elecard

VIDEO COMPRESSION GURU

Elecard Probo SDK v.1.0

Introduction and Specification

Notices

Elecard Probo SDK v.1.0 Introduction and Specification

First edition: May 15th, 2017.

Date modified: May 16, 2018.

For information, contact Elecard.

Tel: +7-3822-488-580.

More information can be found at: https://www.elecard.com/

For Technical Support, please contact the Elecard Technical Support Team: tsup@elecard.com.

Elecard provides this publication “as is” without warranty of any kind, either expressed or implied.

This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Elecard may make improvements and/ or changes in the product(s) and/or the program(s) described in this publication at any time.

Other company, product, trademarks, and service names are trademarks or service marks of other companies or corporations.

Copyright ©2017 - 2018 Elecard. All rights reserved.

CONTENTS

1. Introduction

1.1. Preface

Elecard Probo SDK is a new approach in addressing Transport Stream monitoring tasks. In contrast to other players of the market, Elecard does not offer an expensive monitoring solution, but presents cost-efficient monitoring technology. Elecard Probo SDK is the right choice for you when:

- standard monitoring solutions are above allocated budget;
- GUI and notification system do not meet your requirements;
- you would like your Fault Management system to control several monitoring products and make decision based on data gathered;
- you would like to quickly and easily scale up an existing monitoring system.

Elecard Software Development Kit allows creating your own monitoring systems or adding new features to existing projects. Elecard offers quick and easy way to roll out your own system for video quality analysis by offering software probe and you will focus on the second half of the task – processing and visualization of received statistics.

We have travelled a long way scrutinizing stream structures and refining the probe software for you, and now we are proud to offer a new technology for video quality monitoring that saves your time, money and efforts when creating your own monitoring system.

1.2. Description

Elecard Probo SDK is a software development kit designed for implementation of distributed monitoring systems and intended for ongoing control over transport streams and encoded content quality.

Distributed monitoring system is a system consisting of a set of probes located across the network and one or several aggregation centers for statistics presentation. Architecturally, Probo SDK-based products are client-server applications.

Probo probe is a client console application that runs on Windows or Linux operating systems. The application is installed at any point of content production and delivery network to monitor IPTV parameters and OTT streams. A probe can be installed within hardware and software tools for signal generation and processing: transcoders, multiplexers, crypto machines, serializers etc. The probe detects stream errors and sends detailed statistics to the aggregation server. The Probo probe application is included in a software development kit as a set of executable files and libraries designed for different operating systems. The probe registers all main QoS parameters: CC errors, TR 101 290, IAT, Delay Factor (DF), Media Loss Rate (MLR), loss of a service, bitrate calculation and many other options along with detection of QoE parameters, such as VideoFreeze, thumbnails capture and EPSNR (Estimated Peak to Signal Ratio is a metric, developed in Elecard to measure encoded video quality without reference video sequence). The probe can extract service data which has been added to transport streams: SCTE and Closed Caption. See detailed description of the system features in the Section "Registered Parameters".

Aggregation server is a server application (to be created or to be upgraded with Probo related functionality if already exists) that receives data from the probes implemented in monitoring points. Aggregation server prepares data for presentation (graphs, tables, lists etc.) in a browser or specific applications, draws reports and lists available for download, notifies customers via e-mail, SNMP, sms and etc., and makes decision on hardware (SNMP and other methods) and broadcast control. The server part is supposed to be designed and implemented by a Probo SDK user in compliance with the requirements of a broadcast system and covering the needs arising during network utilization. SDK

includes a server application sample controlling the probe and receiving data from it. The application is provided to make Application Programming Interface (API) easier to understand.

Before starting to examine a software development kit, it is recommended to have a look at a sample project implemented using Elecard Probo SDK. The Elecard Boro service helps estimate technical capacity of a monitoring system and understand tasks performed by client and server sides:

- A demo project with several probes (read only);
- Create your account and launch the probe (Boro service Quick Start manual).

1.3. Components of Software Development Kit

1. **Elecard Probo SDK Introduction and Specifications** (the current document).

   The document describes a system in general, its specifications, requirements to software environment, registered parameters and deployment instructions.

2. **Elecard Probo SDK Application Programming Interface (API)**.

