8.1 Device Detection

CountBOX CCV provides an ability of network device detection using UPnP protocol. Device information, obtained through UPnP, contains the following parameters:

— ModelName — CountBOX

— PresentationURL — device IP address with protocol information included

— ModelNumber — firmware revision

— SerialNumber — serial number.

The folder **DeviceBrowser** folder of the **examples.zip** archive contains **DeviceBrowser** class source code, which is used for CountBOX CCV device network discovery. The **Test** folder contains a sample application using that class.

8.2 Web API

CountBOX CCV handling is performed by means of GET- or POST-requests via HTTP/HTTPS. The necessary requirements for device functioning include:

1. Cookies enabled
2. Certificate authentication disabled.

Valid requests list:

1. Authentication
2. Obtaining video analytics parameters
3. Video analytics parameters setup
4. DB cleanup
5. Obtaining/setup device system settings
6. Obtaining/setup network settings
7. Obtaining/setup of host name
8. Obtaining/setup web server settings
9. Updating firmware
10. Report generation
11. Error reset.

Each request has the following format:

*<protocol>://<device IP–address or domain name>:<port>*1*/<path to script>?<parameters>*2. Hereinafter, the *<device_url>* will be used instead of *<protocol>://<device IP–address or domain name>:<port>*

Responses to the requests (unless otherwise stated) are JSON objects and contain **error** and **error_id** fields3. The values of these fields can determine, whether the request is successful or not.

1 The port is specified, if its value is different from the default (**80** for http protocol; **443** for https protocol).

2 For certain requests parameters are not given.

3 The description of possible error codes **error_id** can be found in **Appendix B, "Possible error codes in error_id field".**

1. If both values have zero length, the request is successful

2. In other cases, there is a request, and the **error** value contains error description and **error_id**

contains error code.

To shorten the notation, hereinafter the values of **error** and **error_id** fields will not be given.

Responses to requests may have additional fields that are not described in the documentation. The lines in the output parameters are encoded using UTF-8.

8.2.1 Authentication

Authentication supposes the session creation, transmitting login and password hash to the device. To authenticate follow the steps given below:

1. Perform the similar format request:

<device_url>/new_session.html.

The result of this request is a JSON–object, which contains **device_salt**, **session_salt** (a randomly generated session line) and **api_version** (a line with **"3.1"** value) fields. As a result of this request, the server will return a cookie, which permits identification of the user after authorization

2. Calculate **secret_hash** value using the **sha256** algorithm. This is the hash value based on the concatenation of the user login, user password and **device_salt** (see point 1) values. E.g. ***secret_hash = SHA256 (password + login + device_salt)***
3. Calculate **pass_sha256** value using **sha256** algorithm. This is the hash value based on the concatenation of **secret_hash** (see step 2) and **session_salt.** E.g. **pass_sha256 = SHA256 (secret_hash + session_salt)**
4. Perform the similar format request:

<device_url>/scripts/login.php?login=<user_name>&pass_sha256=<pass_sha256>,

assuming the *<user_name>* — user name (admin or user), the *<pass_sha256>* — value generated under point 3.

Unless otherwise stated, the transmission of cookies generated under point 4 is needed for other Web API requests.

Source code: StatisticsBoxSDK\Device.cs, function Device.Connect.

8.2.2 Parameters of video analytics reception

Video analytics parameters include:

1. The line
2. The circle defining the size of a person in the frame.

The format of obtaining video analytics parameters request:

<device_url>/scripts/algo_params.php[?return_defaults=<defaults_for>],

assuming the *return_defaults* — option may be used to return default video analytics parameters. defaults_for possible values:

1. **all** — returns default values for both the line and circle
2. **line** — returns default values for the line only
3. **target_circle** — returns default values for the circle only. Server response format:

{"circle":"{{x1,y1},{x2,y2}}","line":"{{{x1,y1},{x2,y2},{x3,y3},{x4,y4},.....,{xN,yN}}}" },

assuming the *circle* parameter defines the center of the circle and one point on the circle, the *line*

parameter is a set of points defining the line.

Point coordinates are real values within the range 0..1 with "**.**"(dot) character as a decimal delimiter.

