Description** |
| sequenceNum | 32 bits (4 bytes) unsigned | Used for ordering and ack’ing messages. |
| port | 8 bits (1 byte) unsigned | Logical ID number indicating the type of message. |
| data | Array of 1 or more 8 bit (1 byte) unsigned characters. | The payload of the message. |

# **The Use of Port Numbers**

The port number within the body of the message identifies the type of the message. Specified port numbers are described below

|  |  |  |
| --- | --- | --- |
| **Port Number** | **Tag** | **Description** |
| 0 | NET\_RESULTS\_PORT | Used by the instrument when transmitting individual assay results or dumping all saved records. |
| 1 | NET\_CONTROL\_PORT | Used when sending or receiving ‘open’, ‘ack’ or status messages. |
| 2 | NET\_CONSOLE\_PORT | R&D use only |
| 3 | NET\_TIME\_DOMAIN\_PORT | R&D use only. |
| 4 | NET\_DEBUG\_PORT | R&D use only. |
| 5 | NET\_COMMAND\_PORT | Used when sending commands to the instrument and command responses from the instrument. |
| 6 | NET\_UPDATE\_PORT | Used when downloading a new flash image to the instrument, for updating the system software. |

# **Control Port (*NET\_CONTROL\_PORT*)**

The control port is used to indicate an Open request from the instrument and to acknowledge transmissions.

Once the VerifyNow instrument determines that serial communication has been enabled, either during power-up or following activation by the user (in the Enabling Serial Communication **Section 5.2**), it will begin sending out *open* messages over the serial port every 15 seconds until it receives a response. NET\_CONTROL\_PORT (equal to 1) is the port number used for sending the *open* message. Any message transmitted or received using NET\_CONTROL\_PORT can be referred to as a control message.

The sequenceNum of the *open* message will be 0. The first byte within the data field (byte 0) is the command (cmd) byte, which will have the value of 0. The remaining bytes within the data field contain a series of keyword=string pairs, delimited by spaces, describing some of the basic properties of the instrument.

# **Open Message**

For an Open record, the Protocol Packet Sequence Number field is the starting sequence number for the instrument, always zero, and its data content is several keyword=value sets separated by spaces. The specific information provided is the instrument serial number, system state, the software Part Number, the revision number of the software part number, the version – a subversion revision number – and the build date for the software.

The instrument software expects an Open record in reply from the interfacing software with its initial sequence number. This no presumption is made about the value of the sequence number given. The Data field content, if any, on this reply is ignored. The sample interface program provided, saves this information for augmentation of all following data records from the instrument.

|  |  |  |
| --- | --- | --- |
| ***Keyword*** | ***String Content*** | ***Example*** |
| Serial | Instrument Serial Number | 3154 |
| systemState | Single character representing the system state: I (Idle), S (sleep), A (assay), Q (QC), E (EQC), W (WQC), R (patient recall), M (maintenance mode), or U (starting up) | I |
| Pn | Software Part Number loaded | 40072 |
| Rev | Software Revision Number | G |
| Version | Software Archive Version Number | S284 |
| Build | Date and Time of Software Build | Mon Oct 4 2020 |

The body of the *open* message:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 0 | <keyword=string> pairs |

cmd

port

sequenceNum

data

The instrument will not attempt to send any other messages over the serial port until it receives a response from the *open* message.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | x | x | x | 1 | 0 |

port

sequenceNum

cmd=data [0]

The instrument always sends this record first when started from a power-up. It will try every 30 seconds to send the record until it gets a reply. The instrument will not consider the network as usable until it receives a valid reply. Any data to be sent over the interface before this period will be discarded. Any interfacing program must re-synchronize its sequence number and discard any unprocessed data when it receives the Open record. An opposite condition can exist when the instrument has been using the interface and the interfacing program has been restarted. In this case, the interfacing program should send a Status request record to the instrument to obtain the serial number, *etc*.

# **Ack (Acknowledgement)**

For the Ack record, the Protocol Packet Sequence Number is the sequence number being acknowledged for successful receipt. The Data packet is empty.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | 1 | 1 |

port

sequenceNum

cmd=data [0]

Any non-control message sent by the instrument is considered “ack’d” if its sequence number is less than the sequence number of the incoming *ack* message. Thus, a single incoming *ack* message may ack more than one message sent by the instrument.

If the instrument has not received an *ack* message within 3 seconds of transmitting a non-control message, then the non-control message will be re-transmitted one more time.

# **Transmitting “Live” Test Results**

With serial communication enabled and a communication session established between the instrument and an attached device (i.e. an *open response* message has been received), the instrument will automatically and unconditionally transmit the result of any test it performs over the serial port. This *result* message will contain a port number of 0 (NET\_RESULTS\_PORT) and a series of keyword=string pairs enclosed by an XML <Result /> tag in the data field.

The body of the *result* message:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | 0 | “<Result “, keyword=string pairs, “ />” |

data

port

sequenceNum

# **Status**

The Status record is used by the remote software to request status information from the instrument, the reply will be sent on this stream. The reply Status record contains the same data fields as the open request.

# **Result Port (*NET\_RESULT\_PORT*)**

During live operation, the data content can contain patient results, EQC results, or WQC result records. They are structured as XML records as described in the instrument DTD below. The instrument can also function in re-transmit mode where all past assay, alarm, error, miscellaneous, and statistical records are transmitted over this port. To manually invoke re- transmission, within the instrument maintenance screens, the Net Results folder function exists. Activation of this function will send all retained past assay, EQC, WQC, Alarm, and Error records to the host over the NET\_RESULTS\_PORT in an xml structured DumpRecords element. *See the instrument DTD below*

# **Command Port (*NET\_COMMAND\_PORT*)**

The instrument has a command processor task which receives commands sent on this stream. The commands that can be used are limited. While several commands are used for system testing/diagnostic purposes, only one has been implemented in the supplied sample program. This is the ‘dump records results’ command. This command will place the instrument into re-transmit mode as described above in the NET\_RESULT\_PORT section and reply with a NET\_CONTROL\_PORT status record, “Dump Complete’ when the re-transmission is complete. For instrument integrity, limit any command interfacing to this command.

# **Data Element Attributes**

|  |  |
| --- | --- |
| **Data Element** | **Description** |
| **state** | **state**: Alarm and Error records only. This indicates the instrument state when the alarm or error occurred. It can have a single character value with the values interpreted below: |
| I | Idle |
| S | Startup |
| A | Assay |
| Q | QC |
| E | EQC |
| W | WQC |
| R | Recall |
| M | Maintenance |
| U | Unknown |
| **Alarm:type** | Class of alarm as one of the following strings: |
| UnspecifiedType | |
| Logic | |
| Heater | |
| Resource | |
| Mechanical | |
| Cartridge | |
| Open | |
| Timeout | |
| Data | |
| Test | |
| pn | the part number for the software in the instrument. Normal production has a part number of XXXX. rev – is the Revision of the software part number |
| version | Revision of the software part number.  version – is the Release number of the software as ‘Snnn’ |
| build | is a date and time string when the software release was built. |

|  |  |  |
| --- | --- | --- |
| **Data Element** | **Description** | **Output Example** |
| VerifyNow instrument type | For clinical use - ***not for RUO (research use only) use*** | ruo="no" |
| type (or *Assay type*) | PRUTest or Aspirin Test | Assay type="PRTUest" |
| date (testDate) | Formatted date in the form: “dd-mmm-yyyy” | testDate="14-Oct-2020" |
| time (testTime) | Formatted 24 hour clock time in the form: “hh:mm:ss” | testTime="16:38:09 |
| value (for PRUTest and Aspirin Test) | For PRUTest and Aspirin Test the value is a decimal number (PRU and ARU, respectively) | wqc="No" value="0266" |
| value (for WQC) | For WQC using PRUTest cartridge or Aspirin Test cartridge, the value is a decimal number (PRU and ARU, respectively) | wqc="Yes" value="0575" |
| fail | For Aspirin and PRUTest, indicates a bad control channel value. | The ‘value’ attribute is suppressed in this case. |
| serial | The instrument serial number | serial'"1201" |
| lcf | Lot correction factor from the Lot barcode - for WQC and Patient tests | This is only presented with the data on an instrument's Net Folder operation. |
| lot | The 4-digit lot number of the cartridge used for the assay | lot="3999" |
| opid | If the instrument has been configured to use passwords, the Operator identifier | opid=" ". If opid is activated, the opid will be displayed, e.g. opid="1234Abcd". Note: opid may contain up to 24 alphanumeric characters |
| ptid | If the instrument has been configured to use Patient Identification, the Patient identifier | If ptid is activated, the ptid will be displayed, e.g. ptid="1234Abcd". Note: ptid may contain up to 24 alphanumeric characters |
| number | Error Number | See **Appendix** for Error and Alarm number and string descriptions |
| code | Alarm and Error records only, this is the string description of the error | code="NoErrors". If there is an error message, the code will be displayed, e.g. code="24" |

|  |  |  |
| --- | --- | --- |
| **EQC Only** | | |
| **Data Element** | **Description** | **Output Example** |
| **type (or *Assay type*)** | For EQC | type="EQC" |
| **o1** | The optical level 1 value seen during the EQC assay. Normal is 0.1≤o1≤1.5 | o1="0.95" |
| **o2** | The optical level 2 value seen during the EQC assay. Normal is 0.8≤o2≤3.6 | o2="1.44" |
| **v1** | The maximum vacuum draw level during the assay. Normal is v1≥5.25 | v1="8.37" |
| **v2** | The vacuum leak value during the assay. Normal is v2≤0.30 | v2="0.05" |
| **p1** | The maximum relieved pressure during the assay. Normal is –0.25≤p1≤0.50 | p1="0.00" |
| **f1** | The flag frequency counter for the assay. Normal is 13<f1<17 | f1="14" |
| **f2** | The computed cycle duration for the assay. Normal is 11≤f2≤16 | f1="14" |
| **em\*** | This is an **error** source **mask** which is zero unless the EQC result value is FAIL | em="0000". This value is a 4-character hexadecimal number |
| **value** | Result value for EQC the value is PASS or FAIL | value="PASS" |

\*em interpretation:

* Character 0 is a 4-bit value indicating the channel failing the f2 check, if any.
* Character 1 is a 4-bit value indicating the channel failing the optical level checks o1 and o2, if any.
  + Bits 0 and 2 represent the Proximal and Distal-1 o1 level checks.
  + Bits 1 and 3 represent the Proximal+1 and Distal o2 level checks.
* Character 2 is a 4-bit value indicating the channel failing a saturation check.
* Character 3 is the number of spotcode sensors that failed the EQC check if any. If more than 15 fail, the number is limited to 15 for reporting

