Net Protocol Suite API Reference Manual

Generated: October 13, 2015, 15:05

Document disclaimer

Trademarks and servicemarks

Copyright

Contents

List of Figures

List of Tables

1. Introduction

Teledyne LeCroy's Net Protocol Suite API (Application Programming Interface) is a C- based library, which allows you to extract link packets in net trace files, open projects, record traces and run jammer scenarios.

2. Functions

2.1 Operations on trace file

The API supports operations such as setting the trace file name of an project, setting the number of segments, setting the size of segment buffers, running jammer scenarios and recording new traces.

2.2 Packet navigation

The API supports operations such as iterating through trace packets, extracting full raw frame data and any metadata such as time stamps.

2.3 Project navigation, recording and jammer

The API supports operations such as set trace file name of a project, set number of segment, set segment buffer size, run a scenario in jammer, record a new trace.

3. System requirements

3.1 Operating System

Windows XP, Windows 7, Windows 8.1, Windows Server 2003 (32-bit), Windows Server 2008 (R2) and Windows Server 2012.

The latest Service Pack available for the Windows OS in use is required.

It is recommended that the 64-bit Windows version of the Operating Systems is installed, as these allow using more RAM memory than the 32 bit ones.

3.2 Required applications

Microsoft Internet Explorer, version 6 or newer. To view the manuals, datasheets and other documents, you would need to install ‘Adobe Acrobat Reader’ (http://get.adobe.com/reader).

3.3 Memory (RAM)

For improved performance of the software, it is recommended that 16GB of RAM is installed on the host machine. Memory as little as 2GB will still allow the software to function, but would impair its performance.

3.4 Non-volatile Storage (SDD or Hard Disk)

Storage space of 200 MB is required for installing the Net Protocol Suite™ software onto the host machine. Additional storage space is needed to operate the software application and to store recorded data in files. Please remember that storing large captured traces can result in multiple gigabytes of file sizes and can quickly fill your available storage space.

3.5 Connectivity

Teledyne LeCroy recommended a Gigabit (1000-Mbps) Ethernet or a USB 3.0 link for connecting with SierraNet analyzers. At minimum, the host machine should have either a 100/1000-Mbps Ethernet connection to the network or a USB 2.0 port.

If multiple analyzers are daisy-chained and connected to the same host machine, one Ethernet connection or one USB port is required for each analyzer.

Please note that there is no connectivity requirement if the analysis application is used to only view pre-recorded traces.

For tips as for how to improve on the performance of the Teledyne LeCroy analysis system and more specifically on the performance of the software, please refer to the User Manual.

4. Configure Visual Studio

- Perform the following setup to use the library and create your own client applications.

Note: In order for your application to be able to connect to analyzers over Ethernet, your application must run a Windows message loop. The following MSDN article has details about how to do this:

```
http://msdn.microsoft.com/en-us/library/windows/desktop/
ms644928(v=vs.85).aspx
```

4.1 Install Net Protocol Suite

1. Download and install Teledyne LeCroy's Net Protocol Suite software.
   • Use the CD that is included with the product.
2. Or download the software from the Teledyne LeCroy website:
   http://teledynelecroy.com/support/softwaredownload/
3. Select **Support - Software Downloads -Protocol Solutions (Analysis Software)**, and select "Ethernet".

4.2 Open Visual Studio

1. Open Microsoft Visual Studio.
2. Create or open a project.

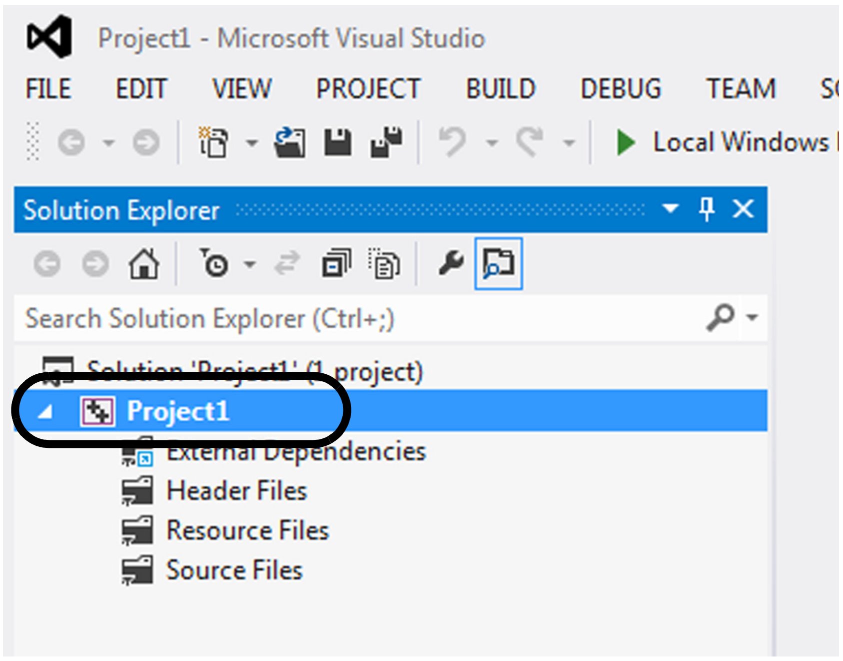


Figure 1: Selecting project name.