The point in the upper left frame corner has (0.0, 0.0) coordinates. The point in the lower right corner has (1.0, 1.0) coordinates. When drawing and calculating the circle, a radius 4:3 aspect ratio is assumed. Sample response:

{"line":"{{{0.017777778,0.477744807},{0.464444444,0.290801187},{0.873333333,0.388724036}}}", "circle":"{{0.500000000,0.501483680},{0.500000000,0.299703264}}","error":""}.

Source code: StatisticsBoxSDK\Device.cs, function Device.GetAlgoParams.

8.2.3 Parameters of video analytics setup

Video analytics parameters (line and circle) are both set up simultaneously. Setting up the line and circle request format:

<device_url>/scripts/algo_params.php?line={{{x1,y1},{x2,y2},{x3,y3},{x4,y4},...,

{xN,yN}}}&target_circle={{x1,y1},{x2,y2}},

assuming the *line* parameter is a set of line points coordinates and
the *target_circle* parameter is a set of circle center coordinates and a point on this circle coordinates. The line can consist of up to 10 segments. It is recommended to set the circle points so that the radius is not less than 16% of the frame height1. **Source code**: StatisticsBoxSDK\Device.cs, function Device.SetAlgoParams

8.2.4 Database cleanup

DB cleanup can be performed by one of the two possible methods:

1. DB full cleanup
2. Removing DB events within a certain time period.

DB full cleanup request format is:

<device_url>/scripts/cleardb.php.

Request format for removing DB events within a certain time period is:

<device_url>/scripts/cleardb.php?how=bytime&from=YYYY-MM-DD&time_from=HH:MM:SS&to=YYYY- MM-DD&time_to=HH:MM:SS.

The request contains the following parameters:

1. *from* — interval start date
2. *time_from* — interval start time
3. *to* — interval end date
4. *time_to* — interval end time.

If *from* and *time_from* parameters are not determined, DB cleanup will be performed from the earliest event to the date and time, shown in *time_to* and *to* respectively.

Source code: StatisticsBoxSDK\Device.cs, function Device.ClearDb.

1 Description of the circle points setting is provided in chapter **5.4.2, "Object Size Setup"**.

8.2.5 Receiving/setting device system setup

Device system settings include:

1. Device name
2. NTP server address
3. Time
4. Timezone.

The request format for getting system settings is:

<device_url>/scripts/device.php.

In addition to system settings, the request allows the user to get the device serial number and to check the time set. The result of the request execution (in JSON format) contains the following settings:

1. **device_name**: device name
2. **date**: date in dd.mm.yyyy format, where **dd** — is the number without the leading zero, **mm** — is the number of a month (beginning with 1) with a leading zero for Jan. – Sept., and **yyyy** — is the year
3. **time**: time in hh:mm format, where **hh** represents hours (from 00 to 23), and **mm** — represents minutes (from 00 to 59)
4. **ntp**: IP address or DNS–name of the time synchronization server (if the server is not defined, the

null is displayed)

5. **timezone**: the time zone. Listed options: Pacific/Honolulu, America/Anchorage, America/Tijuana, America/Phoenix, America/Mazatlan, America/Mexico_City, America/Chicago, America/New_York, America/Halifax, America/Santiago, America/Argentina/Buenos_Aires, America/Fortaleza, Atlantic/South_Georgia, Atlantic/Azores, Etc/UTC, Europe/Dublin, Europe/Amsterdam, Europe/Brussels, Europe/Kiev, Europe/Tallinn, Asia/Baghdad, Europe/Kaliningrad, Europe/Moscow, Asia/Baku, Asia/Tashkent, Asia/Ashgabat, Asia/Yekaterinburg, Asia/Almaty, Asia/Bangkok, Asia/Novosibirsk, Asia/Omsk, Asia/Hong_Kong, Asia/Krasnoyarsk, Asia/Irkutsk, Asia/Tokyo, Australia/Canberra, Asia/Yakutsk, Asia/Vladivostok, Asia/Magadan, Pacific/Fiji
6. **sn**: the device serial number or a failure of getting serial number message
7. **time_error**: the field, which indicates possible issues with setting time. Possible options are listed below:

a) **no_time_error** — no error
b) **incorrect_time_no_ntp** — incorrect time setting is detected, and NTP server is not defined
c) **incorrect_time_ntp_sync_failed** — incorrect time setting is detected, and synchronization via NTP failed
d) **incorrect_time_ntp_synchronized** — incorrect time setting is detected, following synchronization via NTP that was successfully completed.