   The document describes the client-server application programming interface in detail. The interface is based on JSON RPC. The system server-side components should be developed based on API description.

3. **Probo Probe Software (demo-version).**

   The client software includes a set of executable files and libraries created in C/C++ and complied for 32/64-bit Windows OS and 32/64-bit Linux OS. The "System Requirements" Section describes requirements for operating systems in detail. The "Available versions of SDK" Section describes functionality limitations of the demo-version.

4. **Server Software for Communication with Probo Probes.**

   A simple sample of the server software created in Ruby language illustrates inter-application communication that is described in "Elecard Probo SDK Application Programming Interface (API)". The example can be deployed on Windows, Linux and MacOS. The "Server Sample Deployment" Section describes application setup in detail.

5. **Technical Support**

   1 month of "Professional" technical support is included in the SDK package.

   Technical support under Professional program covers:

   a. Technical consulting on basic features and operations with the Software product;
   b. Detection and first priority fixation of malfunctions.

1.4. Requirements to Developer Software Environment

For seamless SDK usage please make sure to consider operating systems and dependencies stated below.

The client side (**Probo probe**) is provided in a binary mode for Intel x86/x64, for the following OSes:

- Windows 7 and higher;
- Linux libc 2.11 and higher.

A server side is a HTTP server processing POST requests (see the server sample).

The server side should be available for a client side over network. The sample is provided as a source code for MRI (Ruby), and requires interpreter version 2.3 and higher. Interpreters are available for different operating systems but the server sample has been examined by Elecard only on Windows OS, macOS and Linux OS.

2. Licensing

2.1. Available Versions of SDK

	Probo SDK Evaluation	Probo SDK Full	Probo Probe Licensed (OEM contract)
API is provided	Partially	Fully	Fully
HASP protection	No	No	Upon request
HTTPS support	No	No	Upon request
Limitation on the number of Streams for analysis	10 streams	100 streams	Based on license
Operating time limit	1 hour	No	No
Concurrent operation of several probes	Yes	Yes	Yes
Video recording by event	No	Partial support	Yes
QoS	Yes	Yes	Yes
Data garbling[1]	Every 5 minutes	1 time per hour	No
QoE	No	Yes	Yes
Thumbnails	No	Logo on a thumbnail	Yes
Video freeze detection	No	Data garbling: 1 time per hour	Yes
EPSNR	No	Data garbling: 1 time per hour	Yes
SCTE-35	No	Data garbling: 1 time per hour	Yes
Closed Captions extraction	No	Data garbling: 1 time per hour	Yes

2.2. License Grant for Commercial Usage and Distribution

Probo SDK license grants the permission to use the product for R&D purposes: to develop and debug your own solution. For commercial use and/or distribution of Elecard components within your product an appropriate license agreement should be executed. The agreement specifies the major technology usage terms and conditions including obligations of both parties, price, payment and delivery terms.
The license price depends on the functionality licensed and the number of probes required for a project. The following packages are available: Quality of Service (QoS) and Quality of Service (QoS) plus Quality of Experience (QoE).

[1] Data garbling – intentional entering of false data in statistics transferred from a probe. It is implemented to limit commercial use of Probo-probe provided in evaluation and R&D SDK versions.

2.3. Package Functionality

QoS:

- ETSI TR 101 290 (priority 1);
- Clock Continuity;
- Signal Loss / Bad Source;
- Profile/Level Conformance;
- Download/Multicast/Payload Rate;
- Min/Max/Average Bitrate;
- Invalid Elementary Stream Detection;
- PSI/PCR Display & Change Detection;
- ES Video Information Change Detection;
- Multiple Multicast Sources Detection;
- Maximum Inter-Packet Arrival Time (IAT);
- Encryption Map Change Detection;
- HLS Warnings, 23 alarms and events;
- TOS/DSCP Change Detection;
- TTL Change Detection;
- Source / Destination MAC Detection.

QoE:

- Signal Loss / Bad Source;
- Video Freeze;
- Estimated PSNR;
- SCTE35 Extraction;
- Closed Captions Extraction;
- Thumbnails capture by events and time interval;
- Stream Record by request and by events (beta).

For more details on registered parameters and their description, see the Section "Registered Parameters".

To get quote for your project, please contact Elecard Sales Department (sales@elecard.com).