3. Right-click on “Project”.
   • The Project menu opens.

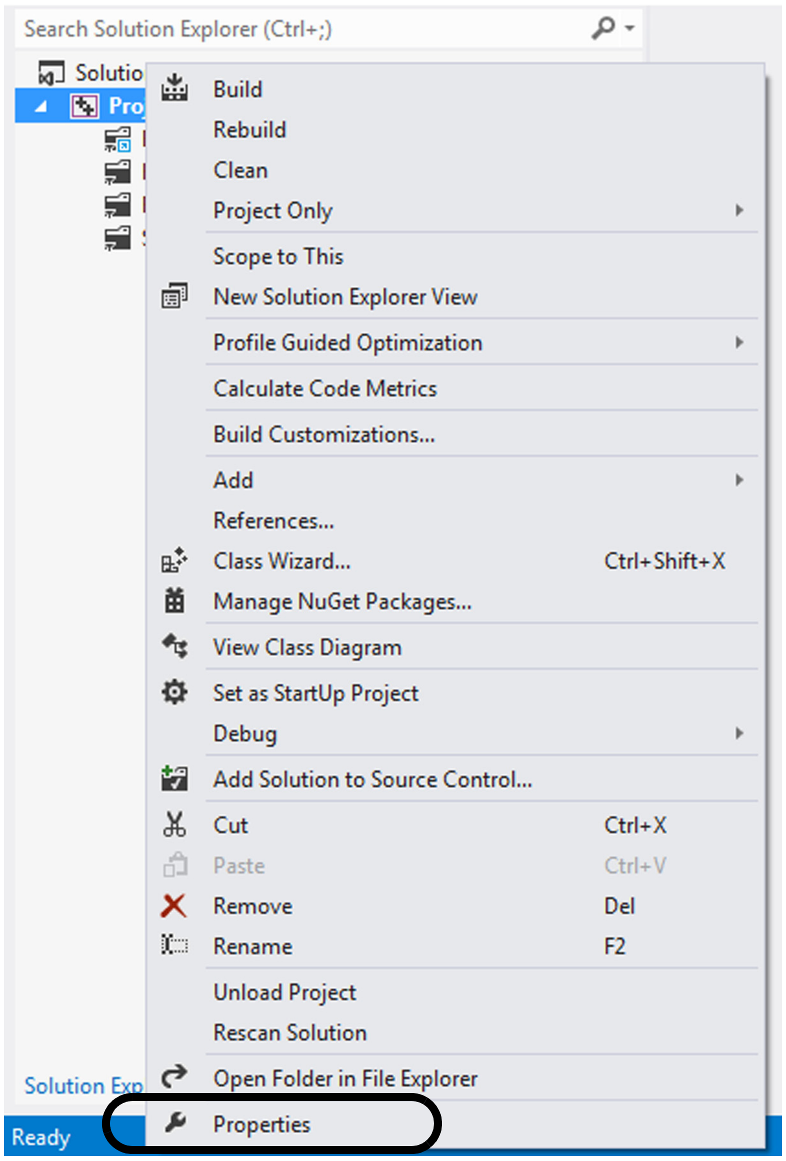
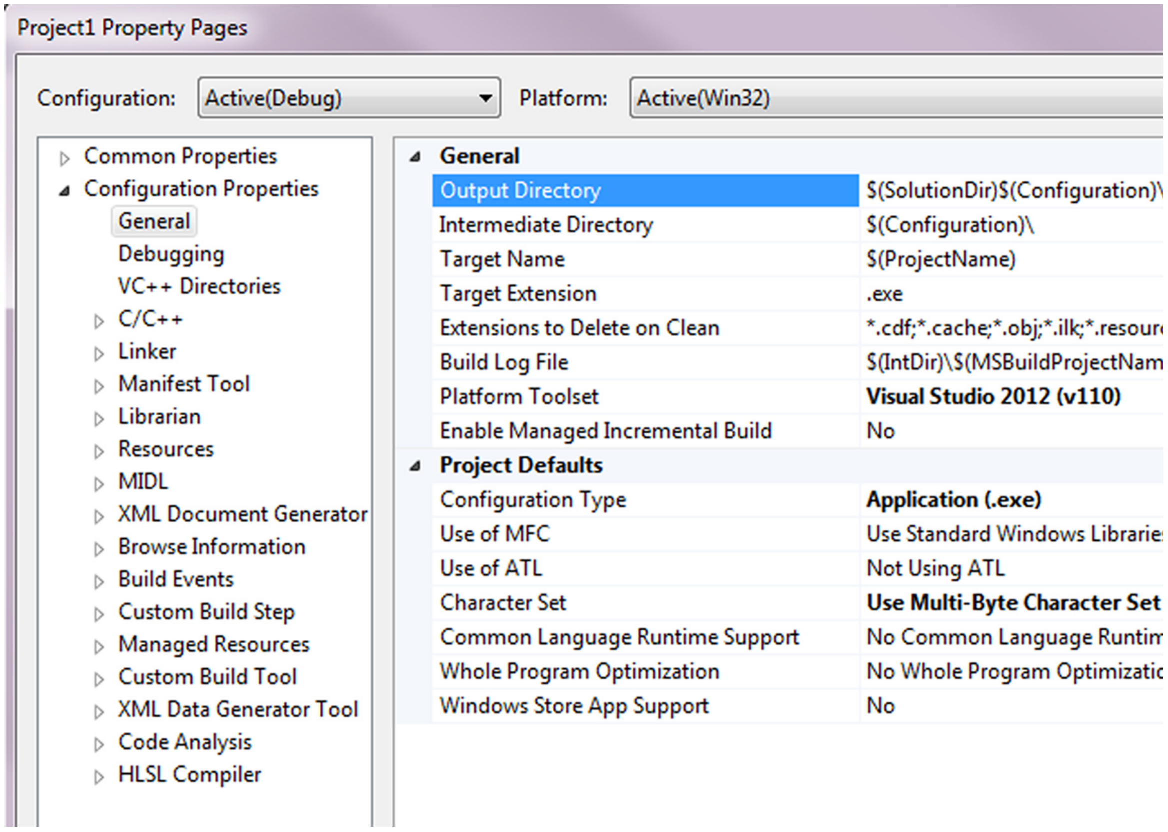


Figure 2: Selecting "properties" menu item.

4. Select “Properties”.
   • Project Property Pages opens.

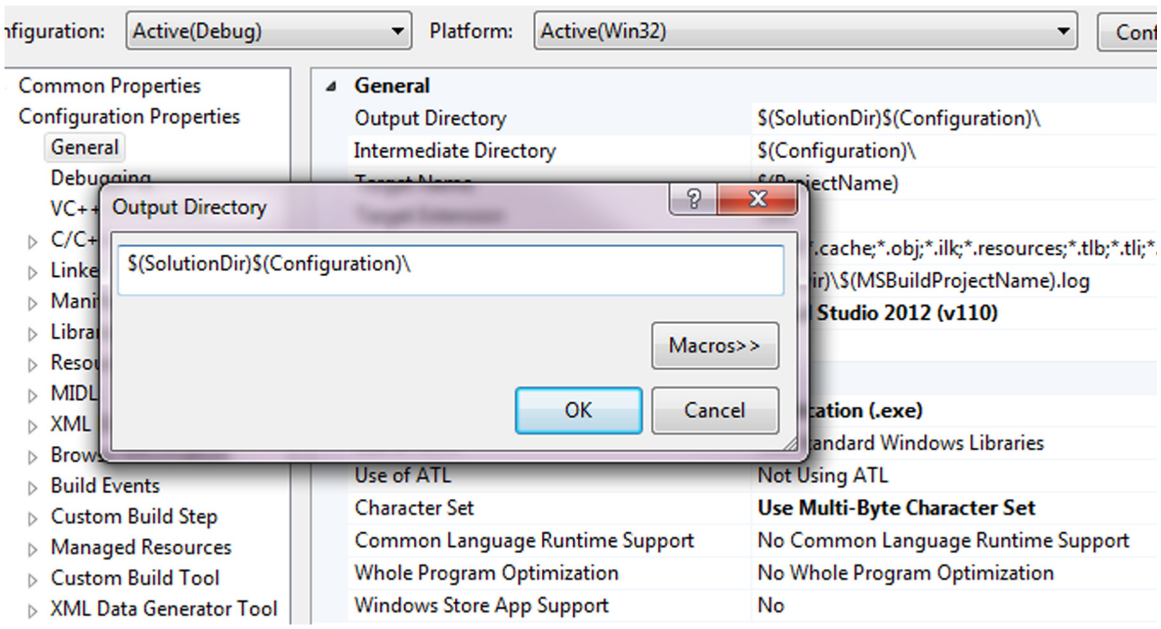


General	
Output Directory	$(SolutionDir)$(Configuration)\
Intermediate Directory	$(Configuration)\
Target Name	$(ProjectName)
Target Extension	.exe
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.ilk;*.resour
Build Log File	$(IntDir)\$(MSBuildProjectNam
Platform Toolset	**Visual Studio 2012 (v110)**
Enable Managed Incremental Build	No
Project Defaults	
Configuration Type	**Application (.exe)**
Use of MFC	Use Standard Windows Librarie
Use of ATL	Not Using ATL
Character Set	**Use Multi-Byte Character Set**
Common Language Runtime Support	No Common Language Runtim
Whole Program Optimization	No Whole Program Optimizatic
Windows Store App Support	No

Figure 3: Property pages.

4.2.1 Select output directory

1. In the drop-down menu to the right, select Browse...
2. Find the directory:
   C:\Users\Public\Documents\LeCroy\Net Protocol Suite\API\SDK\bin
   • Or in the drop-down menu to the right, select Edit.
   • The window "Output Directory" opens.



nfiguration: Active(Debug) Platform: Active(Win32)

Common Properties
Configuration Properties
General
Debugging
VC++
C/C+
Linke
Mani
Libra
Reso
MIDL
XML
Brow
Build Events
Custom Build Step
Managed Resources
Custom Build Tool
XML Data Generator Tool

General	
Output Directory	$(SolutionDir)$(Configuration)\
Intermediate Directory	$(Configuration)\
	ectName)
	.cache;*.obj;*.ilk;*.resources;*.tlb;*.tli;*
	ir)\$(MSBuildProjectName).log
	Studio 2012 (v110)
	ation (.exe)
	andard Windows Libraries
Use of ATL	Not Using ATL
Character Set	**Use Multi-Byte Character Set**
Common Language Runtime Support	No Common Language Runtime Support
Whole Program Optimization	No Whole Program Optimization
Windows Store App Support	No

Output Directory: $(SolutionDir)$(Configuration)\

Figure 4: Defining output directory.

