

# Ellisys Bluetooth Analyzer Remote Control User Guide

**Version:** 2.1

**Release date:** April 17, 2024

## Table of Contents

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Revisions History.....</b>   | <b>5</b> |
| <b>2</b> | <b>Copyright and Intellectual Property.....</b>   | <b>5</b> |
| 2.1      | Copyright Disclaimer  | 5        |
| 2.2      | Intellectual Property Disclaimer  | 5        |
| 2.3      | Open Source Disclaimer  | 5        |
| <b>3</b> | <b>Introduction .....</b>   | <b>6</b> |
| <b>4</b> | <b>Installing the Remote Control plugin .....</b>   | <b>6</b> |
| <b>5</b> | <b>Running the provided samples .....</b>   | <b>8</b> |
| 5.1      | C# / .net sample  | 8        |
| 5.2      | Python sample   | 8        |
| 5.3      | Using a different language or platform  | 9        |
| <b>6</b> | <b>Overriding Connection Properties .....</b>   | <b>9</b> |
| 6.1      | By file   | 9        |
| 6.2      | By command line   | 9        |
| 6.3      | Priority order  | 9        |
| <b>7</b> | <b>API reference .....</b>  | <b>9</b> |
| 7.1      | General-purposes Functions (AnalyzerRemoteControl.ice)                                    | 9        |
| 7.1.1)   | string[] GetAvailableDataSources() .....  | 9        |
| 7.1.2)   | void SelectDataSource(string dataSourceUniqueId) .....                                    | 10       |
| 7.1.3)   | string GetSelectedDataSource().....   | 10       |
| 7.1.4)   | bool IsRecording() .....  | 10       |
| 7.1.5)   | RecordingStatus GetRecordingStatus().....   | 10       |
| 7.1.6)   | void StartRecording() .....   | 10       |
| 7.1.7)   | void StopRecordingAndSaveTraceFile(string filename, bool overwrite) .....                 | 10       |
| 7.1.8)   | void AbortRecordingAndDiscardTraceFile().....   | 11       |
| 7.1.9)   | void ConfigureRecordingOptions(string options) .....                                      | 11       |
| 7.1.10)  | string GetRecordingOptions(bool relevantOnly) .....                                       | 11       |
| 7.1.11)  | void InsertMessage(MessageSeverity severity, string message) .....                        | 11       |
| 7.1.12)  | void InsertComment(string comment, string overviewName).....                              | 12       |
| 7.1.13)  | bool IsLoading() .....  | 12       |
| 7.1.14)  | void StartLoading(string filename).....   | 12       |
| 7.1.15)  | TraceFileInfo GetTraceFileInfo() .....  | 13       |
| 7.1.16)  | void CloseTraceFile() .....   | 13       |
| 7.1.17)  | bool IsModified() .....   | 13       |
| 7.1.18)  | void SaveChanges() .....  | 13       |
| 7.1.19)  | void AddMarkerOnSelectedOverviewItem(AddMarker marker).....                               | 13       |
| 7.1.20)  | void AddMarkerAtTime(long timeInPicoseconds, AddMarker marker) .....                      | 13       |
| 7.1.21)  | GetMarker[] GetMarkers() .....  | 13       |
| 7.1.22)  | void Export(string outputFilename, string exportName, ExportOption[] exportOptions) ..... | 13       |