3. Features of Probo System

3.1. System Requirements

Probo probe is provided in a binary mode for Intel x86/x64, for the following OS:

- Windows 7/8/8.1/10 32/64 bit, Server 2008/2012 32/64 bit;
- Linux libc 2.11 and higher.

Note, that WinPcap driver should be installed in Windows OS to calculate Ethernet[2] parameters.

3.2. System Performance

For details on the system performance, see Probe Performance table below.

IPTV		
	RAM (MB) / Stream	**CPU[3] (MHz) / Stream**
QoS[4] SD	12	41.5
QoS + QoE[5] SD	27.1	72.3
QoS HD	12	59
QoS + QoE HD	75	160
OTT		
	RAM (MB) / Stream	**CPU (MHz) / Stream**
QoS SD	45	41.5
QoS + QoE SD	60.1	72.3
QoS HD	45	59
QoS + QoE HD	108	160

[2] Ethernet parameters include: Inter-packet Arrival Time (IAT), Delay Factor (DF), Media Loss Rate (MLR), Type-of-service (TOS/DSCP), Time to live (TTL), destination MAC, source IP/MAC, mapping. The parameters calculation is based on the third-party library libpcap for Linux applications and on winpcap for Windows applications. The probe does not calculate the above mentioned parameters if the corresponding library has not been installed in your OS. Root privileges are required to start a probe in Linux OS (sudo ./streamMonitor).
[3] It is required to consider physical CPU cores and actual CPU frequency to calculate CPU utilization per the required number of streams.
[4] QoS - all the QoS parameters (Ethernet or OTT, Bitrates, TR290, etc).
[5] QoE - Thumbnails 60/5 + Video freeze detection.

3.3. Network Traffic

Network traffic outgoing from Probo is within the range 10-100 Kb/s per one analyzed stream and depends on several factors:

1. if the thumbnail parameter is enabled, traffic increases by 50-150%;
2. the number of PIDs and services (MPTS) in a stream. More PIDs – higher network traffic;
3. the number of errors and notifications. Errors and notifications increase network traffic.

If Probo cannot send a message to a server for some reason, the message is stored in memory for a while (from 10 to 30 minutes depending on buffer size). As soon as the connection is restored, Probo sends the buffer contents to the server. When it happens, the network traffic goes above the stated traffic values.

Network traffic incoming to Probo is approximately equal to 10-20 Kb/s per probe.

3.4. Supported Formats

Protocols: UDP, RTP[6], HTTP, HLS; File (TS).

Video: MPEG-2, AVC/H.264, HEVC/H.265.

Transport: MPEG-2 TS (SPTS/MPTS).

[6] RTP over UDP (multicast). The stream should meet the RTP Payload Format for MPEG1/MPEG2 Video (rfc2250) specification.

4. Registered Parameters

4.1. Thumbnails

Thumbnails – it is possible to capture video thumbnails within the specified time interval. Thumbnails capture interval should be set up for the detected advertisement period indicated with the SCTE-35 messages.

4.2. Parameters, Events and Errors

Mapping – indicates the number of TS per IP packet. Usually, one IP packet contains 7 transport packets.

TOS/DSCP – Type-of-service, field in the IP header.

TTL – Time to live

Src address/Src MAC – IP and MAC addresses of multicast source.

Dst MAC – Destination MAC. IPv4 multicast packets are delivered using the Ethernet MAC address range 01:00:5e:00:00:00–01:00:5e:7f:ff:ff. See more details following the link.

Maximum Inter-packet Arrival Time (IAT) – a graph representing maximum inter-packet arrival time. Packet jitter can be detected by checking inter-packet arrival time. Maximum IAT is defined as a summary of average IAT and jitter. Maximum IAT value is measured every second, in milliseconds. The parameter is described in more detail in the Section "What is Maximum Inter-packet Arrival Time (IAT)" in "Elecard Boro User Guide".

MinIAT – the minimum inter-packet arrival time registered per a second. Calculated in milliseconds.

AvgIAT – the average inter-packet arrival time recalculated each second. Calculated in milliseconds.