3. Set the path to the output directory. Highlight this text and replace with:

```
C:\Users\Public\Documents\LeCroy\Net Protocol Suite\API\SDK\bin
```

For Windows XP, the directory is:

```
C:\Program Files\LeCroy\Net Protocol Suite\API\SDK\bin
```

4.2.2 Select input file

1. Under Configuration Properties, select "Linker", and under that, select "Input".

2. Select "Additional Dependencies".

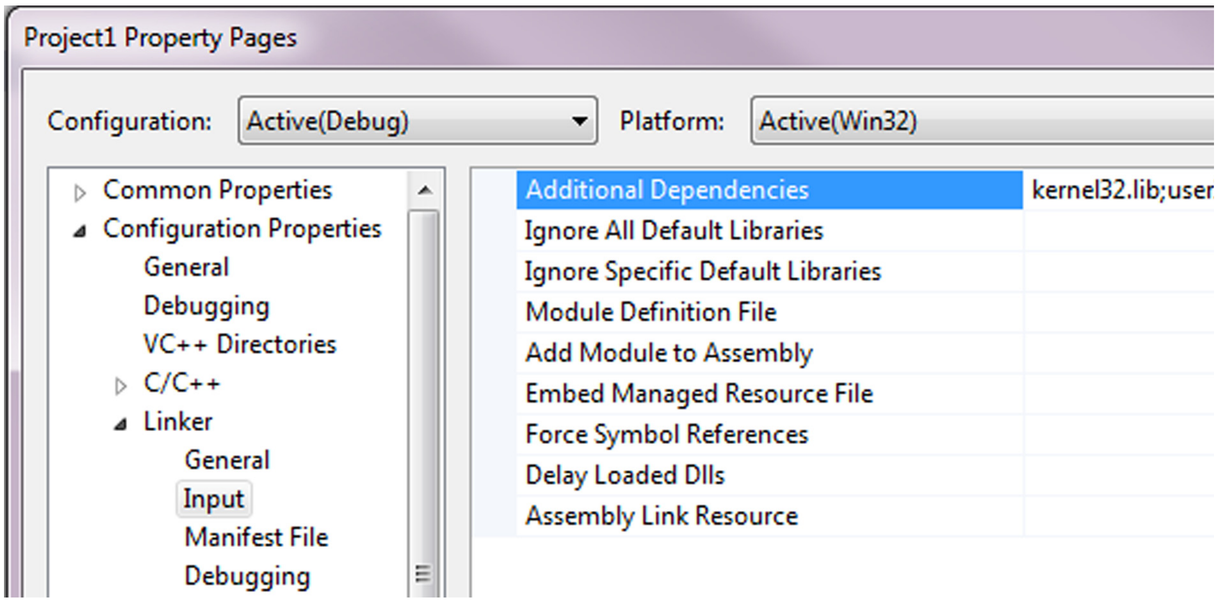


Figure 5: Defining dependencies.

3. Copy this into the field to the right:

```
C:\Users\Public\Documents\LeCroy\Net Protocol Suite\API\SDK\Lib\TLNetAPI.lib
```

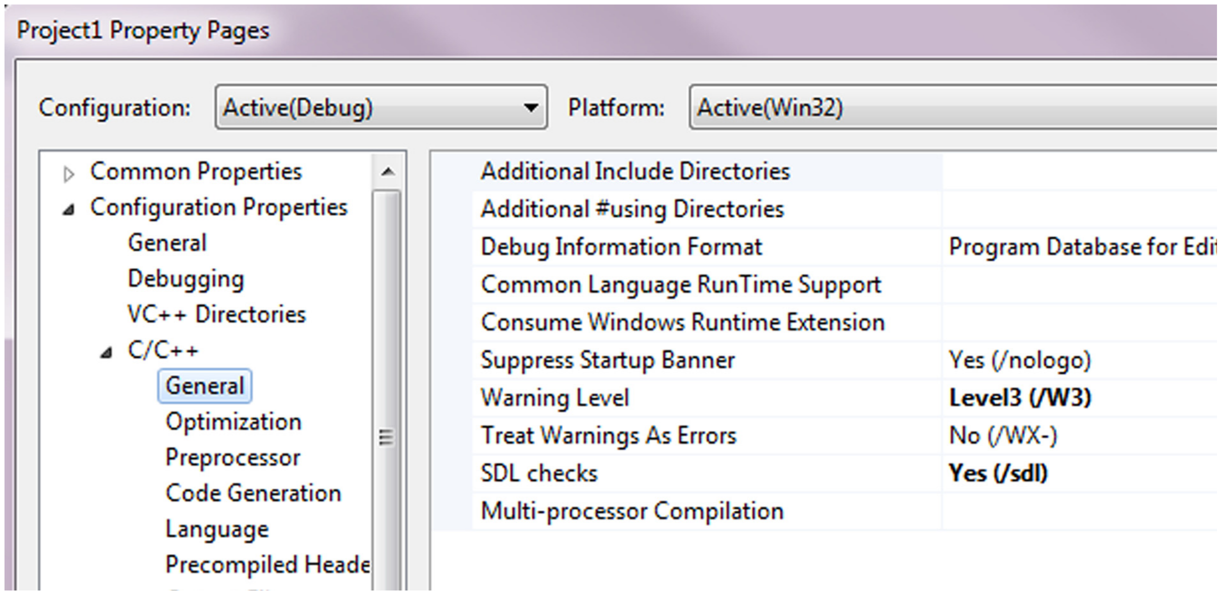
• For Windows XP, use this instead:

```
C:\Program Files\LeCroy\Net Protocol Suite\API\SDK\Lib\TLNetAPI.lib
```

4.2.3 Select directory for include files

1. Under Configuration Properties, select "C/C++", and under that, select "General".

2. Go to "Additional Include Directories".



Additional Include Directories	
Additional #using Directories	
Debug Information Format	Program Database for Edi
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level3 (/W3)**
Treat Warnings As Errors	No (/WX-)
SDL checks	**Yes (/sdl)**
Multi-processor Compilation	

Figure 6: Defining include directories.

3. Enter the following directory:

`C:\Users\Public\Documents\LeCroy\Net Protocol Suite\API\SDK\Include`

4. On the Property Pages window, select "OK" to accept these changes.
   - The contents of the include folder are listed below:

`TLNetAPI.h:` Initializes and releases the API, gets the final internal error code and gets a description of any internal error code functions.

`APIProject.h:` Declares the project functions.

`APIAnalyzerSettings.h:` Declares the project’s analyzer setting functions.

`APIEventHandlers.h:` Declares any callback functions that report the status of recording and jammer processes, and reports any process errors or progress.

`APITrace.h:` Declares all functions.

`APITraceIterator.h:` Defines trace iterating.

`APITracePacket.h:` Defines packet navigating.

4.3 Creating and using objects

1. Create a Trace object and open the trace as follows:

```
EAPIErrorCode error_code= API_OK;
APITrace *trace= NULL;
error_code= APITrace_create(trace );
error_code= APITrace_ Open( trace , trace_file_name.GetBuffer(0) );
```

2. Create a Trace Iterator and pass the already created Trace Object during its construction.

```
APITraceIterator *trace_itr = NULL;
error_code= APITraceIterator_create( trace , trace_itr );
```

3. That's it! You are now ready to use the different methods of trace and iterator.

```
error_code= APITraceIterator_ToFront( trace_itr );
bool has_next= false;
error_code= APITraceIterator_HasNext( trace_itr , has_next );
 if(has_next)
{
   APITracePacket *packet= NULL;
   error_code= APITracePacket_create( packet );
   error_code= APITraceIterator_GetNext( trace_itr , packet );
   EDataType data_type;
   error_code= APITracePacket_getType( packet , data_type );
   .
   .
   .
```

4. Don't forget to destroy all created objects.

```
.
   .
   .
   error_code= APITracePacket_destroy( packet );
}
error_code= APITraceIterator_destroy( trace_itr );
error_code= APITrace_destroy( api_trace );
```

5. Understanding the library interface

5.1 Initialize/release API

Initialize and release the API before you use it. Use the following functions to initialize and release the API.