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Keyword** | **String Content** | **EQC** | **WQC** | **Patient** | **Description** |
| ruo | “Yes” or “No” |  | X | X | Research use only. Always “No” for production instruments. |
| wqc | “Yes” or “No” |  | X | X | Wet Quality Control |
| mcf | Default is “1.0000” |  | X |  | Motor Correction Factor |
| lcf | Decimal value, e.g. “0.98” |  | X | X | Lot Correction Factor |
| Type | “EQC”, “PRUTest”, “AA” | X | X | X | Assay or test type |
| serial | Up to 8 chars | X | X | X | Instrument serial number |
| opid | Up to 24 chars | X | X | X | Operator ID |
| ptid | Up to 24 chars |  |  | X | Patient ID |
| elcf1 | Decimal value, e.g. “0.98” |  |  | X | Extended lot correction factor 1, P2Y12 only. |
| elcf2 | Decimal value, e.g. “0.98” |  |  | X | Extended lot correction factor 2, PRUTest only. |
| testDate | dd-mmm-yyyy, e.g. “04-Apr-2017” | X | X | X | The date of the test. |
| testTime | hh:mm:ss, e.g. “13:10:48” | X | X | X | The time of the test (24-hour format). |
| lot | 4 chars |  | X | X | The cartridge lot number. |
| lowPower | “Yes” or “No” |  | X | X | Indicates if data samples were captured at low emitter power. |
| value | “PASS”, “FAIL” for EQC, 4-character result (e.g. “0123”) otherwise | X | X | X | The result of the test. |
| code | Variable length error string, e.g. “LowPressure” | X |  |  | EQC error code. Used only when value=”FAIL”. |
| em | 4-digit hex string, e.g. “F000” | X |  |  | EQC error mask. Used only when value=”FAIL”. |
| o1 | Decimal value, e.g. “0.98” | X |  |  | Value of optical level 1. |
| o2 | Decimal value, e.g. “0.98” | X |  |  | Value of optical level 2. |
| v1 | Decimal value, e.g. “8.16” | X |  |  | The maximum vacuum draw level. |
| Type | Typically, one of “EQC”, “PRUTest”, “AA” | X | X | X | Assay or test type |
| serial | Up to 8 chars | X | X | X | Instrument serial number |
| f1 | 2-digit value | X |  |  | Flag frequency counter. |
| f2 | 2-digit value | X |  |  | Computed cycle duration. |

The number and type of keyword=string pairs in the data field will vary based on the specific test whose results are being reported. The types of tests are EQC (Electronic Quality Control), WQC (Wet Quality Control) and Patient. **Table 6** summarizes the keywords that are applicable to each type of test.

**Sample XML output for a Patient PRUTest:**

<Result

ruo=”No”

wqc=”No”

lcf=”0.92”

elcf1=”1.00”

elcf2=”0.86”

type=”P2Y12”

serial=”2752”

testDate=”14-Oct-2020”

testTime=”14:28:00”

lowPower=”No”

value=”0220” />

**Sample XML output for a WQC PRUTest:**

<Result

ruo=”No”

wqc=”Yes”

mcf=”1.0000”

lcf=”1.01”

type=”P2Y12”

serial=”2207”

testDate=”14-Oct-2020”

testTime=”14:28:00”

lot=”0264”

lowPower=”No”

value=”0269” />

**Sample XML output for an EQC Test:**

<Result

type=”EQC”

serial=”6031”

opid=” ”

testDate=”14-Oct-2020”

testTime=”14:28:00”

o1=”0.96”

o2=”1.78”

v1=”8.32”

v2=”0.05”

p1=”0.05”

f1=”15”

f2=”15”

value=”PASS”

em=”0000” />

# **Transmitting Past Results and Other Information**

Results from past tests (EQC, WQC and Patient) may be sent over the serial port on-demand by the user via the instrument UI.

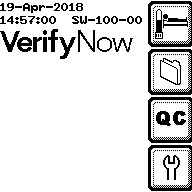
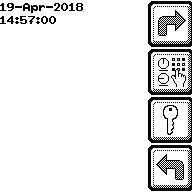
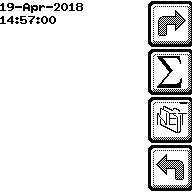
  

Figure 11

Note that the NET icon highlighted in the rightmost panel of Figure 11 will only appear if serial communication has been enabled and a communication session has been established.

Pressing the NET icon will actually transmit a lot more than just past test results over the serial port. Also transmitted will be system statistics, and all alarm, error and miscellaneous records. This is effectively a “dump records” operation, which will be discussed later. All messages transmitted as part of this operation are formatted with an XML tag and keyword=string pairs

|  |  |  |
| --- | --- | --- |
| **XML tag** | **Keywords** | **Description** |
| <SystemStatistics | serial  LastEqcStatus  LastEqcRan  LastQcRan  Eqcs  assays  wqcs  lowAssays  lowWqcs  errors  alarms | Instrument serial number, last EQC status, date/time of last EQC and WQC, total number of assays, wqcs, low-power assays, low-power WQCs, errors and alarms |
| <AssayCounts | serial  lowPower  wqc  type  count | Instrument serial number, low-power (y/n),WQC (y/n), assay type, count |
| <ErrorCounts | serial  error  number  count | Instrument serial number, error string, error ID, count |
| <ErrorStates | serial  state  count | Instrument serial number, system state when error occurred, count |
| <AlarmStates | serial  state  count | Instrument serial number, system state when alarm occurred, count |
| <AssayRecords> | n/a | Outer tag surrounding all other Patient assay results. |
| <Assay |  | Patient assay results. (wqc=”No”) |
| <WqcRecords | n/a | Outer tag surrounding all other WQC assay results. |
| <Assay |  | WQC assay results (wqc=”Yes”) |
| <EqcRecords | n/a | Outer tag surrounding all other EQC assay results |
| <Assay |  | EQC assay results (type=”EQC”, wqc=”No”) |
| <AlarmRecords | n/a | Outer tag surrounding all other alarm records |
| <Alarm | state  type  code  codename  reason  opid  lowPower | System state when alarm occurred, alarm type (e.g. “heater”), alarm code ID, alarm code string, alarm reason string, operator ID, low-power (y/n) |
| <ErrorRecords | n/a | Outer tag surrounding all other error records |
| <Error | state  code  number  opid  lowPower | System state when error occurred, error code string, error ID, operator ID, low-power (y/n) |
| <MiscRecords | n/a | Outer tag surrounding all other miscellaneous records |
| <Misc | state  type  data | System state when miscellaneous data was collected, type=”Temperature History”, data contains up to 10 temperature ADC values converted to volts. |

# **Handling the *status request* Message**

Besides the *open response* and *ack* messages, there is another control message that the instrument may receive on the serial port: the *status request* message. This is an unsolicited control message sent by the attached device requesting some status information from the instrument. The command byte for this message is 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | 1 | 2 |

sequenceNum

port

cmd=data[0]

The instrument responds to the status request message by sending the same keyword=string pairs that were sent in the open response message. (Refer back to Table 5.) The port and command byte of this transmitted status response message are 1 (NET\_CONTROL\_PORT) and 2, respectively, the same as the received message.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | 1 | 2 | <keyword=string> pairs |

cmd

port

sequenceNum

data

# **Processing Input *command* Messages**

A *command* message is an unsolicited, non-control message received on the serial port. The body of a *command* message uses port 5 (NET\_COMMAND\_PORT) and its data field contains a command string.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | 5 | <command\_string> |

data

port

sequenceNum

As expected, the instrument transmits an ack message in response to a command message. The <command string> contained within the message generally instructs the instrument to either transmit some information over the serial port or perform some diagnostic action.

When performing certain diagnostic actions, the instrument also transmits status messages via NET\_CONTROL\_PORT. These status messages are virtually the same as the status response message. The only difference is that an additional keyword=string pair is placed at the beginning of the data field to hold the status message.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | 1 | 2 | “status=<msg>” | Remaining <keyword=string> pairs |

cmd

data

port

sequenceNum

This table describes the various commands and the status messages associated with them.

|  |  |  |
| --- | --- | --- |
| **Command** | **Status Message(s)** | **Description** |
| “chaininfo” | “Chaining”, “ChainInfo Done” | Uses NET\_DEBUG\_PORT to transmit flash memory info and the number and size of the following record types:   * System statistics * System parameters * Patient tests * WQC tests * EQC tests * Passwords * Alarms * Errors * Misc |
| “erase” | “Erasing”, “Erase Complete. Restart!”, or “Refused” | Erases the flash memory of the instrument. System statistics and parameters are retained. Uses NET\_DEBUG\_PORT to transmit additional status messages. |
| “dump system” | “Dumping System”, or “Dump Complete” | Transmits system parameters, records (refer to Table 7) and passwords (if enabled) via NET\_DEBUG\_PORT. Each individual message is XML-tagged with keyword=value pairs. |
| **Command** | **Status Message(s)** | **Description** |
| “dump results” | “Dumping Records”, or “Dump Complete” | Transmits the records shown in Table 7 via NET\_RESULTS\_PORT. |
| “dump records”, or “dump <anything\_else>” | “Dumping Records”, or“ Dump Complete” | Transmits the records shown in Table 7 via NET\_DEBUG\_PORT. |
| “backlight <on/off>” | None | Turns the instrument’s LCD backlight on or off. |
| “heater <on/off>” | None | “on”: disables temperature alarms  “off”: enabled temperature alarms |
| “mixer <on/off>” | None | Turns the mixer motor on or off. |
| “startmixer <freq>” | None | Turns on the mixer motor at the specified frequency. |
| “enablemixer <on/off>” | None | Turns beep tone on or off. |
| “ccd <on/off>” | None | Turns the barcode scanner on or off. |
| “eqctest <on/off>” | None | Turns the EQC test flag on or off. |
| “doublebeep” | None | Causes the instrument to play a double-beep tone. |
| “beep <on/off>” | None | Turn the beep tone on or off. |
| “pressure <on/off>” | None | Turn the pressure valve on or off. |
| “vacuum <on/off>” | None | Turn the vacuum valve on or off. |
| “pump <on/off>” | None | Turn the pump motor on or off. |
| “fillon <value>” | None | Overrides the fill-on parameter used during cartridge fill operation. |
| “filloff <value>” | None | Overrides the fill-off parameter used during cartridge fill operation. |
| “update begin <key>” | None | Indicates the start of a software download operation. |
| “update end” | None | Indicates the end of a software download operation. |

# **Challenges**

1. Once a communication session has been established with an attached device, there is no termination protocol, i.e. the instrument always assumes that the attached device remains connected.
2. Because of a), the instrument will continue trying to send messages over the serial port, even though the attached device may have been disconnected, turned off, crashed, etc. This introduces a vulnerability.
3. Each message sent by the instrument requires the use of a buffer pulled from a pre-allocated pool of memory. The buffer is normally returned to the pool when the message has been ack’d. If a message needs to be sent, but no buffers are available in the memory pool, the instrument software will wait indefinitely until a buffer becomes available. This is a situation that should be avoided because the instrument could appear sluggish or non-responsive. Now consider the case where the attached device is slow or unable to send an ack. One way of dealing with this would be to have the instrument periodically re-transmit the un-ack’d message. However, this cannot be done indefinitely because it will eventually lead to the exhaustion of the memory pool. The solution chosen for the instrument is to have it re-transmit an un-ack’d message just once; after that, the message is simply dropped. This reduces the possibility of exhausting the pool of message buffers, which helps to keep the instrument running normally.