MDI Media Delivery Index [Delay factor (DF): Media Loss Rate (MLR)] – an index indicating the quality of video streaming delivery network. The network is sensitive to jitter and data loss. It provides accurate measurement of a stream jitter which defines bitrate fluctuation from the expected values and Media Loss Rate (MLR). Bitrate fluctuation caused by jitter and MLR can be considered as depth of virtual buffer used to buffer received packets of a stream.

Several broadcasters – several broadcasters in one multicast group.

EIT – EIT data is sent to the server.

ProgramSpecificInformation – PAT, PMT and SDT programs description of the analyzed stream is transferred to the server. A service table can be created according to the received data. Information about the contained stream types (the type field), encryption, stream correctness and PCR presence is passed in this event as well.

PCR – (Program Clock Reference) detects synchronized signals in the selected stream. PSI (program specific information) event contains PCR information.

PcrError – an error occurs when PCR timestamps are not found in a stream.

Encoded stream – an elementary stream containing encoded elements. PSI (program specific information) event contains encoded stream information. Video freeze analysis, thumbnails capture and EPSNR calculation are not performed for such streams.

Invalid elementary stream – (Invalid ES) – if ES video content with the specified PID is received by a probe but no video captions (SPS, PPS) are detected during 10-20 seconds then this stream is marked with the Invalid ES sign (invalid data or encoded stream). PSI (program specific information) event

contains invalid ES information. Video freeze analysis, thumbnails capture and EPSNR calculation are not performed for such streams.

VideoInformation – video streams captions are sent to the server. The description contains the following parameters: coding format, resolution, frames per second (FPS), frame size (height-to-weight ratio) and etc.

Download rate – download rate over HTTP/HTTPS.

Multicast Rate – multicast network bitrate of incoming UDP/RTP stream.

Bitrate – current bitrate of all elementary streams contained in MPEG TS. The bitrate is measured as the average value of current bitrate per 1 sec.

Min/Max bitrate – minimum and maximum values of bitrate of elementary streams contained in MPEG TS.

Average bitrate – average bitrate of elementary video streams calculated during 5, 20 and 60 sec.

Info/Stop – the following events are registered: data occurrence at the probe input and the task stop.

BadSource – events that the probe can not receive data for further analysis are registered. The following criteria are applied for different protocols:

- UDP/RTP – no input data for more than one second;
- HLS – the segment download is impossible for the reasons:
    - A playlist has no changes. 3 attempts of the playlist downloading are performed with the interval equal to duration of the last segment. If after three attempts no changes appear in the downloaded playlists, BadSource is registered.
    - for HTTP/HTTPS – zero download rate for particular time (5 sec. in average). During this time data is taken from the input buffer by the probe.

VideoFreeze – analyses video freeze. The analysis is performed in two steps. Size ratio of I frames to P frames is measured in the first step. If the ratio exceeds the stated threshold, both neighboring I frames are decoded and compared by pixels in the second step.

EPSNR – statistical estimation of the digital video content distortion during encoding. It is expressed in dB and defined as a ratio of peak mean square video signal to mean square deviation of the output signal from the original one. EPSNR (Estimated Peak Signal to Noise Rate) value estimation is based on encoded video stream data, i.e no original video content (not encoded) is required. EPSNR is used to estimate encoders performance quality. To estimate it the following values may be used: 25-30 dB – low quality, 45-50 dB – high quality.

SCTE35 – an ad timestamp is registered according to the standard ANSI/SCTE-35. For example, (SCTE35 00:01:01.157 {"event_id"=>662, "duration"=>242, "out_of_network_indicator"=>true, "pts_time"=>89742.159644}).

ClosedCaption – subtitles from video streams are sent to the server. Standards CEA-608 and CEA-708 are supported.

Error – a group of system and general errors:

- **Buffer overflow, data skipped** – data is flushed before decoding. Such situation occurs when the system resources are overloaded. The data is flushed after stream integrity analysis (TR 101 290) and bitrate calculation, thus, the error does not influence the results of stream integrity analysis. This statement is also applicable to OTT, data flush before decoding does not influence calculation of segments download rate and analysis of OTT errors. Data flush can influence calculation of the parameters: VideoFreeze errors, thumbnails capture, EPSNR.

- **Resumption** – if the probe is restarted due to an unexpected stop error, the event is recorded. A parent process monitors all probes performance and uses backward recovery in case of a fatal error.