In the case of **incorrect_time_ntp_synchronized**, the response will also contain the field **time_difference**, which is equal to the number of seconds of the time difference between the device and NTP server before synchronization. The error resetting operation is described in chapter **8.2.11, "Error resetting"**.

Source code: StatisticsBoxSDK\Device.cs, function Device.GetDeviceSettings.

Request format for setting device system settings is:

<device_url>/scripts/device.php?device_name=<dev_name>&datetime=<dev_date_time>&

timezone=<dev_timezone>&ntp=<ntp_server>,

assuming:

1. *device_name* — the device name
2. *datetime* — the date and time in yyyy–mm–dd hh:mm format
3. *timezone* — the time zone (listed options are provided above)

4. *ntp* — IP address or DNS–name of time synchronization server or null if synchronization is not required.

The result of this request will reset the check box to **"Use hostname as device name"**.

Source code: StatisticsBoxSDK\Device.cs, function Device.SetDeviceSettings.

8.2.6 Receiving/Establishing Network Settings

Network settings include:

1. The device IP address
2. Subnet mask
3. The gateway address
4. The DNS server address
5. The parameter, reporting about getting network settings from the DHCP server.

The request format for receiving network settings is:

<device_url>/scripts/net.php.

The result of the request will be JSON–object with fields:

1. **dhcp_key** — the parameter informing that network settings for current device are received from a DHCP server. Possible options: **on** — network settings are received from a DHCP server, **off** – network settings are set manually
2. **ip_device** — the device IP address
3. **mask** — the subnet mask
4. **gateway** — the gateway IP address by default (0.0.0.0, unless it is preset)
5. **dns** — the DNS server IP address (0.0.0.0, unless it is preset)

6. **mask_manual_edit** — editing the subnet mask manually (option **on**) or not (option **off**). The parameter is used only when **dhcp_key = on**
7. **gateway_manual_edit** — editing the gateway IP address by default manually (option **on**) or not (option **off**). The parameter is used only when **dhcp_key = on**

8. **dns_manual_edit** — editing DNS server IP address manually (value **on**) or not (value **off**). The parameter is used only when **dhcp_key = on**
9. **mac** — the device MAC address.

Source code: StatisticsBoxSDK\Device.cs, function Device.GetNetSettings.

To establish network settings, perform one of the following queries. The format of the request for setting the network settings without using DHCP:

<device_url>/scripts/net.php?dhcp_key=off&dns=<dns_server>&gateway=<gateway_address>

&ip_device=<dev_ip_address>&mask=<dev_mask>,

assuming:

1. *dns* — the DNS server IP address
2. *gateway* — the gateway IP address by default
3. *ip_device* — the device IP address
4. *mask* — the subnet mask.

The validation of parameters should be inspected by the user. The format of the request for setting network configuration with using DHCP is:

<device_url>/scripts/net.php?dhcp_key=on.

For this request, the user can optionally set parameters, whose values will be used instead of the ones received from the DHCP server. These parameters include:

1. DNS
2. Gateway
3. Mask.

The result of the request for setting network settings will restart the network service, which takes about 5 seconds. If the device IP address has been modified, further access to the device will be performed with a new IP address.

Source code: StatisticsBoxSDK\Device.cs, function Device.SetNetSettings.

8.2.7 Receiving/setting the hostname

The format of the request for obtaining the **hostname** is:

<device_url>/scripts/hostname.php.

The result will be JSON–object with the field **hostname***.*

Source code: StatisticsBoxSDK\Device.cs, function Device.GetHostname. The format of the request for setting the **hostname** is:

<device_url>/scripts/hostname.php?hostname=<hostname>,

assuming the *hostname* is a new hostname.

