Programmer's Guide

Wireless Protocol Suite

Automation Server Protocol

Table of Contents

Purpose

This document describes a protocol that provides remote control of the Wireless Protocol Suite, hereafter referred to as “the analyzer” Utilizing the protocol, a remote client can bypass the Microsoft Windows user interface and interact directly with the analyzer.

This document is intended for people with knowledge of TCP socket communications who want to automate the process of testing Bluetooth® and *Bluetooth* low energy devices using the analyzer application. The test process automation is achieved by writing a client application that communicates over a TCP network socket connection with the Automation Server component.

Hardware & Software Requirements and Typical System Topology

- For any hardware and software requirements of the analyzer, refer to the user manual for these requirements.
- TCP/IP network

The typical system topology includes one (1) server machine running one (1) or more instances of the analyzer application and one (1) instance of the Automation Server (reference the analyzer documentation for Windows Operating Systems requirements) and one (1) or more client machines running one (1) or more client applications. The server, analyzer and client applications may all be run from the same machine.

The client application is platform independent. The only requirement for the client application is that a TCP socket connection can be established with the server machine in order to communicate using the Automation Server ASCII protocol described later in this document.

The server machine must be in a logged-in state with a user profile that has the security privileges to launch the Automation Server. The Automation Server may be launched from the Desktop folder created as part of the Wireless Protocol Suite installation:

1. In Windows Explorer, browse to the Desktop folder and locate the Wireless Protocol Suite folder for the installation of interest.
2. Open folder “Help and Tools”
3. Open folder “Automation Server”
4. Double-click the “Start Server” shortcut.

Once the Automation Server launches and initializes, it continuously polls for socket connection requests from the clients. The known IP address of the server machine and available port(s) on the server machine must be provided for the client to establish a socket connection with the server machine.

Analyzer Automation Arguments

The following arguments can be edited using the FTSAutoServer.exe.config file, which must be in the same directory as the FTSAutoServer.exe.

IPAddr

IPAddr is the IP address for the server machine where the analyzer and the Automation Server are running. This address format is the same as the standard IPv4 address notation.

Port

Port identifies any available port on the server machine that the client can use to communicate with the server. If more than one client connection is to be established, this port will be the first available port used. For instance, if port 6000 is configured as the starting port and 10 connections are to be made, then ports 6000-6009 must be available in order to successfully establish all the client connections.

ClientConnections

ClientConnections is the maximum number of client connections that can be established with the Automation Server.

MaxListSize

MaxListSize defines the maximum number of status responses that can be written to the Automation Server window before the oldest responses are deleted.

SleepTime

SleepTime is the number of seconds a given client communication thread sleeps before wakening to send another command. The shorter the sleep period, the faster the client communication can be made.

Command Responses

Command response will be returned for each command upon delivery to the automation server. Depending on the command, it may or may not be delivered to the analyzer. The format for the response is: “<command>;<status>;<timestamp>;[<state>][<reason>]” where parameters in brackets are optional, depending on the command and command processing:

Parameter	Description
command	The command the response refers to – generally upper-case ASCII but checks against a given value should be performed in a case-insensitive manner.
status	The status of the command – “SUCCEEDED” means the command was passed to the analyzer; “FAILED” means the command was not passed to the analyzer or the analyzer could not process the command.
timestamp	Timestamp for when the response was processed by the automation server in the form “Timestamp=<date and time in current culture formatting>”
state (optional)	Some commands may return a state parameter in the form “State=<state>” where the state value will be command dependent.
reason (optional)	Some commands may return a reason parameter in the form “Reason=<reason>” where the reason value will be command dependent.

In general:

- The Automation Server does not block waiting on a response to any command. The client is responsible for the block based on whether the response of a command is required before the next command can be sent.
- A failure notification is always sent if a command is not delivered to the analyzer successfully.
- A success notification is always sent if a command is delivered to the analyzer successfully. The word “delivered” is important. A success notification does not mean that the command was completed successfully, only that it was delivered to the analyzer successfully and that the action was started. The successful delivery of a command to the analyzer does not guarantee a successful execution of the command.
- Checks for command returns should be made in a case-insensitive manner.

Commands

Multiple clients can connect to the Automation Server. Once a connection is established with the server, the client communicates with the server using commands that are delimited ASCII strings. The following commands are supported by the Automation Server. Upon receipt of these commands, the Automation Server packages these commands to send to the analyzer application. The general command format is:

“<command>;[parameters]”

- Commands are case-insensitive.
- Unless otherwise noted, all command parameters are required.
- Unless parameters are of the form “key=value”, the parameter order must be retained.
- The optional parameter “datasource=<value>” where “value” represents the targeted datasource index may be added. Otherwise, the default datasource index will be 0.

Start FTS

The Start FTS command tells the Automation Server to launch the analyzer application. The Automation Server can only communicate with one instance of the analyzer at a time: If an instance has already been started, a FAILED response will be returned. A SUCCEEDED response is returned from the Automation Server if the application has been successfully launched. While the application is initializing, the automation server application will poll the analyzer initialization state (see “Is Initialized below) every 3 seconds until the application is initialized or until 2 minutes has elapsed.