Declarations and prototypes

```
TLNetAPI.h
```

5.1.1 APIInitialize()

Prototype

```
APIInitialize()
```

Description
This function initializes the API. It must be called before any other API function calls.

Parameters
None.

Returned value
It always returns zero.

5.1.2 APIUninitialize()

Prototype

```
APIUninitialize()
```

Description
This function releases the API. It must be called after the last API function call. If any API function is called after this, it returns an error code that means API has not been initialized.

Parameters
None.

Returned value
It always returns zero.

5.2 Enumerations

All API enumerations are located in:

`SDK/Includes/APIEnumerations.h`

5.3 Version/Info Methods

5.3.1 APIGetVersion ()

Prototype

```
APIGetVersion(int *major, int *minor, int *build_no)
```

Description
This function gets the version of the software.

Parameters

major: major software version
minor: minor   manor software version
build_no: build number of software

Table 1: Error codes for APIGetVersion ().

API error code	Value	Description
API_EXCEPTION	1	An exception occurred.

5.3.2 APIGetHardwareInfo()

Prototype

```
APIGetHardwareInfo(unsigned short serial_number, unsigned short * product_id)
```

Description
It gets product ID of a device.

Parameters

serial_number: serial number of the board
product_id: product id of the board
If no board with the input serial number exists, the return value is 33.
(API_ERROR_INVALID_ARGUMENT). It is possible to get the complete description of the error code by calling APIError_GetLastError() and APIError_GetDescription() functions.

Product id: it can be one of the following values:

Value	Product Id:
0x3102	M408
0x3502	M168
0x1402	M8-4
0x1802	M164

Table 2: Error codes for APIGetHardwareInfo ().

API error code	Value	Description
API_EXCEPTION	1	An exception occurred.

5.4 Error handling

All API functions return an integer value in code from the API ErrorCode enumeration. If any error occurs, a proper error code is returned. (If no code is returned, then that means that no error occurred.) In some cases, more details are available about the error. APIError_GetLastError() can be called to return the internal error code. The description of an internal error code, returned by APIError_GetLastError(), can be collected by calling APIError_GetDescription().

Declarations and prototypes

```
TLNetAPI.h
```

5.4.1 APIError_GetLastError()

Prototype

```
APIError_GetLastError(): int
```

Description

If any error occurs during an API function call, the function returns an error in code from the EAPIErrorCode enumeration. In some cases, more details are available about the error. This function returns the last occuring internal error code. The returned value is an integer. To collect a complete description, call APIError_GetDescription().

Parameters

None.

Returned value

It returns the last internal API error code.

5.4.2 APIError_GetDescription()

Prototype

```
APIError_GetDescription(int error_code, char *error_description,
                        int max_length_error_description): void
```

Description

It returns a description of an error code.

Parameters

`error_code:` An integer value as error code.

`error_description:` An allocated buffer that is passed to collect the description of the error.

`max_length_error_description:` Maximum length of error description buffer.

Returned value

None.

5.5 Project methods

Declarations and prototypes

```
APIProject.h
```

5.5.1 APIProject_create ()

Prototype

```
APIProject_create(APIProject* *api_project)
```

Description
It creates a new object of project type.

Parameters

`api_project:` A pointer reference to a created project object.

Table 3: Error codes for APIProject_create ().

API error code	Value	Description
API_ERROR_INSUFFICIENT_MEMORY	3	There is insufficient memory to create a new API object.

5.5.2 APIProject_destroy()

Prototype

```
APIProject_destroy(APIProject* *api_project)
```

Description
This function tries to destroy the input project object. First, it tries to save and close the project. If it cannot save or close, or if the project is in use, this function cannot destroy the input object, and it will return an error. If the project is in use, and API cannot destroy it, nothing happens.

Parameters

`api_project:` Pointer to the project object created by APIProject_create().

Table 4: Error codes for APIProject_destroy().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.

5.5.3 APIProject_Open()

Prototype

```
APIProject_Open(APIProject *api_project, const char *file_name)
```

Description
It opens a project.

Parameters

`api_project:` Pointer of the project object created by APIProject_create().

`file_name:` Null-terminated project file name.

Table 5: Error codes for APIProject_Open().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_CAN_NOT_OPEN_PROJECT	25	It cannot open the project.

5.5.4 APIProject_Close()

Prototype

```
APIProject_Close(APIProject *api_project)
```

Description
It disconnects from all boards and closes the opened project by calling APIProject_open().

Parameters

`api_project:` Pointer of the project object opened by APIProject_Open().

Table 6: Error codes for APIProject_Close().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_CAN_NOT_CLOSE_PROJECT	23	It cannot close project.
API_ERROR_CAN_NOT_SAVE_PROJECT	24	It cannot save project.
API_ERROR_CAN_NOT_DISCONNECT_FROM_DEVICE	26	It cannot disconnect from a device.

5.5.5 APIProject_Save()

Prototype

```
APIProject_Save(APIProject *api_project)
```

Description
It saves the project.

Parameters

`api_project:` Pointer of the opened project object by APIProject_create().

Table 7: Error codes for APIProject_Save().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_CAN_NOT_SAVE_PROJECT	24	It cannot save the project.

5.5.6 APIProject_GetChainCount()

Prototype

```
APIProject_GetChainCount(APIProject *api_project, int *chain_count)
```

Description
It returns the number of included chains in the project.

Parameters

`api_project:` Pointer of the project object opened by APIProject_create().

`chain_count:` The number of chains in the project.

Table 8: Error codes for APIProject_GetChainCount().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.5.7 APIProject_GetDeviceCount()

Prototype

```
APIProject_GetDeviceCount(APIProject *api_project, int chain_index,
                          int *device_count)
```

Description
It returns the number of devices in project’s chain.

Parameters

`api_project:` Pointer of the project object opened by APIProject_create().

`chain_index:` Index of the chain. It is a zero-based index.

`device_count:` Number of devices in index(th) chain of the project.

Table 9: Error codes for APIProject_GetDeviceCount().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.5.8 APIProject_GetPairPortCount()

Prototype

```
APIProject_GetPairPortCount(APIProject *api_project, int chain_index,
                            int device_index, int *pair_port_count)
```

Description
It returns the number of pair ports supported by a device in a project's chain.

Parameters

`api_project:` Pointer of the project object opened by APIProject_create().

`chain_index:` Index of the chain. It is a zero-based index.

`device_index:` Index of the device. It is zero-based index.

`pair_port_count:` Number of the device's pair-port in the project.

Table 10: Error codes for APIProject_GetPairPortCount().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.5.9 APIProject_GetAnalyzerSettings()

Prototype

```
APIProject_GetAnalyzerSettings(APIProject *api_project, int chain_index,
                               APIAnalyzerSettings *api_analyzer_settings)
```

Description
It returns the analyzer settings of a specific chain in the opened project.

Parameters

`api_project:` Pointer of the project object opened by APIProject_create().

`chain_index:` Index of the chain. It is a zero-based index.

`api_analyzer_settings:` Refers to the pointer of the analyzer settings of the project's chain_index.

Table 11: Error codes for APIProject_GetAnalyzerSettings().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_CHAIN_INDEX	4	Invalid chain index.

5.5.10 APIProject_GetPortConfiguration()

Prototype

```
APIProject_GetPortConfiguration(APIProject *api_project, int chain_index,
                                int device_index,
                                ELinkConfiguration *port_config)
```

Description

It returns the port configuration of a device in a chain.