# **APPENDIX**

# **Data Format**

**As Sent by the Instrument**

The following Document Type Definition defines the potential content of the NET\_RESULTS\_PORT stream. The stream does not contain the root element as it is expected that the file will be incorporated into such a root container by the LIS/HIS interface logic.

<!ELEMENT Result EMPTY>

<!ATTLIST Result

type CDATA #REQUIRED  
lowPower CDATA #IMPLIED  
wqc CDATA #IMPLIED  
date CDATA #REQUIRED  
time CDATA #REQUIRED  
result CDATA #IMPLIED  
value CDATA #REQUIRED  
cu CDATA #IMPLIED  
fail CDATA #IMPLIED  
serial CDATA #REQUIRED  
lot CDATA #REQUIRED  
opid CDATA #IMPLIED  
mcf CDATA #IMPLIED  
lcf CDATA #IMPLIED  
elcf1 CDATA #IMPLIED  
elcf2 CDATA #IMPLIED  
ptid CDATA #IMPLIED  
o1 CDATA #IMPLIED  
o2 CDATA #IMPLIED  
v1 CDATA #IMPLIED  
v2 CDATA #IMPLIED  
p1 CDATA #IMPLIED  
f1 CDATA #IMPLIED  
f2 CDATA #IMPLIED  
code CDATA #IMPLIED  
em CDATA #IMPLIED>

<!ELEMENT DumpRecords  
DumpRecords  
 SystemStatistics  
AssayRecords  
WqcRecords  
EqcRecords  
AlarmRecords  
ErrorRecords  
MiscRecords>

<!ATTLIST DumpRecords

Pn CDATA #REQUIRED  
rev CDATA #REQUIED  
version CDATA #REQUIRED  
date CDATA #REQUIRED  
build CDATA #REQUIRED  
serial CDATA #REQUIRED>

<!ELEMENT SystemStatistics   
AssayCounts\*  
ErrorCounts\*  
ErrorStates\*  
 AlarmCounts\*   
AlarmStates\*>

<!ATTLIST SystemStatistics  
LastEqcStatus CDATA #REQUIRED  
LastEqcRan CDATA "undefined"  
LastQcRan CDATA "undefined"  
eqcs CDATA "0"  
wqcs CDATA "0"  
LowAssays CDATA "0"  
lowWqcs CDATA "0"  
errors CDATA "0"  
alarms CDATA "0">

<!ELEMENT AssayCounts EMPTY>  
<!ATTLIST AssayCounts CDATA #REQUIRED  
lowPower CDATA #REQUIRED  
wqc CDATA #REQUIRED  
type CDATA #REQUIRED  
count CDATA #REQUIRED>

<!ELEMENT ErrorCounts EMPTY>  
<!ATTLIST ErrorCounts  
 error CDATA #REQUIRED  
 number CDATA #REQUIRED  
 count CDATA #REQUIRED>

<!ELEMENT ErrorStates EMPTY>  
<!ATTLIST ErrorStates  
 state CDATA #REQUIRED  
 count CDATA #REQUIRED>

<!ELEMENT AlarmCounts EMPTY >  
<!ATTLIST AlarmCounts  
 alarm CDATA #REQUIRED  
 alarm CDATA #REQUIRED>

<!ELEMENT AlarmStates EMPTY >  
<!ATTLIST AlarmStates  
 state CDATA #REQUIRED  
 count CDATA #REQUIRED>

<!ELEMENT AssayRecords (Assay | RecordError)\*  
<!ELEMENT WqcRecords (Assay | RecordError)\*  
<!ELEMENT EqcRecords (Assay | RecordError)\*  
<!ELEMENT AlarmRecords (Alarm | RecordError)\*  
<!ELEMENT ErrorRecords (Error | RecordError)\*  
<!ELEMENT MiscRecords (Misc | RecordError)\*>

<!ELEMENT Assay EMPTY>  
<!ATTLIST Assay

type CDATA #REQUIRED  
ruo CDATA #REQUIRED  
wqc CDATA #REQUIRED  
date CDATA #REQUIRED  
time CDATA #REQUIRED  
value CDATA #REQUIRED  
lot CDATA #REQUIRED  
lcf CDATA #IMPLIED  
opid CDATA #IMPLIED  
ptid CDATA #IMPLIED  
lowPower CDATA #REQUIRED  
o1 CDATA #IMPLIED  
o2 CDATA #IMPLIED  
v1 CDATA #IMPLIED  
v2 CDATA #IMPLIED  
p1 CDATA #IMPLIED  
p2 CDATA #IMPLIED  
f1 CDATA #IMPLIED  
f2 CDATA #IMPLIED  
code CDATA #IMPLIED  
em CDATA #IMPLIED>

<!ELEMENT Alarm EMPTY>  
<!ATTLIST Alarm

type CDATA #REQUIRED  
state CDATA #REQUIRED  
code CDATA #REQUIRED  
reason CDATA #REQUIRED  
date CDATA #REQUIRED  
time CDATA #REQUIRED  
opid CDATA #IMPLIED  
lowPower CDATA #REQUIRED>

<!ELEMENT Error EMPTY>  
<!ATTLIST Error

state CDATA #REQUIRED  
code CDATA #REQUIRED  
number CDATA #REQUIRED  
date CDATA #REQUIRED  
time CDATA #REQUIRED  
opid CDATA #IMPLIED  
lowPower CDATA #REQUIRED>

<!ELEMENT Misc EMPTY>  
<!ATTLIST Misc

state CDATA #REQUIRED  
testDate CDATA #REQUIRED  
testTime CDATA #REQUIRED  
type CDATA #REQUIRED  
data CDATA #REQUIRED>

<!ELEMENT RecordError EMPTY >  
<!ATTLIST RecordError

type CDATA #REQUIRED  
 reason CDATA #REQUIRED>

# **Sequence Diagrams**

Establishing a Communication Session



At power up or after serial communication has been enabled, the instrument sends an *open* message every 15 seconds until it receives a response.

# **Basic send + ack**



A transmitted message is considered ack’d when the sequence number of the received *ack* message is greater than the sequence number of the transmitted message.

Note that an un-acked message will only be re-transmitted once.

# **ack’ing Multiple Messages with Single ack**



A single *ack* message may ack more than one transmitted message.

# **Processing *command* Messages**



Received commands are processed immediately if their sequence number has the expected value. If their sequence number is greater than the expected value, the command is placed in a wait buffer until a command with the expected sequence number is received. A single, transmitted *ack* message can ack multiple received command messages.

# **Alarm and Error records only**

String description of the error from the following. For Errors.

|  |  |  |
| --- | --- | --- |
| **String Value** | **Error Number** | **Description** |
| NoErrors | 0 |  |
| Bubbles | 1 | Bubbles detected during calculation analysis |
| InsufficientSample | 2 | Unable to fill first holding well |
| LotExpired | 3 | Cartridge Lot expired |
| CartSpotCode | 4 | Cartridge Spot code read error |
| InvalidBarCodeLabel | 5 | Pouch bar code does not contain required information |
| PrinterTimeout | 6 | Printer not responding. Check cables and ready status. |
| NoPaperMessage | 7 | Printer out of paper. Insert paper and make printer ready. |
| PrinterOffline | 8 | Printer not ready. Make printer Ready. |
| HumidityError | 9 | Humidity sensor indicates cartridge exposed too long |
| FillTimeout | 10 | Unable to fill assay wells in required time |
| FillAirLeak | 11 | Air leak detected during fill |
| Removed | 12 | Cartridge or vacutainer removed during assay process |
| MeasurementTimeout | 13 | Measurement data collection did not complete in required time |
| NeedSchedEqc | 14 | Scheduled EQC needed. Assay not allowed |
| EQCFailedHold | 15 | EQC failed. Assay not allowed. |
| MixerFrequency | 16 | Assay mixer frequency is out of tolerance (EQC, and Assays) |
| LowVacuum | 17 | EQC Vacuum test unable to draw enough |
| VacuumLeak | 18 | EQC Unable to hold vacuum |
| LowPressure | 19 | EQC Unable to achieve required pressure |
| HighCompare | 20 | EQC High compare is out of tolerance |
| LowProximal | 21 | Low voltage on proximal channel |
| LowDistal | 22 | Low voltage on distal channel |
| DCagreement | 23 | Channel DC agreement bad |
| SlopeAgreement | 24 | Channel slope agreement is out of tolerance |
| MeanAgreement | 25 | Channel mean agreement is out of tolerance |
| PrematureVacutainer | 26 | Vacutainer inserted with cartridge SCR-64 |
| BadCartridgeType | 27 | Bad Cartridge Type. Not supported for assay. |
| SignalSaturation | 28 | Signal Saturated. Possible low Hematocrit |
| EmitterSaturation | 29 | EQC unable to obtain proper emitter saturation |
| AlgorithmFailure | 30 | Algorithm Development Error. Internal to IL. |
| QC\_ChannelFailure | 31 | Algorithm Development Error. Internal to IL. |
| DoorOperation | 32 | Assay analysis indicates a door opened during assay or closed late. |