Failure to launch CPAS results in a FAILED notification response from the server. The failure notification response may or may not be accompanied by a reason parameter. For instance, if a failure occurs during initialization of the analyzer application, no reason information may be available to the Automation server. Therefore, the timeout will occur based on the SleepTime setting configured in the application configuration file explained above. If no SleepTime value is specified, then a default of 10 seconds will automatically be used by the Automation server.

Command format: “Start FTS;<fts.exe directory>;<personality key>”

Parameter	Description
fts.exe directory	The directory that contains the instance of fts.exe to launch. This parameter must refer to a valid directory or be “none” to launch the fts.exe instance that is co-located with the running Automation Server.
personality key	Key to the personality used to form the fts.exe command line where valid

Parameter	Description
	values are presented in the table below.

The table below presents valid personality keys; they are not case-sensitive. In general, if the hardware specified by the personality key is not present, the analyzer will launch in view mode and subsequent datasource commands will fail.

Personality Key	Description
SODERA	This mode requires that a Sodera be attached to the host computer.
SODERA_80211_Coex	This mode requires that a Sodera and 802.11 be attached to the host computer with the sync cable properly connected between them.
SODERALE	This mode requires that a Sodera LE be attached to the host computer.
X240	This mode requires that a X240 be attached to the host computer.
DoubleX240	This mode requires that two X240s be attached to the host computer. Each should be configured for capture of a different technology.
TripleX240	This mode requires that three X240s be attached to the host computer. Each should be configured for capture of a different technology.
80211	This mode requires that a 802.11 be attached to the host computer.
TWOWIFI	This mode requires that a two 802.11s be attached to the host computer with the sync cable properly connected between them.
VIEW	This mode launches the analyzer console with no file open or live mode selected.

Note: For configurations that include one or more X240 devices, the capture technology should be configured by using the appropriate datasource prior to starting capture via automation. Configuration of capture technology and initiating firmware update are not supported via the automation interface.

Response “START FTS;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Is Initialized

The Is Initialized command checks the initialization state of the analyzer.

Command format: “Is Initialized”

Response “IS INITIALIZED;<status>;<timestamp>;<reason>” where a reason will be supplied for SUCCEEDED and FAILED status.

The reason value will be

- “no” – The analyzer is not ready to receive further commands.
- “yes” – The analyzer is ready to receive further commands.

Stop FTS

The Stop FTS command tells the Automation Server to shut down the analyzer application (if one is running) that was previously launched by the Automation Server. Only an instance of the analyzer application launched by the Automation Server can be shut down using this command. If no analyzer application was successfully launched, then a failure notification response will be sent from

the server. If the attempt to shut down the analyzer application times out, then a failure notification response is sent from the server.

Command format: “Stop FTS”

Response “STOP FTS;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Get Datasource HWND

The Get Datasource HWND command is used to get the handles for all datasources launched by the analyzer for a session. This command does not need to be explicitly issued: If this command has not been issued after the analyzer is started, it will automatically be called for any command that requires it.

Command format: “Get Datasource HWND”

Response “GET DATASOURCE HWND;<status>;[HWND list;]<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status and HWND list will only be supplied for SUCCEEDED status.

HWND list will be a comma-delimited list of HWND values for each datasource in order of datasource ID in the format of “HWND<id>=<hwnd>” where “id” is the datasource ID and “hwnd” is the hexadecimal representation of the datasource main window HWND.

Config Settings

This command instructs the analyzer application to initialize the IO settings for a capture device. Important ideas to remember about Config Settings include:

- The parameters will be parsed for correctness.
- The client can begin capturing or sniffing without setting the IO. The analyzer will use the previously configured values from the datasource INI file or the system defaulted values.
- Valid IO settings are dependent upon the devices being tested.
- For any configuration that you send, you must have a data source key that matches the type of I/O settings in the application.
- Some devices do not support Extension channels or 5 GHz channels. The configuration will fail if the settings are not supported by the device.

Valid DataSource Keys are

- Sodera
- X240
- 802.11

The config settings command format for DataSource Keys “Sodera”, “802.11” and “X240” for all Personality Keys ***except*** “DoubleX240” and “TripleX240” is: “Config Settings;<type>;<datasource key>;<config settings>”

The config settings command format for DataSource Key “X240” using Personality Key “DoubleX240” and “TripleX240” is: “Config Settings;<type>;<datasource key>;<serial number key>;<config settings>”

Where type is

- “IOParameters” - All DataSource Keys
- “HWParameters” – 802.11

Serial number key is in the form of “serial_number=serial number” where the serial number value is serial number of the X240 connected to the target datasource.

Response “CONFIG SETTINGS;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

IOParameters – Sodera and X240

The following IO parameters are valid for Sodera and X240 unless otherwise noted.

analyze

This command is an optional command to allow for configuration of options. If not included, the last option selections will be applied. The command format is:

“analyze=inquiryprocess-<value>|pagingnoconn--<value>|nullsandpolls--<value>|emptyle--<value>|anonymousadv--<value>|meshadv--<value>”

The acceptable values are:

- “on” – Selection checked.
- “off” – Selection not checked

The command must be sent with all parameters. The parameters are:

- inquiryprocess – Control selection of “Analyze Inquiry Process Packets”
- pagingnoconn – Control selection of “Analyze All Paging Without Connection”
- nullsandpolls – Control selection of “Analyze NULL and POLL Packets”
- emptyle – Control selection of “Analyze LE Empty Packets”
- anonymousady – Control selection of “Analyze Anonymous/Unknown Adv. Packets”
- meshadv – Control selection of “Analyze Mesh Advertising Packets”