Parameters

`api_project:` Pointer of the project object by APIProject_create().

`chain_index:` Index of the chain. It is a zero-based index.

`device_index:` Index of the device. It is a zero-based index.

`port_config:` Port configuration of the chain's device.

Table 12: Error codes for APIProject_GetPortConfiguration().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.
API_ERROR_INVALID_CHAIN_INDEX	4	Invalid chain index.

5.5.11 APIProject_Assign()

Prototype

```
APIProject_Assign(APIProject *api_project, unsigned short serial_number,
                  EConnectionType connection_type, int chain_index,
                  int device_index)
```

Description

It connects to a board and assigns it to a device of a chain in the project. If any board needs to be updated, it returns error codes.

Parameters

`api_project:` Pointer of the project object opened by APIProject_Open().

`serial_number:` The board's serial number.

`connect_type:` The board's connection type. It can be USB or TCP.

`chain_index:` Index of chain. It is a zero-based index.

`device_index:` Index of device. It is a zero-based index.

Table 13: Error codes for APIProject_Assign().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_CHAIN_INDEX	4	Invalid chain index.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.
API_ERROR_SELECTED_DEVICE_IS_NOT_UPDATED	27	The selected device is not updated.
API_ERROR_CAN_NOT_ASSIGN_TO_DEVICE	31	It cannot assign to the selected device.

5.5.12 APIProject_StartRecording()

Prototype

```
APIProject_StartRecording(APIProject *api_project, int chain_index, int
                          trigger_settings_count, const char
                          **trigger_settings_names, OnTraceCreatedProc
                          OnTraceCreated, OnReportRecordingStatusProc
                          OnReportRecordingStatus, OnAnalyzerErrorProc
                          OnError, int time_out)
```

Description

It starts the recording process.

Parameters

`api_project:` Pointer of the project object opened by APIProject_Open().

`chain_index:` Index of chain. It is a zero-based index.

`trigger_settings_count:` Number of trigger setting names.

`trigger_setting_names:` Null-terminated Trigger/Filter setting name in the current opened project. If it is Null, the default is used.

`event_handler:` It refers to a callback function to handle event notifications from the APIs. If it is NULL, then events will not be sent to the client. If the user wants to follow or perform any process during a different recoding status, the OnReportRecordingStatus callback function can be used to report changes in the status of the recording process. The OnError callback function is called to report any errors that occur during the recording or uploading processes. When a trace is created, the OnTraceCreated callback function reports that a trace files has been created.

`time_out:` If the timeout is greater than 0, a timer will begin counting down from the timeout value. If the recording is still in progress when the timer expires, the recording will automatically stop. If the timeout is less then or equal to zero, the recording will continue until stopped by other means.

Table 14: Error codes for APIProject_StartRecording().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_CAN_NOT_START_RECORDING	30	It cannot start recording process.
API_ERROR_SELECTED_DEVICE_IS_NOT_UPDATED	27	The selected device is not updated.
API_ERROR_INVALID_TRIGGER_NAME	7	Invalid trigger name.
API_ERROR_INVALID_CHAIN_INDEX	4	Invalid chain index.

The following errors are returned by callback function:

Table 15: Callback Errors for APIProject_StartRecording().

API error code	Value	Description
API_ERROR_RECORDING	16	An error occurred during recording process.
API_ERROR_UPLOADING	17	An error occurred during uploading process.

5.5.13 APIProject_StopRecording()

Prototype

```
APIProject_StopRecording(APIProject *api_project, int chain_index,
                         bool do_not_upload)
```

Description
It stops any recording or uploading process.

Parameters

`api_project:` Pointer of the opened project object by APIProject_Open().

`chain_index:` The chain's index. It is a zero-based index.

`Do_not_upload:` If it is true, it does not upload any trace file, or stop uploading if the uploading was started automatically.

Table 16: Error codes for APIProject_StopRecording().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_CHAIN_INDEX	4	Invalid chain index.
API_ERROR_CAN_NOT_STOP_RECORDING	29	

The following errors are returned by callback function.

Table 17: Callback Errors for APIProject_StopRecording().

API error code	Value	Description
API_ERROR_RECORDING	16	An error occurred during recording process.
API_ERROR_UPLOADING	17	An error occurred during uploading process.

5.5.14 APIProject_StartJammer()

Prototype

```
APIProject_StartJammer(APIProject *api_project, int chain_index, int
                       device_index, int pair_port_index, const char
                       *scenario_name, OnReportMonitoringStatusProc
                       OnReportMonitoringStatus, OnReportJammerStatusProc
                       OnReportJammerStatus, OnJammerErrorProc OnError);
```

Description
It starts the jammer process.

Parameters

`api_project:` Pointer of the project object opened by APIProject_Open().

`chain_index:` The chain's index. It is a zero-based index.

`device_index:` The device's index. It is a zero-based index.

`pair_port_index:` The pair port's index. It is a zero-based index.

`scenario_name:` Null-terminated Scenario name. If it is Null, then the project default is used.

`event_handler:` It refers to a callback function to handle the events. If it is NULL, the events will not be sent to the client.

Table 18: Error codes for APIProject_StartJammer().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_CAN_NOT_START_JAMMER	13	It cannot start jammer process.
API_ERROR_SELECTED_DEVICE_IS_NOT_UPDATED	27	The selected device is not updated.
API_ERROR_INVALID_CHAIN_INDEX	4	Invalid chain index.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.
API_ERROR_INVALID_PAIR_PORT_INDEX	6	Invalid pair port index.
API_ERROR_INVALID_SCENARIO_NAME	28	Invalid scenario name.

The following errors are returned by callback function.

Table 19: Callback Errors for APIProject_StartJammer().

API error code	Value	Description
API_ERROR_JAMMER	18	An error occurred during jammer process.

5.5.15 APIProject_StartJammers()

Prototype

```
APIProject_StartJammers(APIProject *api_project, int chain_index, int
                        device_index, int count, int pair_port_indexes[],
                        const char *scenario_names[],
                        OnReportMonitoringStatusProc
                        OnReportMonitoringStatus[], OnReportJammerStatusProc
                        OnReportJammerStatus[], OnJammerErrorProc
                        OnError[]);
```

Description

It starts the jammer process on a set of ports. If any port has a problem, it does not stop; it starts to jam another port. It returns the last error code if any exists; otherwise, it returns zero.

Parameters

`api_project:` Pointer of the project object opened by APIProject_Open().

`chain_index:` The chain's index. It is a zero-based index.

`device_index:` The device's index. It is a zero-based index.

`count:` Jammer count.

`pair_port_indexes[]:` Array of indexes of pair ports. It is a zero-based index.

`scenario_names[]:` Array of indexes of scenario names. Null terminated Scenario name. If it is Null, project default is used.

`event_handler[]:` It refers to a callback function to handle the events. If it is NULL, the events will not be sent to the client.

Table 20: Error codes for APIProject_StartJammers().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_CAN_NOT_START_JAMMER	13	It cannot start jammer process.
API_ERROR_SELECTED_DEVICE_IS_NOT_UPDATED	27	The selected device is not updated.
API_ERROR_INVALID_CHAIN_INDEX	4	Invalid chain index.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.
API_ERROR_INVALID_PAIR_PORT_INDEX	6	Invalid pair port index.
API_ERROR_INVALID_SCENARIO_NAME	28	Invalid scenario name.

The following errors are returned by callback function.

Table 21: Callback Errors for APIProject_StartJammers().

API error code	Value	Description
API_ERROR_JAMMER	18	An error occurred during jammer process.

5.5.16 APIProject_StopJammer()

Prototype

```
APIProject_StopJammer(APIProject *api_project, int chain_index,
                      int device_index, int pair_port_index)
```

Description

It stops the jammer process.

Parameters

`api_project:` Pointer of the project object opened by APIProject_Open().