|         |   |    |
|---------|---|----|
| 7.1.23) | string[] GetAvailableOverviews()  | 14 |
| 7.1.24) | string GetSelectedOverview()  | 14 |
| 7.1.25) | void SelectOverview(string overviewName)  | 14 |
| 7.1.26) | string[] GetAvailableProtocolLayers()   | 15 |
| 7.1.27) | string GetSelectedProtocolLayer()   | 15 |
| 7.1.28) | void SelectProtocolLayer(string protocolLayerName)  | 15 |
| 7.1.29) | string GetOverviewQuery()   | 15 |
| 7.1.30) | void SetOverviewQuery(string query)   | 15 |
| 7.1.31) | int OverviewRootItem()  | 15 |
| 7.1.32) | string GetOverviewItemDescription(int itemHandle)   | 15 |
| 7.1.33) | string[] GetOverviewItemsDescription(int[] itemHandles)   | 16 |
| 7.1.34) | long GetOverviewItemTimeInPicoseconds(int itemHandle)   | 16 |
| 7.1.35) | long[] GetOverviewItemsTimeInPicoseconds(int[] itemHandles)   | 16 |
| 7.1.36) | byte[] GetOverviewItemData(int itemHandle)  | 16 |
| 7.1.37) | byte[][] GetOverviewItemsData(int[] itemHandles)  | 16 |
| 7.1.38) | string GetOverviewItemXmlReport(int itemHandle)   | 17 |
| 7.1.39) | string[] GetOverviewItemsXmlReport(int[] itemHandles)   | 17 |
| 7.1.40) | string GetOverviewItemXmlReportFiltered(int itemHandle, string[] fieldNameFilters)  | 17 |
| 7.1.41) | string[] GetOverviewItemsXmlReportFiltered(int[] itemHandles, string[] fieldNameFilters)  | 17 |
| 7.1.42) | int GetOverviewItemChildCount(int itemHandle)   | 18 |
| 7.1.43) | int GetOverviewItemChild(int itemHandle, int childIndex)  | 18 |
| 7.1.44) | int[] GetOverviewItemChildren(int itemHandle, int childIndex, int childCount)   | 18 |
| 7.1.45) | int[] SearchOverviewItems(int itemHandle, int childIndex, int childCount, int maxDepth, string[] descriptionFilters, string[] fieldNameFilters, string[] fieldValueFilters)                           | 18 |
| 7.1.46) | void ReleaseAllOverviewItemHandles()  | 19 |
| 7.1.47) | void SelectOverviewItem(int itemHandle)   | 19 |
| 7.1.48) | int GetSelectedOverviewItem()   | 19 |
| 7.1.49) | AppInfo GetAppInfo()  | 20 |
| 7.1.50) | void ExitApp()  | 20 |
| 7.1.51) | RunningTask[] GetRunningTasks()   | 20 |
| 7.1.52) | void AbortRunningTask(string name)  | 20 |
| 7.1.53) | void ConfigureSettings(byte[] settings)   | 20 |
| 7.1.54) | byte[] GetSettings()  | 20 |
| 7.1.55) | void CancelUserInteraction()  | 20 |
| 7.2     | Bluetooth-specific Functions (BluetoothAnalyzerRemoteControl.ice)   | 21 |
| 7.2.1)  | void SplitTraceFileAndContinueRecording(string filename)  | 21 |
| 7.2.2)  | void AddLinkKey(long bdaddr1, long bdaddr2, byte[] linkKey)   | 21 |
| 7.2.3)  | void ConfigureDeviceFilter(DeviceFilterMode mode, DeviceAddress[] deviceAddrs)  | 21 |
| 7.2.4)  | void GetLogicSignalsState(long timeInPicoseconds, out int logicSignalsState)  | 21 |
| 7.2.5)  | void FindLogicSignalsTransition(long fromTimeInPicoseconds, long toTimeInPicoseconds, int signalsMask, LogicSignalTransitionType transitionType, out int foundState, out long foundTimeInPicoseconds) | 21 |
| 7.2.6)  | void GetSpectrumRssi(long timeInPicoseconds, int rfChannelNumber, out double rssi, out long startTimeInPicoseconds, out long stopTimeInPicoseconds)   | 22 |

7.2.7) void GetSpectrumRssiRange(long fromTimeInPicoseconds, long toTimeInPicoseconds, int rfChannelNumber, out Rssi[] rssi, out long startTimeInPicoseconds, out long stopTimeInPicoseconds)..... 22

7.2.8) ChannelsSummary GetChannelsSummary(long fromTimeInPicoseconds, long toTimeInPicoseconds) ..... 22

7.2.9) void ExportAudio(string outputDirectory) ..... 23

## 1 Revisions History

| Date          | Rev | Changes   |
|---------------|-----|---|
| June 21, 2007 | 1.0 | Original version.                               |
| Aug. 16, 2019 | 2.0 | New format.                                     |
| Feb. 17, 2023 | 2.1 | Added Export and CancelUserInteraction methods. |

## 2 Copyright and Intellectual Property

### 2.1 Copyright Disclaimer

Copyright © Ellisys 2024. All Rights Reserved.

No part of this document or any of its contents may be reproduced, copied, modified or adapted, without the prior written consent of Ellisys.

### 2.2 Intellectual Property Disclaimer

This document is provided to you “as is” with no warranties whatsoever, including any warranty of merchantability, non-infringement, or fitness for any particular purpose. Ellisys disclaims all liability, including liability for infringement of any proprietary rights, relating to use or implementation of information in this document. The provision of this document to you does not provide you with any license, express or implied, by estoppel or otherwise, to any intellectual property rights.

### 2.3 Open Source Disclaimer

This Ellisys analysis software automation API plugin is based on a third-party networking library from ZeroC named ICE (<https://zeroc.com/products/ice>). This library is licensed under the GNU GPL v2 or later. This plugin is also licensed under the GPL, and its source code can be requested at [gpl@ellisys.com](mailto:gpl@ellisys.com).

This plugin is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

### 3 Introduction

This Ellisys analysis software automation API plugin is based on a third-party networking library from ZeroC named ICE (<https://zeroc.com/products/ice>). In this context, the Ellisys analysis software is the server and the user script is the client. The Ellisys software will run on a Windows OS and will be accessible from any client on the network. The samples are provided in C# .NET and Python but the ZeroC tools can be used to create wrappers for various languages and platforms. Please refer to the section below for instructions to access the API from other languages and platforms supported by the ZeroC ICE library.