Example – “analyze=inquiryprocess-off|pagingnoconn-off|nullsandpolls-off|emptyle-on|anonymousadv-on|meshadv-off”

btdevice

This command is an optional command to allow for configuration of Bluetooth device information, though it is most often used for **Bluetooth low energy** devices. ***This command will result in the addition of a Bluetooth device address to the datasource device store as well as to the automation handler device list.*** The command format is:

“btdevice=<value>, <value>, … <value>”

The values are in the form “<address>[type=<type>] [irk=<irk>]” where the parameters are:

- “address” – Device address – must be in the form “0xaabbccddeeff”
- “type” – Optional type where value of “random” is an indication the address is a low energy random address.
- “irk” – Valid IRK for the specified address. If this parameter is specified but type is not specified, or type is specified as something other than “random”, the command will be rejected as invalid.

Example – “btdevice=0x001122334455, 0x66778899aabb, 0xc01122334455 type=random”

capturetechnology
This command is an optional command to allow for configuration of the Wireless capture technology capture options. If not included, the last option selections will be applied. ***This command parameter is not allowed for Datasource Key “X240”*** The command format is:

“capturetechnology=bredr-<value>|le-<value>|2m-<value>”

The acceptable values are:

- “on” – Selection checked.
- “off” – Selection not checked

The command must be sent with all parameters. The parameters are:

- bredr – Control selection of “BR/EDR” where those packets will be discarded if unchecked.
- le – Control selection of “LE” where all low energy packets will be discarded if unchecked.
- 2m – Control selection of “2M LE” where 2 Mbps low energy packets will be discarded if unchecked.

Example – “capturetechnology=bredr-off|le-on|2m-on”

Note that the validity of a capture technology selection will depend on the capture hardware capability and license.

ledevice
This command is an optional command to allow for configuration of Bluetooth device information, though it is most often used for **Bluetooth low energy** devices. ***This command will result in the addition of a Bluetooth device address to the datasource device store as well as to the automation handler device list.*** The command format is:

“ledevice=<value>”

The value is a valid Bluetooth device address, with or without leading “0x”, but in hexadecimal format.

Example – “ledevice=0x001122334455”

linkkey
This command is an optional command to allow for configuration of **Bluetooth Classic** link key information. This command must be issued ***after*** two Bluetooth device commands have been successfully issued (“btdevice”, “ledevice”, “master” or “slave”) such that there are at least two valid Bluetooth device addresses on the automation handler device list. This command cannot be issued in combination with “longtermkey” nor “pincode” The command format is:

“linkkey=<value>”

The value is a valid Bluetooth link key with leading “0x”, consisting of 32 hexadecimal characters that will result in a 16-byte link key.

Example – “linkkey=0x00112233445566778899aabbccddeeff”

longtermkey
This command is an optional command to allow for configuration of **Bluetooth low energy** long term key information. This command must be issued ***after*** a Bluetooth device command has been successfully issued (“btdevice”, “ledevice”, “master” or “slave”) such that there is at least one valid Bluetooth device address on the automation handler device list. This command cannot be issued in combination with “linkkey” nor “pincode” The command format is:

“longtermkey=<value>”

The value is a valid Bluetooth long term key with leading “0x”, consisting of 32 hexadecimal characters that will result in a 16-byte long term key.

Example – “longtermkey =0x00112233445566778899aabbccddeeff”

master
This command is an optional command to allow for configuration of Bluetooth device information, though it is most often used for **Bluetooth Classic** devices. ***This command will result in the addition of a Bluetooth device address to the datasource device store as well as to the automation handler device list.*** The command format is:

“master=<value>”

The value is a valid Bluetooth device address, with or without leading “0x”, but in hexadecimal format.

Example – “master=0x001122334455”

pincode
This command is an optional command to allow for configuration of **Bluetooth Classic** PIN code information. This command cannot be issued in combination with “linkkey” nor “longtermkey” The command format is:

“pincode=<value>”

The value is a passphrase consisting of a maximum of 16 ASCII characters from the set "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

Example – “pincode=GeorgePBurdell”

pod
This command is an optional command to allow for configuration of the Wired capture technology capture options on Sodera and X240. This command can only be issued if the license supports wired devices. If not included, the last option selections will be applied. The command format is:

“pod=<value>”

The value is the pod number to enable, and there are two pods.

Examples – “pod=1”, “pod=2”, pod=”1,2”

slave
This command is an optional command to allow for configuration of Bluetooth device information, though it is most often used for **Bluetooth Classic** devices. ***This command will result in the addition of a Bluetooth device address to the datasource device store as well as to the automation handler device list.*** The command format is:

“slave=<value>”

The value is a valid Bluetooth device address, with or without leading “0x”, but in hexadecimal format.