`chain_index:` The chain’s index. It is a zero-based index.

`device_index:` The device’s index. It is a zero-based index.

`pair_port_index:` The pair port index. It is a zero-based index.

Table 22: Error codes forAPIProject_StopJammer().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_CHAIN_INDEX	4	Invalid chain index.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.
API_ERROR_INVALID_PAIR_PORT_INDEX	6	Invalid pair port index.
API_ERROR_CAN_NOT_STOP_JAMMER	12	It cannot stop jammer process.

The following errors are returned by callback function.

Table 23: Callback Errors for APIProject_StopJammer().

API error code	Value	Description
API_ERROR_JAMMER	18	An error occurred during jammer process.

5.5.17 APIProject_StopJammers()

Prototype

```
APIProject_StopJammers(APIProject *api_project, int chain_index,
                       int device_index, int count, int pair_port_indexes[])
```

Description
It stops the jammer process on a set of ports. If any port has a problem, it does not stop and continues to stop jamming other ports. It returns the last error code if any exists. Otherwise it returns zero.

Parameters

`api_project:` Pointer of the project object opened by APIProject_Open().

`chain_index:` The chain's index. It is a zero-based index.

`device_index:` The device's index. It is a zero-based index.

`count:` The jammer count.

`pair_port_indexes[]:` Array of indexes of pair ports. It is a zero-based index.

Table 24: Error codes for APIProject_StopJammers().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_CHAIN_INDEX	4	Invalid chain index.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.
API_ERROR_INVALID_PAIR_PORT_INDEX	6	Invalid pair port index.
API_ERROR_CAN_NOT_STOP_JAMMER	12	It cannot stop jammer process.

The following errors are returned by callback function.

Table 25: Callback Errors for APIProject_StopJammers().

API error code	Value	Description
API_ERROR_JAMMER	18	An error occurred during jammer process.

5.6 Analyzer settings

Declarations and prototypes

```
APIAnalyzerSettings.h
```

5.6.1 APIAnalyzerSettings_create()

Prototype

```
APIAnalyzerSettings_create(APIAnalyzerSettings* *api_analyzer_settings)
```

Description
It creates a new analyzer settings object.

Parameters

`api_analyzer_settings:` It is a pointer reference to created analyzer settings object.

Table 26: Error codes for APIAnalyzerSettings_create().

API error code	Value	Description
API_ERROR_INSUFFICIENT_MEMORY	3	There is insufficient memory to create a new API object.

5.6.2 APIAnalyzerSettings _destroy()

Prototype

```
APIAnalyzerSettings_destroy(APIAnalyzerSettings* *api_analyzer_settings)
```

Description
It destroys the created APIAnalyzerSettings object.

Parameters

`api_analyzer_setting:` Pointer to the APIAnalyzerSettings object.

Table 27: Error codes for APIAnalyzerSettings _destroy().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.

5.6.3 APIAnalyzerSettings_GetSegmentBufferSize()

Prototype

```
APIAnalyzerSettings_GetSegmentBufferSize(APIAnalyzerSettings
                                         *api_analyzer_settings,
                                         int device_index,
                                         int *segment_buffer_size)
```

Description
It returns the size of the segment buffer.

Parameters

`api_analyzer_settings:` Pointer of the analyzer settings.

`device_index:` Zero-based index of the device.

`Segment_buffer_size:` Segment buffer size (specified in KB).

Table 28: Error codes for APIAnalyzerSettings_GetSegmentBufferSize().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.

5.6.4 APIAnalyzerSettings_SetSegmentBufferSize ()

Prototype

```
APIAnalyzerSettings_SetSegmentBufferSize(APIAnalyzerSettings
                                         *api_analyzer_settings,
                                         int device_index, int segment_size)
```

Description
It sets the size of a segment buffer.

Parameters

`api_analyzer_settings:` Pointer of the analyzer settings.

`device_index:` Zero-based index of the device.

`segment_size:` New segment size in Kbytes.

Table 29: Error codes APIAnalyzerSettings_SetSegmentBufferSize ().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_SEGMENT_SIZE	9	Invalid segment size.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.

5.6.5 APIAnalyzerSettings_SetTraceFileName ()

Prototype

```
APIAnalyzerSettings_SetTraceFileName(APIAnalyzerSettings
                                     *api_analyzer_settings,
                                     const char* file_name)
```

Description
It sets the file name of the analyzer settings.

Parameters

`api_analyzer_settings:` Pointer of the analyzer settings.

`file_name:` A null terminate file name of analyzer settings object.

Table 30: Error codes APIAnalyzerSettings_SetTraceFileName ().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.6.6 APIAnalyzerSettings_ GetTraceFileName()

Prototype

```
APIAnalyzerSettings_GetTraceFileName(APIAnalyzerSettings
                                     *api_analyzer_settings,
                                     char **trace_file_name)
```

Description
It returns the trace file name of the analyzer settings object.

Parameters

`api_analyzer_settings:` Pointer of the analyzer settings.

`file_name:` Returned trace file name. This pointer must not be deleted or destroyed by the user.

Table 31: Error codes APIAnalyzerSettings_ GetTraceFileName().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.6.7 APIAnalyzerSettings_SetNumberOfSegment ()

Prototype

```
APIAnalyzerSettings_SetNumberOfSegment(APIAnalyzerSettings
                                       *api_analyzer_settings,
                                       int segment_number)
```

Description
It sets the number of segments. If the segment buffer size is not valid according to the device property and new segment number, it will change to the valid maximum segment buffer size, depending on the quantity of current segments.

Parameters

`api_analyzer_settings:` Pointer of the analyzer settings.

`segment_number:` Number of segments.

Table 32: Error codes for APIAnalyzerSettings_SetNumberOfSegment ().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_NUMBER_OF_SEGMENT	10	Invalid number of segments.

5.6.8 APIAnalyzerSettings_GetNumberOfSegment ()

Prototype

```
APIAnalyzerSettings_GetNumberOfSegment(APIAnalyzerSettings
                                       *api_analyzer_settings,
                                       int *number_of_segment)
```

Description
It returns the number of segments.

Parameters

`api_analyzer_settings:` Pointer of the analyzer settings.

`segment_number:` Number of segments.

Table 33: Error codes for APIAnalyzerSettings_GetNumberOfSegment ().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.6.9 APIAnalyzerSettings_ SetTrigMode()

Prototype

```
APIAnalyzerSettings_SetTrigMode(APIAnalyzerSettings *api_analyzer_settings,
                                ETriggerMode trig_mode)
```

Description
It sets trigger mode.

Parameters

`api_analyzer_settings:` Analyzer settings pointer.

`trig_mode:` Trigger mode can be one of these values:

SNAPSHOT	0
PATTERN	1

Table 34: Error codes for APIAnalyzerSettings_ SetTrigMode().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_TRIG_MODE	11	Invalid trigger mode.

5.6.10 APIAnalyzerSettings_GetTrigMode ()

Prototype

```
APIAnalyzerSettings_GetTrigMode(APIAnalyzerSettings *api_analyzer_settings,
                                ETriggerMode *trig_mode)
```

Description
It returns trigger mode.

Parameters

`api_analyzer_settings:` Analyzer settings pointer.

`trig_mode:` Returns trigger mode.

SNAPSHOT	0
PATTERN	1

Table 35: Error codes for APIAnalyzerSettings_GetTrigMode ().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.6.11 APIAnalyzerSettings_SetTrigFilterSetting ()

Prototype

```
APIAnalyzerSettings_SetTrigFilterSetting(APIAnalyzerSettings
                                         *api_analyzer_settings,
                                         int device_index,
                                         const char *trig_setting_name)
```

Description
It sets trigger filter setting.

Parameters

`api_analyzer_settings:` Pointer of the analyzer settings.

`trig_setting_name:` Null-terminated Trigger/Filter setting name in the currently opened project.