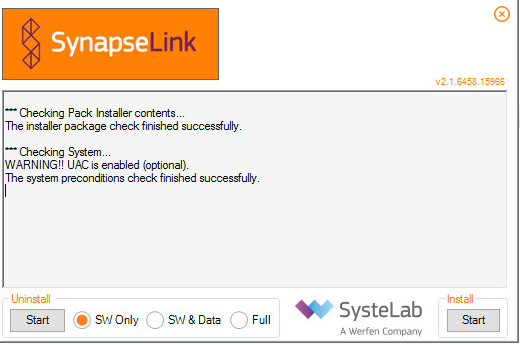
String description of the error from the following. For Alarms:

|  |  |  |
| --- | --- | --- |
| **String Value** | **Alarm Number** | **Description** |
| AD\_Object | 1 | ports.c AD object # passed to ports.c out of range. |
| AlgStgError | 2 | alg\_task.c Alg\_task: Unable to obtain storage for result parameter block. |
| AlgTimeout | 3 | user.c Algorithm task timeout with no return data. |
| Alg\_Storage\_Fail | 4 | alg\_task.c Unable to obtain workspace for filtered data. |
| AsdRelinquishTimeout | 5 | asd\_task.c Unable to acquire control of asd\_task. |
| Asd\_NoStart | 6 | asd\_task.c Asd\_ Sample conversions not startable. |
| Asd\_Resource | 7 | asd\_task.c Asd\_ Free sample request pool exhausted. |
| Asd\_Stall | 8 | asd\_task.c Asd\_ Sample conversions nodt complete in time. |
| BadPagePtr | 9 | prt\_task.c Invalid page address passed to EndPage. |
| Bit\_Assert\_Error | 10 | panels.c Bit\_ routine invoked against a non-bitImage object. |
| Can\_Assert\_Error | 11 | panels.c Can\_ routine invoked against a non-canvas object. |
| Can\_Resource | 12 | panels.c Can\_ Unable to acquire memory for canvas. |
| CorruptedMessage1 | 13 | rcd\_task.c result record corrupted. |
| CorruptedMessage4 | 14 | htr\_task.c Temperature set values' record corrupted. |
| CorruptedMessage5 | 15 | asd\_task.c Sample record corrupted. |
| CorruptedPageList | 16 | prt\_task.c Print document page list corrupted. |
| CorruptedVariable | 17 | clkdrv.c prt\_task.c user.c Detected corrupted variable. |
| Dsp\_Empty\_Panel\_List | 18 | display.c Dsp\_ no panel in active list. |
| Fld\_Assert\_Error | 19 | panels.c Fld\_ routine invoked against a non-field object. |
| HeaterUnstable | 20 | htr\_task.c Heater unstable. Temperature exceeds acceptable bounds. |
| KeyboardStuck | 21 | keyboard.c Keypad key stuck active. |
| Lbl\_Assert\_Error | 22 | panels.c Lbl\_ routine invoked against a non-label object. |
| Lbx\_Assert\_Error | 23 | panels.c Lbx\_ routine invoked against a non-listbox object. |
| LotCorrectionError | 24 | user.c Lot Correction factor is zero. |
| MeasureTimeoutTest | 25 | user.c Measurement Timeout. |
| Mem\_Assert\_Error | 26 | panels.c Mem\_ routine invoked against a non-memo object. |
| NoAlgRecord | 27 | user.c Result parameter record not returned from algorithm task. |
| NoDocumentHead | 28 | prt\_task.c Unable to obtain storage to start a print Document. |
| NoPageAlloc | 29 | prt\_task.c Unable to obtain storage for a print page. |
| NoParmBlock | 30 | user.c Expected result parameters not returned from algorithm section. |
| Oem\_Vacutainer | 31 | oem\_task.c Oem\_ Vacutainer missing when expected. |
| Panel\_Storage\_Fail | 32 | panels.c Unable to obtain workspace for bit image data. |
| Pnl\_Assert\_Error | 33 | display.c panels.c Pnl\_ routine invoked against a non- panel object. |
| Ports\_EmitterLevel | 34 | ports.c Ports: Bad R selection for emitter level. |
| PrintLineStorage | 35 | prt\_task.c Print\_Line: Unable to obtain string storage. |
| Rcd\_Storage | 36 | rcd\_task.c Unable to acquire storage for result records. |
| Sky\_Assert\_Error | 37 | panels.c Sky\_ routine invoked against a non-Softkey object. |
| Stg\_ClearError | 38 | sysparms.c GetStorage stgclear operation corrupted memory. |
| Stg\_Corrupted | 39 | sysparms.c FreeStorage returned area corrupted. |
| Stg\_FreeError | 40 | sysparms.c Attempted to free non-existent storage. |
| Stg\_FreeErrorZero | 41 | sysparms.c Attempted to free storage with no areas allocated. |
| Stg\_HeapCorrupt | 42 | sysparms.c Heap is corrupt. |
| Stg\_Storage | 43 | sysparms.c Storage tracking table full. |
| TooManyResults | 44 | user.c Electronic QC analysis returned more than one result. |
| Txt\_Resource | 45 | panels.c Txt\_ Unable to acquire memory for a textType. |
| XmsInError | 46 | panels.c Unable to move extended memory buffer to real memory buffer. |
| **String Value** | **Alarm Number** | **Description** |
| XmsOutError | 47 | panels.c Unable to move real memory buffer to extended memory buffer. |
| FlashWriteError | 48 | flash.c Unable to write to flash memory. |
| FlashReadError | 49 | flash.c Unable to read from flash memory. |
| FlashEraseError | 50 | flash.c Unable to erase a flash memory block. |
| ALG\_general | 51 | alg\_task.c General Recovery invoked in Alg\_task. |
| ASDRecovery | 52 | asd\_task.c Recovery invoked for Asd\_Task. |
| BARRecovery | 53 | bar\_task.c Recovery invoked in bar\_task. |
| BFRRecover | 54 | buffers.c Recovery invoked in Buffer\_Print. Attempting to continue. |
| CLKRecover | 55 | clkdrv.c Recovery invoked in Clk\_Task. Attempting to continue. |
| CONRecover | 56 | condrv.c Recovery invoked in Conodrv. Attempting to continue. |
| HTRRecovery | 57 | htr\_task.c Recovery invoked for Htr\_Task. |
| IgorRecovery | 58 | alg\_task.c Recovery invoked while in Igor. |
| NETRecovery | 59 | net\_task.c Recovery invoked for Net\_Task. |
| OEMRecovery | 60 | oem\_task.c Recovery invoked for OEM\_Task. |
| Stg\_FreeError | 40 | sysparms.c Attempted to free non-existent storage. |
| Stg\_FreeErrorZero | 41 | sysparms.c Attempted to free storage with no areas allocated. |
| Stg\_HeapCorrupt | 42 | sysparms.c Heap is corrupt. |
| Stg\_Storage | 43 | sysparms.c Storage tracking table full. |
| TooManyResults | 44 | user.c Electronic QC analysis returned more than one result. |
| Txt\_Resource | 45 | panels.c Txt\_ Unable to acquire memory for a textType. |
| XmsInError | 46 | panels.c Unable to move extended memory buffer to real memory buffer. |
| XmsOutError | 47 | panels.c Unable to move real memory buffer to extended memory buffer. |
| FlashWriteError | 48 | flash.c Unable to write to flash memory. |
| FlashReadError | 49 | flash.c Unable to read from flash memory. |
| FlashEraseError | 50 | flash.c Unable to erase a flash memory block. |
| ALG\_general | 51 | alg\_task.c General Recovery invoked in Alg\_task. |
| ASDRecovery | 52 | asd\_task.c Recovery invoked for Asd\_Task. |
| BARRecovery | 53 | bar\_task.c Recovery invoked in bar\_task. |
| BFRRecover | 54 | buffers.c Recovery invoked in Buffer\_Print. Attempting to continue. |
| CLKRecover | 55 | clkdrv.c Recovery invoked in Clk\_Task. Attempting to continue. |
| CONRecover | 56 | condrv.c Recovery invoked in Conodrv. Attempting to continue. |
| HTRRecovery | 57 | htr\_task.c Recovery invoked for Htr\_Task. |
| IgorRecovery | 58 | alg\_task.c Recovery invoked while in Igor. |
| NETRecovery | 59 | net\_task.c Recovery invoked for Net\_Task. |
| OEMRecovery | 60 | oem\_task.c Recovery invoked for OEM\_Task. |
| PRDRecover | 61 | prtdtask.c Recovery invoked in PrtdTask. Attempting to continue. |
| PRTRecover | 62 | prt\_task.c Recovery invoked in Prt\_Task. Attempting to continue. |
| Pnl\_Passwords\_BadFile | 63 | user.c Corrupted password file. |
| QFPlotRecover | 64 | alg\_task.c Recovery invoked in QuadradicFit\_Print. Attempting to continue. |
| RCDRecovery | 65 | rcd\_task.c Recovery invoked for Record Task. |
| RSLRecovery | 66 | user.c Recovery invoked in Results Show: |
| UIPlotRecover | 67 | alg\_task.c Recovery invoked in PlotUIIntArray. Attempting to continue. |
| USRRecovery | 68 | user.c Recovery invoked for UserTask. Shutdown will follow. |
| XMDRecover | 69 | xmodem.c Recovery invoked in Xmd\_Task. Attempting to continue. |
| BadAlign | 70 | user.c Expected 'Alignment' exception did not occur. |
| BadBrkpt | 71 | user.c Expected 'break point' exception did not occur. |
| BadDebug | 72 | user.c Expected 'debug' exception did not occur. |
| BadDvz | 73 | user.c Expected 'divide by zero' exception did not occur. |
| **String Value** | **Alarm Number** | **Description** |
| BadFpedenormal | 74 | user.c Expected 'FPU denormal' exception did not occur. |
| BadFpedvz | 75 | user.c Expected 'FPU divide by zero' exception did not occur. |
| BadFpeinvalid | 76 | user.c Expected 'FPU invalid operation' exception did not occur. |
| BadFpeoverflow | 77 | user.c Expected 'FPU overflow' exception did not occur. |
| BadFpeprecision | 78 | user.c Expected 'FPU precision' exception did not occur. |
| BadFpestack | 79 | user.c Expected 'FPU stack' exception did not occur. |
| BadFpeunderflow | 80 | user.c Expected 'FPU underflow' exception did not occur. |
| BadInvop | 81 | user.c Expected 'invalid operation' exception did not occur. |
| BadNpxabs | 82 | user.c Expected 'FPU absent' exception did not occur. |
| BadOvrfl | 83 | user.c Expected 'integer overflow' exception did not occur. |
| BadSpur | 84 | user.c Expected 'Spurious' exception did not occur. |
| NotAlign | 85 | user.c Expected 'Alignment' exception returned the above. |
| NotBrkpt | 86 | user.c Expected 'break point' exception returned the above. |
| NotDebug | 87 | user.c Expected 'debug' exception returned the above. |
| NotDvz | 88 | user.c Expected 'divide by zero' exception returned the above. |
| NotFpedenormal | 89 | user.c Expected 'FPU denormal' exception returned the above. |
| NotFpedvz | 90 | user.c Expected 'FPU divide by zero' exception returned the above. |
| NotFpeinvalid | 91 | user.c Expected 'FPU invalid operation' exception returned the above. |
| NotFpeoverflow | 92 | user.c Expected 'FPU overflow' exception returned the above. |
| NotFpeprecision | 93 | user.c Expected 'FPU precision' exception returned the above. |
| NotFpestack | 94 | user.c Expected 'FPU stack' exception returned the above. |
| NotFpeunderflow | 95 | user.c Expected 'FPU underflow' exception returned the above. |
| NotInvop | 96 | user.c Expected 'invalid operation' exception returned the above. |
| NotNpxabs | 97 | user.c Expected 'FPU absent' exception returned the above. |
| NotOvrfl | 98 | user.c Expected 'integer overflow' exception returned the above. |
| NotSpur | 99 | user.c Expected 'Spurious' exception returned the above. |
| Log\_Storage | 100 | rcd\_task.c Unable to obtain storage for Log records |
| FlagTimeout | 101 | user.c Timeout trying to position mixer flag on cartridge insertion |
| NotOvrfl | 98 | user.c Expected 'integer overflow' exception returned the above. |
| NotSpur | 99 | user.c Expected 'Spurious' exception returned the above. |
| Log\_Storage | 100 | rcd\_task.c Unable to obtain storage for Log records |
| FlagTimeout | 101 | user.c Timeout trying to position mixer flag on cartridge insertion |
| BufferOverRun | 102 | buffers.c Attempted overrun storing more data in buffer than allowed |
| Index\_Failure | 103 | buffers.c Attempted to retrieve non-existant buffer |
| Heater\_Timeout | 104 | htr\_task.c Time-out during heating cycle SCR-59 |
| UnsupportedCartrigeType | 105 | alg\_task.c Cartridge Type not supported. SCR-97 |
| CharBitImage\_NoStg | 106 | bitlogic.c Unable to get storage for character bit image. |
| CharBitImage\_NoBit | 107 | bitlogic.c Unable to find a bit for Attach |
| FlatPressure | 108 | oem\_task.c Integrated Pneumatics: no pressure change seen in well fill |
| BadTextEntry | 109 | language.c GetTextEntry: invalid text index |
| NoTimer | 110 | No Timer allocated (display.c,... |
| StackCorrupt | 111 | Dispatcher found stack corrupted |
| FanFailed | 112 | Not currently monitored. |
| NetNoBuffer | 113 | net\_task.c No free buffer for lineBufferType |
| NetBadBuffer | 114 | net\_task.c Bad buffer parameter passed |
| UpdateFail | 115 | cmd\_task.c new update command while in update mode. |
| CMDRecovery | 116 | cmd\_task.c Recovery invoked in command task |
| HtrNotCalibrated | 117 | Oem\_task.c. Assay attempted without temperature calibration of unit. |
| **String Value** | **Alarm Number** | **Description** |
| EmitterMalfunction | 118 | user.c eqc failed saturation or corrosion test. SCR0357 |
| FlashProgrammingVoltage | 119 | flash.c low programming voltage detected in write or erase flash memory |
| LibraryConvergence | 120 | alg\_task.c non-linear solver failed to converge. |
| FlashTypeError | 121 | flash.c unknown flash type at FlashOperation |