Example – “slave=0x001122334455”

Example – Sodera or X240
Example configurations for a Sodera or X240:

“CONFIG SETTINGS;IOParameters;Sodera;btdevice=0x98460ab68b10,0x20ee285bb1e6;longtermkey=0x9308D18E1C87E06BFD2FCDB0CF334E91”

“CONFIG SETTINGS;IOParameters;X240;master=0x98460ab68b10;slave=0x20ee285bb1e6;linkkey=0x9308D18E1C87E06BFD2FCDB0CF334E91”

IOParameters – 802.11 and X240

The following IO parameters are valid for 802.11 and X240 unless otherwise noted.

channel
This command is an optional command to allow for configuration of the capture channel, which will also set the capture frequency. The command format is:

“channel= <value>”

extensionchannel
This command is only valid for the datsource key “802.11” This command is an optional command to allow for configuration of a 40 MHz wide channel when capturing traffic in an 802.11n environment. The “direction” of the change, increase or decrease, is channel dependent. If not included, the last option selections will be applied. This parameter is only valid if the “channel” or “frequency” parameter is included in the command. The command format is:

“extensionchannel= <value>”

Example – “extensionchannel=+1”

channelwidth
This command is only valid for the datsource key “X240” This command is an optional command to allow for configuration of the channel width, which may result in a 20, 40 or 80 MHz wide channel when capturing traffic in an 802.11n environment. The “direction” of the change, increase or decrease, and allowable resulting width is channel dependent. If not included, the last option selections will be applied. This parameter is only valid if the “channel” or “frequency” parameter is included in the command. The command format is:

“channelwidth= <value>”

Example – “channelwidth=-40”

frequency
This command is an optional command to allow for configuration of the capture frequency in MHz, which will also set the capture channel. The command format is:

“frequency= <value>”

Example – “frequency=2457”

The following table lists acceptable combinations of channel/frequency, extension channel and channel width.

Channel	Frequency (MHz)	ExtensionChannel (20 MHz) (802.11 only)	Channel Width (X240 only)
1	2412	+1	20, 40
2	2417	+1	20, 40
3	2422	+1	20, 40
4	2427	+1	20, 40
5	2432	-1, +1	-40, 20, 40
6	2437	-1, +1	-40, 20, 40
7	2442	-1, +1	-40, 20, 40
8	2447	-1, +1	-40, 20, 40
9	2452	-1, +1	-40, 20, 40
10	2457	-1	-40, 20
11	2462	-1	-40, 20
12	2467	-1	-40, 20
13	2472	-1	-40, 20
36	5180	+1	20, 40, 80
40	5200	-1	20, 40, 80
44	5220	+1	20, 40, 80
48	5240	-1	20, 40, 80
52	5260	+1	20, 40, 80
56	5280	-1	20, 40, 80
60	5300	+1	20, 40, 800
64	5320	-1	20, 40, 80
100	5500	+1	20, 40, 80
104	5520	-1	20, 40, 80
108	5540	+1	20, 40, 80
112	5560	-1	20, 40, 80
116	5580	+1	20, 40, 80
120	5600	-1	20, 40, 80
124	5620	+1	20, 40, 80
128	5640	-1	20, 40, 80
132	5660	+1	20, 40, 80
136	5680	-1	20, 40, 80
140	5700	0	20, 40, 80
149	5745	+1	20, 40, 80
153	5765	-1	20, 40, 80
157	5785	+1	20, 40, 80
161	5805	-1	20, 40, 80
165	5825	-1	20

fcsfilter
This command is not supported.

capturetype
This command is not supported.

enablewepdecryption
This command is not supported.

Example – 802.11
Example configuration for an 802.11:

"CONFIG SETTINGS; IOParameters;80211;Channel=2"

Example – X240
Example configuration for an X240:

"CONFIG SETTINGS; IOParameters;X240;channel=7;channel width=40"

HWParameters – 802.11
The following HW parameters are valid for 802.11.

devindex
This command is an optional command to allow for selection of the 802.11 device the command is intended for. The command format is:

"devindex= <value>"

The value is the 0-based index of the device and cannot be negative and cannot be equal-to or greater than the number of devices.

Example – "devindex =1"

Example – 802.11
Example configuration for 802.11 hardware parameters:

"CONFIG SETTINGS; HWParameters;80211;devindex=1"

Start Capture
The Start Capture command starts the capture of data in the analyzer.

Command format: "Start Capture"

Response "START CAPTURE;<status>;<timestamp>;[<reason>]" where a reason will only be supplied for FAILED status.

Stop Capture
The Stop Capture command stops the capture of data in the analyzer.

Command format: "Stop Capture"

Response "STOP CAPTURE;<status>;<timestamp>;[<reason>]" where a reason will only be supplied for FAILED status.

Save Capture

The Save Capture command saves the captured data after a capture was started, then stopped.

Command format: “Save Capture;<capture file path>”

Response “SAVE CAPTURE;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

It should be noted that Sodera and X240 have the capacity to buffer frames for processing and it might take some time to clear those buffers. In order to avoid losing those frames or not including them in a capture file, the sequence of commands recommended to save all frames captured would be

- Stop Record
- Is Analyze Complete – Repeat until status indicating completion has been received (see Is Analyze Complete).
- Stop Analyze
- Query State – Repeat until successful status with CAPTURE ACTIVE NO DATA or CAPTURE STOPPED reason.
- Is Processing Complete – Repeat until status indicating completion has been received (see Is Processing Complete).
- Save Capture – Wait until status has been reported.

Clear

The Clear command clears the capture buffer. The following is an example of the command string for this command.