The automation API can be used for very diverse and advanced tasks such as:

- *Common:* Controlling the capture (start / stop capture), saving traces and loading traces.
- *Common:* Selecting the data source, to control multiple analyzers accessible from the same machine.
- *Common:* Accessing the Overviews for parsing the traffic and detecting protocol conditions (either during live capture or with a pre-captured trace).
- *Common:* Adding markers at specific time or on specific protocol items, to flag conditions to be reviewed later manually.
- *Bluetooth:* Injecting known link keys.
- *Bluetooth:* Parsing logic signals captured with the integrated logic analyzer, for example to detect an electrical condition on an external signal, then look at the protocol traffic at that specific time.
- *Bluetooth:* Parsing spectrum information, for example to determine if a retransmission is related to an interference.
- *Bluetooth:* Exporting captured audio traffic.

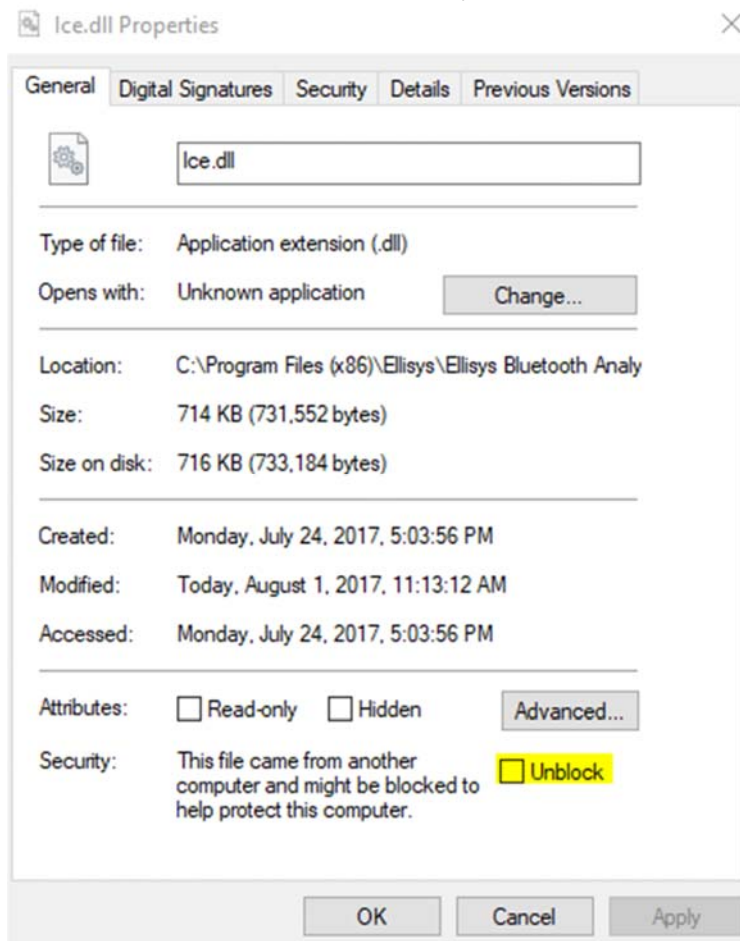
### 4 Installing the Remote Control plugin

The remote control plugin is available on Ellisys website and is linked from the user manual of the analysis software. Please follow these steps to install the plugin:

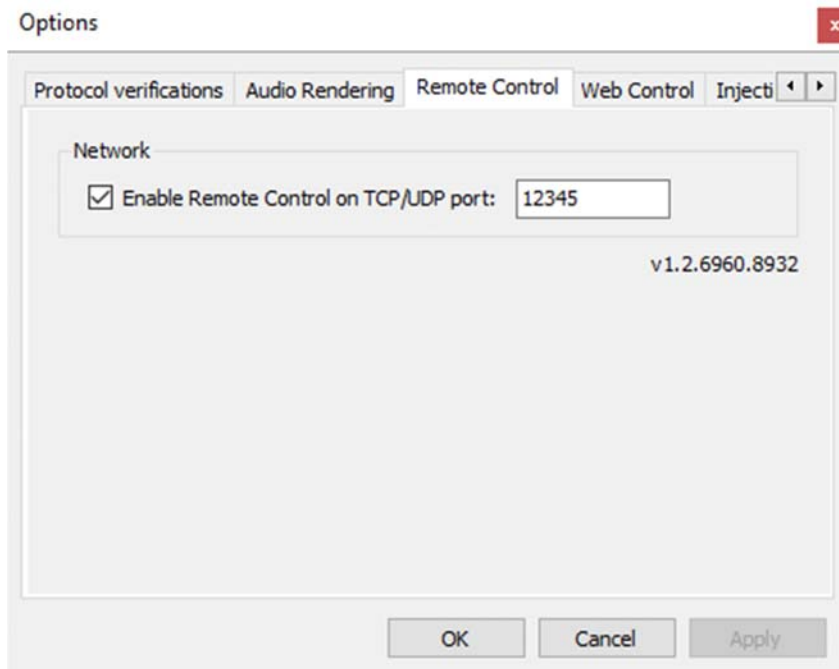
- 1) Install the latest analysis software release.
- 2) Copy the `Binary\RemoteControl` folder and its content to the installation directory, typically `C:\Program Files (x86)\Ellisys\...Analyzer\RemoteControl`.