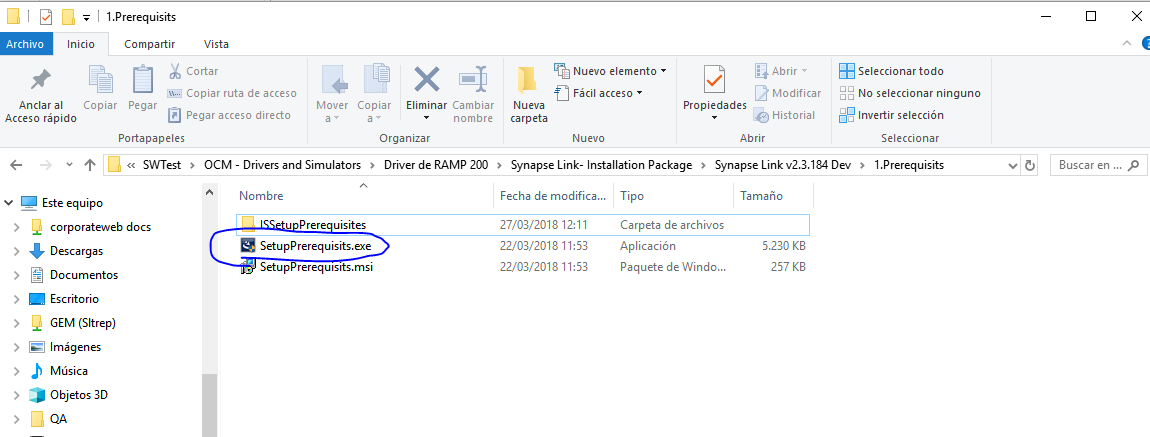
# **Synapse Link Installation**

Please find the instructions to install Synapse Link.

Copy this installation package folder in the machine. Before start (Just in case is not make sure that the machine did not have before Synapse products) if so, it could have installation problems. Go to: Pack Installer -> Press Setup  Select Full and press Uninstall Start (Full)

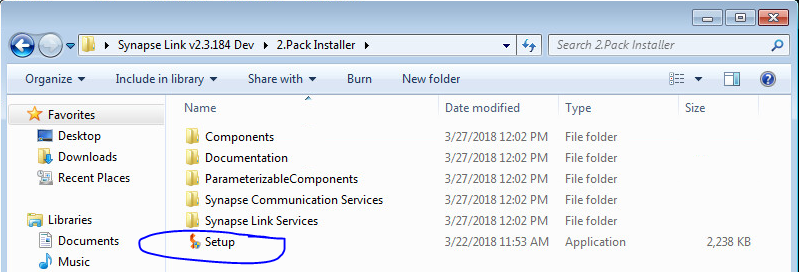


Once the machine is clean, start the installation (use the .exe file, not the msi)



This option will detect if there is something that needs to be installed before start… follow the instructions if it is needed.

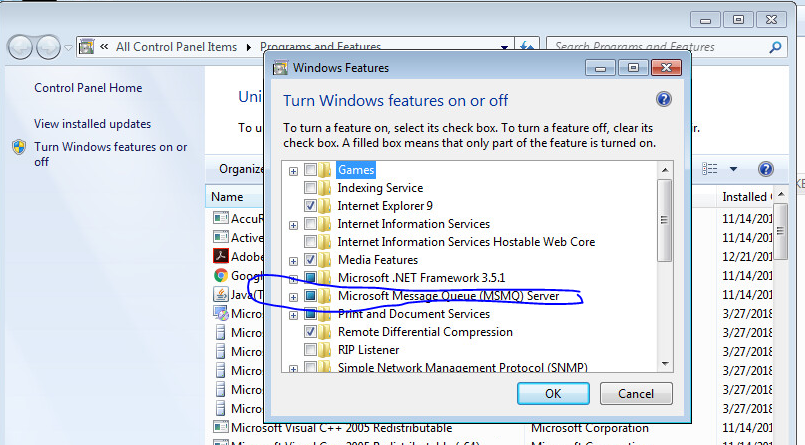
Once the prerequisites are installed, go again to the second folder and press Setup. If the Synapse Communications (version 2.5.58.0 or higher) is not installed in the current machine, ask for it to Synapse.



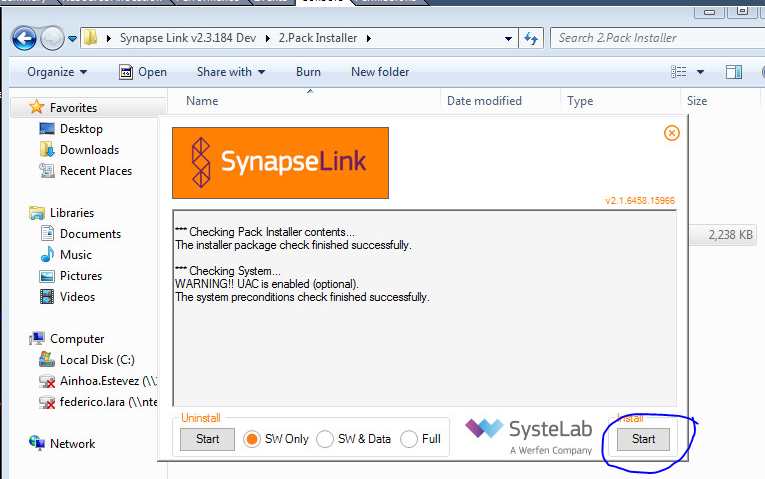
A pop up will be shown, if there is a warning or an error, it needs to be fixed before continuing, if not, Synapse will not install.



If needed: If the pop up asks for it, go to windows  Control Panel  Programs and Features  Turn Windows features on or off  active the option Microsoft message Queue.



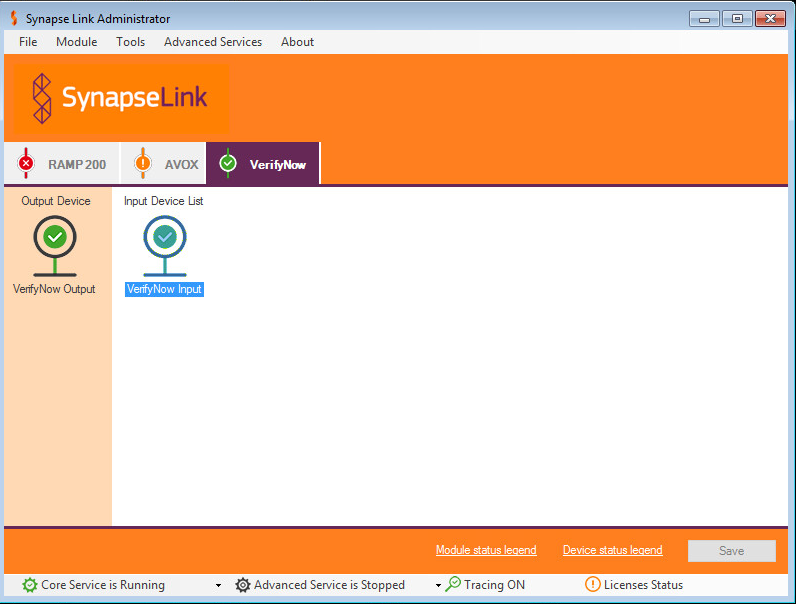
When this is ready, start installing Synapse Link: Go to folder >>>> Synapse Link >>>> 2.Package Installer and select Install.