Command format: “Clear”

Response “CLEAR;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

It should be noted that Sodera and X240 have the capacity to buffer frames for processing and it might take some time to clear those buffers. In order to avoid a potential crash of the analyzer when frames are cleared, the sequence of commands recommended to clear would be

- Stop Record
- Is Analyze Complete – Repeat until status indicating completion has been received (see Is Analyze Complete).
- Stop Analyze
- Query State – Repeat until successful status with CAPTURE ACTIVE NO DATA or CAPTURE STOPPED reason.
- Is Processing Complete – Repeat until status indicating completion has been received (see Is Processing Complete).
- Save Capture – If desired. Wait until status has been reported.
- Clear

Set Resolving List

The Set Resolving List command is for creation of a device-address resolving list when capturing data from BR/EDR devices. If there is the potential for capture of data from devices with addresses that

have the same UAP and LAP as those already in the device database, the resolving list may be used to give an address priority.

Command format: “Set Resolving List;<address list>”

Where “address list” is

- “BD_ADDR0,BD_ADDR1,…,BD_ADDRn” – a comma-separated list of valid BD_ADDRs in the form “0x001122334455”
- “” – an empty list to clear the resolving list.

Response “SET RESOLVING LIST;<status>;<timestamp>;<reason>” where a reason will be supplied for SUCCEEDED and FAILED status.

The reason value will be

- "resolving_list_set=clear" – The resolving list was successfully cleared.
- “resolving_list_set=all| BD_ADDR0|BD_ADDR1|…|BD_ADDRn” – All requested BD_ADDRs were added to the list.
- “resolving_list_set=some| BD_ADDR0|BD_ADDR1|…|BD_ADDRn|resolving_list_error=error0|error1|…|errorn” – Some requested BD_ADDRs were added to the list; some BD_ADDRs could not be added to the list with the accompanying errors.
- “resolving_list_set=none|resolving_list_error=error0|error1|…|errorn” – No BD_ADDRs were added to the resolving list.

Note – the resolving list is indexed on the UAP and LAP portions of the BD_ADDR and the keys are unique. That means that the resolving list can only hold one address per value of UAP and LAP. If the specified address list contains BD_ADDRs such that two or more of them have the same UAP and LAP, they will be added to the list such that only the last one with those values will be on the list. However, the reason value will still show that the earlier valid BD_ADDRs in the request were added and the error list will show that that BD_ADDR was replaced.

Start Sniffing

The Start Sniffing command starts the capture and processing of data. This command is equivalent to “Start Record” followed by “Start Analyze” for Sodera and X240 – please refer to the sections for those commands. It is equivalent to “Start Capture” for 802.11.

Command format: “Start Sniffing”

Response “START SNIFFING;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Stop Sniffing

The Stop Sniffing command stops the capture of data. This command is equivalent to “Stop Record” followed by “Stop Analyze” for Sodera and X240 - please refer to the sections for those commands. The following is an example of the command string for this command.

Command format: “Stop Sniffing[;maxwait=<milliseconds>]”

The “maxwait” parameter is for specification of the maximum amount of time to wait on analyze to complete. “milliseconds” represents an unsigned 32-bit decimal value. If this parameter is omitted, the default wait time is 120000 milliseconds.

Response “STOP SNIFFING;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Start Record

The Start Record command starts the capture of data by all configured capture hardware.

Command format: “Start Record”

Response “START RECORD;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Stop Record

The Stop Record command stops the capture of data by all configured capture hardware.

Command format: “Stop Record”

Response “STOP RECORD;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Start Analyze

The Start Analyze command starts the processing of data by the analyzer.

Command format: “Start Analyze”

Response “START ANALYZE;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Stop Analyze

The Stop Analyze command stops the processing of data by the analyzer. It should be noted that Sodera and X240 have the capacity to buffer frames for processing and it might take some time to clear those buffers. In order to avoid losing those frames or not including them in a capture file, the sequence of commands recommended to stop capture and processing of data would be

- Stop Record
- Is Analyze Complete – Repeat until status indicating completion has been received (see Is Analyze Complete).
- Stop Analyze

Command format: “Stop Analyze”

Response “STOP ANALYZE;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Is Analyze Complete

The Is Analyze Complete command checks the state of the analyze process.

Command format: “Is Analyze Complete”

Response “IS ANALYZE COMPLETE;<status>;<timestamp>;<reason>”

For SUCCEEDED status, reason will be analyze_complete=<value> where value is

- yes – Analyze is complete and all data have been sent to the analyzer.
- no – Analyze is not complete – data are still queued to be sent to the analyzer or record has not been stopped.

Is Processing Complete

The Is Processing Complete command checks the state of the frame compilation process.

Command format: “Is Processing Complete”

Response “IS PROCESSING COMPLETE ;<status>;<timestamp>;<reason>”

For SUCCEEDED status, reason will be

- true – Processing is complete, and all frames have been compiled.
- false – Processing is not complete – frames are still being compiled.

Query State

The Query State command checks the state of the analyzer.

Command format: “Query State”

Response “QUERY STATE;<status>;<timestamp>;<reason>”

For SUCCEEDED status, reason will be

- IDLE – The analyzer is in an idle state.
- CAPTURE ACTIVE WITH DATA – Capture is active and not paused.
- CAPTURE ACTIVE NO DATA – Capture is active but paused.
- CAPTURE STOPPED – Capture is not active.

Open Capture File

The Open Capture File command is a request for the analyzer to open the specified capture file.

Command format: “Open Capture File;<capture file path>[;notify=<value>]”

The “notify” parameter is used to signal a desire to delay the command response until file processing has completed.