Please careful that Windows tends to block files downloaded from the Internet. Please

make sure to unblock the files in the Properties as follow:



- 3) Start the analysis application and go to menu "Tools", then "Options...". If step 2 above was successful you should see a panel named "Remote Control".



- 4) Check the box "Enable Remote Control on TCP/UDP port", and optionally change the port according to your network parameters. The software is now ready to accept incoming commands.

The port can be overridden by the command line with `/remote_control_port=54321`.

## 5 Running the provided samples

### 5.1 C# / .net sample

- 1) The `Samples` folder contains a solution compatible with Visual Studio 2017 or higher.
- 2) Follow these steps to run the sample:
  - a. Start the analysis software previously configured in the above section. Please ensure an analyzer unit is connected to this computer.
  - b. Compile and execute the provided sample from Visual Studio.
  - c. The analysis software should start recording and will wait a key before to stop recording.
- 3) The provided sample can be used as starting point for custom control software.

### 5.2 Python sample

- 1) Install the latest Python environment from <https://www.python.org/downloads/>



- 2) Install the ZeroC ICE package with the command: `python -m pip install zeroc-ice`
- 3) Follow these steps to run the sample:
  - a. Start the analysis software previously configured in the above section. Please ensure an analyzer unit is connected to this computer.
  - b. Run the sample with the command: `python sample.py`
  - c. The analysis software should start recording and will wait a key before to stop recording.
- 4) The provided sample can be used as starting point for custom control software.

### 5.3 Using a different language or platform

The sample is provided with language-agnostic definition of the Ellisys API. The ICE files can be converted to the various supporting languages such as Ruby, JavaScript, and more, with a tool named slice provided by ZeroC. Please consult ZeroC documentation for creating a wrapper for your language and platform of choice:

<https://doc.zeroc.com/ice/latest/the-slice-language/slice-compilation>

## 6 Overriding Connection Properties

### 6.1 By file

ZeroC ICE server offers plenty of parameters to optimize different applications. It is possible to override the default properties by placing a file named `Ice.props` in the `RemoteControl` folder. The file shall be of this format:

<https://doc.zeroc.com/display/Ice35/Configuration+File+Syntax>

### 6.2 By command line

It is also possible to override properties with the command line. Any ICE property can be set on the command line, such as `/Ice.ThreadPool.Server.SizeMax=8`.

### 6.3 Priority order

The priority is given as follow:

1. Defaults in the application
2. File `Ice.props`
3. Command line

## 7 API reference

### 7.1 General-purposes Functions (AnalyzerRemoteControl.ice)

#### 7.1.1) `string[] GetAvailableDataSources()`

Returns the available analyzers accessible by the analysis software.

Equivalent to list of the Record menu > Select analyzer dialog.

**Returns:** Array of available data sources.

#### 7.1.2) void SelectDataSource(string dataSourceUniqueId)

Select the active data source for capturing data. This is typically a string returned from the GetAvailableDataSources() function.

Equivalent to selecting an analyzer from the Record menu > Select analyzer dialog.

**Parameters:**

- dataSourceUniqueId: unique ID of the data source to select

#### 7.1.3) string GetSelectedDataSource()

Returns the currently selected analyzers, if any.

**Returns:** Analyzer currently selected.

#### 7.1.4) bool IsRecording()

Indicates if a recording is currently running.

**Returns:** true if the analyzer is currently capturing data following the call of the StartRecording() method, false otherwise

#### 7.1.5) RecordingStatus GetRecordingStatus()

Returns information on the running recording, such as the analyzer, the duration and trace size.

**Returns:** Status of the recording

#### 7.1.6) void StartRecording()

Starts the analyzer capture with the current settings. The settings can be either configured manually in the GUI software or configured with the ConfigureSettings() method.

Equivalent to Record > Start recording in the GUI software.

#### 7.1.7) void StopRecordingAndSaveTraceFile(string filename, bool overwrite)

Stops the analyzer capture and save the resulting file to the specified filename.

Equivalent to Record > Stop recording followed by File > Save in the GUI software.

**Parameters:**

- filename: path to the file to be saved, based on the remote computer
- overwrite: destination filename will be replaced if already exists

#### 7.1.8) void AbortRecordingAndDiscardTraceFile()

Aborts the ongoing capture and discard any captured data.

Equivalent to Record > Stop recording followed by File > New in the GUI software.

#### 7.1.9) void ConfigureRecordingOptions(string options)

Configure the recording options of the data source. The provided settings can be saved from the Recording Options dialog of the GUI software or can be created manually.

The format of the options string is a modified JSON format with the following customizations to the standard JSON format:

- The root object braces can be omitted. For example, "{ field:value }" can be optionally be expressed as "field:value" for brevity.
- The name double quotes can be omitted. For example, "field":"value" can optionally be expressed as field: "value" for brevity.
- Encapsulated parent objects having a single child object can use the '.' hierarchical notation. For example, "a:{ b:value }" can optionally be expressed as "a.b:value" for brevity.