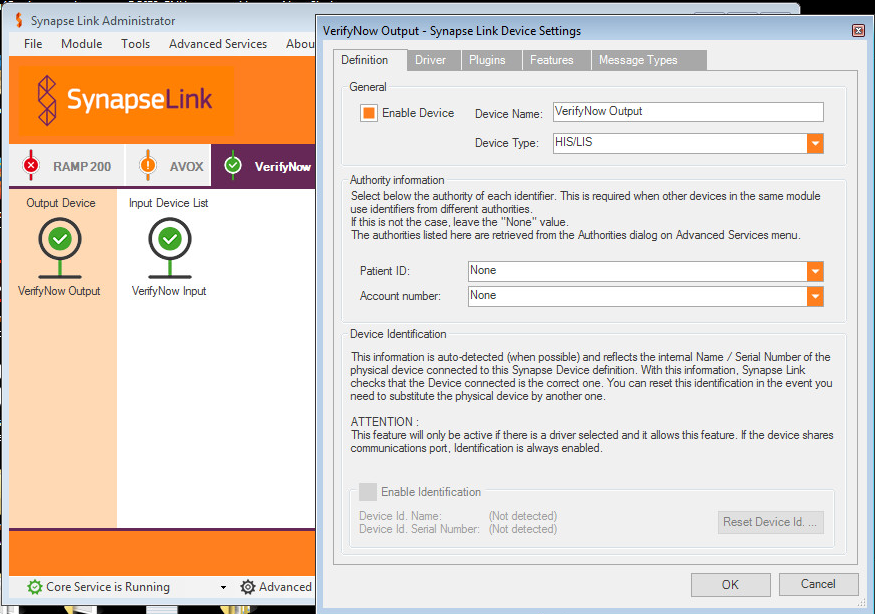
When it is installed, go to Synapse Link Administrator. Create a module specific for each driver that uses Synapse Link and configure single Input-Output for VerifyNow.

In the case of VerifyNow, the following Synapse drivers are needed:

* IL / GEMWebPlus Mode/ ASTM / provider (at the output)
* IL / VerifyNow / XML / client (at the input)

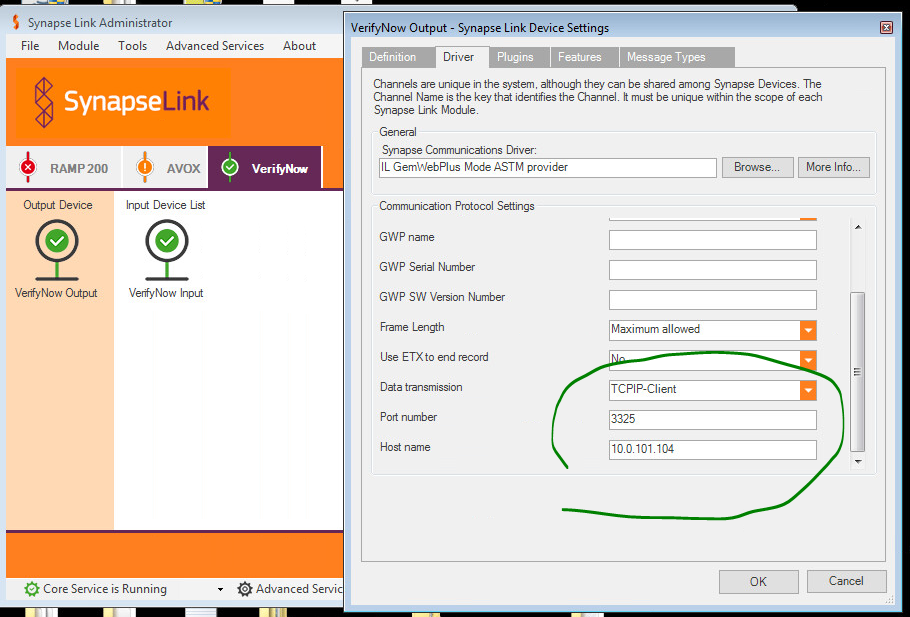


Configure Output selecting the driver from IL / GEMWebPlus Mode / ASTM Mode / provider

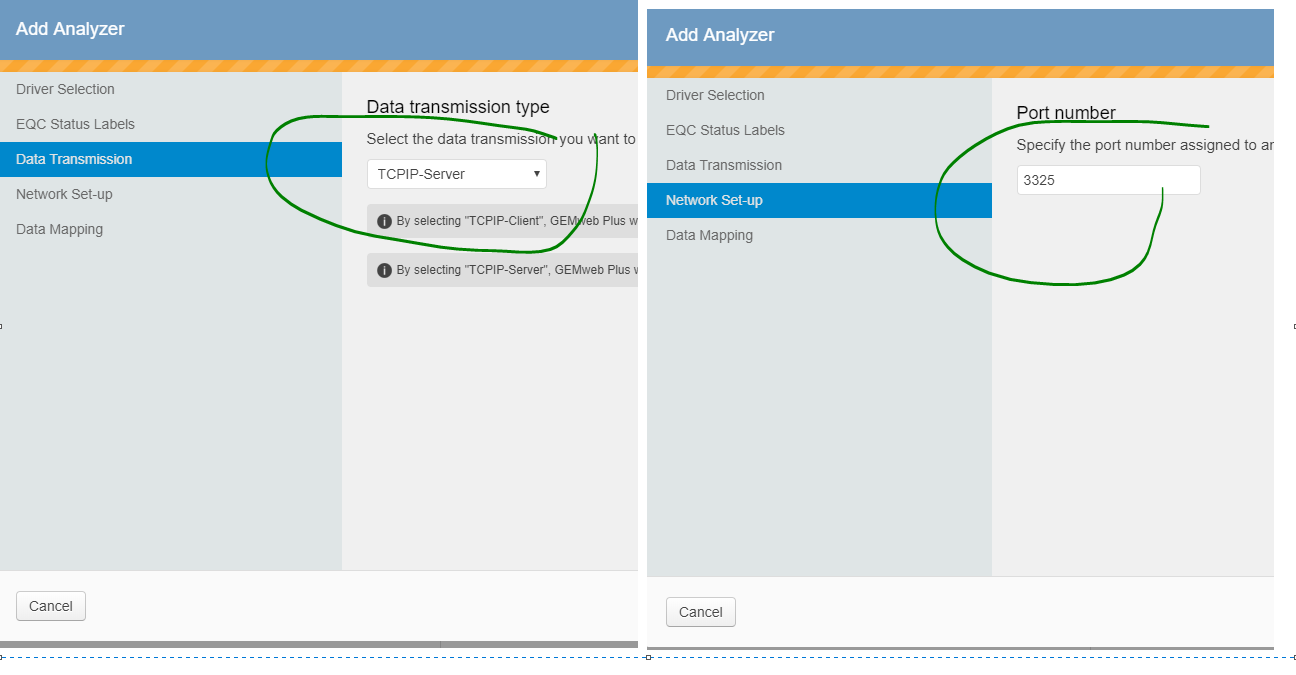


At the output-driver tab, select the IP & port that are set also at GWP.

Synapse Link output--- is connected to ----> GWP

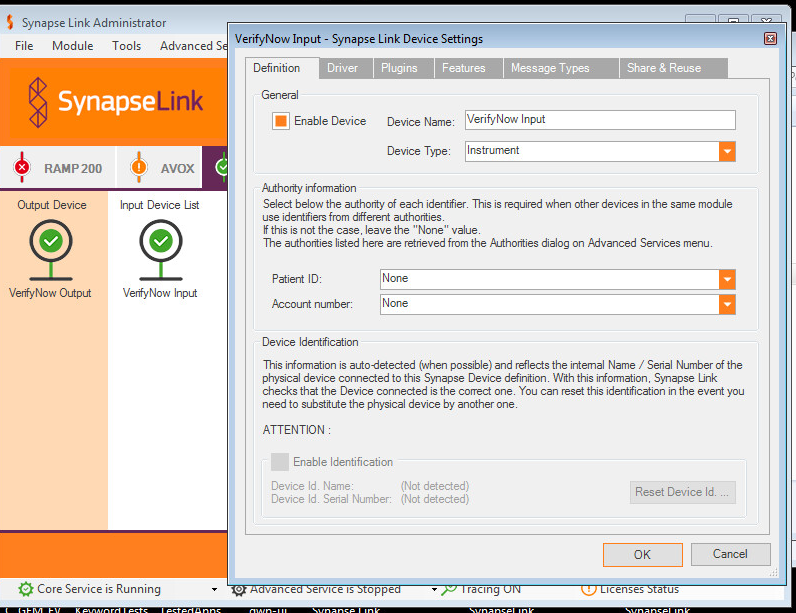


Configure the driver in GWP according to the same set in the output tab of the Synapse Link module