The values are

- 0 – Do not wait to for open to complete before sending response.
- 1 – Wait until open is complete to send response.

Example – “Open Capture File;F:\My Capture Files\CaptureFile.cfa;notify=1”

Response “OPEN CAPTURE FILE;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

By default, the successful notification response is sent immediately after the capture file is opened successfully. The frame compiler may run for several minutes after a large capture file is opened. If an HTML Export command or Export command is sent before the frame compiler completes, the exported file will not contain all the frames.

If it is important to wait for the frame compiler to complete before proceeding the optional “Notify” parameter with a value of “1” can be used to delay sending the success notification.

If transitioning from a live-capture mode to file-view mode and the command fails, the datasource may be closed.

Add Bookmark

The Add Bookmark command adds a bookmark to a frame in an open capture file. If a bookmark already exists for that frame, the command will fail – see Modify Bookmark.

Command format: “Add Bookmark;string=<bookmark>;frame=<frame>”

Where “bookmark” is any string and “frame” is the a 1-based frame that exists in the file.

Example – “Add Bookmark;string=Frame 1 Bookmark;frame=1”

Response “ADD BOOKMARK;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Modify Bookmark

The Modify Bookmark command modifies a bookmark for a frame in an open capture file. If the frame does not have a bookmark, one will be added. If the modification is an empty string, the existing bookmark will be deleted.

Command format: “Modify Bookmark;string=<bookmark>;frame=<frame>”

Where “bookmark” is any string and “frame” is the a 1-based frame that exists in the file.

Example – “Modify Bookmark;string=New Frame 1 Bookmark;frame=1”

Response “MODIFY BOOKMARK;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Close Capture File

The Close Capture File command closes the capture file.

Command format: “Close Capture File”

Example – “Close Capture File”

Response “CLOSE CAPTURE FILE ;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Go Live
The Go Live command is not supported.

Exit Live Mode
The Exit Live Mode command is not supported.

Sync Status
The Sync Status command is not supported.

ReSync Status Now
The ReSync Status Now command is not supported.

HTML Export
The HTML Export command exercises the HTML Export menu option.

Command format:
“HTML Export;summary=<summary>;databytes=<bytes>;decode=<decode>;[frames=<frames>;]
[frame range upper=<frame number>;][frame range lower=<frame number>;file=<export path>"

For “summary” the values are

- 0 – Do not include, default
- 1 – Include

For “bytes” the values are

- 0 – Do not include
- 1 – Include, default

For “decode” the values are

- 0 – Do not include
- 1 – Include, default if “bytes” is “include”

Valid values for “frames” are

- “all” – Export all frames, all other frame parameters ignored
- “selected” – Export selected frames, default

For “frame number”

- Valid values are 0 to the highest frame.
- if “frames” is set to “all”. “frame range upper” and “frame range lower” are ignored
- If “frames” is set to “selected”, “frame range upper” defaults to highest frame and “frame range lower” defaults to 0.

If the export path represents just a file name, the configured log directory will be prepended.

Response “HTML EXPORT;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Examples

“HTML Export;summary=1;databytes=0;decode=0;file=f:\MyExport.htm”

“HTML Export;summary=1;databytes=0;decode=0;file=f:\MyExport.htm”;frame range upper=48”

There are several important ideas to remember about HTML Export.

- BE AWARE THAT THIS CAN CAUSE THE ANALYZER TO SPEND CONSIDERABLE TIME GENERATING THE HTML FILE.
- If no html file is specified, a default one will be generated and saved in the log file directory created during installation of the analyzer application.
- If an html file is specified, then this file is created and saved in the log file directory created during installation of the analyzer application.
- All layers for each frame will be exported. The user cannot select the layers.
- If the capture file is empty, no html will be generated, and an error will be reported.
- If the range selected is outside the current range of processed frames, no html will be generated, and the last process frame will be included in the reason.
- If all frames need to be exported, then select some unknown large number for the upper range or set “frames” to “all” or repeatedly send Is Processing Complete command until frame compilation is done prior to sending HTML Export command.

Export

The Export command exercises the Export CSV menu option.

Command format: “Export;file=<path>;[mode=<mode>;][tab=<technology:layer>]”

For “path”

- If the export path represents just a file name, the configured log directory will be prepended.
- The file extension must be “.csv”

Valid values for “mode” are

- 0 – Wait for frame compilation to complete, default
- 1 – Do not wait for frame compilation to complete

Note that it is important to understand that if the “mode” parameter is not specified, or has a value”0”, and the sniffer is still running and capturing new frames, then the Export command may wait indefinitely.

The tab currently selected on the Frame Display is what gets exported. A specified tab can be selected using the optional Tab= “<technology>:<layer>” where the layer name generally matches the tab names in Frame Display.