Table 36: Error codes for APIAnalyzerSettings_SetTrigFilterSetting ().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_TRIGGER_NAME	7	Invalid trigger name.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.

5.6.12 APIAnalyzerSettings_SetTrigPosition ()

Prototype

```
APIAnalyzerSettings_SetTrigPosition(APIAnalyzerSettings
                                    *api_analyzer_settings,
                                    int device_index, int trig_position)
```

Description
It sets the trigger position.

Parameters

`api_analyzer_settings:` Analyzer settings pointer.

`device_index:` Zero-based index of the device.

`Trig_position:` Project trigger’s position. It can be 1 to 99.

Table 37: Error codes for APIAnalyzerSettings_SetTrigPosition ().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_TRIGGER_POSITION	8	Invalid trigger position. It must be between 1 and 99.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.

5.6.13 APIAnalyzerSettings_GetTrigPosition ()

Prototype

```
APIAnalyzerSettings_GetTrigPosition(APIAnalyzerSettings
                                    *api_analyzer_settings,
                                    int device_index, int *trig_position)
```

Description
It returns the trigger position.

Parameters

`api_analyzer_settings:` Pointer of the analyzer settings.

`device_index:` Zero-based index of the device.

`trig_position:` Project trigger’s position. It can be 1 to 99.

Table 38: Error codes for APIAnalyzerSettings_GetTrigPosition ().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_DEVICE_INDEX	5	Invalid device index.

5.7 Trace methods

Declarations and prototypes

```
APITrace.h
```

5.7.1 APITrace_create()

Prototype

```
APITrace_create(APITrace* *api_trace)
```

Description
It creates a new trace object.

Parameters

`api_trace:` It is a pointer reference to a created trace object.

Table 39: Error codes for APITrace_create().

API error code	Value	Description
API_ERROR_INSUFFICIENT_MEMORY	3	There is insufficient memory to create a new API object.

5.7.2 APITrace_destroy()

Prototype

```
APITrace_destroy(APITrace* *api_trace)
```

Description
It destroys the created trace object. It does not close the trace file. It calls APITrace_Close() function before destroying the api_trace object.

Parameters

`api_trace:` Pointer to the trace object.

Table 40: Error codes for APITrace_destroy().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_CAN_NOT_SAVE_TRACE	21	It cannot save the trace.
API_ERROR_CAN_NOT_CLOSE_TRACE	22	It cannot close the trace.

5.7.3 APITrace_Open()

Prototype

```
APITrace_Open(APITrace *api_trace, const char *file_name)
```

Description
It opens a trace. If trace file needs to be updated, it will be updated and then opened. The update process updates the trace file without creating any backup from the updated trace file.

Parameters

`api_trace:` Pointer to the trace object.

`file_name:` Null-terminated string providing the full pathname to the trace file.

Table 41: Error codes for APITrace_Open().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_CAN_NOT_UPDATE_TRACE	20	It cannot update the trace.
API_ERROR_CAN_NOT_OPEN_TRACE	19	It cannot open the trace.

5.7.4 APITrace_Close()

Prototype

```
APITrace_Close(APITrace *api_trace)
```

Description
It closes the opened trace object. If the trace has any modification, it is saved and then closed.

Parameters

`api_trace:` Pointer to the trace object.

Table 42: Error codes for APITrace_Close().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_CAN_NOT_SAVE_TRACE	21	It cannot save the trace.
API_ERROR_CAN_NOT_CLOSE_TRACE	22	It cannot close the trace.

5.7.5 APITrace_GetStartDateTime()

Prototype

```
APITrace_GetStartDateTime(APITrace *api_trace, time_t *start_date_time)
```

Description
It returns the start date-time of the trace object.

Parameters

`api_trace:` Pointer to the trace object.

`start_date_time:` Start date-time of the trace.

Table 43: Error codes for APITrace_GetStartDateTime().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.7.6 APITrace_GetEndDateTime()

Prototype

```
APITrace_GetEndDateTime(APITrace *api_trace, time_t *end_date_time)
```

Description
It returns the end date-time of the trace object.

Parameters

`api_trace:` Pointer to the trace object.

`end_date_time:` End date-time of the trace.

Table 44: Error codes for APITrace_GetEndDateTime().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.7.7 APITrace_GetTriggerDateTime()

Prototype

```
APITrace_GetTriggerDateTime(APITrace *api_trace, time_t *trig_date_time)
```

Description
It returns the trigger date-time of the trace object.

Parameters

`api_trace:` Pointer to the trace object.

`trig_date_time:` Trig datetime of the trace.

Table 45: Error codes for APITrace_GetTriggerDateTime().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.8 Packet methods

Declarations and prototypes

```
APITracePacket.h
```

5.8.1 APITracePacket_create()

Prototype

```
APITracePacket_create(APITracePacket* *api_trace_packet)
```

Description
It creates a new trace object.

Parameters

`api_trace_packet:` It is a pointer reference to a created packet object.

Table 46: Error codes for APITracePacket_create().

API error code	Value	Description
API_ERROR_INSUFFICIENT_MEMORY	3	There is insufficient memory to create a new API object.

5.8.2 APITracePacket_destroy()

Prototype

```
APITracePacket_destroy(APITracePacket* *api_trace_packet)
```

Description
It destroys the created packet object.

Parameters

`api_trace_packet:` Pointer to the trace object.

Table 47: Error codes for APITracePacket_destroy().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.

5.8.3 APITracePacket_GetTimeStamp()

Prototype

```
APITracePacket_GetTimeStamp(APITracePacket *api_trace_packet,
                            __int64 *timestamp)
```

Description
It returns time stamp of the packet in picosecconds. One picosecond is equal to 1/1000 nanosecond.

Parameters

`api_trace_packet:` Pointer to the trace packet object.

`timestamp:` Timestamp of the packet in picoseconds.

Table 48: Error codes for APITracePacket_GetTimeStamp().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.8.4 APITracePacket_GetChannel()

Prototype

```
APITracePacket_GetChannel(APITracePacket *api_trace_packet, int *channel)
```

Description
It returns the packet's channel.

Parameters

`api_trace_packet:` Pointer to the trace packet object.

`channel:` Channel of the packet.

Table 49: Error codes for APITracePacket_GetChannel().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.8.5 APITracePacket_GetSpeed()

Prototype

```
APITracePacket_GetSpeed(APITracePacket *api_trace_packet,
                        EAnalyzerSpeed *speed)
```

Description
It returns the packet's speed.

Parameters

`api_trace_packet:` Pointer to the trace packet object.

`speed:` Speed of the packet.

Table 50: Error codes for APITracePacket_GetSpeed().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.8.6 APITracePacket_GetType()

Prototype

```
APITracePacket_GetType(APITracePacket *api_trace_packet, EDataType *type)
```

Description
It returns the packet's protocol.

Parameters

`api_trace_packet:` Pointer to the trace packet object.

`type:` It returns the type of the packet. The Packet Type enumeration is given below:

ConnectDisconnect	0
EthernetFrame	1
FCFrame	2
EthernetOrderedSet	3
FCOrderedSet	4
AutoNegotiation	5
TrainingSequence	6

Table 51: Error codes for APITracePacket_GetType().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

Table 52: Packet type enumeration.

Packet Type	Value (enum)
Connect/Disconnect	0
Ethernet Frame	1
FC Frame	2
Ethernet Ordered Set	3
FC Ordered Set	4
Auto Negotiation	5
Training Sequence	6

5.8.7 APITracePacket_GetData()

Prototype

```
APITracePacket_GetData(APITracePacket *api_trace_packet,
                       unsigned char* *data, int *data_length)
```

Description

The ‘raw_data’ parameter will be set to the address of the raw data buffer of the packet. The return value shows the size of the buffer in bytes. The packet object manages the raw data buffer. Do not attempt to manually delete the buffer. The buffer is valid as long as the packet object is alive and the trace is open.

Parameters

`api_trace_packet:` Pointer to the trace packet object.