Source code: StatisticsBoxSDK\Device.cs, function Device.SetHostname.

8.2.8 Receiving/setting web server characteristics

Web server parameters include:

1. Port
2. Protocol.

The format of the request for getting web server parameters is:

<device_url>/scripts/webserver.php.

The result of the request will be a JSON–object with the following fields:

1. **port** — number of the port listened by the web server
2. **protocol** — the protocol used by the web server (https or http).

Source code: StatisticsBoxSDK\Device.cs, function Device.GetWebserverSettings. The format of the request for setting web server parameters is:

<device_url>/scripts/webserver.php?port=<port>&protocol=<protocol>,

assuming:

1. *port* — the number of ports listened by the web server
2. *protocol* — the protocol used by the web server (https or http).

Source code: StatisticsBoxSDK\Device.cs, function Device.SetWebserverSettings.

8.2.9 Uploading device updates

Uploading updates on the device is performed by a POST-request. The format of a POST–request for uploading updates is:

<device_url>/scripts/upload_update.php.

The request is carried out by sending a form containing a file upload field with the **file** name (Content- Disposition:form-data;name="file";filename="<arbitrary file name>").

Source code: StatisticsBoxSDK\Device.cs, function Device.UploadUpdate.

8.2.10 Generating a report

The report contains the crossing data. In parameters of report requests, the date is set in the yyyy-mm-dd format, where **yyyy** is year, **mm** — is the month number with or without the leading zero, **dd** — is the date with or without the leading zero. The time is set in the hh:mm:ss format, assuming **hh** represents hours (from 0 to 23) with or without the leading zero, **mm** — represents minutes with the leading zero (i.e. from 00 to 59), **ss** — represents seconds with the leading zero (from 00 to 59).

There are 3 types of statistics request:

1. The format of a request by date allows to get the pass statistics within a certain time period (in days):

device_url>/scripts/xmlGen.php?[export_type=<format>&][how=bydate&]from=yyyy-mm-dd[&to=yyyy-mm-dd],

assuming the parameter *from* specifies the period start date. An optional parameter *to* is the period end date. If the parameter is not specified, *to* is considered to be the date of the last pass. For this type of request the *how* parameter can be omitted

2. The format of a request by date and time allows to get pass statistics within a period limited by date

and time:

<device_url>/scripts/xmlGen.php?[export_type=<format>&]how=bytime&from=yyyy-mm-dd&time_from=hh:mm:ss[&to=yyyy-mm-dd&time_to=hh:mm:ss],

assuming *from* specifies the period start date, *time_from* — the period start time. Optional parameters to, time_to specify the period end date and time accordingly. If these parameters are not specified, the date and time of the last pass are used

3. The format of a request by pass identifier allows to get pass statistics with identifiers within the specified range:

<device_url>/scripts/xmlGen.php?[export_type=<format>&]how=byid&id_from=<id_from>[&id_to=<i

d_to>],

assuming *id_from* is a pass identifier, complying with the start range. Optionally the user can specify *id_to* as a pass identifier of a period end. If *id_to* is not specified, it is considered to be equal to the last pass identifier.

An optional parameter *export_type* (listed options: xml (by default), csv) specifies the format of the generated report. If an invalid value *export_type* is specified, HTTP-code reply will be **"400 Bad Request"**.

The result of these requests will be an XML/CSV-file with pass statistics. If the parameters are malformed, or are not sufficient to generate the report, the answer will contain a description of the error[1]. If the parameters are correct but no passes were recorded within the given period, the document will not contain any pass records.

Source code: StatisticsBoxSDK\Device.cs, function Device.GetReport.

8.2.11 Error reset

There can be mistakes requiring a user's intervention in the device operation process: for example,

time_error[2]. If the error has been corrected by the user, it must be reset. The format of the request for error resetting is:

<device_url>/scripts/clear_error.php.

1 Description of the error message is provided in chapter **5.5.2**, **"XML file Report"** and in **Appendix B, "Possible error codes in error_id field".**

2 Description of this error is provided in chapter **8.2.5, "Receiving/setting device system setup"**.