Export Technologies

Valid values for “technology” are

- “Classic” – Classic baseband
- “LE” – low energy baseband
- “80211” – 802.11 radio

Export Layers

Valid values for “layer” are

"6LowPan"	"6LoWPAN UDP"	"802.11 AMP"	"802.11 MAC"	"802.11 Radio"
"802.1D - STP"	"802.1X"	"A2DP"	"AMP Manager"	"ARP"
"ASHA"	"Async PPP"	"ATT"	"AVCTP"	"AVDTP"
"AVDTP Media"	"AVDTP Recover"	"AVDTP Report"	"AVDTP Signaling"	"AVRCP"
"AVRCP Browsing"	"Baseband"	"BCCMD"	"BIP"	"BlueCore Serial Protocol"
"BlueCore Serial Protocol 1"	"BlueCore Serial Protocol 2"	"Bluetooth FHS"	"Bluetooth PRP"	"Bluetooth USB"
"Bluetooth Virtual Transport"	"BNEP"	"BPP"	"BT SLIP"	"BT-HID"
"CAPI"	"Chipcard Bulk Messages"	"CIP"	"CMTP"	"Context Missing"
"CSB"	"CSRMesh MASP"	"CSRMesh MCP"	"CSRMesh MTP"	"CTN"
"Data"	"DHCP/BOOTP"	"DHCPv6"	"DLR"	"DNS"
"Encaps. Ethernet"	"Encaps. HDLC mod 128 (LAPB)"	"Encaps. HDLC mod 8 (LAPB)"	"Encaps. SDLC mod 128"	"Encaps. SDLC mod 8"
"Encapsulated AsyncPPP"	"Ethernet"	"Extended Inquiry Response"	"FAX"	"Frame Info"
"FTP"	"H4DS"	"Hands-Free"	"HCI"	"HCI SCO/eSCO"
"HCI UART"	"HCI UART 1"	"HCI UART 2"	"HCRP Control"	"HCRP Data"
"HDP_11073"	"Headset"	"HID Keyboard"	"HID Mouse"	"HID OVER GATT"
"HS/HF"	"HTTP"	"iAP"	"iAP1 Accessory Equalizer"	"iAP1 Digital Audio"
"iAP1 Display Remote"	"iAP1 Extended Interface"	"iAP1 General"	"iAP1 iPod Out"	"iAP1 Location"
"iAP1 RF Tuner"	"iAP1 Simple Remote"	"iAP1 Sports"	"iAP1 Storage"	"iAP1 USB"
"iAP1 USB Host Mode"	"iAP2"	"IBM NetBIOS"	"ICMP"	"ICMPv6"
"IEC-61850"	"IGMP"	"IPsecESP"	"IPsecNAT"	"IPv4"
"IPv6"	"IPX NetBIOS"	"IPX RIP"	"IPX SAP"	"IPX SPX"
"ISAKMP"	"ISO-DEP"	"L2CAP"	"LE ADV"	"LE BB"
"LE BIS"	"LE CIS"	"LE DATA"	"LE LL"	"LE PKT"
"LE Test Mode"	"LLC 802.2"	"LLDP"	"LMP"	"Logic Signals"
"LPMP"	"LTP"	"MAP"	"Mass Storage Bulk Only Transport"	"MCAP Control"
"Mesh"	"Mesh Beacon"	"Mesh Provisioning"	"Mesh Proxy"	"Metadata"
"Metadata "	"Metadata "	"Metadata "	"Metadata "	"Metadata "
"Metadata "	"Metadata "	"Mifare"	"Mifare Memory"	"MWS WCI-2"
"MWS WCI-2 1"	"MWS WCI-2 2"	"NBDS"	"NBNS"	"NBSS"
"NCM Data"	"NCP"	"NMEA-0183"	"NMEA_0183"	"Non-Captured Info"
"OBEX"	"OPP"	"Padding"	"PBAP"	"PPP"

"PreConnection-FHS"	"PTP/MTP"	"PTS"	"Q.933"	"RA USBCIP"
"RFCOMM"	"RNDIS"	"RTCP"	"RTP"	"SCO/eSCO"
"SCSI Primary Commands"	"SDIO"	"SDIO/SPI"	"SDIO/SPI/HCI"	"SDP"
"SIM ACCESS"	"SIM Application"	"SIP"	"SLIP"	"SMB"
"SMP"	"SMTP"	"SNAP"	"SNMP"	"SPP"
"SSDP"	"Sybase TDS"	"SYNC"	"SyncML"	"TCP"
"TCS"	"Three-Wire UART"	"Three-Wire UART 1"	"Three-Wire UART 2"	"TLS"
"TYPE 1"	"TYPE 1 Memory"	"TYPE 2"	"TYPE 2 Memory"	"TYPE 3"
"TYPE 3 Memory"	"TYPE 4"	"TYPE 4 Memory"	"UDI"	"UDP"
"UMA"	"Undecoded L2CAP Frame"	"Undecoded Mesh"	"Undecoded RFCOMM Frame"	"URB"
"USB"	"USB 1"	"USB 2"	"USB Data and Handshake Messages"	"USB Pre Messages"
"USB Setup"	"USB Split Message"	"USB Token Message"	"UWB MAC"	"UWB Packet Encapsulator"
"UWB Radio"	"VCP"	"VDP"	"Virtual Sniffer"	"VJC"
"VJU"	"WB Speech"	"WiMedia"	"Wireless USB"	"WUSB"

Response “EXPORT;<status>;<timestamp>;[<reason>]” where a reason will only be supplied for FAILED status.

Example – “Export;file=export1.csv;tab=Classic:SCO/eSCO”

Note, if your capture file does not contain the specified layer name OR you make an error in entering the layer name then the export command will export the currently selected layer. In other words, there will not be an error notification for this case.

Note that Export can cause the analyzer to spend considerable time generating the file.