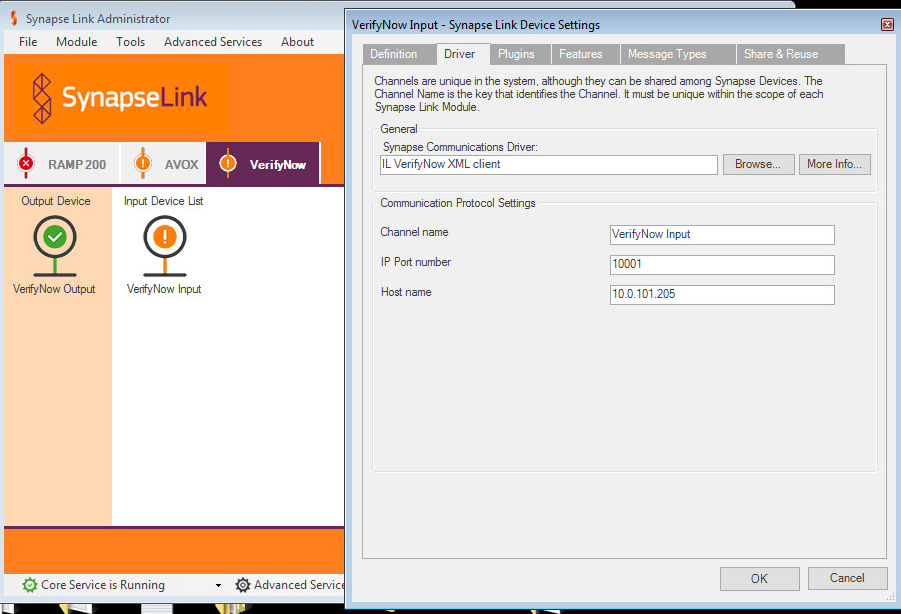


Configure Input selecting the driver from IL / VerifyNow / XML / client

Simulator/VerifyNow instrument --- is connected to - Synapse Link Input

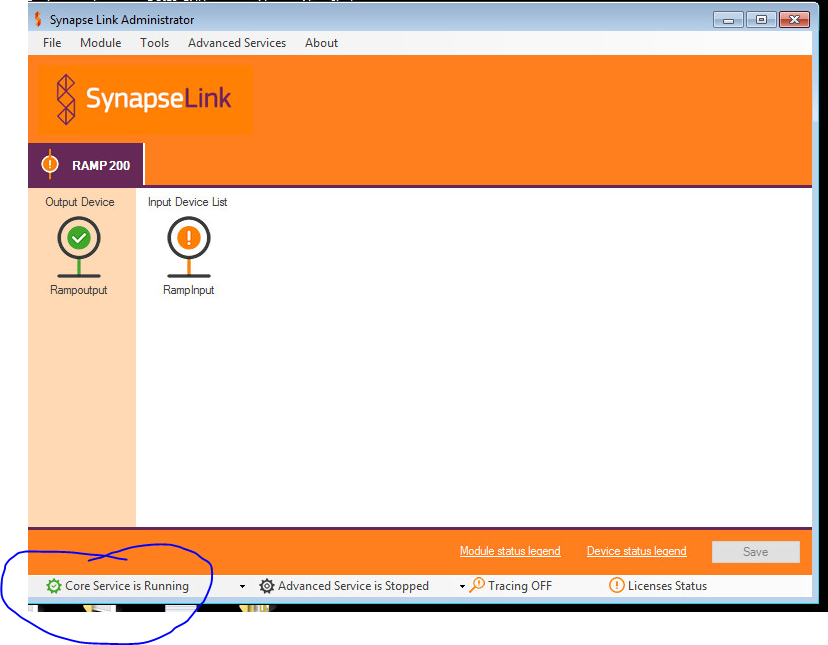


At the input-driver tab, use the IL / VerifyNow / XML / client and set the port & IP of the Lantronix device (use the LANtronix or another converter configuration)



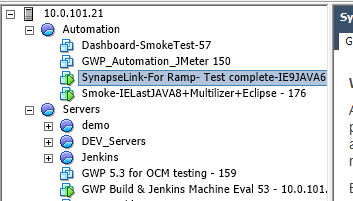
In this case, 10001 and 10.0.101.205 are port and IP of the LANTRONIX CONVERTER serial installed with the instrument

After this configuration, SAVE IT and make sure that the start core service is running.



Finally, add a License if needed.

NOTE: This configuration has been set up in the server 10.0.101.178, inside the VSPHERE 10.0.101.21



IMPORTANT NOTE: Due to the VerifyNow proprietary protocol there is NO option to connect a VerifyNow Simulator with Synapse link

# **GEMweb Plus 500 Connectivity Guide – VerifyNow**

**Purpose**

This appendix section offers an installation guide for understanding how to connect a VerifyNow instrument to GWP.

**Pre-installation procedure**

The following applications and frameworks must be installed first:

|  |  |
| --- | --- |
| Licenses | Active |
| GEM Web Plus 500 or Higher activated with 10 or 32 analyzers license | **✓** |
| OCM License installed | **Not required** |
| Specific driver License installed | **Not required** |

Connectivity of VerifyNow to the network requires use of the Lantronix adapter.

Synapse link is required to connect to GEMweb Plus. The analyzer connects to Synapse and Synapse connects to GEMweb Plus.

See preceding section of this Appendix for instructions.

Synapse configuration instructions are included in this document in the last chapter. However, for the most updated installation and configuration instructions, please refer to Werfen Clinical Software

NOTE:

QC samples from the VerifyNow cannot be displayed in the Quality Management section of GEMweb Plus, due to missing information in the QC test result data received from the device.

## **VerifyNow Installation Procedure**

Driver requirements

The driver is preinstalled in GEMweb Plus V5.5.0 and above. Set here the Instrument and driver version used.

|  |  |  |
| --- | --- | --- |
| Instrument version | Driver Name | Driver Version |
| VerifyNow | VerifyNow SYNAPSE | 1.15 |

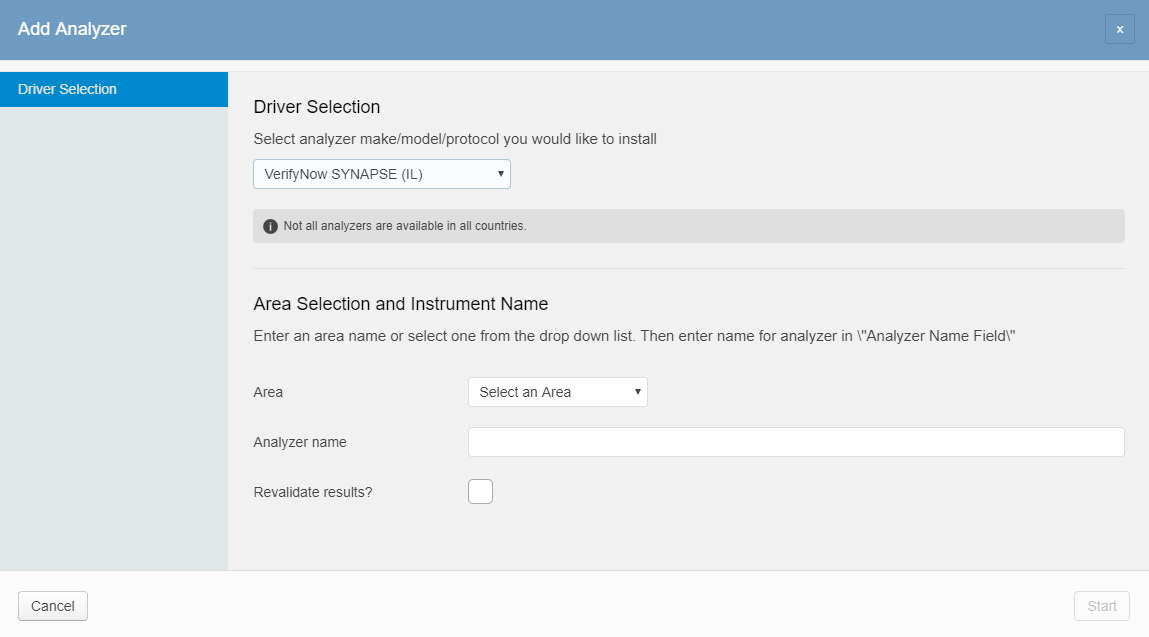
**Configuration of the VerifyNow Network Adapter**

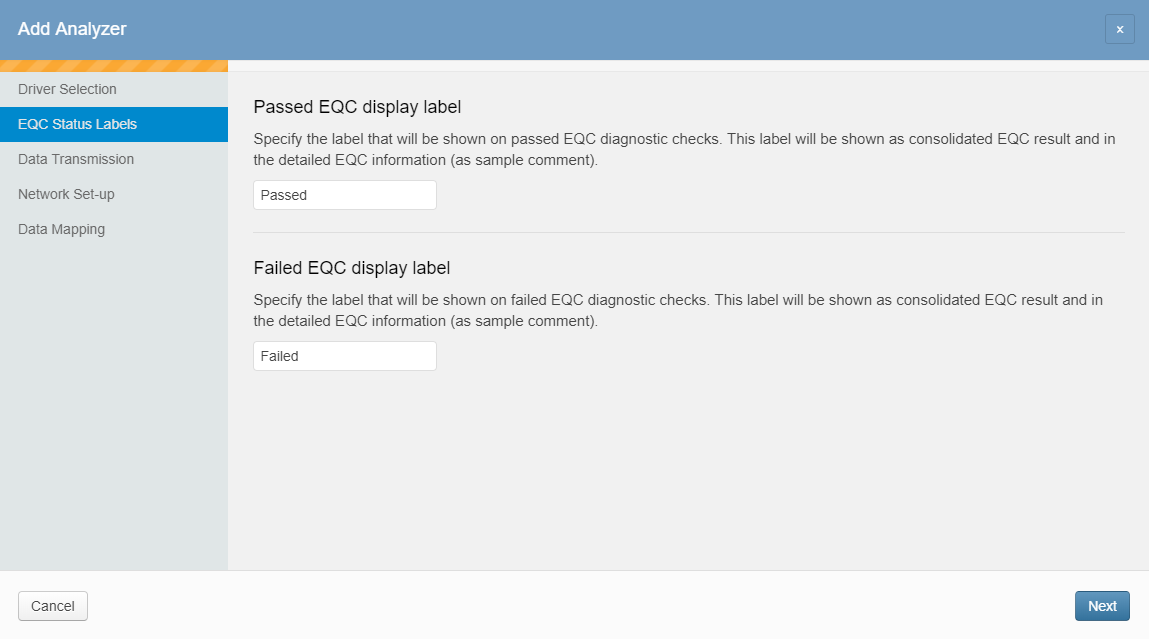
Refer to **Section 3** of this document for instructions.