TR_101_290_errors (priority 1) – a group of errors of the first priority according to ETSI TR 101 290 (a technical report devoted to the inspection of transport stream in compliance with MPEG-2 TS. The standard has been produced by ETSI committee).

- **TS_Sync_Loss** – an error occurring when two or more successive Sync_Byte_Errors are detected (see below). This error disappears after five or more successive sync bytes are received (synchronization is achieved again).
- **Sync_Byte_Error** – occurs when a sync byte 0x47 is missing in the successive packet (after 188 or 204 bytes).
- **PAT_Error** – occurs under the conditions described below:
  - PID 0x0000 does not appear every 0,5 sec. (configurable parameters).
  - PID 0x0000 does not contain a table_id 0x00 (i.e. a PAT).
  - Scrambling_control_field is not equal to 00 for PID 0x0000.
- **Continuity_Count** – occurs under the conditions described below:
  - Incorrect packet order,
  - One and the same packet is received successively more than twice,
  - Packets loss.
- **PMT_Error** – occurs under the conditions described below:
  - Sections with **table_id** 0x02, (i.e. a PMT), cannot be found (configurable parameters) on the PID which is referred to in the PAT at least every 0,5 sec.;
  - **Scrambling_control_field** is not equal to 00 for all PIDs containing table_id 0x02 (i.e. PMT).

  "0" value in thresholds configurations disables PMT Error detection.
- **PID_error** – occurs when data for the selected PID cannot be found during a specified period (default interval is 5 s). It corresponds to partial loss of service or to errors occurred in PAT/PMT. The error can be configured and generated separately for video and audio elementary streams. "0" value in thresholds configurations disables AV PID error / PID error detection.

TR_101_290_errors (priority 2) – a group of errors of the second priority according to ETSI TR 101 290. The parameter is under development.

- **Transport_error** – is registered if the **Transport_error_indicator** field in TS header contains "1".

ClockContinuity – timestamps discontinuity is detected for a video stream. ClockContinuity monitors continuity of the PTS/DTS timestamps, detects backward time shifts and sudden skips in the stream (it is usually related to packet loss and/or, as a result, stream splicing). In contrast to "(ETSI TR 101 290 Second priority 2.5 PTS_error) PTS repetition period more than 700 ms" the ClockContinuity is synchronization timestamps continuity analysis rather than the data presentation within the stated interval (maxPTSInterval). The "0" value in thresholds configurations disables ClockContinuity detection.

4.3. HLS Events and Errors

HlsEvent – the event of data download over the HLS protocol is recorded. Download time and date, caption, size, duration and sequence number of the segment are registered. Download time and a file size define download speed.

Profile changed (HlsBandwidthSwitched) – the event of switching to profile with different bitrate is recorded. It is applied only for the probe in the "Player" mode.

The number of profiles changed (HlsNumberOfProfilesChanged) – the number of profiles in a Master playlist is changed.

Minimum profiles (HlsMinimumProfiles) – the number of profiles stated in the Master playlist is less than the minimum value stated in thresholds configuration.

Profiles sequence divergence (HlsSequenceDivergence) – Media playlists contain divergence in the #EXT-X-MEDIA-SEQUENCE fields.

Profile streamtype changed (HlsProfileStreamTypeChanged) – profile information contained in the Master playlist is changed.

Profile duplicate bandwidth (HlsDuplicateBandwidth) – the Master playlist has two similar maximum bitrates stated for different profiles (the BANDWIDTH fields).

Profile invalid resolution (HlsInvalidResolution) – the Master playlist has zero resolution in the RESOLUTION field.

(LowDownloadrate) – download time of a segment exceeds the segment duration.

Download bitrate low (HlsDownloadSpeed = "Warning") – download bitrate is low. If download speed is lower than the stated Download speed warning (download_speed_warning) thresholds, notification is automatically generated. It is expressed in % and calculated as download time / segment duration >= warning threshold (%). Warning threshold can not exceed error threshold.

Download bitrate too low (HlsDownloadSpeed = "Error") – download bitrate is too low. If download speed is lower than the stated Download speed error (download_speed_error) thresholds, a notification is automatically generated. It is expressed in % and calculated as download time / segment duration >= error threshold (%).