The provided string can contain only the fields to be updated.

Equivalent to Record > Recording Options > Load in the GUI software.

##### Parameters:

- options: recording options in extended JSON format

#### 7.1.10) string GetRecordingOptions(bool relevantOnly)

Retrieve the current recording options in JSON format.

Equivalent to Record > Recording Options > Save in the GUI software.

##### Parameters:

- relevantOnly: indicates if the function shall return only the fields which are set with non-default values, or if default values shall be returned as well.

**Returns:** JSON formatted-string containing the recording options

#### 7.1.11) void InsertMessage(MessageSeverity severity, string message)

Insert a message in the Message Log overview at the current time. This function can only be called during recording.

The message can either be raw text, or hierarchically formatted text in JSON or XML. The JSON format is as follow:

```
{
  fields: [
```

```

    { field:"field1", value:[
      { field:"field2", value:"value2" },
      { field:"field3", value:"value3" }
    ]
  }
]
}

```

The XML format is as follow:

```

<fields>
  <field name="field1">
    <field name="field2">value2</field>
    <field name="field3">value3</field>
  </field>
</fields>

```

**Parameters:**

- severity: indicates if this message is an information, a warning or an error
- message: the message that will be inserted, either in raw text, JSON or XML

#### 7.1.12) void InsertComment(string comment, string overviewName)

Insert a comment item in the selected overview at the current time. This function can only be called during recording.

**Obsolete:** This method is deprecated, use InsertMessage instead.

**Parameters:**

- comment: text of the comment
- overviewName: name of the overview where the comment shall be inserted

#### 7.1.13) bool IsLoading()

Indicates if the software is loading a trace.

**Returns:** true if the file is loading, false otherwise.

#### 7.1.14) void StartLoading(string filename)

Starts loading the specified filename.

Equivalent to File > Open in the GUI software.

**Parameters:**

- filename: path to the file to be loaded, based on the remote computer

#### 7.1.15) `TraceFileInfo GetTraceFileInfo()`

Returns information on the loaded trace file, such as the size, start date, analyzer used and software version.

**Returns:** Information on the loaded trace file

#### 7.1.16) `void CloseTraceFile()`

Close the loaded trace file. Changes will be lost if any, unless saved explicitly with `SaveChanges()`.

Equivalent to File > New in the GUI software.

#### 7.1.17) `bool IsModified()`

Indicates if the open file has changes. The changes can be saved with `SaveChanges()`.

**Returns:** true if the open file has changes, false otherwise

#### 7.1.18) `void SaveChanges()`

Save changes in the currently open file.

Equivalent to File > Save in the GUI software.

#### 7.1.19) `void AddMarkerOnSelectedOverviewItem(AddMarker marker)`

Adds a marker on the currently selected item in the Overview.

Equivalent to main toolbar > Markers > Mark selected item in the GUI software.

**Parameters:**

- marker: information on the marker to be added

#### 7.1.20) `void AddMarkerAtTime(long timeInPicoseconds, AddMarker marker)`

Adds a marker at the specified time.

Equivalent to Instant Timing > Right Click Menu > Add new marker here in the GUI software.

**Parameters:**

- timeInPicoseconds: the time in picoseconds of the marker to be added
- marker: information on the marker to be added

#### 7.1.21) `GetMarker[] GetMarkers()`

Get all markers of the trace file.

#### 7.1.22) `void Export(string outputFilename, string exportName, ExportOption[] exportOptions)`

Export the specified data to the output filename with the specified options. The options are optional and are specific to the export mode. The following export modes are supported:

- filtered\_trace\_time\_range
  - Equivalent export to “Filtered trace based on time range” in the GUI.
  - Export Options:
    - StartTime: relative start time, in picoseconds
    - MaxSize: maximum exported trace size, in bytes
    - MaxDuration: maximum exported trace duration, in picoseconds
    - MaxItems: maximum number of items in the export trace
- filtered\_trace\_active\_overview
  - Equivalent export to “Filtered trace based on active overview” in the GUI.
  - Export Options:
    - StartTime: relative start time, in picoseconds
    - MaxSize: maximum exported trace size, in bytes
    - MaxDuration: maximum exported trace duration, in picoseconds
    - MaxItems: maximum number of items in the export trace
- bluetooth\_audio
  - Equivalent export to “Bluetooth Audio” in the GUI.
  - Export Options:
    - SynchroBufferMilliseconds: synchro buffer size, in milliseconds
    - AddRawPayloadFiles: boolean value indicating if raw files shall be stored in addition to the decoded files
- bluetooth\_mobile\_phone\_data
  - Equivalent export to “Bluetooth Mobile Phone Data” in the GUI.
  - Export Options:
    - No options

#### 7.1.23) string[] GetAvailableOverviews()

Get the list of available overviews.

**Returns:** the list of available overviews

#### 7.1.24) string GetSelectedOverview()

Retrieve the active overview.

**Returns:** the name of the active overview

#### 7.1.25) void SelectOverview(string overviewName)

This method makes a given overview active. All the methods for navigating the overview tree structure will work in the active overview so it is required to first ensure the overview is active. For example SelectOverview(“USB 3.0”) will make the USB 3.0 Overview active.