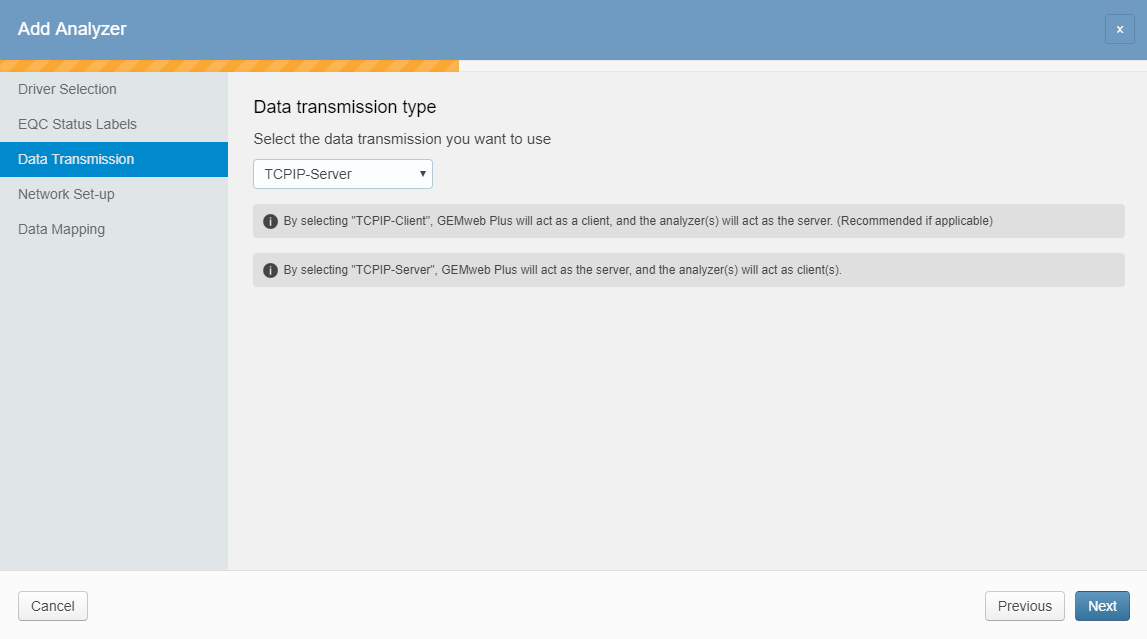
### **GWP Configuration**

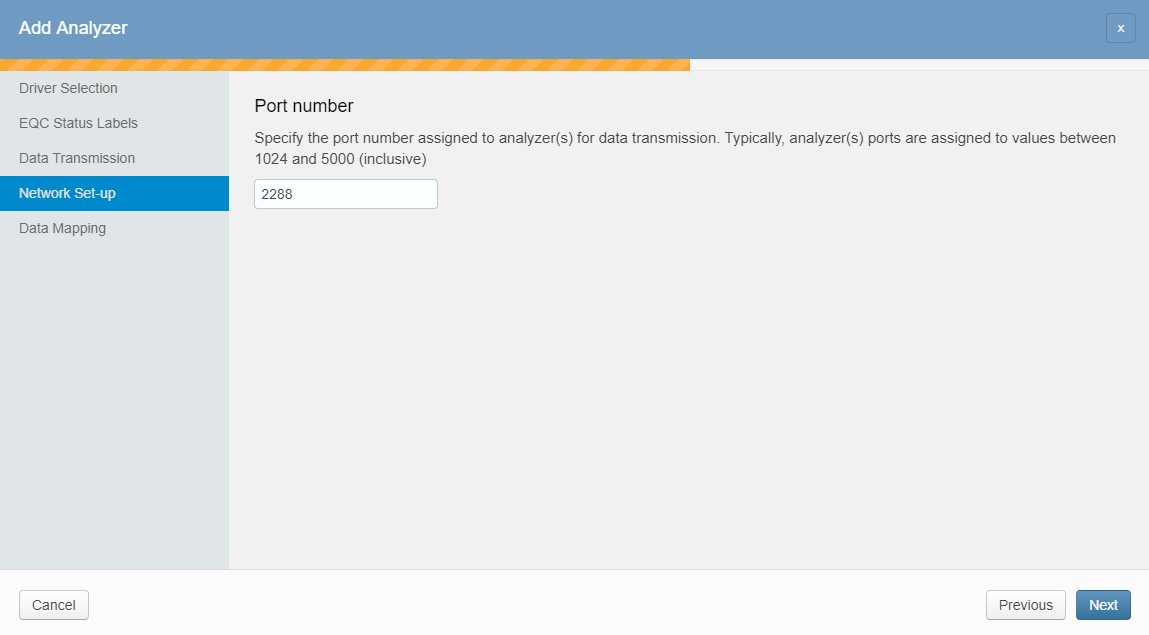
**Analyzer Details Instrument configuration**

Make sure the instrument is correctly added in GWP see screenshots for help:



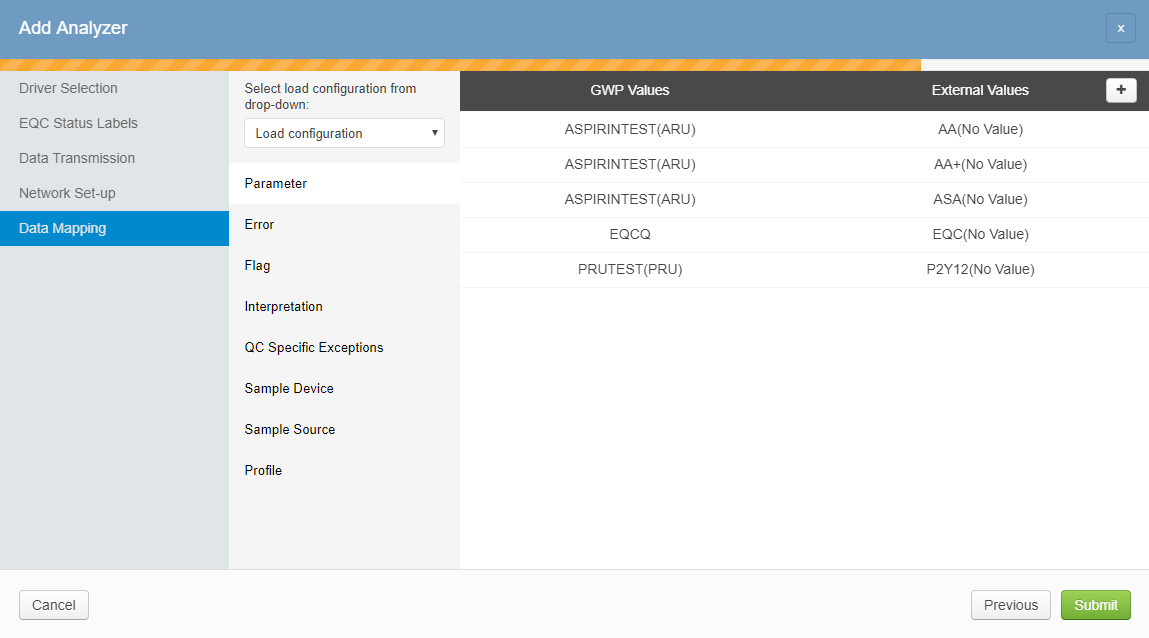






### **Custom Mapping**

Make sure to map the GWP Values with the External Values correctly.



# **Sending Quality Control data to GWP**

VerifyNow sends in an automatic way to GWP any Sample (Patient or QC) run on the instrument. No further actions need to be performed to get the data.

As there is some missing information in the data received in the QC Samples sent by VerifyNow (instrument QC samples do not contain the level value field), the QC Management could not be used in GWP. QC Samples are received and show in GWP as they were sent by the instrument, but the absence of the level value implies that no Lots would be managed in GWP.

# **Host Connection Information**

**HIS/LIS connectivity**

If the default data mapping is used, and host connection using the ASTM GEMweb Plus Mode is created, the analyzer will send the results to the LIS/HIS through GEMweb Plus using the format and codification included in this chapter.

**Example of Patient Sample**

H|@^\|||GEMweb Plus^1.0^GWP Server^938023^GWP.5.4.0.54.0.31.0|||||||P|LIS2-A|20181004113224P|1||0000000023||^^|||  
O|1|^|b98a24d9-d636-400f-8e64-5dcf21ada925||||^||^||||||BLD^^U|||||||20181004094941|||F  
C|1|P|FIELD^Source Instrument^7002|G  
R|1|Aspirin Test^^IL^Aspirin Test|412|ARU||||F||^^||20181004094911|VerifyNow 110^ICU^SynapseLinkforVerifyNow ASTM^^^  
L|1|N

**Example of LQC Sample**

H|@^\|||GEMweb Plus^1.0^GWP Server^938023^GWP.5.4.0.54.0.31.0|||||||P|LIS2-A|20181004113339 P|1|  
O|1|1^--^^^|ea36b6e0-d42b-493f-abc8-34d1882011f6||||^||^||||||QC^EQC|||||||20181001174315|||F  
C|1|P|FIELD^Source Instrument^7002|G  
R|1|Aspirin Test^^IL^Aspirin Test|350|ARU||N||F||^^||20180831035420|VerifyNow 110^ICU^SynapseLinkforVerifyNow ASTM^^^  
L|1|N

# **Instrument Mapping**

Default Mapping

By default, VerifyNow mapping, Errors and QC Specific Exceptions will be interpreted in GWP as:

PARAMETERS

|  |  |
| --- | --- |
| GWP Values | External Values |
| ASPIRINTEST(ARU) | AA(No Value) |
| EQCQ | EQC(No Value) |
| PRUTEST(PRU) | PRUTest(No Value) |

QC Specific Exceptions

|  |  |
| --- | --- |
| GWP Values | External Values |
| 40- QC Failed (A) | A |

SAMPLE SOURCE

|  |  |
| --- | --- |
| GWP Values | External Values |
| Blood | BLD |

## **Custom Mapping**

Note down the data mapping configuration if different than the default one. Refer to section 2.3 for more information about the default mapping.

Was the default mapping used?  YES  NO

If answer to the previous question is NO, please fill in the following tables for your own future references

PARAMETERS

|  |  |
| --- | --- |
| GWP Values | External Values |
|  |  |
|  |  |

ERRORS

|  |  |
| --- | --- |
| GWP Values | External Values |
|  |  |
|  |  |

FLAGS

|  |  |
| --- | --- |
| GWP Values | External Values |
|  |  |
|  |  |

INTERPRETATION

|  |  |
| --- | --- |
| GWP Values | External Values |
|  |  |
|  |  |

QC Specific Exceptions

|  |  |
| --- | --- |
| GWP Values | External Values |
|  |  |
|  |  |

SAMPLE DEVICE

|  |  |
| --- | --- |
| GWP Values | External Values |
|  |  |
|  |  |

SAMPLE SOURCE

|  |  |
| --- | --- |
| GWP Values | External Values |
|  |  |
|  |  |