Actual bitrate (HlsActualBitrate = "Error") – average bitrate for a downloaded segment is higher or lower than the bitrate stated in a playlist. Actual bitrate min (actual_bitrate_min) corresponds to the lowest threshold and Actual bitrate max (actual_bitrate_max) corresponds to the highest threshold, in percentage. The Actual bitrate min error is generated when the size of a downloaded segment / stated duration <= stated bitrate of a profile (%). The Actual bitrate max error is generated when size of a downloaded segment / stated duration >= stated bitrate of a profile (%).

Bad segment size (HlsBadSegmentSize) – incorrect segment size. A segment bitrate (segment size / duration) exceeds the maximum bitrate specified in the BANDWIDTH field of a Master playlist by 50.

Manifest sequence discontinuity (HlsSequenceNumberDiscontinuity) – loss of one or more playlists and HLS data is detected. The error is detected only if the subsequent number of the received playlist differs from the previous one by more than one point, and data loss is detected. This error may be caused by OTT content generation and distribution issues or insufficient performance of a probe.

Static manifest (HlsStaticManifest) – the media playlist has not been updated during subsequent downloads. The exact number of download attempts is set by a user in the Number of identical playlist field (sequance_age). A pause equal to download duration of the last segment is made between download attempts.

Manifest error (HlsManifestError) – an error occurs while processing a playlist. The playlist content that could not be parsed is sent back.

Unknown manifest (HlsUnknownManifest) – unknown manifest. All possible causes of an error are sent back.

Manifest size (HlsManifestSize) – a playlist size exceeds the Manifest size (manifest_size) threshold stated by a user.

Manifest download failure (HlsFailedDowloadPlaylist) – an error occurs if a receiver can not download a playlist, and an additional **HlsCurlError** or **HlsHTTPError** is registered.

Key download failure (HlsFailedDownloadKey) – an error defining that key is not received for an encrypted segment, and additional **HlsCurlError** or **HlsHTTPError** is registered.

Curl error (HlsCurlError) –code and description of an error for HLS receipt returned by the libcurl module. For more details see libcurl project.

HTTP error (HlsHTTPError) – an error defining a failure of HLS receipt. The HTTP error code is retuned.

Skip segment (HlsSkipSegment) – a segment is skipped, and a queue of downloaded segments exceeds the stated value. Insufficient performance and a segment download time exceed the segment processing time.

5. Usage

5.1. Introduction

Before starting to read through the development kit it is highly recommended that you look at a project implemented based on Elecard Probo SDK. Elecard Boro service allows evaluating the monitoring system functionality and getting understanding of separation between the client and server sides. Follow the links below:

- A demo project with several probes (read only);
- Create your account and launch the probe (Boro service Quick Start manual).

It is recommended that you work with the sample included in the SDK by going through two steps:

1. Deploy the server sample;
2. Launch and setup a probe that will send data to the server.

The server sample is complementing the document "Probo SDK API" and will help to get better understanding of communication between the server and client applications.

Integrating probes into an existing fault management system or creating new products/systems based on Elecard Probo SDK is considered as the result of SDK examination.

Elecard field engineers will be happy to help with any questions you might have while setting up the probe (tsup@elecard.com) or you might want to consider using Elecard integration services (sales@elecard.com).

5.2. Server Sample Deployment

Linux (Ubuntu OS)

Dependencies installation:

- Download the latest version of Probo SDK;
- Extract zip-archive into a software development folder;
- Open a terminal.

Subsequently enter the following commands in the terminal:

- sudo apt install ruby
- cd ".../[Elecard Probo SDK]/Sample Applications/Simple Server/"
- gem install bundler
- bundle install

Launching a server sample

- cd ".../[Elecard Probo SDK]/Sample Applications/Simple Server/"
- ruby ./main.rb

Linux (CentOS)

The application deployment and start are performed in the same manner as for Ubuntu OS, except ruby packet installation (sudo yum install ruby). The packet 2.4.2. is not currently available in CentOs. That is why it is required to apply alternative installation methods, specifically using Ruby Version Manager. See an example of installation on CentOS by using RVM.

Windows

Dependencies installation for Windows OS:

- Download the latest version of Probo SDK;
- Extract zip-archive into a software development folder;
- Download and install interpreter MRI 2.4.2 (ruby installer);
- Launch the terminal by clicking `Start Command Prompt with Ruby.lnk` in Start menu.