**Parameters:**

- overviewName: Name of the overview to select

#### 7.1.26) `string[] GetAvailableProtocolLayers()`

Get the list of available protocol layers.

**Returns:** the list of available protocol layers

#### 7.1.27) `string GetSelectedProtocolLayer()`

Retrieve the current protocol layer in the active overview.

**Returns:** the selected protocol layer

#### 7.1.28) `void SelectProtocolLayer(string protocolLayerName)`

This method will switch the protocol layer in the active overview. It is equivalent to clicking the protocol layers buttons in the GUI.

**Parameters:**

- `protocolLayerName`: Name of the protocol layer to select, as displayed in the popup on the button in the GUI

#### 7.1.29) `string GetOverviewQuery()`

Retrieve the current query in the active overview.

**Returns:** the current query if any, empty string otherwise

#### 7.1.30) `void SetOverviewQuery(string query)`

Applies the specified query string to the active overview. This enables filtering the overview root items with the specified conditions.

Equivalent to Search > Filter in the GUI software.

**Parameters:**

- `query`: the query to apply

#### 7.1.31) `int OverviewRootItem()`

The root item is a conceptual item that contains all the items displayed in the overview.

The overview content is a tree structure where each item can have sub-items, called children. Fully traversing such a tree structure can be implemented as a recursive iteration. The root item is not a visible item and is just used of the purpose of iterating the tree structure.

**Returns:** handle to the active overview's root item

#### 7.1.32) `string GetOverviewItemDescription(int itemHandle)`

This method returns the string of the Item column for a given item handle.

**Parameters:**

- itemHandle: handle of the item, either from OverviewRootItem() or GetOverviewItemChild()

**Returns:** string of the Item column for a given item handle

7.1.33) `string[] GetOverviewItemsDescription(int[] itemHandles)`

This method returns the strings of the Item column for the specified item handles.

**Parameters:**

- itemHandles: handles of the items

**Returns:** strings of the Item column for the specified item handles

7.1.34) `long GetOverviewItemTimeInPicoseconds(int itemHandle)`

This method returns the time in picoseconds for a given item handle.

**Parameters:**

- itemHandle: handle of the item, either from OverviewRootItem() or GetOverviewItemChild()

**Returns:** time in picoseconds for a given item handle

7.1.35) `long[] GetOverviewItemsTimeInPicoseconds(int[] itemHandles)`

This method returns the times in picoseconds for the specified items handles.

**Parameters:**

- itemHandle: handle of the items

**Returns:** time in picoseconds for the specified items handles

7.1.36) `byte[] GetOverviewItemData(int itemHandle)`

This method returns the data displayed in the Raw data view for the specified item handle.

**Parameters:**

- itemHandle: handle of the item, either from OverviewRootItem() or GetOverviewItemChild()

**Returns:** data displayed in the Raw data view for the specified item handle

7.1.37) `byte[][] GetOverviewItemsData(int[] itemHandles)`

This method returns the data displayed in the Raw data view for the specified items handles.

**Parameters:**

- itemHandles: handles of the items



**Returns:** data displayed in the Raw data view for the specified item handle

#### 7.1.38) `string GetOverviewItemXmlReport(int itemHandle)`

This method returns the content of the Details view as XML for the specified item handle. This XML can then be decoded by using any standard XML parser.

**Parameters:**

- `itemHandle`: handle of the item, either from `OverviewRootItem()` or `GetOverviewItemChild()`

**Returns:** content of the Details view as XML for the specified item handle

#### 7.1.39) `string[] GetOverviewItemsXmlReport(int[] itemHandles)`

This method returns the content of the Details view as XML for the specified items handles. This XML can then be decoded by using any standard XML parser.

**Parameters:**

- `itemHandle`: handle of the items

**Returns:** content of the Details view as XML for the specified items handles

#### 7.1.40) `string GetOverviewItemXmlReportFiltered(int itemHandle, string[] fieldNameFilters)`

This method returns the content of the Details view as XML for the specified item handle as above, but only the fields matching the `fieldNameFilters` will be returned.

**Parameters:**

- `itemHandle`: handle of the item, either from `OverviewRootItem()` or `GetOverviewItemChild()`
- `fieldNameFilters`: list of inclusion filters. The filters can be specific field names, and accept wildcards such as `*` and `?`, or a regular expression, in which case the string shall start by `regex:` followed by the regular expression. More information about the supported regex format: <https://docs.microsoft.com/en-us/dotnet/api/system.text.regularexpressions.regex>

**Returns:** content of the Details view as XML for the specified item handle

#### 7.1.41) `string[] GetOverviewItemsXmlReportFiltered(int[] itemHandles, string[] fieldNameFilters)`

This method returns the content of the Details view as XML for the specified items handles as above, but only the fields matching the `fieldNameFilters` will be returned.