`data:` It will be set to the address of the packet’s data buffer.

`data_length:` Data size in bytes.

- The packet object manages the raw data buffer. Do not attempt to manually delete the buffer. The buffer is valid as long as the packet object is alive and the trace is open.

Table 53: Error codes for APITracePacket_GetData().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

The format for each packet type is as follows. Unless otherwise specified, the byte ordering is little-endian.

Table 54: Connect/Disconnect

DWord	Description
0	Connect: 1/Disconnect: 0

Table 55: Ethernet Frame.

DWord	Description
0	Ethernet Frame
1	Ethernet Frame
2	Ethernet Frame
...	...
N	Ethernet Frame

Table 56: FC Frame

DWord	Description
0	SOF
1	FC Frame
2	FC Frame
...	...
N	EOF

Table 57: Ethernet Ordered Set

DWord	Description
0	Order Set Data (64 bits)
1	
2	Order Set Count

Table 58: FC Ordered Set

DWord	Description
0	Order Set Data (32 bits)
1	Order Set Count

Table 59: Auto Negotiation (Packed Format)

DWord	Description
9	Count

Table 60: Training Sequence

DWord	Description
0	Control Field + Status Fields
1	Count

5.8.8 APITracePacket_GetBookmark()

Prototype

```
APITracePacket_GetBookmark(APITracePacket *api_trace_packet,
                           char* *bookmark)
```

Description
It returns the packet's bookmark.

Parameters

`api_trace_packet:` Pointer to the trace packet object.

`bookmark:` Current bookmark of the packet.

Returned value
It returns packet bookmark strings (only the bookmark title). The packet object manages the returned buffers. Do not attempt to manually delete the buffers. The buffers are valid as long as the packet object is alive and the trace is open.

Table 61: Error codes for APITracePacket_GetBookmark().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.8.9 APITracePacket_SetBookmark()

Prototype

```
APITracePacket_SetBookmark(APITracePacket *api_trace_packet,
                           const char *bookmark)
```

Description
It places a new bookmark on the packet. If the packet already has an existing bookmark, it is replaced with the new one.

Parameters

`api_trace_packet:` Pointer to the trace packet object.

`bookmark:` Name of the bookmark.

Table 62: Error codes for APITracePacket_SetBookmark().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.9 Trace iterator methods

Declarations and prototypes

```
APITraceIterator.h
```

5.9.1 APITraceIterator_create()

Prototype

```
APITraceIterator_create(APITrace *api_trace, APITraceIterator*
                        *api_trace_iterator)
```

Description
It creates a new trace iterator object. The iterator will be initialized to the beginning of the trace.

Parameters

`api_trace:` Trace object.

`Api_trace_iterator:` Reference pointer of created iterator object.

Table 63: Error codes for APITraceIterator_create().

API error code	Value	Description
API_ERROR_INVALID_TRACE	14	Invalid API trace object.
API_ERROR_INSUFFICIENT_MEMORY	3	There is insufficient memory to create a new API object.

5.9.2 APITraceIterator_destroy()

Prototype

```
APITraceIterator_destroy(APITraceIterator* *api_trace_iterator)
```

Description
It destroys the created trace iterator object.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

Table 64: Error codes for APITraceIterator_destroy().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.

5.9.3 APITraceIterator_HasNext()

Prototype

```
APITraceIterator_HasNext(APITraceIterator *api_trace_iterator,
                         bool *has_next)
```

Description
It returns true if iteration has more packets when traversing in forward direction; otherwise, returns false.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

`has_next:` If there is a next packet, it will be true; otherwise it will be false.

Table 65: Error codes for APITraceIterator_HasNext().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.9.4 APITraceIterator_HasPrevious()

Prototype

```
APITraceIterator_HasPrevious(APITraceIterator *api_trace_iterator,
                             bool *has_previous)
```

Description
It returns true if iteration has more packets when traversing in backward direction; otherwise, returns false.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

`has_previous:` If there is any previous packet, it will be true; otherwise it will be false.

Table 66: Error codes for APITraceIterator_HasPrevious().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.9.5 APITraceIterator_GetNext()

Prototype

```
APITraceIterator_GetNext(APITraceIterator *api_trace_iterator,
                         APITracePacket *api_trace_packet)
```

Description
It returns the iterator packet and moves the iterator forward.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

`api_trace_packet:` Next packet container.

Returned value
It returns next packet of iterator and moves the iterator forward.

Table 67: Error codes for APITraceIterator_GetNext().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.9.6 APITraceIterator_GetPrevious()

Prototype

```
APITraceIterator_GetPrevious(APITraceIterator *api_trace_iterator,
                             APITracePacket *api_trace_packet)
```

Description
It returns previous packet of iterator and moves the iterator backward.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

`api_trace_packet:` Previous packet container.

Table 68: Error codes for APITraceIterator_GetPrevious().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.9.7 APITraceIterator_PeekNext()

Prototype

```
APITraceIterator_PeekNext(APITraceIterator *api_trace_iterator,
                          APITracePacket *api_trace_packet)
```

Description
It returns next packet of iterator without moving the iterator forward.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

`api_trace_packet:` Previous packet container.

Table 69: Error codes for APITraceIterator_PeekNext().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.9.8 APITraceIterator_PeekPrevious()

Prototype

```
APITraceIterator_PeekPrevious(APITraceIterator *api_trace_iterator,
                              APITracePacket *api_trace_packet)
```

Description
It returns previous packet of iterator without moving the iterator backward.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

`api_trace_packet:` Previous packet container.

Table 70: Error codes for APITraceIterator_PeekPrevious().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.9.9 APITraceIterator_GetAt()

Prototype

```
APITraceIterator_GetAt(APITraceIterator *api_trace_iterator, int index,
                       APITracePacket *api_trace_packet)
```

Description
It returns packet in the input index.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

`index:` Index of record. It is a zero-based index.

`api_trace_packet:` Packet container.

Table 71: Error codes for APITraceIterator_GetAt().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.
API_ERROR_INVALID_INDEX_OF_PACKET	15	Invalid index of packet.

5.9.10 APITraceIterator_GetCount()

Prototype

```
APITraceIterator_GetCount(APITraceIterator *api_trace_iterator, int *count)
```

Description
It returns the number of packets.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

`count:` Packet count.

Table 72: Error codes for APITraceIterator_GetCount().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.9.11 APITraceIterator_ToBack()

Prototype

```
APITraceIterator_ToBack(APITraceIterator *api_trace_iterator)
```

Description
It moves the iterator pointer to the last packet.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

Table 73: Error codes for APITraceIterator_ToBack().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.9.12 APITraceIterator_ToFront()

Prototype

```
APITraceIterator_ToFront(APITraceIterator *api_trace_iterator): void
```

Description
It moves the iterator pointer to the first packet.

Parameters

`api_trace_iterator:` Pointer to the trace iterator object.

Table 74: Error codes for APITraceIterator_ToFront().

API error code	Value	Description
API_ERROR_INVALID_OBJECT	2	The API object is not valid.
API_EXCEPTION	1	An exception occurred.

5.10 Examples

The following is a simple example for using the API to iterate through all the packets in a trace file. For complete examples, please refer to the API examples named "Trace" and "Project" which are provided with the Net Protocol Suite installer. This example is located in the directory.

```
"...\API\Examples\Project" and "...\API\Examples\Trace"
```

6. How to Contact Teledyne LeCroy

Type of Service	Contact
Call for technical support	US and Canada: 1 (800) 909-2282 Worldwide: 1 (408) 653-1260
Fax your questions	Worldwide: 1 (408) 727-6622
Write a letter	Teledyne Lecroy Protocol Solutions Group Customer Support 3385 Scott Blvd. Santa Clara, CA 95054-3115
Send e-mail	psgsupport@lecroy.com
Visit Teledyne LeCroy’s website	teledynelecroy.com/