Subsequently enter the following commands in the terminal:

- cd "...\[Elecard Probo SDK]\Sample Applications\Simple Server\"
- gem install bundler
- bundle install

Launching a server sample

- cd "...\[Elecard Probo SDK]\Sample Applications\Simple Server\"
- ruby main.rb

The server sample is started with all available interfaces and is waiting for connection on **the port 8084 by default**. The port or interface can be changed by entering parameters at the application start. See an example for Windows:

```
ruby main.rb --binding=192.168.1.1 --port=8888
```

5.3. Starting a Probe

Before starting a probe, it is required to make sure that a server sample is installed and started, as described in the previous Section. Note, that probes can be started locally (from the PC where the server has been started) and / or remotely. The number of probes to be started is not limited. If it is required to start several probes at the same host, create a folder for each probe and copy the files from the received SDK to it.

Do not start the application several times simultaneously from the same catalogue, it causes the probe malfunction, because applications get concurrent access to some files. For example, if you work with Windows 64bit, create as many folders (the probe copies) win64 as the number of probes to be simultaneously started at the same host.

Starting a probe:

- Open the probe folder (...\[Elecard Probo SDK]\Components\Probo Application\[operating system]) matching your operating system in SDK and copy it to local or remote PC.
- Edit the following fields in the *monitor.cfg* file:
  - "AppDescription" – enter the probe name;
  - "server" – specify IP address and the server port (8084[7] is set by default);
  - "proxy" – delete a comment from this line and set proxy-server, if required.
- Starting a probe
  - Linux: open a console, go to the catalogue containing the probe and perform the *sudo ./streamMonitor* command;
  - Windows: start[8] a probe as an Administrator.

Press **ctrl+c** in the console to stop the probe correctly.

A configuration file does not contain tasks to be analyzed at the first start. To set tasks to the probe, use a sample server or edit the *monitor.cfg* file. See detailed information on how to set tasks by editing a configuration file described in the "Configuration File" Section in "Elecard Boro User Guide".

[7] A port can be changed at a server sample start. See more details in Server Sample Deployment.

[8] Install WinPcap driver for the probe proper operation on Windows OS before the first start.

5.4. Managing a Server

Available commands:

- help -- show help message;
- version -- show software version;
- info probe `probe_id` -- show information about the probe with the specified ID;
- info task `task_id` -- show information about the task with the specified ID;
- task start `probe_id` `url` -- create a new task for the probe:
    - --bind_interface=127.0.0.1 -- bind interface for data receiving, 127.0.0.1 is set by default;
    - --thumbnail=yes|no -- capture thumbnails, “No” is set by default;
    - --video_freeze=yes|no -- detect video freeze, “No” is set by default;
    - --name=task_name -- specify a task (stream) name, null is set by default.
- task stop `probe_id` `task_id` -- stop a task / tasks for the specified probe;
- task stop `probe_id` all -- stop all tasks for the specified probe;
- status server -- show server status;
- probe restart `probe_id` -- restart the probe with the specified ID;
- status probe all -- show status for all connected probes;
- status probe `probe_id` -- show status of the probe with the specified ID;
- status task `task_id` -- show status of the task with the specified ID;
- exit -- exit the application.

If the probe has been successfully started, a task to analyse a stream can be assigned. Perform the *status probe all* command to define ID of the required probe. To start the task, perform the *task start* command. See the example of a task start below:

```
task start 22079 udp://239.0.0.2:1234 --bind_interface=10.10.30.235 --thumbnail=yes --freeze=yes --name=My_first_task
```

5.5. Application Programming Interface (API)

Find a separate document “Application programming interface (API)” included into Elecard Probo SDK package. In addition to these documents, the package also contains server sample in source code for visual representation of data exchange.

5.6. Probe Utilization Know-Hows

A lot of questions on how to use probe are covered in the document “Elecard Boro User Guide”. It contains the following chapters:

- How to launch and setup the probe;
- Probe launch as a service (daemon process);
- Configuration file examples;
- Recommendations for network adapter;
- IAT calculation information;
- Possible issues with multicast receipt and workarounds;
- What happens when hardware resources are insufficient for the task.