**Parameters:**

- `itemHandles`: handle of the items
- `fieldNameFilters`: list of inclusion filters. The filters can be specific field names, and accept wildcards such as `*` and `?`, or a regular expression, in which case the string shall start by `regex:` followed by the regular expression. More information about the supported regex format: <https://docs.microsoft.com/en-us/dotnet/api/system.text.regularexpressions.regex>

**Returns:** content of the Details view as XML for the specified items handles

#### 7.1.42) `int GetOverviewItemChildCount(int itemHandle)`

This method returns the child count for the specified item handle.

**Parameters:**

- `itemHandle`: handle of the item, either from `OverviewRootItem()` or `GetOverviewItemChild()`

**Returns:** child count for the specified item handle

#### 7.1.43) `int GetOverviewItemChild(int itemHandle, int childIndex)`

This method returns the child item handle corresponding to a given index for the specified item handle.

**Parameters:**

- `itemHandle`: handle of the item, either from `OverviewRootItem()` or `GetOverviewItemChild()`

**Returns:** child item handle corresponding to a given index for the specified item handle

#### 7.1.44) `int[] GetOverviewItemChildren(int itemHandle, int childIndex, int childCount)`

This method returns the quantity of requested children items handles starting from the specified index for the specified item handle.

**Parameters:**

- `itemHandle`: handle of the item, either from `OverviewRootItem()` or `GetOverviewItemChild()`
- `childIndex`: index of the first child to be returned
- `childCount`: count of children to be returned

**Returns:** child item handle corresponding to a given index for the specified item handle

#### 7.1.45) `int[] SearchOverviewItems(int itemHandle, int childIndex, int childCount, int maxDepth, string[] descriptionFilters, string[] fieldNameFilters, string[] fieldValueFilters)`

This method searches the specified filters in the children items of the specified parent item, going through the specified max depth.

The value and field name filters can be used as follow:

- If `fieldNameFilters` is null, then the item will be kept if any field value matches any value filter.
- If `fieldNameFilters` is not null and `fieldValueFilters` is null, then the item will be kept if any field name matches any name filter.

- If both `fieldNameFilters` and `fieldValueFilters` are not null, then both arrays must be of the same size. If a field matches a field name filter, the field's value will be matched against the corresponding index in the value filters. So in other terms, `fieldNameFilters[0]` is related to `fieldValueFilters[0]`, `fieldNameFilters[1]` is related to `fieldValueFilters[1]`, etc. The item will be retained if any field matches any filter.

**Parameters:**

- `itemHandle`: handle of the item, either from `OverviewRootItem()` or `GetOverviewItemChild()`
- `childIndex`: index of the first child of the item to search from
- `childCount`: quantity of children of the item to search into
- `maxDepth`: maximum depth to go through the children. For example, a value of 1 will return only the direct children of the item.
- `descriptionFilters`: inclusion filters matching the item description. The filters accepts wildcards such as `*` and `?`, or a regular expression, in which case the string shall start by `regex:` followed by the regular expression. Can be null if no filtering is required.
- `fieldNameFilters`: inclusion filters matching a fields names in the report. The filters accepts wildcards such as `*` and `?`, or a regular expression, in which case the string shall start by `regex:` followed by the regular expression. Can be null if no filtering is required.
- `fieldValueFilters`: inclusion filters matching fields values in the report. The filters accepts wildcards such as `*` and `?`, or a regular expression, in which case the string shall start by `regex:` followed by the regular expression. Can be null if no filtering is required. More information about the supported regex format: <https://docs.microsoft.com/en-us/dotnet/api/system.text.regularexpressions.regex>

**Returns:** children items handles for the specified parent item handle, through the specified max depth, and matching the specified filters

**7.1.46) void ReleaseAllOverviewItemHandles()**

This method releases all item handles allocated by `GetOverviewItemChild()` or `GetOverviewItemChildren()`. It is useful to release allocated handles after allocating a large quantity of handles to avoid excessive memory consumption in the analysis application.

**7.1.47) void SelectOverviewItem(int itemHandle)**

Highlights the specified item in the active overview.

**Parameters:**

- `itemHandle`: Item to be highlighted

**7.1.48) int GetSelectedOverviewItem()**

Returns the selected overview item handle.

**Returns:** Item handle highlighted in the active overview

#### 7.1.49) ApplInfo GetAppInfo()

Returns information about the remote application, such as version, file extension, etc.

**Returns:** Application info.

#### 7.1.50) void ExitApp()

Exits the remote application.

#### 7.1.51) RunningTask[] GetRunningTasks()

Returns the currently running tasks of the software, such as recording, saving, filtering, loading, etc.

Equivalent to the Tasks view.

**Returns:** Array of running tasks.

#### 7.1.52) void AbortRunningTask(string name)

Aborts the specified task if running.

Equivalent to canceling a task in the Tasks view.

**Parameters:**

- name: Name of the task to be aborted

#### 7.1.53) void ConfigureSettings(byte[] settings)

Configure the software with the specified settings. The provided settings can be saved directly from the GUI software in order to create a few different configurations, or can be created dynamically.

Equivalent to File > Import and Export settings > Import settings in the GUI software.