Get ComProbe Information

The Get ComProbe Information command is used to get the serial number and hardware version of the Sodera or X240.

Command format: “Get ComProbe Information”

Example – “Get ComProbe Information”

Response “GET COMPROBE INFORMATION;<status>;<timestamp>;[<reason>]” where reason will be supplied for SUCCEEDED and FAILED status.

For SUCCEEDED status, reason will be “serial_number=<hardware serial number>|hardware_version=<hardware version>”

Get ComProbe Firmware Information

The Get ComProbe Firmware Information command is used to get the firmware versions of the Sodera or X240.

Command format: “Get ComProbe Firmware Information”

Example – “Get ComProbe Firmware Information”

Response “GET COMPROBE FIRMWARE INFORMATION ;<status>;<timestamp>;[<reason>]” where reason will be supplied for SUCCEEDED and FAILED status.

For SUCCEEDED status, reason will be “firmware_status=<firmware status>|firmware_status_description=<firmware status description>|firmware_version=<firmware version>|fpga_version=<FPGA version>|pic_version=<PIC version>”

Get Capture State

The Get Capture State command is used to get the capture state of the Sodera or X240.

Command format: “Get Capture State”

Example – “Get Capture State”

Response “GET CAPTURE STATE;<status>;<timestamp>;[<reason>]” where reason will be supplied for SUCCEEDED and FAILED status.

For SUCCEEDED status, reason will be “<capture state> where capture state can be

- Capture Stopped – Capture has stopped.
- Capture Stopping – Capture is stopping.
- Capture Starting – Capture is starting.
- Capture Active – Capture is active.
- Invalid Capture State – Invalid capture state.

Configure BD_ADDRs

The Configure BD_ADDRs command is used to pre-select BD_ADDR values for the analyze operation of the Sodera or X240.

Command format: “Configure BD_ADDRs;<address list>”

Address list is a comma-delimited string of BD_ADDRs to be selected for analyze.

Example – “Configure BD_ADDRs;0x001122334455,0xaabbccddeeff”

Response “CONFIGURE BD_ADDRs;<status>;<timestamp>;[<reason>]” where reason will be supplied for SUCCEEDED and FAILED status.

For SUCCEEDED status, reason will be “<status>” where status may be

- “addresses_configured=none” – The supplied address list was empty, and nothing was configured or there was a command processing error.
- “addresses_configured=some|<BD_ADDR>|<BD_ADDR>|…|<BD_ADDR>|configuration_error=No device found for addresses|<BD_ADDR>|<BD_ADDR>|…|<BD_ADDR>” - Some addresses were configured and others were not.
- “addresses_configured=all||<BD_ADDR>|<BD_ADDR>|…|<BD_ADDR>” – All addresses were successfully configured.

Data Extraction Plugin Commands

This section describes automation commands and statuses for the Data Extraction Plugin:

- Automation commands are sent by the client to the plugin.
- Automation status is sent by the plugin to the client.
- The Open Capture File command must precede plugin commands.

Plugin Commands

Plugin command format:

"Plugin Command;plugin name=<name>;command=<command>[;parameters=<parameters>]"

Plugin Responses

A plugin response has the following formats

"PLUGIN COMMAND;Plugin Name=<name>;<status>;<timestamp>;<error code>;description=<description>

Status values are "FAILED" or "SUCCEEDED"

A status message is sent immediately after the command is received. If the command will take time to execute, one or more additional status messages are sent later.

The "error code" and "description" fields indicate the meaning of each SUCCEEDED or FAILED message respectively.

Status	Result	Description
SUCCEEDED	10	Command completed
SUCCEEDED	11	Command accepted
SUCCEEDED	30	Extraction completed
SUCCEEDED	40	File opened
SUCCEEDED	41	File closed
SUCCEEDED	60	In live mode
SUCCEEDED	63	In standby mode
FAILED	9010	Invalid syntax (explanation)
FAILED	9011	Unknown command
FAILED	9012	Missing parameter (explanation)
FAILED	9013	Invalid parameter (explanation)
FAILED	9014	Too many parameters
FAILED	9020	Path does not exist
FAILED	9021	Invalid basename
FAILED	9030	No data found to extract
FAILED	9040	File open error
FAILED	9041	File close error
FAILED	9042	File write error
FAILED	9043	File does not exist

Status	Result	Description
FAILED	9044	File not open
FAILED	9060	Already in live mode
FAILED	9061	Already not in live mode
FAILED	9062	Unable to enter live mode
FAILED	9063	Unable to exit live mode
FAILED	9998	Still executing previous user command
FAILED	9999	Still executing previous automation command

Plugin - Data and Audio Extraction

open files after extraction
This command tells the plugin to open files after extraction – parameter values are “yes” or “no”

two mono files
This command sets the plugin option according to the parameter value - parameter values are “yes” or “no” where “yes” means produce two mono files and “no” means produce one stereo file.

convert to linear pcm
This command sets the plugin option according to the parameter value - parameter values are “yes” or “no”

output path
This command sets the plugin option according to the parameter value - parameter values is a valid path.

output base name
This command sets the plugin option according to the parameter value - parameter values is a non-empty base name that is used as the base file name. Default is the base name of the capture file.

extract
This command sets the plugin option according to the parameter value - parameter values are profile names “BIP”, “FTP”, “MAP”, “OPP” or “PBAP”