**Parameters:**

- settings: content of a file created from File > Import and Export settings > Export settings

#### 7.1.54) byte[] GetSettings()

Retrieve the current settings.

Equivalent to File > Import and Export settings > Export settings in the GUI software.

**Returns:** Settings in binary form

#### 7.1.55) void CancelUserInteraction()

Cancels any message that might block the UI waiting for an user input.

## 7.2 Bluetooth-specific Functions (BluetoothAnalyzerRemoteControl.ice)

### 7.2.1) void SplitTraceFileAndContinueRecording(string filename)

This method will save the ongoing trace to the specific file path and will continue recording. It is equivalent to clicking the "Save & Continue" button in the GUI.

**Parameters:**

- filename: path to the file to be saved, based on the remote computer

### 7.2.2) void AddLinkKey(long bdaddr1, long bdaddr2, byte[] linkKey)

This method will add the specified link key to the devices database for the specified connection. The views will not reload after the call of this function. If a loaded trace is affected by this new link key, it shall be reloaded manually.

**Parameters:**

- bdaddr1: BDADDR of the first device
- bdaddr2: BDADDR of the second device
- linkKey: 16 bytes composing the link key

### 7.2.3) void ConfigureDeviceFilter(DeviceFilterMode mode, DeviceAddress[] deviceAddrs)

This method sets the device filter. It is equivalent to specifying a device filter from the GUI.

**Parameters:**

- mode: specifies if the filter is keep all, exclude background, keep only or keep involving
- deviceAddrs: list of devices to keep. No device shall be specified in case of keep all and exclude background.

### 7.2.4) void GetLogicSignalsState(long timeInPicoseconds, out int logicSignalsState)

This method returns the state of all logic signals at the specified time. The returned logic signals state contain the state of all signals. Bit 0 corresponds to logic signal 0, bit 1 to logic signal 1, etc.

**Parameters:**

- timeInPicoseconds: time at which the logic signals state shall be retrieved
- out logicSignalsState: state of all logic signals at the specified time

### 7.2.5) void FindLogicSignalsTransition(long fromTimeInPicoseconds, long toTimeInPicoseconds, int signalsMask, LogicSignalTransitionType transitionType, out int foundState, out long foundTimeInPicoseconds)

This method searches a certain transition in the specified time range, for the specified signals. The desired signals should be set to 1 in the signalsMask to be included. For example if signals 0, 2, and 3 should be searched, the corresponding binary mask shall be 1101 (bits 0, 2 and 3 set), which is 0xD in hexadecimal.

**Parameters:**

- fromTimeInPicoseconds: time from which to search for the logic transition
- toTimeInPicoseconds: time to which to search for the logic transition
- signalsMask: mask indicating which signals to search
- transitionType: specifies if the searched transition is rising edge, falling edge, or any edge
- out foundState: state of all logic signals at the specified time
- out foundTimeInPicoseconds: time at which the condition is found

7.2.6) void GetSpectrumRssi(long timeInPicoseconds, int rfChannelNumber, out double rssi, out long startTimeInPicoseconds, out long stopTimeInPicoseconds)

This method returns the RSSI level in the captured spectrum at the specified time and on the specified channel. The RSSI value is in dBm.

**Parameters:**

- timeInPicoseconds: Time at which the spectrum RSSI shall be retrieved
- rfChannelNumber: RF channel number, from 0 to 78
- out rssi: Signal strength in dBm
- out startTimeInPicoseconds: Start time in picoseconds of the RSSI sample
- out stopTimeInPicoseconds Stop time in picoseconds of the RSSI sample

7.2.7) void GetSpectrumRssiRange(long fromTimeInPicoseconds, long toTimeInPicoseconds, int rfChannelNumber, out Rssi[] rssi, out long startTimeInPicoseconds, out long stopTimeInPicoseconds)

This method returns the RSSI level in the captured spectrum in the specified time range and on the specified channel.

**Returns:**

- fromTimeInPicoseconds: time from which to get RSSI samples
- toTimeInPicoseconds: time to which to get RSSI samples
- rfChannelNumber: RF channel number, from 0 to 78
- out rssi: Array of signal strength samples in dBm
- out startTimeInPicoseconds: Start time in picoseconds of the first RSSI sample
- out stopTimeInPicoseconds Stop time in picoseconds of last RSSI sample

7.2.8) ChannelsSummary GetChannelsSummary(long fromTimeInPicoseconds, long toTimeInPicoseconds)

This method returns the Bluetooth channels summary as displayed in the Instant Channels view for the specified time range.

**Parameters:**

- fromTimeInPicoseconds: time from which to get channels summary
- toTimeInPicoseconds: time to which to get channels summary

**Returns:** Bluetooth channels summary as displayed in the Instant Channels view

#### 7.2.9) void ExportAudio(string outputDirectory)

This method exports all audio available in the trace to the specified directory. It is equivalent to the File menu > Export > Bluetooth Audio function in the GUI.

**Parameters:**

- outputDirectory: path to the directory to export audio files, based on the remote computer